

# Predicting Neighbor Label Distributions in Dynamic Heterogeneous Information Networks

Yuchi Ma\* · Ning Yang\* · Lei Zhang ·  
Philip S. Yu

Received: date / Accepted: date

**Abstract** In Dynamic Heterogeneous Information Networks (DHINs), predicting neighbor label distribution is important for a variety of applications. For example, when a user changes job, the composition of the user's friends can change, hence the profession distribution of his/her social circle may change. If we can accurately predict the change of the distribution, we will be able to improve the quality of personal services for him/her. The challenges of predicting neighbor label distribution mainly come from four aspects: infinite state space of neighbor label distributions, link sparsity, the complexity of link formation preferences, and the stream of DHIN snapshots. To address these challenges, we propose a Latent Space Evolution Model (LSEM) for the prediction of neighbor label distribution, which builds a Neighbor Label Distribution Matrix (NLDM) for each type of labels of neighbors of given nodes. LSEM can predict the next NLDM by reconstructing it from two latent feature matrices estimated by their respective autoregressive models. The experiments conducted on real datasets verify the effectiveness of LSEM and the efficiency of the proposed algorithm.

---

\*These authors equally contributed to this study and share the first authorship.

Yuchi Ma  
College of Computer Science, Sichuan University, China,  
E-mail: scu.Richard.Ma@gmail.com

Ning Yang (corresponding author)  
College of Computer Science, Sichuan University, China,  
E-mail: yangning@scu.edu.cn

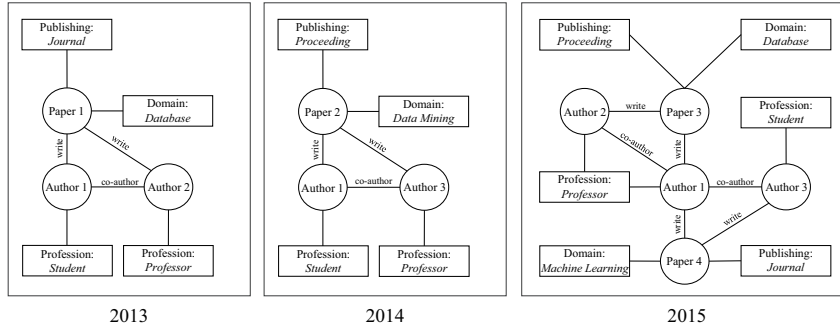
Lei Zhang  
College of Computer Science, Sichuan University, China,  
E-mail: Leizhang@scu.edu.cn

Philip S. Yu  
Department of Computer Science, University of Illinois at Chicago, USA,  
Institute for Data Science, Tsinghua University, Beijing, China,  
E-mail: psyu@uic.edu

**Keywords** Dynamic Heterogeneous Information Network · Neighbor Label Distribution Prediction · Latent Space Evolution Model

## 1 Introduction

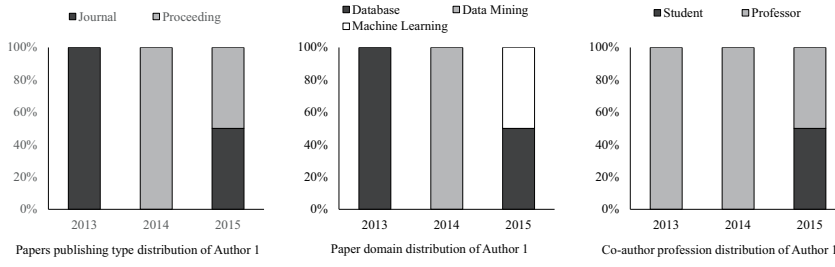
In recent years, prediction in dynamic heterogeneous information networks (DHINs) has become a hot topic owing to the development of social networks (e.g., Facebook, Twitter and Foursquare). Existing works mainly focus on predicting the rating [27,30,9,33] or the formation [11,20,21,42,39] of a single link. Some other works focus on predicting the labels of a single node [48,5,17,3,10]. In this paper, we aim to address the problem of the prediction of the neighbor label distributions of given nodes in a DHIN.



**Fig. 1** A stream of co-author network snapshots

In a DHIN, a node might have neighbors with different types in different periods. For example, Fig. 1 shows a stream of co-author network snapshots, where the nodes and labels are represented by circles and rectangles, respectively. As we can see from Fig. 1, there are two types of nodes, Author nodes and Paper nodes, and each type of nodes may have multiple types of labels. For example, in Fig. 1, an Author node has a Profession label (e.g., the label "Student" of Author 1), while a Paper node has a domain label (e.g., the label "Database" of Paper 1) and a publishing category label (e.g., the label "Journal" of Paper 1). For a given node, we can build a distribution for each type of the labels of its neighbor nodes, which we call neighbor label distribution. For example, there are three types of neighbor label distributions of Author 1 in Fig. 1, where each type of neighbor label distribution is represented by a 100% stacked as shown in Fig. 2. As we can see from Fig. 2, the neighbor label distributions are time-evolving.

Predicting neighbor label distributions is valuable for a variety of applications. For example, it is important for a scholar to keep up with the academic frontier by predicting the distributions of the research interests of leading scholars and following them. For more examples, a location-based social network may want to predict regions a user will go to tomorrow so as to



**Fig. 2** Neighbor label distributions

recommend venues for the user. A movie rating network may want to predict the spending power distribution of the potential consumers on a new TV play for making a proper price of the play. A social media network may want to predict the interested topic distribution of each user so as to design his/her personalized start page for each log-in.

However, predicting neighbor label distributions is not easy due to the following challenges.

- **Infinite state space of neighbor label distributions:** The number of possible states of a neighbor label distribution is infinite, because fractions in a neighbor label distribution are real numbers. Temporal models such as Markov chain-based models and time varying relational classifier [10, 34] cannot serve our goal because they often assume a finite state space.

- **Sparsity of links:** In real-world networks, the number of links between a given node and its neighbors is relatively small compared with the huge volume of nodes, e.g., "publishing" in DBLP and "checking in" in Foursquare. The sparsity of links makes it difficult to mine sufficient meaningful patterns for individuals.

- **The complexity of link formation preference:** A neighbor label distribution essentially reflects the link formation preference of a given node to the nodes with a specific type, and the evolution of the distribution results from the preference shifting. For example, in a co-author network, the paper domain distribution of an author changes with his/her research interests. Some previous researches [7, 19] indicate that the link formation preference might be very complicated as it can be affected by multiple factors, e.g., social circle, profession, age, and so on. Therefore, it is difficult to accurately model the link formation preference of a node by using specific features.

- **Stream of DHIN snapshots:** As we can see from Fig. 1, neighbor label distributions of a given node evolve over a stream of DHIN snapshots. Sometimes we may want to get real-time response to each snapshot. For example, the recommender of a video website might want to predict the genre distribution of users' playlists so as to arrange personalized start pages for them in real time. Many traditional methods are not applicable to make a prediction by only using the latest information.

To address these challenges, we propose a Latent Space Evolution Model (LSEM) for the prediction of neighbor label distribution, based on which the

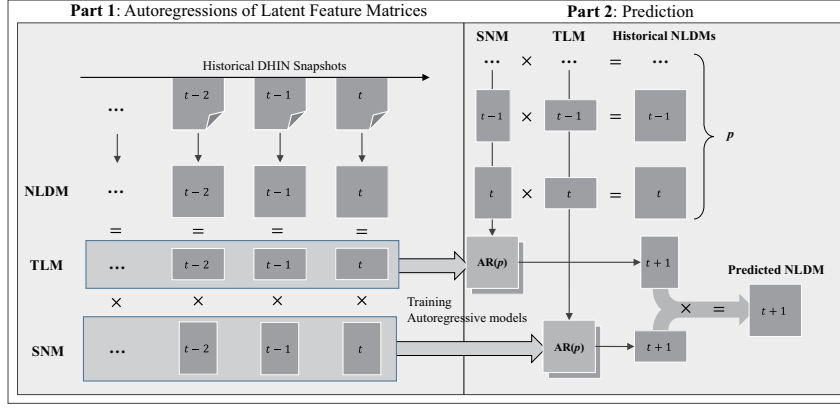


Fig. 3 Overview of LSEM

time cost of the predictions is linear with the number of nodes and independent of the number of DHIN snapshots. The main idea of LSEM is to model neighbor label distributions of given nodes as a Neighbor Label Distribution Matrix (NLDM) and make prediction of next NLDM by autoregressions of latent features. We call the given nodes *source nodes* and the labels *target labels*. Fig. 3 gives an overview of LSEM. In the first part, to deal with the link sparsity and the complexity of link formation preference, LSEM factorizes each historical NLDM into a Source Node latent feature Matrix (SNM) and a Target Label latent feature Matrix (TLM). A row vector of SNM implicitly represents the link formation preference of a source node, while a row vector of TLM represents the latent feature of a target label. By the factorization, we can model the link formation preference as a dense latent feature vector, even if the links around a node in a DHIN snapshot are sparse. To deal with the stream of DHIN snapshots, we fit the historical TLMs and SNMs by two  $p$ -order autoregressive models. As shown in the second part in Fig. 3, LSEM makes a prediction of the next NLDM by reconstructing it from the next SNM and TLM which are estimated from the latest  $p$  historical SNMs and TLMs by the autoregressive models, respectively.

