arXiv:1701.05291v1 [cs.AI] 19 Jan 2017

Heterogeneous Information Network Embedding for Meta Path based Proximity

Zhipeng Huang, Nikos Mamoulis
The University of Hong Kong
{zphuang, nikos}@cs.hku.hk

ABSTRACT

A network embedding is a representation of a large graph in a low-dimensional space, where vertices are modeled as vectors. The objective of a good embedding is to preserve the proximity (i.e., similarity) between vertices in the original graph. This way, typical search and mining methods (e.g., similarity search, kNN retrieval, classification, clustering) can be applied in the embedded space with the help of off-the-shelf multidimensional indexing approaches. Existing network embedding techniques focus on homogeneous networks, where all vertices are considered to belong to a single class. Therefore, they are weak in supporting similarity measures for heterogeneous networks. In this paper, we present an effective heterogeneous network embedding approach for meta path based proximity measures. We define an objective function, which aims at minimizing the distance between two distributions, one modeling the meta path based proximities, the other modeling the proximities in the embedded vector space. We also investigate the use of negative sampling to accelerate the optimization process. As shown in our extensive experimental evaluation, our method creates embeddings of high quality and has superior performance in several data mining tasks compared to state-of-the-art network embedding methods.

Keywords

heterogeneous information network; meta path; network embedding

1. INTRODUCTION

The availability and growth of large networks, such as social networks, co-author networks, and knowledge base graphs, has given rise to numerous applications that search and analyze information in them. However, for very large graphs, common information retrieval and mining tasks such as link prediction, node classification, clustering, and recommendation are time-consuming. This motivated a lot of interest [6, 22, 31] in approaches that *embed* the network into a low-dimensional space, such that the original vertices of the graph are represented as vectors. A good embedding preserves the proximity (i.e., similarity) between vertices in the original graph. Search and analysis can then be applied on the embedding with the help of efficient algorithms and indexing approaches for vector spaces.

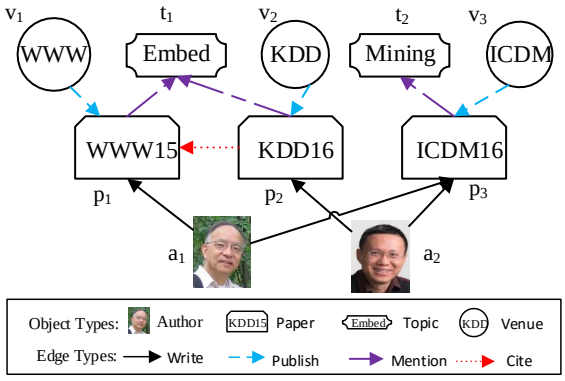


Figure 1: An example of HIN

Heterogeneous information networks (HINs), such as DBLP [15], YAGO [26], DBpedia [2] and Freebase [4], are networks with nodes and edges that may belong to multiple types. These graph data sources contain a vast number of interrelated facts, and they can facilitate the discovery of interesting knowledge [11, 14, 17, 20]. Figure 1 illustrates a HIN, which describes the relationships between objects (graph vertices) of different types (e.g., author, paper, venue and topic). For example, `Jiawei Han` (a_1) has *written* a `WWW` paper (p_1), which *mentions* topic "`Embed`" (t_1).

Compared to homogeneous networks, the relationships between objects in a HIN are much more complex. The proximity among objects in a HIN is not just a measure of closeness or distance, but it is also based on *semantics*. For example, in the HIN in Figure 1, author a_1 is close to both a_2 and v_1, but these relationships have different semantics. a_2 is a co-author of a_1, while v_1 is a venue where a_1 has a paper published.

Meta path [29] is a recently proposed proximity model in HINs. A meta path is a sequence of object types with edge types in between modeling a particular relationship. For example, $A \rightarrow P \rightarrow V \rightarrow P \rightarrow A$ is a meta path, which states that two authors (A) are related by their publications in the same venue (V). Based on meta paths, several proximity measures have been proposed. For example, *PathCount* [29] counts the number of meta path instances connecting the two objects, while *Path Constrained Random Walk (PCRW)* [14] measures the probability that a random walk starting from one object would reach the other via a meta path instance. These measures have been shown to have better perfor-

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

mance compared to proximity measures not based on meta paths, in various important tasks, such as k-NN search [29], link prediction [27, 28, 35], recommendation [34], classification [12, 13] and clustering [30].

Although there are a few works on embedding HINs [7, 21], none of them is designed for meta path based proximity in general HINs. To fill this gap, in this paper, we propose HINE, which learns a transformation of the objects (i.e., vertices) in a HIN to a low-dimensional space, such that the meta path based proximities between objects are preserved. More specifically, we define an appropriate objective function that preserves the relative proximity based rankings the vertices in the original and the embedded space. As shown in [29], meta paths with too large lengths are not very informative; therefore, we only consider meta paths up to a given length threshold l. We also investigate the use of negative sampling [19] in order to accelerate the optimization process.

We conduct extensive experiments on four real HIN datasets to compare our proposed HINE method with state-of-the-art network embedding methods (i.e., LINE [31] and DeepWalk [22]), which do not consider meta path based proximity. Our experimental results show that our HINE method with PCRW as the meta path based proximity measure outperforms all alternative approaches in most of the qualitative measures used.

The contributions of our paper are summarized as follows:

- This is the first work that studies the embedding of HINs for the preservation of meta path based proximities. This is an important subject, because meta path based proximity has been proved to be more effective than traditional structured based proximities.
- We define an objective function, which explicitly aims at minimizing the the distance between two probability distributions, one modeling the meta path based proximities between the objects, the other modeling the proximities in the low-dimensional space.
- We investigate the use of negative sampling to accelerate the process of model optimization.
- We conduct extensive experiments on four real HINs, which demonstrate that our HINE method is the most effective approach for preserving the information of the original networks; HINE outperforms the state-of-the-art embedding methods in many data mining tasks, e.g., classification, clustering and visualization.

The rest of this paper is organized as follows. Section 2 discusses related work. In Section 3, we formally define the problem. Section 4 describes our HINE method. In Section 5, we report our experimental results. Section 6 concludes the paper.

