Int J Data Sci Anal
DOI 10.1007/s41060-017-0046-1

REGULAR PAPER

Meta-path-based link prediction in schema-rich heterogeneous information network

Xiaohuan Cao[1] · **Yuyan Zheng**[1] · **Chuan Shi**[1] · **Jingzhi Li**[2] · **Bin Wu**[1]

Received: 5 August 2016 / Accepted: 4 February 2017

Abstract Recent years have witnessed the boom of heterogeneous information network (HIN), which contains different types of nodes and relations. Complex structure and rich semantics are unique features of HIN. Meta-path, the sequence of object types and relations connecting them, has been widely used to mine this semantic information in HIN. Link prediction is an important data mining task to predict the potential links among nodes, which are required in many applications, e.g., filling missing links. The contemporary link prediction is usually based on simple HIN whose schema is bipartite or star schema. In these works, the meta-paths should be predefined or enumerated. However, in many real networked data, it is hard to describe their network structures with simple schema. For example, the RDF-formatted Knowledge Graph which includes tens of thousands types of objects and links is a kind of schema-rich HIN. In this kind of schema-rich HIN, it is impossible to enumerate meta-paths so that the contemporary work is invalid. In this paper, we study link prediction in schema-rich HIN and propose a novel method named *Li*nk *P*rediction with *a*utomatic meta *P*ath (LiPaP). The LiPaP designs an algorithm called automatic meta-path generation to automatically extract meta-paths from schema-rich HIN in the approximate order of relevance and adopt a supervised method with likelihood function to learn the weights of extracted meta-paths. Extensive experiments on real knowledge database, Yago, demonstrate that LiPaP is an effective, steady and efficient approach.

Keywords Heterogeneous information network · Link prediction · Similarity measure · Meta-path

1 Introduction

Nowadays, the study of heterogeneous information network (HIN) becomes more and more popular in data mining area [7], where the network includes different types of nodes and relations. Many data mining tasks have been exploited on this kind of network, such as clustering [25], classification [11]. Among those tasks in HIN, link prediction is a fundamental and important problem that attempts to estimate the existence probability of a link between two nodes, based on observed links and attributes of nodes. Researchers would use the link prediction methods as the basis to solve problems in many data mining tasks, like data clearness and recommendation.

Some works have been done to predict link existence in HIN. Because of the unique semantic characteristic and structural information of HIN, meta-path [25], a sequence of relations connecting two nodes, is widely used for link prediction. Utilizing meta-path, a lot of works usually employ a two-step process to solve the link prediction problem in HIN. The first step is to extract meta-path-based feature vectors, and the second step is to train a regression or classification model to compute the existence probability of a link [2,22,23,30]. For example, Sun et al. [22] proposed PathPredict to solve the problem of co-author relationship prediction. But these link prediction works should have a basic assumption: the meta-paths can be predefined or enumerated in a simple HIN. When the HIN is simple, we can easily and

This paper is an extension version of the PAKDD2016 Long Presentation paper "Link Prediction in Schema-Rich Heterogeneous Information Network" [3].

✉ Chuan Shi
shichuan@bupt.edu.cn

[1] Beijing University of Posts and Telecommunications, Beijing, China

[2] Southern University of Science and Technology, Shenzhen, China

Published online: 17 February 2017

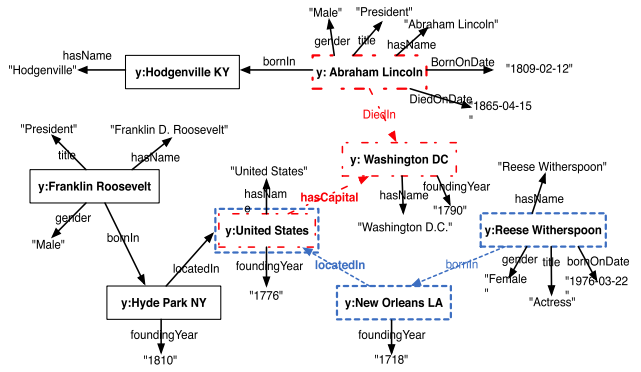

Fig. 1 A snapshot of the RDF structure extracted from DBpedia

manually enumerate some meaningful and short meta-paths [25]. For example, a bibliographic network with star schema is used in [22,23,30], and only several meta-paths are enumerated or predefined.