Our previous work [25] presents the problem of prediction of neighbor label distribution in an HIN, and proposes an Evolution Factor Model (EFM) for it. There are two major differences between LSEM and EFM. First, LSEM is an incremental model which makes a prediction based on the latest  $p$  snapshots, while EFM is a batch model which makes a prediction based on the whole history of an HIN. Second, LSEM takes into consideration the evolution of link formation preference of source nodes, as it is designed for a stream of DHIN snapshots, while EFM does not as it is for an HIN of which the history is treated holistically.

Our contributions can be summarized as follows:

1. We propose a Latent Space Evolution Model (LSEM) for the prediction of neighbor label distribution in DHINs. The time complexity of prediction

algorithm based on LSEM is linear with the number of source nodes and independent of the number of DHIN snapshots.

2. We propose two  $p$ -order autoregressive models for the estimation of the next SNM and TLM, from which the NLDM can be reconstructed.
3. We compare LSEM with six baseline methods on four real datasets. The results verify the effectiveness and efficiency of the proposed model and algorithm.

The rest of this paper is organized as follows. We give the preliminaries in Section 2. We describe NLDM and formalize the problem of neighbor label distribution prediction in Section 3. We present the details of LSEM in Section 4. We present the experimental results and analysis in Section 5. Finally, we discuss related works in Section 6, and conclude in Section 7.

## 2 Preliminaries

In this paper, a vector is denoted by a boldface lowercase letter (e.g.,  $\mathbf{v}$ ), where the  $i$ th cell is denoted by  $x^{(i)}$ . A matrix is denoted by a boldface capital letter (e.g.,  $\mathbf{X}$ ), where the cell at  $i$ th row and  $j$ th column of  $\mathbf{X}$  is denoted by  $x^{(i,j)}$ .

We denote a heterogeneous information network by  $G = \langle V, E \rangle$ , where  $V$  is the node set and  $E$  is the link set. A Dynamic Heterogeneous Information Network (DHIN) can be segmented into a stream of DHIN snapshots denoted by  $\mathcal{G} = \langle G_1, G_2, \dots, G_t, \dots \rangle$ , where  $G_t$  represents the snapshot of  $G$  at time  $t$  [38]. For example, in the co-author network case, each  $G_t$  is a network comprised of all papers published in year  $t$ , as well as the authors linking to them. The neighbor label distribution is formally defined as follow:

**Definition 1 (Neighbor Label Distribution):** For a given node  $v$  of type  $V_1$ , its neighbor label distribution over labels of type  $L$  of neighbor nodes of type  $V_2$  is a vector

$$(x^{(1)}, \dots, x^{(j)}, \dots, x^{(M)}),$$

where  $M$  is the number of difference labels of label type  $L$ , and an entry  $x^{(j)}$ ,  $1 \leq j \leq M$ , is the occurrence fraction of label  $j$  of type  $L$ , and  $\sum_{j=1}^M x^{(j)} = 1$ . The given node  $v$  is called **source node**, and the labels of type  $L$  are called **target labels**.

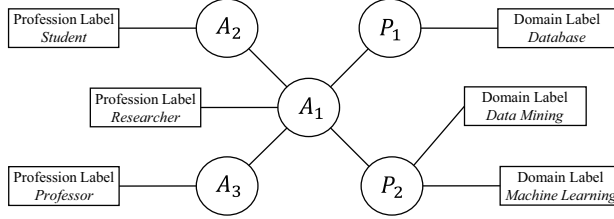
## 3 Neighbor Label Distribution Matrix

For a given node, it has a neighbor label distribution uniquely corresponding to each possible combination of a neighbor node type and a neighbor label type. This corresponding relation can be modeled by a 2-hop meta-path [37] from a node type to a label type, which we call preference meta-path. The definition of the preference meta-path is given in Definition 2.

**Definition 2 (Preference Meta-Path):** A preference meta-path, denoted by  $\mathcal{R}$ , is a 2-hop relation from a node type to a label type,

$$\mathcal{R} = V_1 \rightarrow V_2 \rightarrow L, \quad (1)$$

where  $V_1$  is a source node type,  $V_2$  is a neighbor node type, and  $L$  is a label type.



**Fig. 4** A snapshot of a co-author network

A preference meta-path uniquely defines a type of neighbor label distributions. For example, in Fig. 4, the paper domain distribution of authors can be defined by the preference meta-path,  $\mathcal{R} = Author \rightarrow Paper \rightarrow Domain$ , and an instance of it is  $A_1 \rightarrow P_2 \rightarrow Machine\ Learning$ .

Based on the concept of preference meta-path, we propose Neighbor Label Distribution Matrix (NLDM) to model the neighbor label distributions of source nodes. The formal definition of NLDM is given as follow:

**Definition 3 (Neighbor Label Distribution Matrix (NLDM)):** The NLDM of a preference meta-path  $\mathcal{R} = V_1 \rightarrow V_2 \rightarrow L$  at time  $t$ , denoted by  $\mathbf{X}_t(\mathcal{R})$ , is a matrix of  $\mathbb{R}^{N \times M}$ , where  $N$  is the number of source nodes of type  $V_1$  and  $M$  is the number of different labels of type  $L$ . The  $i$ th row vector in  $\mathbf{X}_t(\mathcal{R})$ , denoted by  $\mathbf{X}_t^{(i)}(\mathcal{R})$ , represents a neighbor label distribution, i.e.,


$$\mathbf{X}_t^{(i)}(\mathcal{R}) = (x^{(i,1)}, \dots, x^{(i,j)}, \dots, x^{(i,M)}),$$

where  $x^{(i,j)}$ ,  $1 \leq j \leq M$ , is the  $j$ th entry in  $\mathbf{X}_t^{(i)}(\mathcal{R})$ .  $x^{(j)}$  can be calculated as  $x^{(j)} = \frac{PC^{i,j}(\mathcal{R})}{\sum_{l=1}^M PC^{i,l}(\mathcal{R})}$ , where  $PC^{i,j}(\mathcal{R})$  is the number of instances defined by the preference meta-path  $\mathcal{R}$  from the node  $i$  of type  $V_1$  to the label  $j$  of label type  $L$ .


Hereinafter, for simplicity, we denote an NLDM by  $\mathbf{X}_t$  and the  $i$ th row vector in  $\mathbf{X}_t$  by  $\mathbf{X}_t^{(i)}$  in an unambiguous context.

As an illustration, Fig. 5 shows two neighbor label distributions of node  $A_1$  in the co-author network shown in Fig. 4. Fig. 5(a) shows the neighbor label distribution of  $A_1$  defined by the preference meta-path  $\mathcal{R}_1 = Author \rightarrow Author \rightarrow Profession$ . Fig. 5(b) shows the other neighbor label distribution of  $A_1$  defined by the preference meta-path  $\mathcal{R}_2 = Author \rightarrow Paper \rightarrow Domain$ .

$\mathcal{R}_1 = \text{Author} \rightarrow \text{Author} \rightarrow \text{Profession}$			$\mathcal{R}_2 = \text{Author} \rightarrow \text{Paper} \rightarrow \text{Domain}$		
Profession	$PC^{A_1, J}(\mathcal{R}_1)$	$\mathbf{x}^{(j)}$	Domain	$PC^{A_1, J}(\mathcal{R}_2)$	$\mathbf{x}^{(j)}$
Student ( $j = 0$ )	1	1/2	Database ( $j = 0$ )	1	1/3
Professor ( $j = 1$ )	1	1/2	Data Mining ( $j = 1$ )	1	1/3
Researcher ( $j = 2$ )	0	0	Machine Learning ( $j = 2$ )	1	1/3



(a)  $\mathbf{X}_t^{(A_1)}(\mathcal{R}_1) = (\frac{1}{2}, \frac{1}{2}, 0)$



(b)  $\mathbf{X}_t^{(A_1)}(\mathcal{R}_2) = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$

**Fig. 5** Examples of neighbor label distribution vectors

Now the problem of the prediction of neighbor label distributions can be reduced to the following prediction problem.

**Problem Statement:** Given a preference meta-path  $\mathcal{R}$  and the latest  $p$  NLDMs  $\{X_{t-p+1}(\mathcal{R}), \dots, X_t(\mathcal{R})\}$  until current time  $t$ , we want to predict the NLDM at time  $t + 1$ ,  $X_{t+1}(\mathcal{R})$ .

#### 4 Latent Space Evolution Model

In this section, we present the details of Latent Space Evolution Model (LSEM), which consists of two parts, autoregressions of latent feature matrices and reconstruction of NLDM for prediction.

##### 4.1 Autoregressions of Latent Feature Matrices

For the prediction of Neighbor Label Distribution Matrix (NLDM), a straight forward idea is regression, however, which can not be directly applied due to the two challenges mentioned above, sparsity of links and the complexity of link formation preference. To overcome these issues, we first factorize each historical NLDM to latent feature matrices, and then utilizes two  $p$ -order autoregressive models to estimate the future latent matrices from which the future NLDM can be reconstructed.