2. RELATED WORK

2.1 Heterogeneous Information Networks

The heterogeneity of nodes and edges in HINs bring challenges, but also opportunities to support important applications. Lately, there has been an increasing interest in both academia and industry in the effective search and analysis of information from HINs. The problem of classifying objects in a HIN by authority propagation is studied in [12]. Follow-up work [13] investigates a collective classification problem in HINs using meta path based dependencies. PathSelClus [30] is a link based clustering algorithm for HINs, in which a user can specify her clustering preference by providing some examples as seeds. The problem of link prediction on HINs has been extensively studied [27, 28, 35], due to its important applications (e.g., in recommender systems). A related problem is entity recommendation in HINs [34], which takes advantage of the different types of relationships in HINs to provide better recommendations.

2.2 Meta Path and Proximity Measures

Meta path [29] is a general model for the proximity between objects in a HIN. Several measures have been proposed for the proximity between objects w.r.t. a given meta path $\mathcal{P}$. PathCount measures the number of meta path instances connecting the two objects, and PathSim is a normalized version of it [29]. Path constrained random walk (PCRW) was firstly proposed [14] for the task of relationship retrieval over bibliographic networks. Later, [17] proposed an automatic approach to learn the best combination of meta paths and their corresponding weights based on PCRW. Finally, HeteSim [25] is recently proposed as an extension of meta path based SimRank. In this paper, we focus on the two most popular proximity measures, i.e., PathCount and PCRW.

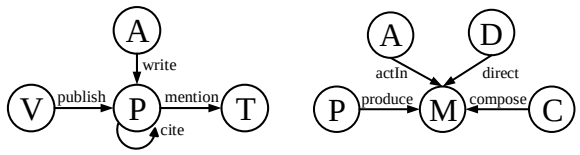
2.3 Network Embedding

Network embedding aims at learning low-dimensional representations for the vertices of a network, such that the proximities among them in the original space are preserved in the low-dimensional space. Traditional dimensionality reduction techniques [3, 8, 24, 32] typically construct the affinity graph using the feature vectors of the vertexes and then compute the eigenvectors of the affinity graph. Graph factorization [1] finds a low-dimensional representation of a graph through matrix factorization, after representing the graph as an adjacency matrix. However, since these general techniques are not designed for networks, they do not necessarily preserve the global network structure, as pointed out in [31].

Recently, DeepWalk [22] is proposed as a method for learning the latent representations of the nodes of a social network, from truncated random walks in the network. DeepWalk combines random walk proximity with the SkipGram model [18], a language model that maximizes the co-occurrence probability among the words that appear within a window in a sentence. However, DeepWalk has certain weaknesses when applied to our problem settings. First, the random walk proximity it adopts does not consider the heterogeneity of a HIN. In this paper, we use PCRW, which extends random walk based proximity to be applied for HINs. Second, as pointed out in [31], DeepWalk can only preserve second-order proximity, leading to poor performance in some tasks, such as link recover and classification, which require first-order proximity to be well-preserved.

LINE [31] is a recently proposed embedding approach for large-scale networks. Although it uses an explicit objective function to preserve the network structure, its performance suffers from the way it learns the vector representations. By design, LINE learns two representations separately, one preserving first-order proximity and the other preserving second-order proximity. Then, it directly concatenates the two vectors to form the final representation. In this paper, we propose a network embedding method, which does not distinguish the learning of first and second-order proximities and embeds proximities of all orders simultaneously.

GraRep [6] further extends DeepWalk to utilize high-order proximities. GraRep does not scale well in large networks due to the expensive computation of the power of a matrix and the involvement of SVD in the learning process. SDNE [33] is a semi-supervised deep model that captures the non-linear structural information over the network. The source code of SDNE is not available, so this ap-



(a) DBLP (b) MOVIE

Figure 2: Schemas of two heterogeneous networks

proach cannot be reproduced and compared to ours. Similarly, [5] embeds entities in knowledge bases using an innovative neural network architecture and TriDNR [21] extends this embedding model to consider features from three aspects of the network: 1) network structure, 2) node content, and 3) label information. Our current work focuses on meta path based proximities, so it does not consider any other information in the HIN besides the network structure and the types of nodes and edges.

3. PROBLEM DEFINITION

In this section, we formulate the problem of heterogeneous information network (HIN) embedding for meta path based proximities. We first define a HIN as follows:

DEFINITION 1. ***(Heterogeneous Information Network)*** *A heterogeneous information network (HIN) is a directed graph $G = (V, E)$ with an object type mapping function $\phi : V \rightarrow \mathcal{L}$ and a link type mapping function $\psi : E \rightarrow \mathcal{R}$, where each object $v \in V$ belongs to an object type $\phi(v) \in \mathcal{L}$, and each link $e \in E$ belongs to a link type $\psi(e) \in \mathcal{R}$.*

Figure 1 illustrates a small bibliographic HIN. We can see that the HIN contains two authors, three papers, three venues, two topics, and four types of links connecting them. For the ease of discussion, we assume that each object belongs to a single type; however, our technique can easily be adapted to the case where objects belong to multiple types.

DEFINITION 2. ***HIN Schema***. *Given a HIN $G = (V, E)$ with mappings $\phi : V \rightarrow \mathcal{L}$ and $\psi : E \rightarrow \mathcal{R}$, the schema T_G of G is a directed graph defined over object types $\mathcal{L}$ and link types $\mathcal{R}$, i.e., $T_G = (\mathcal{L}, \mathcal{R})$.*

The HIN schema expresses all allowable link types between object types. Figure 2(a) shows the schema of the HIN in Figure 1, where nodes A, P, T and V correspond to author, paper, topic and venue, respectively. There are also different edge types in the schema, such as '*publish*' and '*write*'. Figure 2(b) shows the schema of a movie-related HIN, with M, A, D, P, C representing movie, actor, director, producer and composer, respectively.

In a HIN G, two objects o_1, o_2 may be connected via multiple edges or paths. Conceptually, each of these paths represents a specific direct or composite relationship between them. In Figure 1, authors a_1 and a_2 are connected via multiple paths. For example, $a_1 \rightarrow p_3 \rightarrow a_2$ means that a_1 and a_2 are coauthors of paper p_3, and $a_1 \rightarrow p_1 \rightarrow p_2 \rightarrow a_2$ means that a_2's paper p_2 cites a_1's paper p_1. These two paths represent two different relationships between author a_1 and a_2.

