A Label Inference Method Based on Maximal Entropy Random Walk over Graphs

Jing Pan[1,2], Yajun Yang[1,2(✉)], Qinghua Hu[1,2], and Hong Shi[1,2]

[1] School of Computer Science and Technology, Tianjin University, Tianjin, China
{panjing,yjyang,huqinghua,serena}@tju.edu.cn
[2] Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin, China

Abstract. With the rapid development of Internet, graphs have been widely used to model the complex relationships among various entities in real world. However, the labels on the graphs are always incomplete. The accurate label inference is required for many real applications such as personalized service and product recommendation. In this paper, we propose a novel label inference method based on maximal entropy random walk. The main idea is that a small number of vertices in graphs propagate their labels to other unlabeled vertices in a way of random walk with the maximal entropy guidance. We give the algorithm and analyze the time and space complexities. We confirm the effectiveness of our algorithm through conducting experiments on real datasets.

Keywords: Label inference · Random walk · Maximal entropy

1 Introduction

With the rapid development of Internet, graphs have been widely used to model the complex relationships among various entities in real applications including social network, web graph and biology. These graphs are always with labels. For an example of social network, every people is represented by a vertex in the graph. The vertices may be with the labels to indicate the hobbies, occupations or other attributes for their representing people. As another example of protein interaction network, the labels on the vertices indicate the types of the protein. These graphs are called as labeled graphs. However, due to the limitation of information collection and the existence of noise, the labels on graphs are always incomplete, i.e., only a fraction of the vertices are attached with the labels on the graph and the remaining ones are unknown. For the example, in Sina Weibo, which is the largest micro-blog platform in China, only $20\,\% - 30\,\%$ users exhibit the labels about their occupations. Therefore, inferring the labels for the vertices in graphs is a critical problem in many applications.

Consider a concrete application of social network, e.g., Sina Weibo, an internet enterprise tries to provide more personalized service for the users according to the users' information such as their hobbies, occupations and so on. However, only a small amount of information or labels are explicitly provided by rarely users themselves due to the privacy or other concerns, then an effective label

F. Li et al. (Eds.): APWeb 2016, Part I, LNCS 9931, pp. 506–518, 2016.
DOI: 10.1007/978-3-319-45814-4_41

inference mechanism is required to infer the missing labels for the remaining users. By the accurate label inference, more personalized services such as product and content recommendation may be provided for the targeting customers. It improves the user experience effectively.

Over the last several years, there have been an increasing interest in label inference from the graphs. The first category of these works focuses on building a classifier to distinguish the vertices with the distinct labels in a network [8,12,14]. For these methods, the sufficient background knowledge is necessary for extracting features from the users [6,18]. Specially, an actual and reliable sub-network is required as a training dataset including all the kinds of labels. However, such sub-network is difficult to find out, even it may be not exist. The other category is a community detection method to identify the group affiliation for the individuals in the network [17]. The individuals in the same group are deem as with the uniform labels. Homophily [13] is a common assumption in these methods to infer user labels, i.e., the similar users tend to interact with each other directly. Hence the users with the similar labels are aggregated naturally to a community by one-hop interactions.

In this paper, we study the problem of identifying the labels for all the vertices in the labeled graphs. The main idea of our label inference mechanism is that a small number of labeled vertices propagate their labels to the entire network in a way of random walk under the guidance by the maximal entropy model. Different to the previous methods, our method does not require too much background knowledge or training sub-networks. On the other hand, homophily assumption is not necessary, that is, it allows the users with the same label appear in the network dispersedly but not in some common community.

The main contributions of this paper are summarized as below. First, we propose a novel label inference model based on the maximal entropy random walk. Second, we give the algorithm to infer labels with our model and analyze the time and space complexities of our algorithm. Our algorithm can handle both undirected and directed graphs. Finally, we confirm the effectiveness of our algorithm through conducting experiments on real datasets by compared with SVM, Homophily, and traditional random walk algorithms.

The rest of the paper is organized as follow. Section 2 defines the problem. Section 3 proposes the random walk model with the maximal entropy guidance. The label inference algorithm is given in Sect. 4. The experimental results on real networks are presented in Sect. 5. The related works are introduced in Sect. 6. Finally, we conclude this paper in Sect. 7.