We can factorize an NLDM  $\mathbf{X}$  to a Source Node latent feature Matrix (SNM) and a Target Label latent feature Matrix (TLM) by solving the following optimization problem:

$$\operatorname{argmin}_{\mathbf{S}, \mathbf{T}} \frac{1}{2} \|\mathbf{X} - \mathbf{S} \times \mathbf{T}^T\|_F^2 + \frac{\lambda}{2} (\|\mathbf{S}\|_F^2 + \|\mathbf{T}\|_F^2), \quad (2)$$

where  $\|\cdot\|_F$  denotes the Frobenius norm,  $\mathbf{S} \in \mathbb{R}^{N \times D}$  is the SNM,  $\mathbf{T} \in \mathbb{R}^{M \times D}$  is the TLM,  $N$  is the number of source nodes,  $M$  is the number of difference target labels,  $D$  is the dimensionality of latent features,  $\lambda$  is the balance parameter, and  $\|\mathbf{S}\|_F^2$  and  $\|\mathbf{T}\|_F^2$  are the regularization terms which avoid overfitting and guarantee the row vectors in  $\mathbf{S}$  and  $\mathbf{T}$  are dense.

A row vector of SNM implicitly represents the link formation preference of a source node, and a row vector of TLM represents the latent feature of a target label. Note that even if the links of a node in a DHIN snapshot are sparse, the latent features of the source node and the target label can still make sense because the SNM and TLM would be dense due to the regularization terms. Similarly, we can generate the historical SNMs and TLMs as follows:

$$\begin{aligned}\mathbf{X}_1 &= \mathbf{S}_1 \times \mathbf{T}_1^T, \\ &\dots \\ \mathbf{X}_{t-1} &= \mathbf{S}_{t-1} \times \mathbf{T}_{t-1}^T, \\ \mathbf{X}_t &= \mathbf{S}_t \times \mathbf{T}_t^T.\end{aligned}$$

We estimate the SNM  $\hat{\mathbf{S}}_{t+1}$  and the TLM  $\hat{\mathbf{T}}_{t+1}$  from the latest  $p$  historical SNMs and TLMs respectively by using the following  $p$ -order autoregressive models:

$$\hat{\mathbf{S}}_{t+1} = \sum_{j=1}^p \phi_S^{(j)} \mathbf{S}_{t-j+1} + \mathbf{C}_S, \quad (3)$$

$$\hat{\mathbf{T}}_{t+1} = \sum_{j=1}^p \phi_T^{(j)} \mathbf{T}_{t-j+1} + \mathbf{C}_T, \quad (4)$$

where  $p$  is the order of the two autoregressive models,  $\mathbf{C}_S$  and  $\mathbf{C}_T$  are constant matrices in which all the entries are filled with  $\phi_S^{(0)}$  and  $\phi_T^{(0)}$ , respectively, and  $\{\phi_S^{(0)}, \dots, \phi_S^{(p)}\}$  and  $\{\phi_T^{(0)}, \dots, \phi_T^{(p)}\}$  are parameters of the two autoregressive models. We learn the two autoregressive models by solving the following optimization problems:

$$\underset{\phi_S^{(0)}, \dots, \phi_S^{(p)}}{\operatorname{argmin}} \frac{1}{2} \sum_{t'=p+1}^t \|\mathbf{S}_{t'} - \hat{\mathbf{S}}_{t'}\|_F^2, \quad (5)$$

$$\underset{\phi_T^{(0)}, \dots, \phi_T^{(p)}}{\operatorname{argmin}} \frac{1}{2} \sum_{t'=p+1}^t \|\mathbf{T}_{t'} - \hat{\mathbf{T}}_{t'}\|_F^2, \quad (6)$$

where  $\hat{\mathbf{S}}_{t'} = \sum_{j=1}^p \phi_j \mathbf{S}_{t'-j} + \mathbf{C}_S$  and  $\hat{\mathbf{T}}_{t'} = \sum_{j=1}^p \phi_j \mathbf{T}_{t'-j} + \mathbf{C}_T$  are the estimated SNM and TLM at time  $t'$ , respectively. The order of the autoregressive models can be determined by minimizing the following residual [6]:

$$e = \sum_{t'=p+1}^t (\|\mathbf{S}_{t'} - \hat{\mathbf{S}}_{t'}\|_F + \|\mathbf{T}_{t'} - \hat{\mathbf{T}}_{t'}\|_F), \quad (7)$$

where  $\hat{\mathbf{S}}_{t'} = \sum_{j=1}^p \phi_j \mathbf{S}_{t'-j} + \mathbf{C}_S$  and  $\hat{\mathbf{T}}_{t'} = \sum_{j=1}^p \phi_j \mathbf{T}_{t'-j} + \mathbf{C}_T$  are the estimated SNM and TLM at time  $t'$ , respectively.

Note that in Equations (5) - (7),  $p$  decides how many historical snapshots will impact the current SNM and TLM. When the setting value of  $p$  is too large,



the earlier historical snapshots become disturbance items, which will result in the increasing of the residual  $e$ . On the contrary, when the setting value of  $p$  is too small, the model will be underfitting, which will also increase the residual  $e$ . Therefore, the optimal  $p$  depends on datasets and is experimentally determined by minimizing the residual  $e$  defined by Equation (7). In our paper, we determine  $p$  based on experimental result as shown in Fig. 11. For the datasets used for our experiments,  $p$  is chosen as 2 for DBLP, Foursquare and Yelp, and 3 for Netflix.

Algorithm 1 gives the learning algorithm of autoregressive models.

---

**Algorithm 1** *Learning Algorithm of Autoregressive Models* ( $\{\mathbf{X}_1, \dots, \mathbf{X}_t\}, D$ )

---

**INPUT:**

$\{\mathbf{X}_1, \dots, \mathbf{X}_t\}$ : The set of NLDMs at  $\{1, \dots, t\}$ ;  
 $D$ : The dimensionality of latent features;

**OUTPUT:**

$\Phi_S = \{\phi_S^{(0)}, \phi_S^{(1)}, \dots, \phi_S^{(p)}\}$ : The set of parameters in the autoregressive model of SNM;  
 $\Phi_T = \{\phi_T^{(0)}, \phi_T^{(1)}, \dots, \phi_T^{(p)}\}$ : The set of parameters in the autoregressive model of TLM;  
 $p$ : the order of the autoregressive models of both SNM and TLM;

- 1: **for**  $i = 1 \rightarrow t$  **do**
  - 2:   Factorize  $\mathbf{X}_i$  into  $\mathbf{S}_i$  and  $\mathbf{T}_{ui}$ ;
  - 3: **end for**
  - 4: Determine  $p$  by minimizing the residual defined by Equation (7);
  - 5: Apply gradient descent method to learn  $\Phi_S$  of the autoregressive model defined by Equation (3) by minimizing the objective function defined by Equation (5);
  - 6: Apply gradient descent method to learn  $\Phi_T$  of the autoregressive model defined by Equation (4) by minimizing the objective function defined by Equation (6);
- 

#### 4.2 Prediction

After the parameters,  $p$ ,  $\Phi_S$  and  $\Phi_T$ , are learnt, we can predict  $\mathbf{X}_{t+1}$  from the given  $p$  latest NLDMs,  $\{\mathbf{X}_{t-p+1}, \dots, \mathbf{X}_{t-1}, \mathbf{X}_t\}$ . We first generate the SNMs  $\{\mathbf{S}_{t-p+1}, \dots, \mathbf{S}_{t-1}, \mathbf{S}_t\}$  and TLMs  $\{\mathbf{T}_{t-p+1}, \dots, \mathbf{T}_{t-1}, \mathbf{T}_t\}$  from the latest  $p$  historical NLDMs. Then, we estimate  $\hat{\mathbf{S}}_{t+1}$  and  $\hat{\mathbf{T}}_{t+1}$  by using the two learnt autoregressive models defined by Equation (3) and Equation (4). Finally, we can make a prediction of NLDM  $\hat{\mathbf{X}}_{t+1}$  as

$$\hat{\mathbf{X}}_{t+1} = \hat{\mathbf{S}}_{t+1} \times \hat{\mathbf{T}}_{t+1}^T, \quad (8)$$

where  $\hat{\mathbf{S}}_{t+1}$  and  $\hat{\mathbf{T}}_{t+1}$  are the estimated SNM and TLM.

Algorithm 2 gives the neighbor label distribution prediction algorithm. The time complexity of matrix factorization (Line (2)) is  $O(DET)$ , where  $E$  is the number of entries in  $\mathbf{X}_t$ ,  $D$  is the dimensionality of latent features and  $T$  is the fixed maximum iteration times. Because  $D$ ,  $T$  and  $p$  are constants, the time complexity of Line (1) to Line (3) is  $O(E)$ . The time complexity of autoregressive model (Line (4) and Line (5)) is  $O(E)$ . Thus, the overall time

complexity of Algorithm 2 is  $O(E)$ , which indicates the time complexity of Algorithm 2 is linear with the number of source nodes and independent of the number of DHIN snapshots.

---

**Algorithm 2** *Neighbor Label Distribution Prediction Algorithm* ( $\{\mathbf{X}_{t-p+1}, \dots, \mathbf{X}_t\}, D$ )

---

**INPUT:**

$\{\mathbf{X}_{t-p+1}, \dots, \mathbf{X}_t\}$ : The latest  $p$  NLDMs;

**OUTPUT:**

$\hat{\mathbf{X}}_{t+1}$ : The predicted NLDM at time  $t + 1$ ;

- 1: **for**  $i = k - p + 1 \rightarrow k$  **do**
  - 2:     Factorize  $\mathbf{X}_t$  into  $\mathbf{S}_t$  and  $\mathbf{T}_t$ ;
  - 3: **end for**
  - 4: Estimate  $\hat{\mathbf{S}}_{t+1}$  by the autoregressive model defined by Equation (3);
  - 5: Estimate  $\hat{\mathbf{T}}_{t+1}$  by the autoregressive model defined by Equation (4);
  - 6: Predict  $\hat{\mathbf{X}}_{t+1}$  according to Equation (8);
- 

## 5 Experiments

In this section, we present the details of the experiments conducted on real datasets. We first determine the dimensionality  $D$  of latent feature and the order  $p$  of the autoregressive models, then verify the effectiveness and efficiency of LSEM. The experiments are executed on a Windows 7 PC with an Intel Xeon CPU of 3.4GHz and 16 GB RAM, and all programs are implemented in C#. The source code is available on [23].