Each different type of a relationship is modeled by a *meta path* [29]. Specifically, a meta path $\mathcal{P}$ is a sequence of object types $l_1, \ldots, l_n$ connected by link types $r_1, \ldots, r_{n-1}$ as follows:

$$\mathcal{P} = l_1 \xrightarrow{r_1} l_2 \ldots l_{n-1} \xrightarrow{r_{n-1}} l_n.$$

For example, a meta path $A \xrightarrow{write} P \xrightarrow{write^{-1}} A$ represents the coauthor relationship between two authors.

An *instance* of the meta path $\mathcal{P}$ is a path in the HIN, which conforms to the pattern of $\mathcal{P}$. For example, path $a_1 \rightarrow p_3 \rightarrow a_2$ in Figure 1 is an instance of meta path $A \xrightarrow{write} P \xrightarrow{write^{-1}} A$.

DEFINITION 3. ***(Meta Path based Proximity)*** *Given a HIN $G = (V, E)$ and a meta path $\mathcal{P}$, the proximity of two nodes $o_s, o_t \in V$ with respect to $\mathcal{P}$ is defined as:*

$$s(o_s, o_t \mid \mathcal{P}) = \sum_{p_{o_s \rightarrow o_t} \in \mathcal{P}} s(o_s, o_t \mid p_{o_s \rightarrow o_t}) \quad (1)$$

where $p_{o_s \rightarrow o_t}$ is a meta path instance of $\mathcal{P}$ linking from o_s to o_t, and $s(o_s, o_t \mid p_{o_s \rightarrow o_t})$ is a proximity score w.r.t. the instance $p_{o_s \rightarrow o_t}$.

There are different ways to define $s(o_s, o_t \mid p_{o_s \rightarrow o_t})$ in the literature. Here, we only list two definitions that we use as test cases in our experiments.

- According to the **PathCount (PC)** model, each meta path instance is equally important and should be given equal weight. Hence, $s(o_s, o_t \mid p_{o_s \rightarrow o_t}) = 1$ and the proximity between two objects w.r.t. $\mathcal{P}$ equals the number of instances of $\mathcal{P}$ connecting them, i.e., $s(o_s, o_t \mid \mathcal{P}) = |\{p_{o_s \rightarrow o_t} \in \mathcal{P}\}|$.[1]
- **Path Constrained Random Walk (PCRW)** [14] is a more sophisticated way to define the proximity $s(o_s, o_t \mid p_{o_s \rightarrow o_t})$ based on an instance $p_{o_s \rightarrow o_t}$. According to this definition, $s(o_s, o_t \mid p_{o_s \rightarrow o_t})$ is the probability that a random walk restricted on $\mathcal{P}$ would follow the instance $p_{o_s \rightarrow o_t}$.

Note that the embedding technique we introduce in this paper can also be easily adapted to other meta path based proximity measures, e.g., PathSim [29], HeteSim [25] and BPCRW [17].

DEFINITION 4. ***(Proximity in HIN)*** *For each pair of objects $o_s, o_t \in V$, the proximity between o_s and o_t in G is defined as:*

$$s(o_s, o_t) = \sum_{\mathcal{P}} s(o_s, o_t \mid \mathcal{P}) \quad (2)$$

where $\mathcal{P}$ is some meta path, and $s(o_s, o_t \mid \mathcal{P})$ is the proximity w.r.t. the meta path $\mathcal{P}$ as defined in Definition 3.

According to Definition 4, the proximity of two objects equals the sum of the proximities w.r.t. all meta paths. Intuitively, this can capture all kinds of relationships between the two objects. We now provide a definition for the problem that we study in this paper.

DEFINITION 5. ***(HIN Embedding for Meta Path based Proximity)*** *Given a HIN $G = (V, E)$, develop a mapping $f : V \rightarrow R^d$ that transforms each object $o \in V$ to a vector in R^d, such that the proximities between any two objects in the original HIN are preserved in R^d.*

4. HINE

In this section, we introduce our methodology for embedding HINs. We first discuss how to calculate the truncated estimation of meta path based proximity in Section 4.1. Then, we introduce our model and define the objective function we want to optimize in Section 4.2. Finally, we present a negative sampling approach that accelerates the optimization of the objective function in Section 4.3.

[1]For the ease of presentation, we overload notation $\mathcal{P}$ to also denote the set of instances of meta path $\mathcal{P}$.

Table 1: Meta paths and instances connecting a_1 and a_2

Length	$\mathcal{P}$	$p_{o_s \to o_t}$	s_{PC}	s_{PCRW}
2	APA	$a_1 \to p_3 \to a_2$	1	0.25
3	$APPA$	$a_1 \to p_1 \to p_2 \to a_2$	1	0.5
4	$APTPA$	$a_1 \to p_1 \to t_1 \to p_2 \to a_2$	1	0.25
		$a_1 \to p_3 \to t_2 \to p_3 \to a_2$	1	0.25
	$APVPA$	$a_1 \to p_3 \to v_3 \to p_3 \to a_2$	1	0.25
...	...	...	...	...

4.1 Truncated Proximity Calculation

According to Definition 4, in order to compute the proximity of two objects $o_i, o_j \in V$, we need to accumulate the corresponding meta path based proximity w.r.t. each meta path $\mathcal{P}$. For example, in Table 1, we list some meta paths that have instances connecting a_1 and a_2 in Figure 1. We can see that there is one length-2 meta path, one length-3 meta path and two length-4 meta paths that have instances connecting a_1 and a_2. Generally speaking, the number of possible meta paths grows exponentially with their length and can be infinite for certain HIN schema (in this case, the computation of $s(o_i, o_j)$ is infeasible).

As pointed out in [29], shorter meta paths are more informative than longer ones, because longer meta paths link more remote objects (which are less related semantically). Therefore, we use a truncated estimation of proximity, which only considers meta paths up to a length threshold l. This is also consistent with previous works on network embedding, which aim at reserving low-order proximity (e.g., [22] reserves only second-order proximity, and [31] first-order and second-order proximities). Therefore, we define:

$$\hat{s}_l(o_i, o_j) = \sum_{len(\mathcal{P}) \le l} s(o_i, o_j \mid \mathcal{P}) \tag{3}$$

as the *truncated proximity* between two objects o_i and o_j.

For the case of PCRW based proximity, we have the following property:

PROPERTY 1.

$$\hat{s}_l(o_i, o_j) = \sum_{(o_i, o') \in E} p_{o_i \to o'}^{\psi(o_i, o')} \times \hat{s}_{l-1}(o', o_j)$$

where $p_{o_i \to o'}^{\psi(o_i, o')}$ is the transition probability from o_i to o' w.r.t. edge type $\psi(o_i, o')$. If there are n edges from o_i that belong to edge type $\psi(o_i, o')$, then $\psi(o_i, o') = \frac{1}{n}$.