2 Problem Statement

A labeled graph is a simple directed graph, denoted as $G = (V, E, \Sigma, L)$, where V is the set of vertices and E is the set of edges in G. The total number of nodes is n. Each edge $e \in E$ is represented by (u, v), $u, v \in V$. e is called u's outgoing edge or v's incoming edge and v (or u) is called u (or v)'s outgoing(or incoming) neighbor. Σ is the set of all the labels in G. L is a label function mapping V to Σ. Each $L(u)$ is a label in Σ, i.e. $L(u) \in \Sigma$. For example, in Sina Weibo, a user u

Table 1. Important notations

Notation	Description
$G(V, E)$	structure of the graph
Σ, V_s, L	label set, labeled vertex set, label function mapping V to Σ
$T(u)$	label-weight set on vertex u
$w_x(u)$	the probability that vertex u has the label x
$p_{i,j}^{(k)}$	transition probability at the k-th iteration of maximal entropy random walk
$N^+(v_i)$, $N^-(v_i)$	outgoing neighbor set of v_i, incoming neighbor set of v_i
$H(v_j)$	entropy value of v_j
$w_x(v_i, v_j)$	probability that v_i have the label l_x propagated by its incoming neighbor v_i

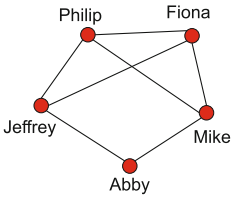


Fig. 1. An example graph from a company

Table 2. Labels on vertices in Fig. 1

ID	Name	Label
1	Fiona	Manager
2	Philip	Programmer
3	Jeffrey	Human Resource
4	Mike	Programmer
5	Abby	Manager

may have an occupation label as "Manager", then L(u) = "Manager". Note that all the labels in Σ are with the same type, e.g., "Manager" and "Programmer" are two labels about the same label type "occupation". In real applications, a user may has several labels with different types. In the following, we first focus on the case that every vertex has a label with the same type and we will discuss how to deal with the general case that every vertex may have the labels with several types in Sect. 4.3. Our work can be easily extended to handle undirected graphs, in which an undirected edge (u, v) is equivalent to two directed edges (u, v) and (v, u) (Table 1).

Figure 1 illustrates an example graph in a social network. There are five users in this graph that are staffs in software company and the labels about their occupations are shown in Table 2. In this example, the occupation of Fiona is "Manager", so she is labeled in "Manager".

Problem Statement: Given a labeled graph $G = (V, E, \Sigma, L)$ and the labeled vertex set V_s where $V_s \subset V$, i.e., only the labels on the vertices in V_s is known, and our work is to infer $L(u)$ for every vertex $u \in V - V_s$.

3 Inference Model Based on Maximal Entropy Random Walk

3.1 Framework

In this section, we introduce the framework of our label inference model as shown in Fig. 2. The input is $G(V, E, \Sigma, L)$ and V_s, which describe the structure of the