### 5.1 Datasets

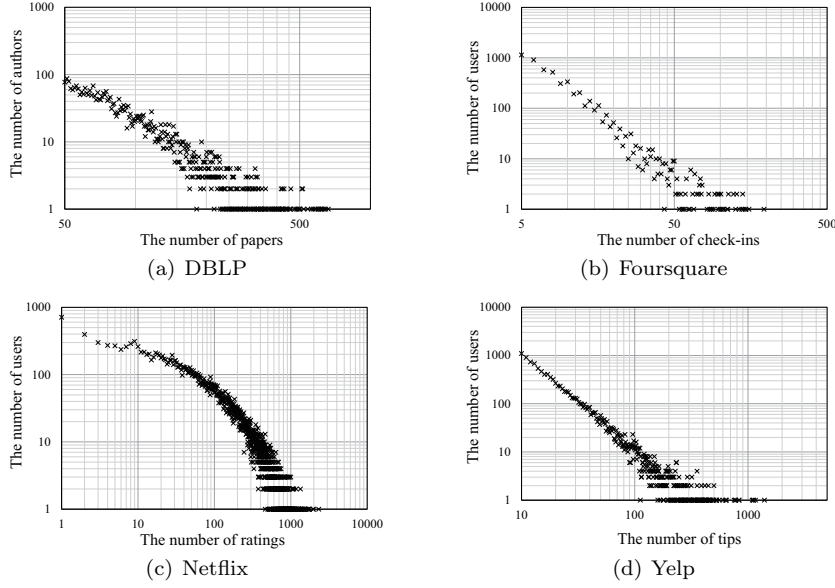
We conduct our experiments on four real datasets, two location-based social networks (Foursquare and Yelp), a co-author network (DBLP), and a movie rating network (Netflix), which are summarized in Table 1.

#### 5.1.1 DBLP

DBLP dataset is collected and maintained by the DBLP team [40]. The dataset indexes more than 2,800,000 articles.

On DBLP, given a preference meta-path  $\mathcal{R} = \text{Author} \rightarrow \text{Paper} \rightarrow \text{Domain}$  and a group of authors, we want to predict the paper domain distributions of the authors. There are 25 domain labels on Computer Science. To obtain the domain labels of these papers, we first divide the titles of papers into keywords by using "NLTK" [28], which is a toolkit for building Python programs to work with human language data. Then we build a dictionary [24] which maps

Dataset	Preference meta-path			#Snapshot	#Latent feature
	$V_1$	$V_2$	L		
DBLP	#Author	#Paper	#Domain	14	11
	13,153	1,202,530	25		
Foursquare	#User	#Venue	#Region	24	22
	5,317	46,065	8		
Netflix	#User	#Movie	#Genre	24	6
	21,439	17,770	28		
Yelp	#User	#Business	#Category	28	9
	11,214	57,476	14		

**Table 1** Summary of Datasets.**Fig. 6** Overviews of datasets

keywords to domain labels. At last, we extract the domain labels from keywords based on the dictionary. We select 13,153 authors who published more than 50 papers, and use their 1,202,530 papers which were published from 2000 to 2011. We use a year as the unit of time, thus the dataset is segmented into 12 snapshots each of which corresponds to one year of 2000 to 2011. Fig. 6(a) shows that the number of authors follows a heavy-tailed distribution over the number of their published papers, which indicates the dataset is nature.

### 5.1.2 Foursquare

We joint two Foursquare datasets which are collected by Zhang *et al.* [45, 46, 16] and Bao *et al.* [4], respectively. One contains 5,392 users, 48,756 tips,

38,921 venues and 76,972 social links, and the other one contains 221,128 tips generated by 49,062 users at 46,065 venues in New York City (NYC).

On Foursquare, given a preference meta-path  $\mathcal{R} = User \rightarrow Venue \rightarrow Region$  and a group of users, we want to predict the venue region distributions of the users' check-ins. We select 5,317 users who have more than 5 check-ins, and use their 59,498 check-ins in NYC from May. 1<sup>st</sup>, 2009 to May. 1<sup>st</sup>, 2011. We use a month as the unit of time, and segment the dataset into 24 snapshots. To obtain the region labels of venues, we divide NYC into 8 regions in terms of administrative divisions. Fig. 6(b) shows that the number of users follows a heavy-tailed distribution over the number of check-ins, which again indicates the dataset is nature.

### 5.1.3 Netflix

Netflix dataset [1] contains more than 100,000,000 rating records from about 480,000 customers over about 17,000 movie titles. The labels of "Movie" contain 28 genres crawled from the website IMDb [12].

On Netflix, given a preference meta-path  $\mathcal{R} = User \rightarrow Movie \rightarrow Genre$  and a group of users, we want to predict the genre distributions of the users' rated movies. We select 21,439 users, and use their 2,704,432 ratings on 6,583 movies from Jan. 1<sup>st</sup>, 2000 to Jan. 1<sup>st</sup>, 2002. We use a month as the unit of time, and divide the dataset into 24 snapshots. Fig. 6(c) shows that the number of users follows a heavy-tailed distribution over the number of movies, which also indicates the dataset is nature.

### 5.1.4 Yelp

Yelp dataset [2] contains more than 27,000,000 reviews and 649,000 tips by 687,000 users for 86,000 businesses. The labels of "Businesses" contain over 1000 categories, and we summarize these categories into 14 major categories.

On Yelp, given a preference meta-path  $\mathcal{R} = User \rightarrow Business \rightarrow Category$  and a group of users, we want to predict the category distributions of the users' check-ins. We select 11,214 users who have more than 10 tips, and use their 648,303 tips from Oct. 1<sup>st</sup>, 2009 to Aug. 1<sup>st</sup>, 2016. We use a season as the unit of time, and segment the dataset into 28 snapshots. Fig. 6(d) shows that the number of users follows a heavy-tailed distribution over the number of check-ins, which again indicates the dataset is nature.

## 5.2 Baseline Methods

In order to demonstrate the effectiveness of our model, we compare LSEM with the following six baseline methods:

1. **Evolution Factor Model (EFM)**: An Evolution Factor Model (EFM) is proposed in [25] to infer neighbor label distribution in heterogeneous information networks. EFM considers a network itself as the historical network,

and an update of the network as the current network. For a set of given nodes, EFM first clusters the nodes by using  $K$ -means, and then infers the neighbor label distribution for each cluster by directly applying a regression on the neighbor label distributions. To apply EFM for the prediction of NLDM, it takes the snapshot at time  $t$  as the current network, and the network comprised of all snapshots from time 1 to time  $t - 1$  as the historical network. Then EFM is used to predict the NLDM at time  $t + 1$ .

2. **Matrix Factorization** (MF): MF is proposed by B. Webb [43] to solve the movie recommender problem in Netflix Price. MF assumes the latent features of objects are modeled by vectors, and different types of objects have factors with the same size. When predicting the preferences of the objects of type  $A$  to the objects of type  $B$ , the preferences can be expressed as the product between the latent factor of the objects of type  $A$  and the latent factor of the objects of type  $B$ . To use MF for the prediction of neighbor label distribution, we build a historical NLDM  $\mathbf{X} = \mathbf{X}_t + \dots + \mathbf{X}_{t-p+1}$ , and then factorize  $\mathbf{X}$  by solving the following optimization problem:

$$\underset{\mathbf{S}, \mathbf{T}}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{X} - \mathbf{S} \times \mathbf{T}^T\|_F^2 + \frac{\lambda}{2} (\|\mathbf{S}\|_F^2 + \|\mathbf{T}\|_F^2),$$

where  $\lambda$  is the balance parameter,  $\mathbf{S}$  and  $\mathbf{T}$  are the latent feature matrices of source nodes and target labels. We take the filled matrix  $\widehat{\mathbf{X}} = \mathbf{S} \times \mathbf{T}^T$  as the predicted NLDM.

3. **Biased Matrix Factorization** (Biased MF): Biased MF is proposed by Paterek [27], which is an extension of Matrix Factorization. Biased MF adds biased rates to the objects of either type. To apply Biased MF to the prediction of neighbor label distribution, we build a historical NLDM  $\mathbf{X} = \mathbf{X}_t + \dots + \mathbf{X}_{t-p+1}$ , and then factorize  $\mathbf{X}$  by solving the following optimization problem:

$$\underset{\mathbf{S}, \mathbf{T}, \mathbf{B}_S, \mathbf{B}_T}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{X} - \mathbf{S} \times \mathbf{T}^T - \mathbf{B}_S - \mathbf{B}_T\|_F^2 + \frac{\lambda}{2} (\|\mathbf{S}\|_F^2 + \|\mathbf{T}\|_F^2),$$

where  $\lambda$  is the balance parameter,  $\mathbf{S}$  and  $\mathbf{T}$  are the latent feature matrices of source nodes and target labels, respectively, and  $\mathbf{B}_S$  and  $\mathbf{B}_T$  are the biased matrices of source nodes and target labels, respectively. We take the filled matrix  $\widehat{\mathbf{X}} = \mathbf{S} \times \mathbf{T}^T$  as the predicted NLDM.