PROOF. Assume that $p = o_i \to o' \to \cdots \to o_j$ is a meta path instance of length l. According to definition of PCRW:

$$s(o_i, o_j \mid p[1:l]) = p_{o_i \to o'}^{\psi(o_i, o')} \times s(o', o_j \mid p[2:l]) \tag{4}$$

where $p[i:j]$ is the subsequence of path instance p from the i-th to the j-th objects, and $p[2:l]$ is a length $l-1$ path instance. Then, by summing over all the length-l path instances for Equation 4, we get Property 1. □

Based on Property 1, we develop a dynamic programming approach (Algorithm 1) to calculate the truncated proximities. Basically, we compute the proximity matrix $\hat{S}_k$ for each k from 1 to l. We first initialize the proximity matrix $\hat{S}_0$ (lines 1-3). Then, we use the transition function in Property 1 to update the proximity matrix for each k (lines 4-9). If we use PCRW as the meta path based proximity measure, $inc(o_s, o', \hat{S}_{k-1}[o', o_t])$ in line 9 equals $p_{o_s \to o'}^{\psi(o_s, o')} \times \hat{S}_{k-1}[o', o_t]$. The algorithm can also be used for PathCount, in which case we set $inc(o_s, o', \hat{S}_{k-1}[o', o_t]) = \hat{S}_{k-1}[o', o_t]$.

Algorithm 1: Calculate Truncated Proximity

Input: HIN $G = (V, E)$, length threshold l
Output: Truncated Proximity Matrix $\hat{S}_l$

```
1  Ŝ_0 ← ∅
2  for o_s ∈ V do
3  |   Ŝ_0[o_s, o_s] ← 1.0
4  for k ∈ [1 ··· l] do
5  |   Ŝ_k ← ∅
6  |   for o_s ∈ V do
7  |   |   for o' ∈ neighbor(o_s) do
8  |   |   |   for (o', o_t) ∈ Ŝ_{k-1} do
9  |   |   |   |   Ŝ_k[o_s, o_t] ←
   |   |   |   |   Ŝ_k[o_s, o_t] + inc(o_s, o', Ŝ_{k-1}[o', o_t])
10 return Ŝ_l;
```

We now provide a time complexity analysis for computing the truncated proximities on a HIN using Algorithm 1. For each object $o_s \in V$, we need to enumerate all the meta path instances within length l. Suppose the average degree in the HIN is D; then, there are on average l^D such instances. Hence, the total time complexity for proximity calculation is $O(|V| \cdot l^D)$, which is linear to the size of the HIN.

4.2 Model

We now introduce our HINE embedding method, which preserves the meta path based proximities between objects as described above. For each pair of objects o_i and o_j, we use Sigmoid function to define their joint probability, i.e.,

$$p(o_i, o_j) = \frac{1}{1 + e^{-v_i \cdot v_j}} \tag{5}$$

where v_i(or v_j) $\in R^d$ is the low-dimensional representation of object o_i (or o_j). $p : V^2 \to R$ is a probability distribution over a pair of objects in the original HIN.

In the original HIN G, the empirical joint probability of o_i and o_j can be defined as:

$$\hat{p}(o_i, o_j) = \frac{s(o_i, o_j)}{\sum_{o' \in V} s(o_i, o')} \tag{6}$$

To preserve the meta path based proximity $s(\cdot, \cdot)$, a natural objective is to minimize the distance of these two probability distributions:

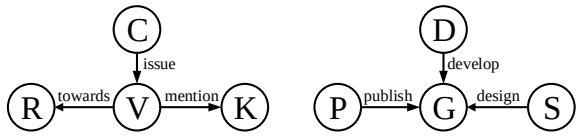
$$O = dist(\hat{p}, p) \tag{7}$$

In this paper, we choose KL-divergence as the distance metric, so we have:

$$O = -\sum_{o_i, o_j \in V} s(o_i, o_j) \log p(o_i, o_j) \tag{8}$$

4.3 Negative Sampling

Directly optimizing the objective function in Equation 8 is problematic. First of all, there is a trivial solution: $v_{i,d} = \infty$ for all i and all d. Second, it is computationally expensive to calculate the gradient, as we need to sum over all the non-zero proximity scores $s(o_i, o_j)$ for a specific object o_i. To address these problems, we adopt negative sampling proposed in [19], which basically samples a small number of negative objects to enhance the influence of positive objects.



(a) YELP (b) GAME

Figure 3: Schemas of Two HINs

Formally, we define an objective function for each pair of objects with non-zero meta path based proximity $s(o_i, o_j)$:

$$T(o_i, o_j) = -\log\left(1 + e^{-v_i v_j}\right) - \sum_{1}^{K} \mathbb{E}_{v' \in P_n(o_i)}[\log\left(1 + e^{v_i v'}\right)] \tag{9}$$

where K is the times of sampling, and $P_n(v)$ is some noise distribution. As suggested by [19], we set $P_n(v) \propto d_{out}(v)^{3/4}$ where $d_{out}(v)$ is the out-degree of v.

We adopt the asynchronous stochastic gradient descent (ASGD) algorithm [23] to optimize the objective function in Equation 9. Specifically, in each round of the optimization, we sample some pairs of objects o_i and o_j with non-zero proximity $s(o_i, o_j)$. Then, the gradient w.r.t. v_i can be calculated as:

$$\frac{\partial O}{\partial v_i} = -s(o_i, o_j) \cdot \frac{e^{-v_i v_j}}{1 + e^{-v_i v_j}} \cdot v_j \tag{10}$$

5. EXPERIMENTS

In this section, we conduct extensive experiments in order to test the effectiveness of our proposed HIN embedding approach. We first introduce our datasets and the set of methods to be compared in Section 5.1. Then, we evaluate the effectiveness of all approaches on five important data mining tasks: network recovery (Section 5.2), classification (Section 5.3), clustering (Section 5.4), k-NN search (Section 5.5) and visualization (Section 5.6). In addition, we conduct a case study (Section 5.7) to compare the quality of top-k lists. We evaluate the influence of parameter l in Section 5.8. Finally, we assess the runtime cost of applying our proposed transformation in Section 5.9.