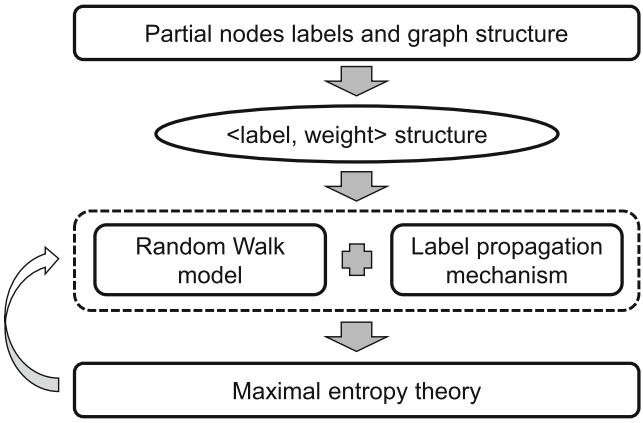


Fig. 2. The framework of label inference model

entire graph $G(V, E)$ and the labeled vertex set respectively. The main idea of our method is propagating the labels from the vertices in V_s to the unlabeled vertices in the way of random walk. It is worth noting that an unlabeled vertex $v \in V - V_s$ may become a vertex with several labels in the random walk process. Σ is the set of all the possible labels in graph G, $\Sigma = \{l_1, \cdots, l_p\}$, then every vertex in G only has a label from Σ. We use a tuple set $T(u) = \{< l_x, w_x(u) > | 1 \leq x \leq p\}$ to denote the labels l_x $(1 \leq x \leq p)$ on vertex u with the corresponding weight $w_x(u)$. $T(u)$ is called label-weight set of u. Here, $w_x(u)$ indicates the probability that vertex u has the label l_x and $\sum_{x=1}^{p} w_x(u) = 1$. For every vertex $u \in V_s$, if it has a label l_{x_1}, then $w_{x_1}(u) = 1$ and $w_x(u) = 0$ for $x \neq x_1$ and $1 \leq x \leq p$. For every vertex $u \in V - V_s$, $w_x(u)$ is initialized as $\frac{1}{p}$ for $1 \leq x \leq p$. It means u may have p labels with the same probability at the beginning. The value of $w_x(u)$ $(1 \leq x \leq p)$ for every vertex is updated iteratively after every one step random walk and $w_x(u)$ becomes larger and larger if u has the label l_x in real world. We discuss how to update $w_x(u)$ in Sect. 3.3. Moreover, the random walk is under the guidance of maximal entropy. After every one step random walk, our algorithm computes the entropy value for every vertex $v \in V - V_s$. For every vertex u in G, the transition probability from u to all its outgoing neighbors in the next one step random walk is calculated according to the entropy values of all the outgoing neighbors of u. Finally, when the convergence is satisfied, the random walk terminates, and the label set $L(u)$ for every vertex $u \in V_s$ can be inferred by its $T(u)$.

3.2 One by One Step Random Walk

A random walk is a mathematical formalization of a path that consists of a succession of random steps. Give a directed graph $G(V, E)$, where the number of the vertex is $n = |V|$. Graph G can be represented as a $n \times n$ symmetric adjacency matrix M, where $m_{i,j}$ is an entry of M at the i-th row and the j-th column.

Then $m_{i,j} = 1$ if $(v_i, v_j) \in E$ otherwise $m_{i,j} = 0$ for $v_i, v_j \in V$. The degree of a vertex v_i is denoted as $d_i = \sum_{j=1}^{n} m_{i,j}$. Let $D = \mathsf{diag}(d_1, \cdots, d_n)$ be a diagonal matrix of vertex degree. A traditional random work on G can be defined utilizing the transition matrix $P = D^{-1}M$ with entries $p_{i,j} = \frac{m_{i,j}}{d_i}$. $p_{i,j}$ is called the transition probability from v_i to v_j by one step random walk and P is called the initial transition probability matrix. Let $P^{(k)}$ denote the transition probability matrix of k-th step random walk, then the transition probability matrix $P^{(k+1)}$ of $(k+1)$-th step random walk can be calculated as $P^{(k+1)} = P^k \times P$. An entry $p_{i,j}^{(k)}$ in $P^{(k)}$ is the transition probability from v_i to v_j by k step random walk.

In our algorithm, the labels are propagated in the way of random walk one by one step. At each iteration, a vertex v_i propagates its labels in $T(v_i)$ by one step random walk. Let v_j is an outgoing neighbor of v_i. The entropy value of v_j is calculated by $T(v_j)$. Then the transition probability $p_{i,j}$ from v_i to v_j by next one step random walk is calculated according to the entropy values of all v_i's outgoing neighbors. Different to the traditional random walk, the transition probability $p_{i,j}$ for one step random walk is not an uniform value in our method, that is, $p_{i,j}$ for the k-th and $(k+1)$-th iteration are different. We use $p_{i,j}^{(k)}$ to denote the transition probability at the k-th iteration. With $p_{i,j}^{(k)}$, all v_j's incoming neighbors propagate their labels to v_j and then $T(v_j)$ is updated. The initial transition probability $p_{i,j}^{(1)}$ from v_i to v_j is given below.

$$p_{i,j}^{(1)} = \frac{I(v_i, v_j)}{\sum_{v_j \in N^+(v_i)} I(v_i, v_j)} \tag{1}$$

Where $N^+(v_i)$ is the outgoing neighbor set of v_i and $I(v_i, v_j)$ is parameter used to reflect the closeness between v_i and v_j. The value of $I(v_i, v_j)$ can be given by the real application requirement. For example, in DBLP dataset, $I(v_i, v_j)$ is the number of the papers in which researcher v_i and v_j are co-author. If there is no requirement about $I(v_i, v_j)$, then $I(v_i, v_j) = 1$ for every pair of vertex v_i and v_j in G.