4. **Tucker Decomposition** (TD): TD [41] decomposes a tensor into a product of a core tensor and matrices along all dimensions. To apply TD to prediction of neighbor label distribution, we construct the tensor and decompose it as shown in Fig. 7. The factor matrices can be computed by solving the following optimization problem:

$$\underset{\mathcal{C}, \mathbf{S}, \mathbf{T}, \mathbf{K}}{\operatorname{argmin}} \frac{1}{2} \|\mathcal{A} - \mathcal{C} \times_S \mathbf{S} \times_T \mathbf{T} \times_K \mathbf{K}\|_F^2 + \frac{\lambda}{2} (\|\mathbf{S}\|_F^2 + \|\mathbf{T}\|_F^2 + \|\mathbf{K}\|_F^2),$$

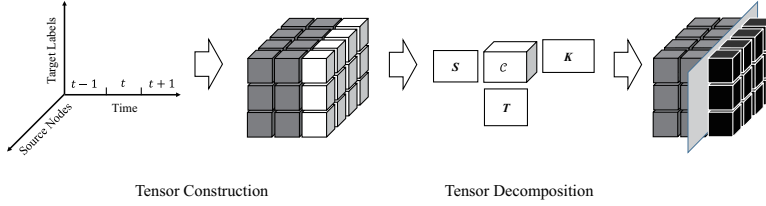
where  $\mathcal{A}$  is the constructed tensor shown in Fig. 7,  $\mathcal{C} \in \mathbb{R}^{D_1 \times D_2 \times D_3}$  is the core tensor,  $\mathbf{S} \in \mathbb{R}^{N \times D_1}$ ,  $\mathbf{T} \in \mathbb{R}^{M \times D_2}$  and  $\mathbf{K} \in \mathbb{R}^{3 \times D_3}$  are the latent feature

matrices of source nodes, target labels and time, respectively,  $\lambda$  is the balance parameter,  $N$  is the number of source nodes, and  $M$  is the number of target labels. We take the sliced matrix at time  $t+1$  along the time dimension of the filled tensor  $\hat{\mathcal{A}} = \mathcal{C} \times_S \mathcal{S} \times_T \mathcal{T} \times_K \mathcal{K}$  as the predicted NLDM.

5. **Context-Aware Tensor Decomposition (CATD)**: CATD [47] decomposes a tensor into a product of a core tensor and factor matrices combining additional matrices built as context constraints. To apply CATD to the prediction of neighbor label distribution, we construct the same tensor as shown in the left part of Fig. 7, and make a prediction of NLDM by solving the following optimization problem:

$$\begin{aligned} \underset{\mathcal{C}, \mathcal{S}, \mathcal{T}, \mathcal{K}}{\operatorname{argmin}} \quad & \frac{1}{2} \|\mathcal{A} - \mathcal{C} \times_S \mathcal{S} \times_T \mathcal{T} \times_K \mathcal{K}\|_F^2 + \frac{\lambda_1}{2} \sum_i \sum_j (\|\mathcal{S}^{i*} - \mathcal{S}^{j*}\|_F^2 \mathcal{S}_s^{(ij)}) + \\ & \frac{\lambda_2}{2} (\|\mathcal{S}\|_F^2 + \|\mathcal{T}\|_F^2 + \|\mathcal{K}\|_F^2), \end{aligned}$$

where  $\mathcal{S}^{i*}$  is the latent feature vector of source node  $i$ ,  $\mathcal{S}_s$  is the relationship indicating matrix in which an entry  $\mathcal{S}_s^{(i,j)}$  is 1 if there is a link between  $i$  and  $j$ , and 0, otherwise. We take the sliced matrix at time  $t+1$  along the time dimension of the filled tensor  $\hat{\mathcal{A}} = \mathcal{C} \times_S \mathcal{S} \times_T \mathcal{T} \times_K \mathcal{K}$  as the predicted NLDM.



**Fig. 7** Tensor construction and decomposition for predicting neighbor label distribution

6. **Meta Path-based Regression (MPR)**: MPR [35, 36] uses a generalized linear model to build the connection between the observed building time of links and the meta path-based features. To apply MPR to the prediction of neighbor label distribution, we extract features based on two meta paths, "Author / User - Paper / Venue / Movie / Business - Domain / Region / Genre / Category" and "Author / User - Author / User - Paper / Venue / Movie / Business - Domain / Region / Genre / Category", and make a prediction of NLDM as follow:

$$\hat{x}_{t+1}^{(i,j)} = \exp\{-\mathbf{v}\boldsymbol{\beta}\},$$

where  $\hat{x}_{t+1}^{(i,j)}$  is an entry of the predicted NLDM  $\hat{\mathcal{X}}_{t+1}$  denoting author/user  $i$  and label  $j$ ,  $\mathbf{v}$  is the meta path-based feature vector extracted from the historical snapshots, and  $\boldsymbol{\beta}$  is the coefficient vector.

### 5.3 Metrics

The performance is measured by two metrics, Root Mean Square Error (RMSE) and Mean Absolute Error (MAE), defined as follows:

$$RMSE = \sqrt{\frac{\|\mathbf{X}_{t+1} - \widehat{\mathbf{X}}_{t+1}\|_F^2}{2N}},$$

$$MAE = \frac{\sum_{i=1}^N \|\mathbf{X}_{t+1}^{(i)} - \widehat{\mathbf{X}}_{t+1}^{(i)}\|_1}{2N},$$

where  $\widehat{\mathbf{X}}_{t+1}$  is the predicted NLDM,  $\mathbf{X}_{t+1}$  is the ground truth,  $\mathbf{X}_{t+1}^{(i)}$  and  $\widehat{\mathbf{X}}_{t+1}^{(i)}$  are the  $i$ th row vectors in  $\mathbf{X}_{t+1}$  and  $\widehat{\mathbf{X}}_{t+1}$ , respectively,  $\|\cdot\|_1$  denotes the  $L_1$ -norm, and  $N$  is the number of source nodes.

### 5.4 Sensitivity of parameters

We now investigate the sensitivity of the parameters of LSEM, i.e., the dimensionality  $D$  of latent feature, the balance parameter  $\lambda$ , and the order  $p$  of the autoregressive models.

#### 5.4.1 The dimensionality of latent features

For each of the four datasets, we tune the dimensionality of latent features, denoted by  $D$ , according to the residual of the matrix factorization on the first snapshot. The residual is calculated as

$$e = \|\mathbf{X}_1 - \mathbf{S}_1 \times \mathbf{T}_1^T\|_F,$$

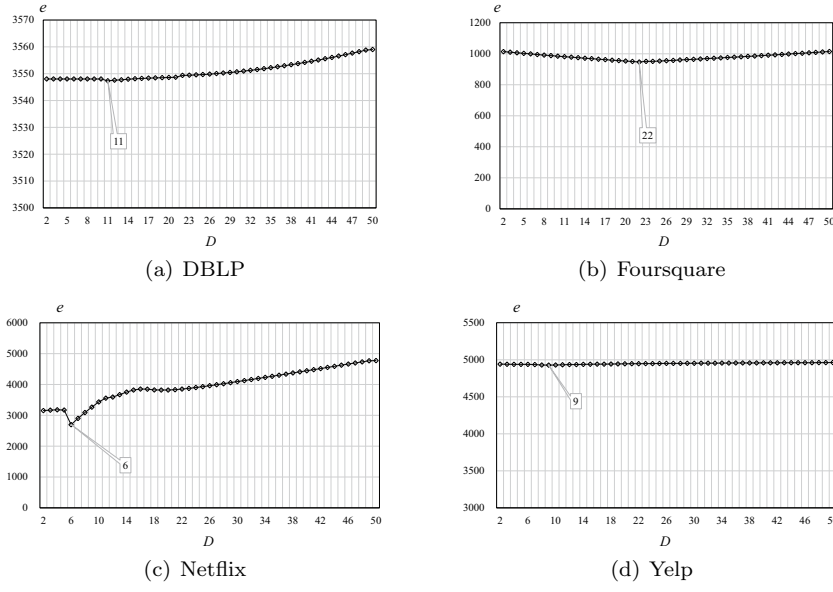
where  $\mathbf{S}_1$  and  $\mathbf{T}_1$  are generated by solving the following optimization problem:

$$\operatorname{argmin}_{\mathbf{S}_1, \mathbf{T}_1} \frac{1}{2} \|\mathbf{X}_1 - \mathbf{S}_1 \times \mathbf{T}_1^T\|_F^2 + \frac{\lambda}{2} (\|\mathbf{S}_1\|_F^2 + \|\mathbf{T}_1\|_F^2).$$