5.1 Dataset and Configurations

Datasets. We use four real datasets in our evaluation. Table 2 shows some statistics about them.

- **DBLP**. The schema of DBLP network is shown in Figure 2(a). We use a subset of DBLP, i.e., DBLP-4-Area taken of [29], which contains 5,237 papers (P), 5,915 authors (A), 18 venues (V), 4,479 topics (T). The authors are from 4 areas: *database*, *data mining*, *machine learning* and *information retrieval*.

- **MOVIE**. We extracted a subset from YAGO [10], which contains knowledge about movies. The schema of MOVIE network is shown in Figure 2(b). It contains 7,332 movies (M), 10,789 actors (A), 1,741 directors (D), 3,392 producers (P) and 1,483 composers (C). The movies are divided into five genres: *action*, *horror*, *adventure*, *sci-fi* and *crime*.

Table 2: Statistics of Datasets

	$\|V\|$	$\|E\|$	Avg. degree	$\|\mathcal{L}\|$	$\|\mathcal{R}\|$
DBLP	15,649	51,377	6.57	4	4
MOVIE	25,643	40,173	3.13	5	4
YELP	37,342	178,519	9.56	4	3
GAME	6,909	7,754	2.24	4	3

- **YELP**. We extracted a HIN from YELP, which is about reviews given to restaurants. It contains 33,360 reviews (V), 1,286 customers (C), 82 food-related keywords (K) and 2,614 restaurants (R). We only extract restaurants from one of the following types: *fast food*, *American*, *sushi bar*. The schema of the HIN is shown in Figure 3(a).

- **GAME**. We extracted from Freebase [4] a HIN, which is related to video games. The HIN consists of 4,095 games (G), 1,578 publishers (P), 2,043 developers (D) and 197 designers (S). All the game objects extracted are of one of the three genres: *action*, *adventure*, and *strategy*. The schema is shown in Figure 3(b).

Competitors. We compare the following network embedding approaches:

- **DeepWalk** [22] is a recently proposed social network embedding method (see Section 2.3 for details). In our experiment settings, we ignore the heterogeneity and directly feed the HINs for embedding. We use default training settings, i.e., window size $w = 5$ and length of random walk $t = 40$.

- **LINE** [31] is a method that preserves first-order and second-order proximities between vertices (see Section 2.3 for details). For each object, it computes two vectors; one for the first-order and one for the second-order proximities separately and then concatenates them. We use equal representation lengths for the first-order and second-order proximities, and use the default training settings; i.e., number of negative samplings. $K = 5$, starting value of the learning rate $\rho_0 = 0.025$, and total number of training samplings $T = 100M$. Same as DeepWalk, we directly feed the HINs for embedding.

- **HINE_PC** is our HINE model using PathCount as the meta path based proximity. We implemented the process of proximity computing in C++ on a 16GB memory machine with Intel(R) Core(TM) i5-3570 CPU @ 3.4 GHz. By default, we use $l = 2$. In Section 5.8, we study the influence of parameter l.

- **HINE_PCRW** is our HINE model using PCRW as the meta path based proximity. All the other configurations are the same as those of HINE_PC.

Unless otherwise stated, the dimensionality d of the embedded vector space equals 10.

5.2 Network Recovery

We first compare the effectiveness of different network embedding methods at a task of link recovery. For each type of links (i.e., edges) in the HIN schema, we enumerate all pairs of objects (o_s, o_t) that can be connected by such a link and calculate their proximity in the low-dimensional space after embedding o_s to v_s and o_t to v_t. Finally, we use the area under ROC-curve (AUC)

to evaluate the performance of each embedding. For example, for edge type *write*, we enumerate all pairs of authors a_i and papers p_j in DBLP and compute the proximity for each pair. Finally, using the real DBLP network as ground-truth, we compute the AUC value for each embedding method.

The results for $d = 10$ are shown in Table 3. Observe that, in general, HINE_PC and HINE_PCRW have better performance compared to LINE and DeepWalk. In order to analyze the reasons behind the bad performance of LINE, we also included two special versions of LINE: **LINE_1st** (**LINE_2nd**) is a simple optimization approach that just uses stochastic gradient descent to optimize just the first-order (second-order) proximity among the vertices. LINE_1st has much better performance than LINE_2nd because second-order proximity preservation does not facilitate link prediction. LINE, which concatenates LINE_1st and LINE_2nd vectors, has worse performance than LINE_1st because its LINE_2nd vector component harms link prediction accuracy. This is consistent with our expectation, that training proximities of multiple orders simultaneously is better than training them separately. Among all methods, HINE_PCRW has the best performance in preserving the links of HINs; in the cases where HINE_PCRW loses by other methods, its AUC is very close to them. HINE_PCRW outperforms HINE_PC, which is consistent with results in previous work that find PCRW superior to PC (e.g., [10]). The rationale is that a random walk models proximity as probability, which naturally weighs nearby objects higher compared to remote ones.

5.3 Classification

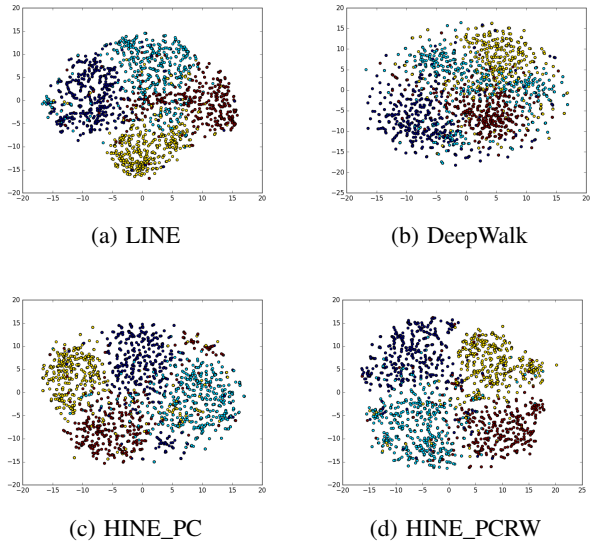
Table 5: Results of Classification with $d = 10$

		LINE	DeepWalk	HINE_PC	HINE_PCRW
DBLP	Macro-F1	0.816	0.839	0.844	**0.868**
	Micro-F1	0.817	0.840	0.844	**0.869**
MOVIE	Macro-F1	0.324	0.280	0.346	**0.365**
	Micro-F1	0.369	0.328	0.387	**0.406**
YELP	Macro-F1	**0.898**	0.339	0.470	0.886
	Micro-F1	**0.887**	0.433	0.518	0.878
GAME	Macro-F1	0.451	**0.487**	0.415	0.438
	Micro-F1	0.518	**0.545**	0.501	0.518
Overall	Macro-F1	0.622	0.486	0.519	**0.639**
	Micro-F1	0.648	0.537	0.563	**0.668**