3.3 Updating Label-Weight Set Under Maximal Entropy Guidance

In this section, we discuss how to compute the transition probability $p_{i,j}^{(k)}$ under maximal entropy guidance and how to update $T(v_j)$ for every vertex $v_j \in V - V_s$. We introduce the maximal entropy for computing the transition probability $p_{i,j}^{(k)}$ which is well known in information theory. Without loss of generality, at k-th iteration, every vertex v_j is with a label-weight set $T(v_j) = \{< l_x, w_x(v_j) > | 1 \leq x \leq p\}$. Then the entropy value $H(v_j)$ of v_j can be calculated as below.

$$H(v_j) = -\sum_{x=1}^{p} w_x(v_j) \times \ln w_x(v_j) \tag{2}$$

Note that $w_x(v_j) \times \ln w_x(v_j) = 0$ if $w_x(v_j) = 0$. $H(v_j)$ indicates the uncertainty about the labels on vertex v_j. The larger $H(v_j)$ results in the more uncertainty

on v_j for label inference. Therefore, the more labels from the other vertices are expected to propagate their labels to v_j for improving the label inference on v_j. There is a special case to be handled with carefully, where v_j has no label, i.e., $w_x(v_i) = 0$ for $1 \leq x \leq p$. In this case, $H(v_j)$ is computed as $H(v_j) = -\sum_{x=1}^{p} \frac{1}{p} \times \ln \frac{1}{p}$. It means the uncertainty on v_i is the maximum. If v_j is an outgoing neighbor of v_i, then the transition probability $p_{i,j}^{(k)}$ from v_i to v_j at k-th iteration is computed as follows.

$$p_{i,j}^{(k)} = \frac{H(v_j)}{\sum_{v_j \in N^+(v_i)} H(v_j)} \tag{3}$$

where $N^+(v_i)$ is the outgoing neighbor set of v_i.

Next, we introduce how to update $T(v_j)$ utilizing the transition probability $p_{i,j}^{(k)}$ at k-th iteration. We use $w_x(v_i, v_j)$ to denote the probability that v_j have the label l_x propagated by its incoming neighbor v_i at this iteration, then $w_x(v_i, v_j)$ can be calculated by

$$w_x(v_i, v_j) = w_x(v_i) \times p_{i,j}^{(k)} \tag{4}$$

By $w_x(v_i, v_j)$, the $w_x(v_j)$ updates utilizing the following equation

$$w_x(v_j) = \frac{\sum_{v_i \in N^-(v_j)} w_x(v_i, v_j)}{\sum_{x=1}^{p} \sum_{v_i \in N^-(v_j)} w_x(v_i, v_j)} \tag{5}$$

where $N^-(v_j)$ is the incoming neighbor set of v_j. Thus for every vertex $v_j \in V_s$, the $T(v_j)$ is also updated after updating every $w_x(v_j)$ $(1 \leq x \leq p)$.

4 Label Inference Algorithm

4.1 Label Inference Algorithm

The algorithm for label inference is shown in Algorithm 1. The input is a labeled graph G and vertex subset V_s, where the labels on every vertex $v \in V_s$ is given. The output is $T(v_j)$ for every vertex in $v_j \in V_r$, where $V_r = V - V_s$. Initially, Algorithm 1 computes the initial transition probability $p_{u,v}^{(1)}$ by Eq. (1) for every vertex $u \in V_s$ and every u's outgoing neighbor v (line 1 to 2). Line 3 to 13 shows the one by one step random walk under maximal entropy guidance. In each iteration, for every vertex $v_j \in V_r$, algorithm first computes its $H(v_j)$. Next, algorithm computes $p_{i,j}^{(k)}$ for all v_j's incoming neighbor v_i, then $w_x(v_i, v_j)$ and $w_x(v_j)$ can be computed. Finally, $T(v_j)$ can be updated with updating $w_x(v_j)$. The algorithm terminates when the convergence is satisfied. The condition of convergence is given by the following equation.

$$\sum_{v_j \in V_r} \sum_{1 \leq x \leq p} |w_x^k(v_j) - w_x^{k-1}(v_j)| \leq \theta \times p \times |V_r| \tag{6}$$

where $w_x^k(v_j)$ is essentially $w_x(v_j)$ at k-th iteration, and θ is a threshold given by user to control the number of iterations for convergence. As discussion in [10], the convergence can be satisfied for a maximal entropy random walk.