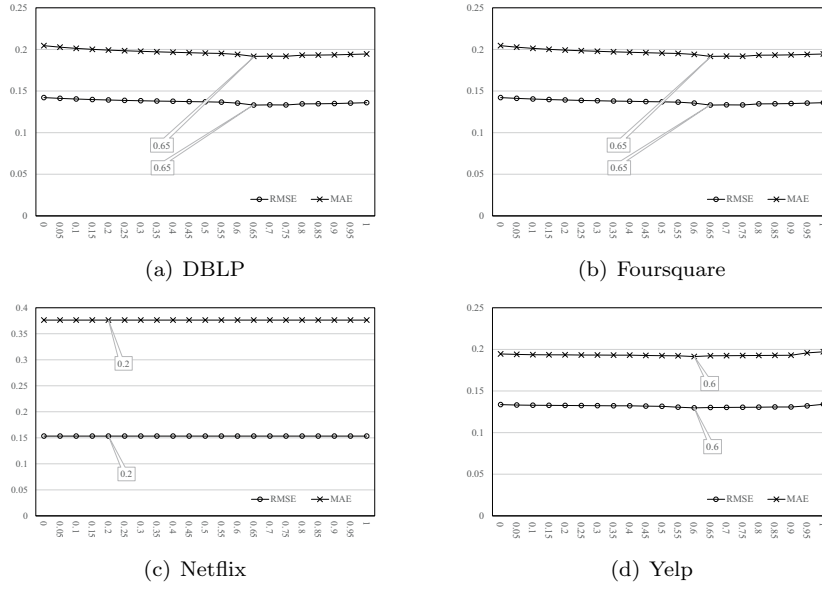
We tune  $D$  from 2 to 50, and as shown in Fig. 8, the residuals on DBLP, Foursquare, Netflix and Yelp reach the minimum at 11, 22, 6 and 9, respectively. So we choose  $D = 11$ ,  $D = 22$ ,  $D = 6$  and  $D = 9$  as the dimensionality of latent features for the four datasets, respectively.

#### 5.4.2 The balance parameter $\lambda$

For each of the four datasets, we determine the balance parameter  $\lambda$ , according to the performance of LSEM when setting different values of  $\lambda$ . We tune  $\lambda$  from 0.0 to 1.0 with step-size 0.05, and use the first 6 snapshots as the training set, and predict  $\mathbf{X}_{t+1}$ ,  $t = 6$  on DBLP. On Foursquare and Netflix, we use the first 18 snapshots as the training set, and predict  $\mathbf{X}_{t+1}$ ,  $t = 18$ . On Yelp, we use



**Fig. 8** Tuning the dimensionality of latent features



**Fig. 9** Tuning balance parameter

the first 22 snapshots as the training set, and predict  $\mathbf{X}_{t+1}, t = 22$ . As shown in Fig. 9, the RMSE and MAE of LSEM on DBLP, Foursquare, Netflix and



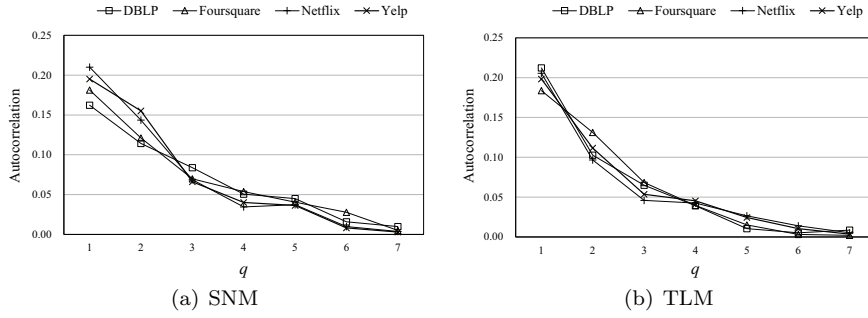
Yelp reach the minimum at 0.15, 0.65, 0.2 and 0.6, respectively. So we choose  $\lambda = 0.15$ ,  $\lambda = 0.65$ ,  $\lambda = 0.2$  and  $\lambda = 0.6$  for the four datasets, respectively.

#### 5.4.3 The order $p$ of autoregressive models

We first verify the stationarity of the sequences of latent features before applying autoregressive model for the estimation of the latent feature matrices, SNM and TLM. We randomly select 300 source nodes from each dataset and compute the autocorrelation of the sequence of each latent feature of each user in the first 12 snapshots of each dataset. For a sequence  $\{s_1, s_2, \dots, s_t\}$ , its autocorrelation  $r_q$  is calculated as:

$$r_q = \frac{\sum_{i=q+1}^t (s_i - \bar{s})(s_{i-q} - \bar{s})}{\sum_{i=1}^t (s_i - \bar{s})^2},$$

where  $q$  is the delay time, and  $\bar{s}$  is the average of the sequence. Then, for each of the latent feature matrices, SNM and TLM, we plot the average autocorrelation curves of each latent feature of each user in the same graph. As we can



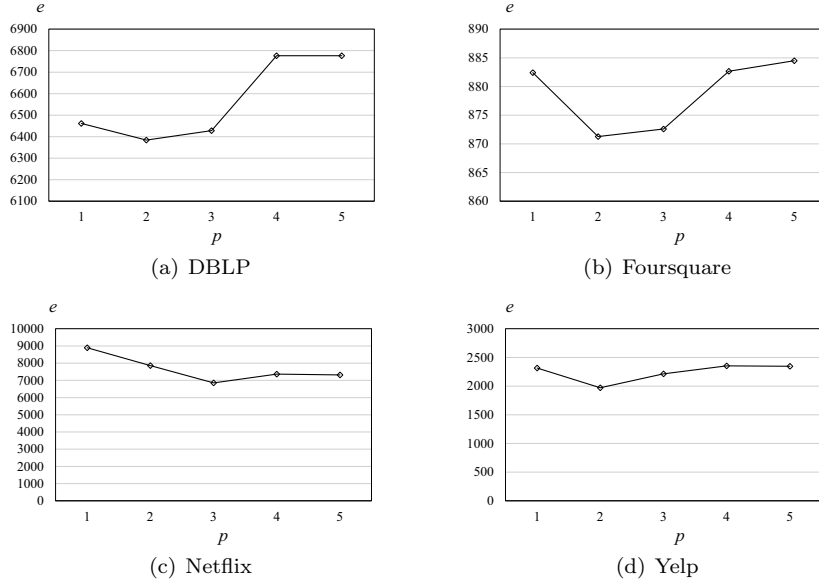
**Fig. 10** Average autocorrelation curves of DBLP, Foursquare, Netflix and Yelp

see from Fig. 10, the autocorrelations all decline with  $q$  and drop to near zero, which shows that the sequences are stationary [6].

We then determine the order  $p$  according to residual analysis on snapshots of DBLP, Foursquare, Netflix and Yelp, respectively, where the residual is calculated as

$$e = \sum_{t'=p+1}^t (\|\mathbf{S}_{t'} - \hat{\mathbf{S}}_{t'}\|_F + \|\mathbf{T}_{t'} - \hat{\mathbf{T}}_{t'}\|_F).$$

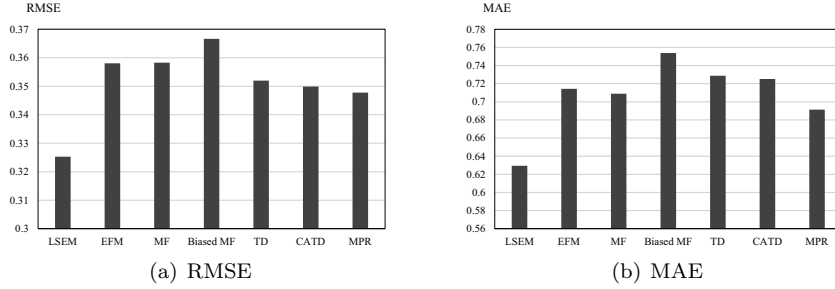
We choose the order  $p = 2$  for DBLP, Foursquare and Yelp, and  $p = 3$  for Netflix, because the residuals on DBLP, Foursquare, Netflix and Yelp reach minimum at  $p = 2$ ,  $p = 2$ ,  $p = 3$  and  $p = 2$  respectively, as shown in Fig. 11.



**Fig. 11** Residual of each  $p$  on DBLP, Foursquare, Netflix and Yelp

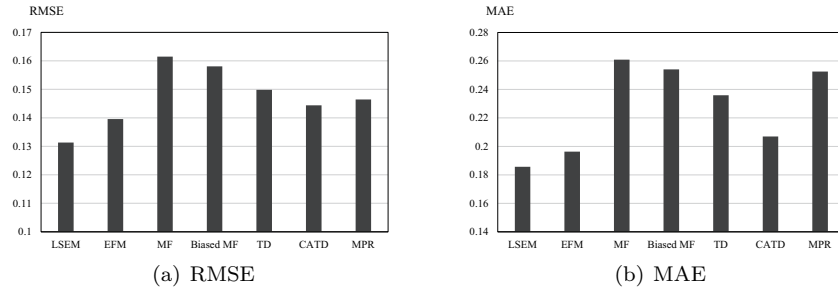
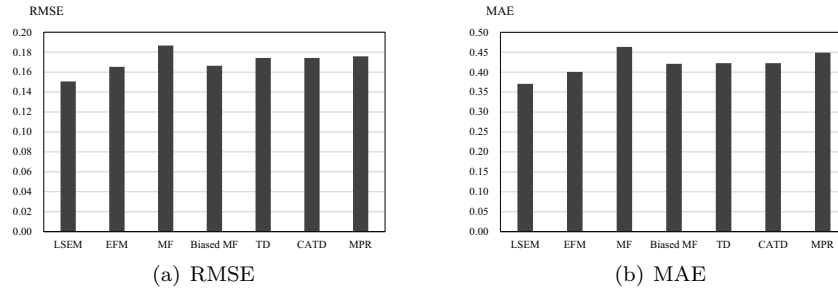
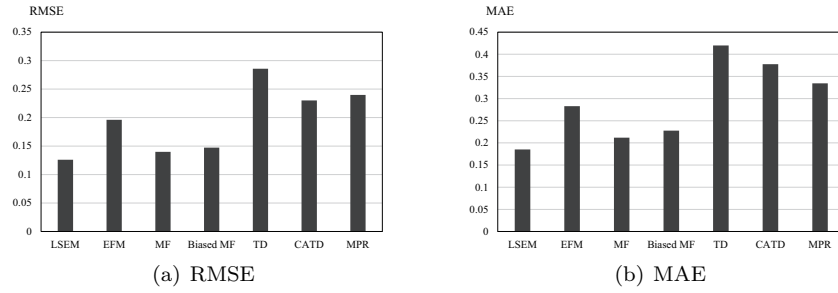
### 5.5 Effectiveness