Table 6: Results of Clustering with $d = 10$

NMI	LINE	DeepWalk	HINE_PC	HINE_PCRW
DBLP	0.3920	**0.4896**	0.4615	0.4870
MOVIE	0.0233	0.0012	**0.0460**	**0.0460**
YELP	**0.0395**	0.0004	0.0015	0.0121
GAME	0.0004	0.0002	**0.0022**	0.0004
Overall	0.1138	0.1229	0.1278	**0.1364**

We conduct a task of multi-label classification. For DBLP, we use the areas of authors as labels. For MOVIE, we use the genres of movies as labels. For YELP, we use the restaurant types as labels. For GAME, we use the type of games as labels. We first use different methods to embed the HINs. Then, we randomly partition the samples into a training and a testing set with ratio $4 : 1$. Finally, we use k nearest neighbor ($k-$NN) classifiers with $k = 5$ to predict the labels of the test samples. We repeat the process for 10 times and compute the average Macro-F1 and Micro-F1 scores to evaluate the performance.



(a) LINE (b) DeepWalk

(c) HINE_PC (d) HINE_PCRW

Figure 6: Visualization Results on DBLP

Table 5 shows the results for $d = 10$. We can see that all methods have better results on DBLP and YELP, than on MOVIE and GAME. This is because the average degree on MOVIE and GAME is smaller, and the HINs are sparser (See Table 2). Observe that HINE_PC and HINE_PCRW outperform DeepWalk and LINE in DBLP and MOVIE; HINE_PCRW performs the best in both datasets. On YELP, LINE has slightly better results than HINE_PCRW (about 1%), while DeepWalk has a very poor performance. On GAME, DeepWalk is the winner, while LINE and HINE_PCRW perform similarly. Overall, we can see that HINE_PCRW has the best (or close to the best) performance, and LINE has quite good performance. On the other hand, DeepWalk's performance is not stable.

We also measured the performance of the methods for different values of d. Figures 5(a) and 5(b) show the results on DBLP. As d becomes larger, all approaches get better in the beginning, and then converge to a certain performance level. Observe that overall, HINE_PCRW has the best performance in classification tasks.

5.4 Clustering

We also conducted clustering tasks to assess the performance of the methods. In DBLP we cluster the authors, in MOVIE we cluster the movies, in YELP we cluster the restaurants, and in GAME we cluster the games. We use the same ground-truth as in Section 5.3. We use normalized mutual information (NMI) to evaluate performance.

Table 6 shows the results for $d = 10$. We can see that the performance of all methods in clustering tasks is not as stable as that in classification. All the embedding methods perform well on DBLP, but they have a relatively bad performance on the other three datasets. On DBLP, DeepWalk and HINE have better performance than LINE. On MOVIE, HINE_PC and HINE_PCRW outperform all other approaches. On YELP, LINE has better performance than HINE_PCRW, and DeepWalk has very poor performance. On GAME, DeepWalk outperforms the others. Overall, we can see that HINE_PCRW outperforms all the other methods in the task of clustering.

Figure 5(c) shows performance of the approaches for different d values on DBLP. Observe that HINE_PCRW in general outper-

Table 3: Accuracy of Network Recovery ($d = 10$)

	Edge Type	LINE_1st	LINE_2nd	LINE	DeepWalk	HINE_PC	HINE_PCRW
DBLP	*write*	0.9885	0.7891	0.8907	**0.9936**	0.9886	0.9839
	publish	0.9365	0.6572	0.8057	0.9022	0.9330	**0.9862**
	mention	**0.9341**	0.5455	0.7186	0.9202	0.8965	0.8879
	cite	**0.9860**	0.9071	0.9373	0.9859	0.9598	0.9517
MOVIE	*actIn*	0.9648	0.5214	0.8425	0.7497	0.9403	**0.9733**
	direct	0.9420	0.5353	0.8468	0.7879	0.9359	**0.9671**
	produce	0.9685	0.5334	0.8599	0.7961	0.9430	**0.9900**
	compose	0.9719	0.5254	0.8574	0.9475	0.9528	**0.9864**
YELP	*issue*	0.9249	0.4419	0.8744	**1.0000**	0.5083	0.9960
	towards	0.9716	0.4283	0.9221	0.5372	0.5155	**0.9981**
	mention	**0.8720**	0.4178	0.7738	0.5094	0.5951	0.8534
GAME	*design*	0.9599	0.4997	0.6781	0.9313	0.9219	**0.9828**
	develop	0.9021	0.5043	0.7736	0.9328	0.8592	**0.9843**
	publish	0.7800	0.4719	0.6786	0.9711	0.7617	**0.9756**
Overall	Average	0.9359	0.5556	0.8185	0.8546	0.8365	**0.9655**

forms all other methods. Only for a narrow range of d ($d = 10$) DeepWalk slightly outperforms HINE_PCRW (as also shown in Table 6). Generally speaking, HINE_PCRW best preserves the proximities among authors in the task of clustering.

5.5 k-NN Search

We compare the performances of three methods, i.e., LINE, DeepWalk and HINE_PCRW, on k-NN search tasks. Specifically, we conduct a case study on DBLP, to compare the quality of k nearest *authors* for venue *WWW* in the embedded space. We first evaluate the quality of k-NN search by counting the average number of papers that the authors in the k-NN result have published in WWW, when varying k from 1 to 100 (Figure 4(a)). We can see that the nearest authors found in the embedding by HINE_PCRW have more papers published in WWW compared to the ones found in the spaces of LINE and DeepWalk.

We then use the top-k author list for venue WWW in the original DBLP network as ground-truth. We use two different metrics to evaluate the quality of top-k lists in the embedded space, i.e., Spearman's footrule $\mathcal{F}$ and Kendall's tau $\mathcal{K}$ [9]. The results are shown in Figures 4(b) and 4(c). We can see that the top-k list of HINE_PCRW is closer to the one in the original HIN.