Algorithm 1. LABEL-INFERENCE $(G(V, E, \Sigma, L), V_s)$

Input: $G(V, E, \Sigma, L)$, V_s.
Output: $T(v_j)$ for every vertex $v_j \in V_r$(or $V - V_s$).

for every vertex $u \in V_s$ **do**
  computes $p_{u,v}^{(1)}$ for every $v \in N^+(u)$;
while $\sum_{v_j \in V_r} \sum_{1 \le x \le p} |w_x^k(v_j) - w_x^{k-1}(v_j)| \ge \theta \times p \times |V_r|$ **do**
  for every vertex $v_j \in V_r$ **do**
    computes $H(v_j)$ by Eq. (2);
  for $v_j \in V - V_s$ **do**
    for $v_i \in N^-(v_j)$ **do**
      computes $p_{i,j}^{(k)}$ by Eq. (3);
      for $x = 1$ to p **do**
        computes $w_x(v_i, v_j)$ by Eq. (4);
    for $x = 1$ to p **do**
      computes $w_x(v_j)$ by Eq. (5);
    updates $T(v_j)$ by $w_x(v_j)$;
return $T(v_j)$ for every vertex $v_j \in V_r$

4.2 Complexity Analysis

We analyze the time and space complexities of Algorithm 1. Let n be the number of the vertices in G. For the time complexity, at each iteration, algorithm needs to compute $H(v_j)$ for every $v_j \in V_r$, thus the time complexity of this step is $O(|V_r|)$. The complexity of computing $p_{i,j}^{(k)}$ is $O(nd)$, where d is the average out-degree of all the vertices in G. Because there are p labels in G, then complexity of computing $w_x(v_j)$ is $O(|V_r| + pnd) = O(pnd)$. Let the number of iterations be k, then the total complexity of Algorithm 1 is $O(kpnd)$. In the worst case, $d = n$ but d is always far less than n in real applications.

For the space complexity, at each iteration, Algorithm 1 needs to calculate and maintain $w_x(v_i, v_j)$ $(1 \le x \le p)$, where v_i is a vertex in G and v_j is an outgoing neighbor of v_i. Then the space complexity is $O(pnd)$.

4.3 Discussion About Multiply Types of Labels in Graph

In the above discussion, we assume that all the vertices in G have a label with the same type, e.g., all users have the label with the same type "occupation" in a social network. Next, we introduce how to deal with the general case where the vertex may have the labels with several types. For example, an user may have two different label types such as "hobby" and "occupation". Generally, if the graph G has q label types, then the graph is considered as q labeled graphs $G_1, \cdots, G_q$, where every $G_i (1 \le i \le q)$ has the same structure (V, E) as the original graph G but only have one label type. Our label inference method can be used on G_q to infer the labels for the q-th label type.

5 Experiments

In this section, we compare our method with other relevant approaches from different aspects to demonstrate the effectiveness of the model we proposed on label inference. In the following, we introduce the datasets, the baseline methods, and report our results. All the experiments are conducted on a 3.2 GHz Intel Core i5 CPU PC with the 16 GB main memory, running on Windows 7 and the programming language is Python.

5.1 Dataset and Experiment Setup

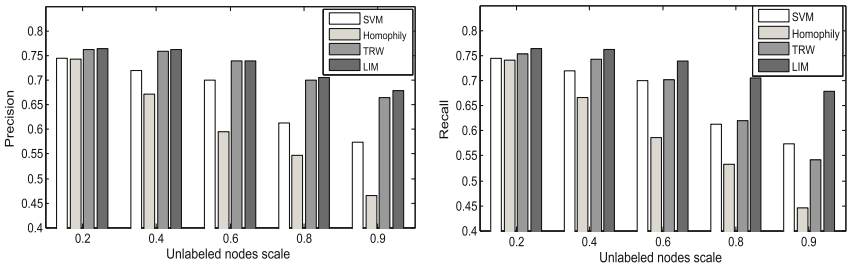
DBLP is a dataset that describes the researcher cooperation network. We treat the research domains as users' labels. We extract the journal information and conference information from DBLP network, and we label these journals and conferences into seven computer domain, including Artificial Intelligence, Data Mining, Computer Security, Programming Language, Computer Architecture, Theoretical Computer Science, and Human-computer Interaction. We extract 1,000, 3,000, 5,000, 7,000, and 10,000 vertices in the DBLP graph respectively to conduct the experiments. Meanwhile, We extract 4, 5, 6, and 7 research domains to be the labels for researchers by combining some domains. We evaluate the performance of experiments by precision and recall.