On DBLP, we use the first 7 snapshots as the training set, and predict  $\mathbf{X}_{t+1}$ ,  $7 \leq t \leq 11$ . On Foursquare and Netflix, we use the first 19 snapshots as the training set, and predict  $\mathbf{X}_{t+1}$ ,  $19 \leq t \leq 23$ . On Yelp, we use the first 23 snapshots as the training set, and predict  $\mathbf{X}_{t+1}$ ,  $23 \leq t \leq 27$ . The results on the four datasets are shown in Fig. 12, Fig. 13, Fig. 14 and Fig. 15 respectively.

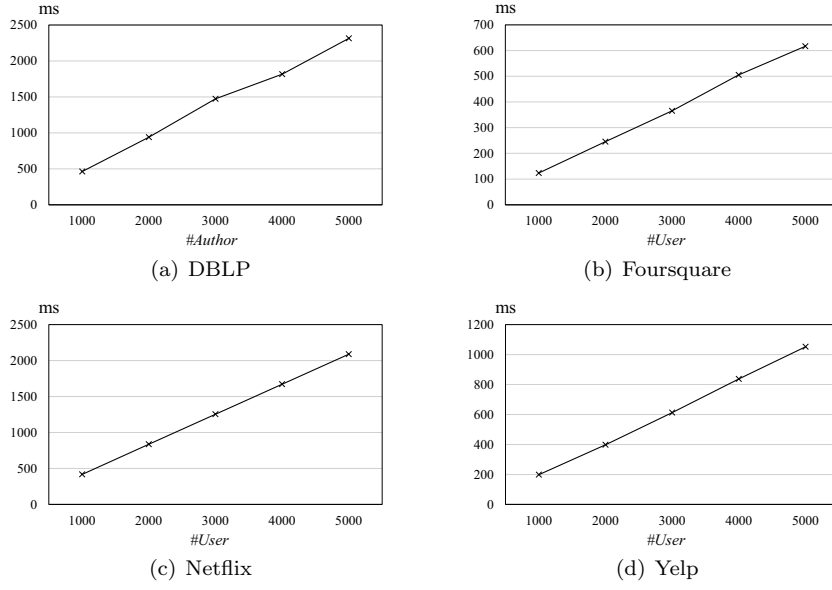


**Fig. 12** Performance comparison on DBLP

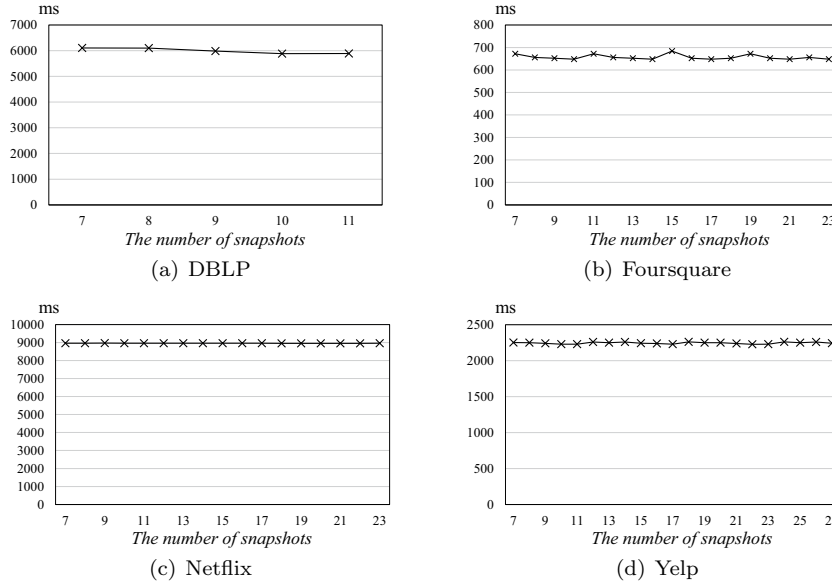
As we can see from the above figures, LSEM outperforms the baseline methods on all the datasets, of which the reasons can be analyzed as follows:

**Fig. 13** Performance comparison on Foursquare**Fig. 14** Performance comparison on Netflix**Fig. 15** Performance comparison on Yelp

1. In contrast with LSEM, EFM makes predictions directly by a regression of NLDMs, which ignores link formation preference of source nodes.
2. MPR, MF, Biased MF, TD, and CATD can make predictions based on models incorporating the link formation preference, but they can not capture the evolution of link formation preference of source nodes.



**Fig. 16** Running time of LSEM of different source nodes



**Fig. 17** Running time of LSEM

### 5.6 Efficiency

As mentioned in Section 4.2, the time complexity of Neighbor Label Distribution Prediction Algorithm (NLDPA, shown in Algorithm 2) is linear with the number of source nodes and independent of the number of DHIN snapshots.

To verify the running time of NLDPA is linear with the number of source nodes, we randomly select 1000, 2000, 3000, 4000 and 5000 source nodes (Author nodes of DBLP, User nodes of Foursquare, Netflix, and Yelp) from each dataset, and run NLDPA for each group of source nodes. Fig. 16 shows the running time of NLDPA is linear with the number of the source nodes on all the four datasets.

To verify the running time of NLDPA is independent of the number of DHIN snapshots, we use the first 7 snapshots as the training set for all the datasets. And we predict  $\mathbf{X}_{t+1}$ , taking  $7 \leq t \leq 11$  on DBLP,  $7 \leq t \leq 23$  on Foursquare and Netflix, and  $7 \leq t \leq 27$  on Yelp. As shown in Fig. 17, the running time of prediction algorithm remains stable with the increasing number of DHIN snapshots on all the four datasets.

## 6 Related Work

In this section, we first clarify the difference between this work and our previous work on the prediction of neighbor label distribution, then briefly introduce the related works from four domains, link prediction, recommending algorithms, user profiling, and node classification.

### Link Prediction

Hasan *et al.* [11] introduce a supervised learning method to predict whether a link will be built in the future. Wang *et al.* [42] propose a local probabilistic graphical model method that can scale to large graphs to estimate the joint co-occurrence probability of two nodes. Taskar *et al.* [39] propose a method to address the problem of link prediction in heterogeneous networks based on the observations of the attributes of the objects. There are a number of other studies which make use of the information about the change in the network over time for link prediction. For example, Zhu *et al.* [14] propose a Markov model-based method to predict links for helping user to surf the Internet, where the website page are clustered according to their hyperlinks. And the Markov model-based method is applied on the hierarchy constructed from the clustering results. Acar *et al.* [8] propose a time-aware link prediction method which constructs a three dimension tensor. The tensor structure includes the time information as the third dimension instead of collapsing the data, which overcomes the difficulty of representing temporal data. Sun *et al.* [36] extend the traditional link prediction to relationship prediction, which not only predicts whether it will happen, but also infers when it will happen. These methods mainly focus on the predictions around a single link, but pay little attention to the prediction of distributions.

### Recommending Algorithms

Recommending algorithms can be classified as two categories: memory-based algorithms and model-based algorithms. The former type of methods make predictions based on homogeneous neighbors of given individuals directly [22, 29, 44], and the latter type of methods make recommendations based on a learned prediction model [30]. However, existing recommending algorithms

mainly focus on predicting the rating of a single link between a user and an item, while in this paper, we aim at the prediction of neighbor label distribution where the states of neighbors are considered holistically.

### User Profiling

User profiling is an important task for many online services. Some early works focus on gender discrimination [18] and author identification [26]. Costa Jr *et al.* [15] propose an approach to predict demographic attributes and personalities based on the answers of users to some specific psychometric tests. Kosinski *et al.* [17] use dimensionality reduction for preprocessing the Likes data, which are then entered into logistic/linear regression to predict individual psychodemographic profiles. Brdar *et al.* [5] propose a multi-level classification model to predict users' genders, ages, marital statuses, job types, and so on. Lately, Zhong *et al.* [48] extract rich semantics of users' check-ins and propose the location to profile (L2P) framework to infer demographics of users. Our work is different from the user profile mining, because we focus on the dynamic characteristics of nodes which evolves over time.

### Node Classification

Most node classification methods classify nodes in dynamic networks. For example, Sharan and Neville propose a relational Bayes classifier [31] and a probability tree [32] to classify nodes in an evolving heterogeneous network with different types of nodes, where time information is included to improve the classifier. Callut *et al.* [13] propose a novel technique, called *D*-walks, to tackle semi-supervised classification problems in dynamic network, which can deal with directed or undirected graphs with a linear time complexity with respect to the number of edges. Recently, a time varying relational classifier [10] is proposed to classify nodes in a heterogeneous network by effectively using nodes connected to the node and the connections from the past. However, these methods cannot serve our goal because they often assume a finite state space.