5.6 Visualization

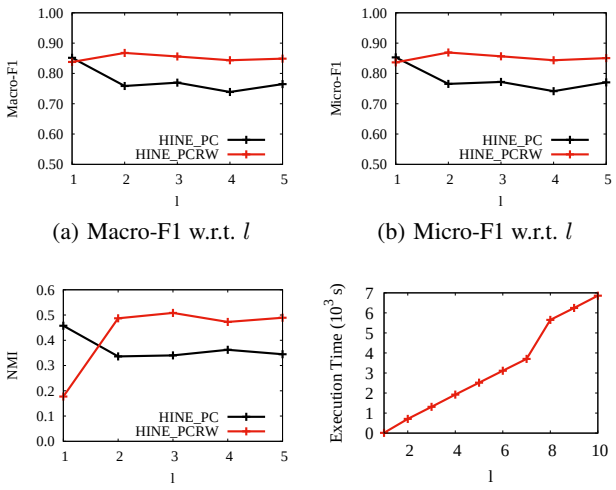
We compare the performances of all approaches on the task of visualization, which aims to layout an HIN on a 2-dimensional space. Specifically, we first use an embedding method to map DBLP into a vector space, then, we map the vectors of authors to a 2-D space using the t-SNE [16] package.

The results are shown in Figure 6. LINE can basically separate the authors from different groups (represented by the same color), but some blue points mixed with other groups. The result of DeepWalk is not very good, as many authors from different groups are mixed together. HINE_PC clearly has better performance than DeepWalk. Compared with LINE, HINE_PC can better separate different groups. Finally, HINE_PCRW's result is the best among these methods, because it clearly separates the authors from different groups, leaving a lot of empty space between the clusters. This is consistent with the fact that HINE_PCRW has the best performance in classifying the authors of DBLP.

5.7 Case Study

We perform a case study, which shows the k-NN objects to a given object in DBLP. Specifically, we show in Table 4 the top-5 closest *venues*, *authors* and *topics* to author *Jiawei Han* in DBLP.

By looking at the results for top-5 venues, we can see that LINE does not give a good top-5 list, as it cannot rank KDD in the top publishing venues of the author. DeepWalk is slightly better, but it ranks ICDM at the 5th position, while HINE_PCRW ranks KDD and ICDM as 1st and 2nd, respectively. Looking at the results for authors, observe that HINE_PCRW gives a similar top-5 list to DeepWalk, except that HINE_PCRW prefers top researchers, e.g., Christos Faloutsos. Looking the results for topics, note that only HINE_PCRW can provide a general topic like "mining".



(a) Macro-F1 w.r.t. l (b) Micro-F1 w.r.t. l

(c) NMI w.r.t. l (d) Execution time w.r.t. l

Figure 7: Performances w.r.t. l

5.8 Effect of the l Threshold

We now study the effect of l on different data mining tasks, e.g., classification and clustering. Figure 7(a) and (b) shows the results of classification on DBLP. We can see that 1) HINE_PCRW has quite stable performance w.r.t. l, and it achieves its best performance when $l = 2$. 2) The performance of HINE_PC is best when $l = 1$. This is because PathCount views meta path instances of different length equally important, while PCRW assigns smaller

weights to the longer instances when performing the random walk. This explains why HINE_PCRW outperforms HINE_PC in these tasks.

Figure 7(c) shows the results of clustering on DBLP. The results are similar those of classification, except that HINE_PC has much better performance than HINE_PCRW when $l = 1$. This is natural, because when $l = 1$, HINE_PCRW can only capture very local proximities. When $l > 1$, HINE_PCRW outperforms HINE_PC.

5.9 Efficiency

We show in Figure 7(d) the running time for computing all-pair truncated proximities on DBLP with the method described in Section 4.1. We can see that our proposed algorithm can run in a reasonable time and scales well with l. Specifically, in our experiments with $l = 2$, the time for computing all-pair proximities is less than 1000s. In addition, note that using ASGD to solve our objective function in our experiments is very efficient, taking on average 27.48s on DBLP with $d = 10$.

6. CONCLUSION

In this paper, we study the problem of heterogeneous network embedding for meta path based proximity, which can fully utilize the heterogeneity of the network. We also define an objective function, which aims at minimizing the distance of two distributions, one modeling the meta path based proximities, the other modeling the proximities in the embedded low-dimensional space. We also investigate using negative sampling to accelerate the optimization process. As shown in our extensive experiments, our embedding methods can better recover the original network, and have better performances over several data mining tasks, e.g., classification, clustering and visualization, over the state-of-the-art network embedding methods.

7. REFERENCES

[1] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola. Distributed large-scale natural graph factorization. In *WWW*, pages 37–48. ACM, 2013.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. In *The Semantic Web*, pages 722–735. Springer, 2007.

[3] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, volume 14, pages 585–591, 2001.

[4] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250, 2008.

[5] A. Bordes, J. Weston, R. Collobert, and Y. Bengio. Learning structured embeddings of knowledge bases. In *AAAI*, 2011.

[6] S. Cao, W. Lu, and Q. Xu. Grarep: Learning graph representations with global structural information. In *CIKM*, pages 891–900. ACM, 2015.

[7] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang. Heterogeneous network embedding via deep architectures. In *SIGKDD*, pages 119–128. ACM, 2015.

[8] T. F. Cox and M. A. Cox. *Multidimensional scaling*. CRC press, 2000.

[9] R. Fagin, R. Kumar, and D. Sivakumar. Comparing top k lists. *SIAM Journal on Discrete Mathematics*, 17(1):134–160, 2003.

[10] Z. Huang, Y. Zheng, R. Cheng, Y. Sun, N. Mamoulis, and X. Li. Meta structure: Computing relevance in large heterogeneous information networks. In *SIGKDD*, pages 1595–1604, 2016.

[11] N. Jayaram, A. Khan, C. Li, X. Yan, and R. Elmasri. Querying knowledge graphs by example entity tuples. *TKDE*, 27(10):2797–2811, 2015.

[12] M. Ji, J. Han, and M. Danilevsky. Ranking-based classification of heterogeneous information networks. In *SIGKDD*, pages 1298–1306. ACM, 2011.

[13] X. Kong, P. S. Yu, Y. Ding, and D. J. Wild. Meta path-based collective classification in heterogeneous information networks. In *CIKM*, pages 1567–1571. ACM, 2012.

[14] N. Lao and W. W. Cohen. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67, 2010.

[15] M. Ley. The dblp computer science bibliography: Evolution, research issues, perspectives. In *SPIRE*, pages 1–10. Springer, 2002.