In order to demonstrate the effectiveness of the label inference method (LIM), we compare our method against a number of baseline approaches. Since our model considers both the local network structure of individual users and the effects from neighbor influence, we use the following approaches to show the performance of our method from different perspectives:

(1) SVM: We use the traditional SVM classifier as the first baseline.
(2) Homophily: From the paper [13], we know that the label of a user is associated with the user's neighbors. Therefore, we use this method as a baseline which can infer labels for users by majority votes on the user's neighbors. And this method is called Homophily.
(3) TRW: Our method is modified by the traditional random walk, so we apply the traditional random walk as another baseline. We refer to this approach as TRW.

5.2 Experimental Results

Exp1-Impact of the Proportion of Unlabeled Vertices. In Fig. 3, we study the performance of our proposed model LIM and the baselines mentioned in Sect. 5.1 on the different proportion of unlabeled vertices. We conduct this experiment in 5,000 vertices graph and we set the label number as 5 classes. We set the unlabeled scale 20 %, 40 %, 60 %, 80 %, and 90 % respectively. We present the result of precision and recall in Fig. 3(a) and (b) respectively. We can discovery that the decline tendency of our method is much slow than other methods along with the change of the scale of unlabeled vertices. And the precision of our

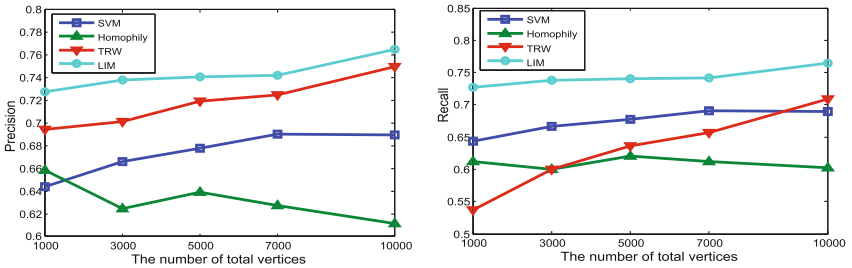


(a) Precision (b) Recall

Fig. 3. Impact of the proportion of unlabeled vertices

method is almost 70 % at the condition of there are 90 % vertices lack of label. We can get the conclusion that the performance is influenced by the proportion of unlabeled vertices.

Exp2-Impact of Vertex Size. We study the performance of our method and the baselines in different vertex size and the results can be seen in Fig. 4. We conduct this experiment in 1,000, 3,000, 5,000, 7,000, and 10,000 vertices graph respectively. And we set the label number as 4 classes. The Fig. 4(a) and (b) show the results of precision and recall of all the methods. We can analyze the results to get that the precision and recall have no relation with the vertex size in most cases. As the number of vertices changing, the performance has no obvious change in most cases.

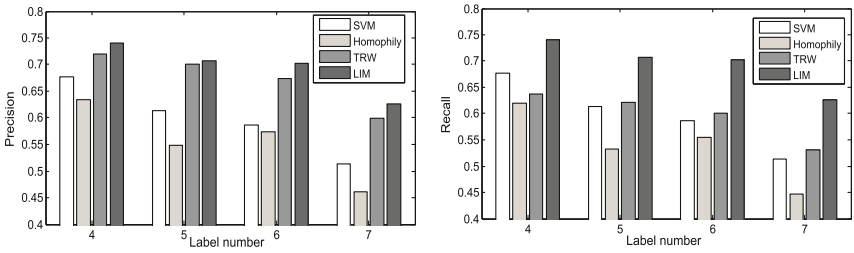


(a) Precision (b) Recall

Fig. 4. Impact of vertex size

Exp3-Impact of Label Number. We study the impact of the number of vertices' labels in Fig. 5. We show the performance of our method and the baselines on 4, 5, 6, and 7 label classes in the graph which includes 5,000 vertices and the proportion of unlabeled vertices is 80 %, which we can have a intuitive understanding on precision and recall according to the Fig. 5(a) and (b) respectively. It is obvious that the larger the label number is, the lower the precision is

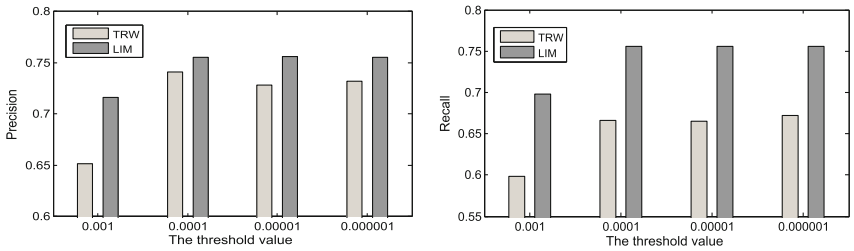
because of the unrelated labels have interference for label inference when unlabeled vertices collect labels from labeled vertices. However, the descent rate of performance of our method is slower than other methods.