## 7 Conclusions

In this paper, we aim at addressing the problem of neighbor label distribution prediction in dynamic heterogeneous information networks. We propose a Latent Space Evolution Model (LSEM), which uses a Neighbor Label Distribution Matrix (NLDM) to represent the state of neighbors of given nodes. LSEM factorizes historical NLDM into Target Label latent feature Matrix (TLM) and Source Node latent feature Matrix (SNM) and predicts the next NLDM by reconstructing it from the next SNM and TLM which are estimated by the proposed autoregressive models. We conduct the experiments on four real datasets, two location-based social networks (Foursquare and Yelp), a co-author network (DBLP), and a movie rating network (Netflix). The results verify the effectiveness and efficiency of the proposed model and algorithm.

## Acknowledgment

This work is supported by National Science Foundation of China through grant 61173099, the Basic Research Program of Sichuan Province with Grant 2014JY0220, and US National Science Foundation through grants CNS-1115234, DBI-0960443, and OISE-1129076.

## References

1. Netflix prize. URL <http://www.netflixprize.com>
2. Yelp dataset challenge. URL [https://www.yelp.com/dataset\\_challenge](https://www.yelp.com/dataset_challenge)
3. Back, M.D., Schmukle, S.C., Egloff, B.: How extraverted is honey.bunny77@hotmail.de? inferring personality from e-mail addresses. *Journal of Research in Personality* **42**(4), 1116–1122 (2008)
4. Bao, J., Zheng, Y., Mokbel, M.F.: Location-based and preference-aware recommendation using sparse geo-social networking data. *GIS '12 Proceedings of the 20th International Conference on Advances in Geographic Information Systems* pp. 199–208 (2012)
5. Brdar, S., Culibrk, D., Crnojevic, V.: Demographic attributes prediction on the real-world mobile data. *Proceedings of the Nokia mobile data challenge 2012 workshop* (2012)
6. Cryer, J.D., Chan, K.S.: *Time series analysis with applications in r*, second edition (2008)
7. Diane, K., Teevan, J.: Implicit feedback for inferring user preference: a bibliography. *ACM SIGIR Forum*. **37**(2) (2003)
8. E, A., DM, D., TG, K.: Link prediction on evolving data using matrix and tensor factorizations. *ICDM 2010 workshops. IEEE Computer Society* pp. 262–269 (2010)
9. Gorrell, G., Webb, B.: Generalized hebbian algorithm for incremental latent semantic analysis. *Proceedings of Interspeech* (2006)
10. Gunes, I., Cataltepe, Z., Gunduz-Oguducu, S.: Ga-tvrc-het: genetic algorithm enhanced time varying relational classifier for evolving heterogeneous networks. *Data Min Knowl Disc* **28**(2), 670C701 (2014)
11. Hasan, M.A., Chaoji, V., Salem, S., Zaki, M.: Link prediction using supervised learning. *SDM'06, Proceedings of SIAM International Conference on Data Mining (workshop on Link Analysis, Counterterrorism and Security)* (2006)
12. IMDb: URL <http://www.imdb.com/> (2015)
13. J, C., K, F., M, S., P, D.: Semi-supervised classification from discriminative random walks. *ECML/PKDD* **5211**, 162C177 (2008)
14. J, Z., J, H., JG, H.: Using markov models for web site link prediction. *Hypertext. ACM* pp. 169–170 (2002)
15. Jr, P.T.C., McCrae, R.R.: Reply to ben-porath and waller. *Psychological Assessment* **4**(1), 20–22 (1992)
16. Kong, X., Zhang, J., Philip, Y.S.: Inferring anchor links across multiple heterogeneous social networks. *CIKM '13 Proceedings of the 22nd ACM international conference on Conference on Information and Knowledge Management* (2013)
17. Kosinski, M., Stillwell, D., Graepel, T.: Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences* **110**(15), 5802–5805 (2013)
18. Labov, W.: *The social stratification of english in new york city*. PhD thesis, Columbia university (1964)
19. Lamar, R.N., Beswick, R.W.: Voice mail versus conventional channels: A cost minimization analysis of individuals' preferences. *Academy of Management Journal* **33**(4) (1990)
20. Leroy, V., Cambazoglu, B.B., Bonchi, F.: Cold start link prediction. *KDD'10, Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (2010)

21. Lichtenwalter, R.N., Lussier, J.T., Chawla, N.V.: New perspectives and methods in link prediction. *KDD'10, Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* **26**(12), 2942–2955 (2010)
22. Linden, G., Smith, B., York, J.: Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* (2003)
23. Ma, Y.: Source code. URL <https://pan.baidu.com/s/1gffUYmJ>
24. Ma, Y.: Keywords-direction dictionary. URL <http://pan.baidu.com/s/1kTmXd4N> (2015)
25. Ma, Y., Yang, N., Li, C., Zhang, L., Yu, P.S.: Predicting neighbor distribution in heterogeneous information networks. *SDM '14* (2014)
26. Mosteller, F., Wallace, D.L.: Inference in an authorship problem: A comparative study of discrimination methods applied to the authorship of the disputed federalist papers. *Journal of the American Statistical Association* **58**(302), 275–309 (1963)
27. Paterek, A.: Improving regularized singular value decomposition for collaborative filtering. In: *Proceedings of KDD cup and workshop 2007*, vol. 2007, pp. 5–8 (2007)
28. Project, N.: Nltk 3.0 documentation. URL <http://www.nltk.org/> (2015)
29. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. *WWW'10, Proceedings of the 19th international conference on World Wide Web* (2001)
30. Savia, E., Puolamäki, K., Kaski, S.: Latent grouping models for user preference prediction. *Machine Learning* **74**(1), 75–109 (2009)
31. Sharan, U., Neville, J.: Exploiting time-varying relationships in statistical relational models. *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis* pp. 9–15 (2007)
32. Sharan, U., Neville, J.: Temporal relational classifiers for prediction in evolving domains. *ICDM'08, Proceedings of the 8th IEEE International Conference on Data Mining* pp. 540–549 (2008)
33. Shi, C., Li, Y., Yu, P.S., Wu, B.: Constrained-meta-path-based ranking in heterogeneous information network. *Knowledge and Information Systems* pp. 1–29 (2016)
34. Sigal, S., Avi, R., Sarit, K., Navot, A.: A hybrid approach of classifier and clustering for solving the missing node problem. In: *AAAI*, pp. 282–289 (2015)
35. Sun, Y., Han, J.: *Mining heterogeneous information networks: Principles and methodologies* (2012)
36. Sun, Y., Han, J., Aggarwal, C.C., Chawla, N.V.: When will it happen? - relationship prediction in heterogeneous information networks. *WSDM'12, Proceedings of the 8th ACM International Conference on Web Search and Data Mining* (2012)
37. Sun, Y., Han, J., Yan, X., Yu, P.S., Wu, T.: Paths: Meta path-based top-k similarity search in heterogeneous information networks. *VLDB' 11* (2011)
38. Sun, Y., Tang, J., Han, J., Chen, C., Gupta, M.: Co-evolution of multi-typed objects in dynamic star networks. *TKDE, IEEE Transaction on Knowledge and Data Engineering* (2013)
39. Taskar, B., Wang, M., Abbeel, P., Koller, D.: Link prediction in relational data. *Learning Statistical Patterns in Relational Data Using Probabilistic Relational Models* **7** (2005)
40. Team, D.: Dblp. URL <http://dblp.uni-trier.de/> (2015)
41. Tucker, L.R.: Some mathematical notes on three-mode factor analysis. *Psychometrika* **31**, 279–311 (1966)
42. Wang, C., Satuluri, V., Parthasarathy, S.: Local probabilistic models for link prediction. *ICDM'07, Proceedings of IEEE International Conference on Data Mining 2007* (2007)
43. Webb, B.: Netflix update: Try this at home. URL <http://sifter.org/~simon/JOURNAL/20061211.html> (2006)
44. Yu, K., Schwaighofer, A., Tresp, V., Xu, X., Kriegel, H.P.: Probabilistic memorybased collaborative filtering. *TKDE, Transaction on Knowledge and Data Engineering* **16**(1), 56–69 (2004)
45. Zhang, J., Kong, X., Philip, Y.S.: Predicting social links for new users across aligned heterogeneous social networks. *ICDM '13 Proceedings of the 13th International Conference on Data Mining* pp. 1289–1294 (2013)



46. Zhang, J., Kong, X., Philip, Y.S.: Transferring heterogeneous links across location-based social networks. WSDM '14 Proceedings of the 7th ACM international conference on Web search and Data Mining (2014)
47. Zheng, Y., Liu, T., Wang, Y., Zhu, Y., Liu, Y., Chang, E.: Diagnosing new york city's noises with ubiquitous data. Ubicomp '14, Proceedings of ACM International Conference on Ubiquitous Computing 2014 (2014)
48. Zhong, Y., Yuan, N.J., Zhong, W., Zhang, F., Xie, X.: You are where you go: Inferring demographic attributes from location check-ins. WSDM'15 Proceedings of the 8th ACM international conference on Web search and Data Mining (2015)