[16] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[17] C. Meng, R. Cheng, S. Maniu, P. Senellart, and W. Zhang. Discovering meta-paths in large heterogeneous information networks. In *WWW*, pages 754–764. ACM, 2015.

[18] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[19] T. Mikolov and J. Dean. Distributed representations of words and phrases and their compositionality. pages 3111–3119, 2013.

[20] D. Mottin, M. Lissandrini, Y. Velegrakis, and T. Palpanas. Exemplar queries: Give me an example of what you need. *PVLDB*, 7(5):365–376, 2014.

[21] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang. Tri-party deep network representation. In *IJCAI*, pages 1895–1901, 2016.

[22] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *SIGKDD*, pages 701–710. ACM, 2014.

[23] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, pages 693–701, 2011.

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

[25] C. Shi, X. Kong, Y. Huang, S. Y. Philip, and B. Wu. Hetesim: A general framework for relevance measure in heterogeneous networks. *TKDE*, 26(10):2479–2492, 2014.

[26] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*, pages 697–706, 2007.

[27] Y. Sun, R. Barber, M. Gupta, C. C. Aggarwal, and J. Han. Co-author relationship prediction in heterogeneous bibliographic networks. In *ASONAM*, pages 121–128. IEEE, 2011.

[28] Y. Sun, J. Han, C. C. Aggarwal, and N. V. Chawla. When will it happen?: relationship prediction in heterogeneous information networks. In *WSDM*, pages 663–672. ACM, 2012.

[29] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *PVLDB*, 4(11):992–1003, 2011.

[30] Y. Sun, B. Norick, J. Han, X. Yan, P. S. Yu, and X. Yu. Pathselclus: Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. *TKDD*, 7(3):11, 2013.

[31] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *WWW*, pages 1067–1077. ACM, 2015.

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

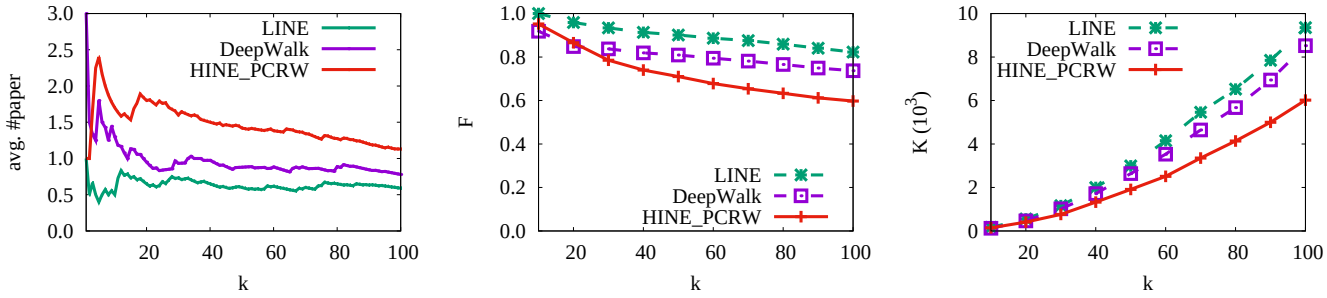
[33] D. Wang, P. Cui, and W. Zhu. Structural deep network embedding. In *SIGKDD*, pages 1225–1234. ACM, 2016.

[34] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han. Personalized entity recommendation: A heterogeneous information network approach. In *WSDM*, pages 283–292. ACM, 2014.

[35] J. Zhang, P. S. Yu, and Z.-H. Zhou. Meta-path based multi-network collective link prediction. In *SIGKDD*, pages 1286–1295. ACM, 2014.

Table 4: Top-5 similar lists for *Jiawei Han*

	LINE			DeepWalk			HINE_PCRW		
k	conf	author	topic	conf	author	topic	conf	author	topic
1	PAKDD	Jiawei Han	clustermap	KDD	Jiawei Han	itemsets	KDD	Jiawei Han	closed
2	PKDD	Gao Cong	orders	PAKDD	Xifeng Yan	farmer	ICDM	Xifeng Yan	mining
3	SDM	Lei Liu	summarizing	PKDD	Ke Wang	closed	SDM	Ke Wang	itemsets
4	KDD	Yiling Yang	ccmine	SDM	Yabo Xu	tsp	PAKDD	Christos Faloutsos	sequential
5	ICDM	Daesu Lee	concise	ICDM	Jason Flannick	prefixspan	PKDD	Xiaoxin Yin	massive



(a) Avg. #paper w.r.t. k (b) $\mathcal{F}$ w.r.t. k (c) $\mathcal{K}$ w.r.t. k

Figure 4: Results of Top-k lists for venue *WWW*

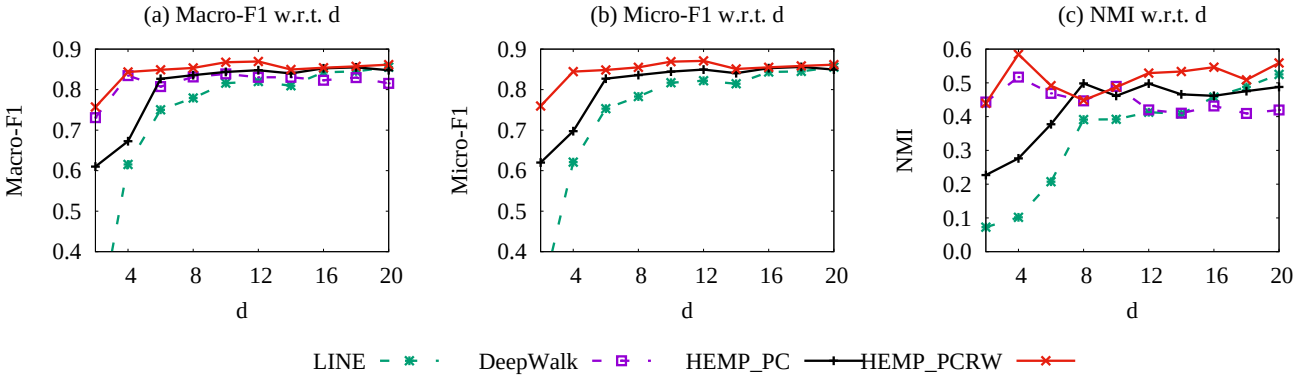


Figure 5: Performance of Classification and Clustering w.r.t. d