(a) Precision (b) Recall

Fig. 5. Impact of label number

Exp4-Impact of Threshold θ. We study the impact of threshold θ which has a detailed introduction in Sect. 4.1. The precision and recall value are different in different θ. We show the performance of this experiment in Fig. 6. We conduct the experiment in 7,000 vertices graph and the unlabeled vertices scale is 80 %. The method we compare to is TRW only. We can see the results from Fig. 6. The threshold affects the iteration convergence times. However, the threshold doesn't have an important impact on the experimental performance.



(a) Precision (b) Recall

Fig. 6. Impact of threshold θ

Exp5-Real Case Study. In Table 3 we give a real case study on DBLP dataset. We can discover that most inference results are accurate on our method. However, The last row which is marked in bold type is wrong. By analyzing the papers of this researcher, both the true label and the inference label are his main domains and the number of published papers are almost same. In this real case study, we should make efforts on this special case to improvement our algorithm in future.

Table 3. Real case study

Name	True label	Inference label
Jiawei Han	Data Mining	Data Mining
Philip S. Yu	Data Mining	Data Mining
Jeffrey Yu	Data Mining	Data Mining
Nate Foster	Programming Language	Programming Language
Onur Mutlu	Computer Architecture	Computer Architecture
Yi Lin	Artificial Intelligence	Artificial Intelligence
Alex Pentland	Human-computer Interaction	Human-computer Interaction
Ronald L. Rivest	**Computer Security**	**Theoretical Computer Science**

6 Related Work

Label inference is an important problem in graphs and there have been an increasing interest in this problem over the last several years. The existing works for label inference problem can be divided into two categories.

The first category of the works [2,20,22] proposes different ideas which based on building feature vectors for vertices mainly from their textual informations and neighbor link informations. By training the feature vectors they build a classifier for inferring labels for vertices. For example, [15] uses the Bayesian classifier and iteration method to update attribute information. These works have a strong dependency on the background knowledge of users. However, the background knowledge isn't sufficient and always outdated which strongly influences the inference result. The second category is the approaches based on digging the graph structure, mainly for surrounding structure mining [4,19]. [5] provides a method on how can we automatically discover role labels for vertices. For vertices in graph, the similarity in graph structure decides the similarity of them. However, it is an unsupervised learning approach which is not applicable on the semi-supervised problem we focus on. [21] uses mining technology on graph structure and the relationship between vertices and their neighbors and combines the two aspects infer social roles for vertices. They abstract five social factors to mining all the vertices' structure information in the graph. In their paper, they propose an optimization framework and propose a probabilistic model to integrate the vertices' structure information and local neighbors' information to infer labels. In [14] studies user attribute inference in university social networks by applying community detection. However, these methods that depend on the surrounding structure maybe give a wrong judge if two users have same label but not in the adjacent structure.

Random Walk model is a good method to mining graph structure [1,9,11,16]. It is well studied in multiclass semi-supervised learning [7,23]. The main idea of these work is to estimate a distribution over the missing labels based on Markov random walk. Meanwhile, there are many works is proposed to improve the performance of random walk which combines with the maximum entropy

theory [3,10]. However, the problem they focus on is link prediction or some else, which is different with the problem our paper focus on.

7 Conclusion

In this paper, we study how to infer labels for unlabeled nodes in graphs or social networks. We first define the label inference problem. Second, we propose the maximal entropy random walk inference model to solve this problem. We improve the result precision by the proposed model. Finally, we confirm the effectiveness of our algorithm through conducting experiments on real datasets. In the next, we will do more works about how to deal with multi-attributes graphs and how to optimize our model to save more time and memories.

Acknowledgments. This work is supported by the grant of the National Natural Science Foundation of China No. 61432011, 61402323 and 61502335.

References

1. Azran, A.: The rendezvous algorithm: Multiclass semi-supervised learning with Markov random walks. In: Proceedings of the 24th International Conference on Machine Learning, pp. 49–56. ACM (2007)
2. Bhagat, S., Cormode, G., Muthukrishnan, S.: Node classification in social networks. In: Aggarwal, C.C. (ed.) Social Network Data Analytics, pp. 115–148. Springer, New York (2011)
3. Burda, Z., Duda, J., Luck, J., Waclaw, B.: Localization of the maximal entropy random walk. Phys. Rev. Lett. **102**(16), 160602 (2009)
4. Fortunato, S.: Community detection in graphs. Phys. Rep. **486**(3), 75–174 (2010)
5. Henderson, K., Gallagher, B., Eliassi-Rad, T., Tong, H., Basu, S., Akoglu, L., Koutra, D., Faloutsos, C., Li, L.: Rolx: structural role extraction & mining in large graphs. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1231–1239. ACM (2012)
6. Hu, X., Liu, H.: Social status and role analysis of Palin's email network. In: Proceedings of the 21st International Conference Companion on World Wide Web, pp. 531–532. ACM (2012)
7. Jaakkola, M.S.T., Szummer, M.: Partially labeled classification with Markov random walks. In: Advances in Neural Information Processing Systems (NIPS), vol. 14, pp. 945–952 (2002)
8. Leuski, A.: Email is a stage: discovering people roles from email archives. In: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 502–503. ACM (2004)
9. Li, R.H., Yu, J.X., Huang, X., Cheng, H.: Random-walk domination in large graphs. In: 2014 IEEE 30th International Conference on Data Engineering (ICDE), pp. 736–747. IEEE (2014)
10. Li, R.H., Yu, J.X., Liu, J.: Link prediction: the power of maximal entropy random walk. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, pp. 1147–1156. ACM (2011)
11. Lovász, L., et al.: Random walks on graphs: a survey. Comb. Paul Erdos is Eighty **2**, 353–398 (1996)

12. McCallum, A., Wang, X., Corrada-Emmanuel, A.: Topic and role discovery in social networks with experiments on enron and academic email. J. Artif. Intell. Res. **30**, 249–272 (2007)
13. McPherson, M., Smith-Lovin, L., Cook, J.M.: Birds of a feather: homophily in social networks. Ann. Rev. Sociol. **27**, 415–444 (2001)
14. Mislove, A., Viswanath, B., Gummadi, K.P., Druschel, P.: You are who you know: inferring user profiles in online social networks. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining, pp. 251–260. ACM (2010)
15. Neville, J., Jensen, D.: Iterative classification in relational data. In: Proceedings of the AAAI-2000 Workshop on Learning Statistical Models from Relational Data, pp. 13–20 (2000)
16. Ribeiro, B., Wang, P., Murai, F., Towsley, D.: Sampling directed graphs with random walks. In: 2012 Proceedings IEEE INFOCOM, pp. 1692–1700. IEEE (2012)
17. Wang, G., Zhao, Y., Shi, X., Yu, P.S.: Magnet community identification on social networks. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 588–596. ACM (2012)
18. Welser, H.T., Cosley, D., Kossinets, G., Lin, A., Dokshin, F., Gay, G., Smith, M.: Finding social roles in Wikipedia. In: Proceedings of the 2011 iConference, pp. 122–129. ACM (2011)
19. Xie, J., Szymanski, B.K.: Community detection using a neighborhood strength driven label propagation algorithm. In: 2011 IEEE Network Science Workshop (NSW), pp. 188–195. IEEE (2011)
20. Zhao, Y., Sundaresan, N., Shen, Z., Yu, P.S.: Anatomy of a web-scale resale market: a data mining approach. In: Proceedings of the 22nd International Conference on World Wide Web, pp. 1533–1544. International World Wide Web Conferences Steering Committee (2013)
21. Zhao, Y., Wang, G., Yu, P.S., Liu, S., Zhang, S.: Inferring social roles and statuses in social networks. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 695–703. ACM (2013)
22. Zheleva, E., Getoor, L.: To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles. In: Proceedings of the 18th International Conference on World Wide Web, pp. 531–540. ACM (2009)
23. Zhu, X., Ghahramani, Z., Lafferty, J., et al.: Semi-supervised learning using Gaussian fields and harmonic functions. In: ICML, vol. 3, pp. 912–919 (2003)