However, in many real networked data, the network structures are more complex, and meta-paths cannot be predefined or enumerated manually. Knowledge Graph is the base of the contemporary search engine [20], where its resource description framework (RDF) [15] $< object, relation, object >$ naturally constructs an HIN. In such an HIN, the types of nodes and relations are huge. For example, DBpedia [1], a kind of Knowledge Graph, has recorded more than 38 million entities and 3 billion facts. In this kind of network, every node may be connected by more than tens of link types, so it is hard to describe them with simple schema, and we call them schema-rich HIN. Figure 1 shows a snapshot of the RDF structure extracted from DBpedia. You can find that there are many types of objects and links in such a small network, e.g., Person, City, Country. Moreover, there are many meta-paths connecting two object types. For example, in Fig. 1, for Person and Country types, there are still two meta-paths existed: $Person \xrightarrow{bornin} City \xrightarrow{locatedIn} Country$ and $Person \xrightarrow{Diedin} City \xrightarrow{hasCapital^{-1}} Country$. Note that Fig. 1 shows one extreme little part of the whole DBpedia network, and there will be huge number of meta-paths connecting Person and Country in a real network, so that meta-paths in this kind of schema-rich HIN are too many to enumerate and it is hard to analyze them whether useful for prediction or not. The present achievements of research could not handle this situation. It is full of challenges to do link prediction task in schema-rich HIN.

To be specific, the challenges of link prediction in schema-rich HIN are mainly from two aspects. (1) The meta-paths cannot be predefined and enumerated manually. As mentioned above, there are tens of thousands of nodes and links in such schema-rich HIN, and the meta-paths in the network have the same order of magnitude. It is impossible to enumerate all useful meta-paths between two node types. (2) It is also not easy to effectively integrate these meta-paths. Even though masses of meta-paths can be found between target nodes, most of them are meaningless or less important for link prediction. Besides, for specific link prediction problem, a meta-path may have different importance in different prediction tasks, so that we need to learn weight for each meta-path, where the weight represents the importance of the path for specific link prediction.

In this paper, we study the link prediction in schema-rich HIN, and based on meta-path we propose the *Li*nk *P*rediction with *a*utomatic meta *P*ath (LiPaP) method. The LiPaP designs a novel algorithm, called automatic meta-path generation (AMPG), to automatically extract useful meta-paths from schema-rich HIN in the order according to the relevance of meta-paths. And then we design a supervised method with likelihood function to learn the weights of meta-paths. On a real Knowledge Graph Yago, we do extensive experiments to validate the performance of LiPaP. Extensive experiments show that LiPaP can effectively solve link prediction in schema-rich HIN through automatically extracting important meta-paths and learning the weights of paths.

The rest of this paper is organized as follows. Section 2 summarizes the related work of our study. And basic concepts and the problem definition are given in Sect. 3. The proposed algorithm and its descriptions are outlined in Sect. 4. And Sect. 5 discusses the experimental results. Concluding remarks are made in Sect. 6.

2 Related work

In this section, we review the related work on machine learning with Knowledge Graph, heterogeneous information networks and link prediction in HIN.

Knowledge Graph

Knowledge Graph [20], developed by Google to optimize the search results, is a knowledge-based system with semantic properties. And Knowledge Graph is derived from text data of knowledge sources. For example, DBpedia [1] is constructed by extracting entities and relations from Wikipedia. Nowadays, there are thousands of Knowledge Graph for study or application, such as DBpedia, Freebase, Yago. The hottest task of Knowledge Graph is Q/A (natural language question and answering) system [27,28,35]. For example, Zou et al. [35] proposed a whole graph data-driven framework to answer natural language questions over Knowledge Graph. The basic requirement of Q/A system is the correctness of Knowledge Graph. But the relations in Knowledge Graph cannot be made sure that the links are totally correct and complete, so that link prediction is essential for Knowledge Graph to check the correctness of links and find out missing links.

Heterogeneous Information Network

Heterogeneous information network (HIN), with different types of nodes and relations, has richer semantic information than general network with single type. Therefore, HIN provides a new orientation to manage networked data. And kinds of data mining tasks for HIN are realized recently. These research developments include similarity measure [12,24], clustering [25,26], classification [9,11], link prediction [2,22], ranking [14,34], recommendation [8,18], information fusion [10,19]. But these tasks just work on simple HINs with simple schema. The data mining tasks for schema-rich HIN are few to be done, and the existing methods for simple HIN are not appropriate to the situation of schema-rich HIN. We could take more attention to the study of schema-rich HIN.

Link Prediction in HIN

With the prevalence of HIN, link prediction in HIN has attracted many researchers. Using the meta-path feature, some works have been done [2,22,23,30] in simple HIN. Sun et al. [22] proposed PathPredict to solve the problem of co-author relationship prediction by extracting meta-path-based feature and building logistic regression-based model. Cao et al. [2] designed an iterative framework to predict multiple types of links collectively in HIN. Zhang et al. [33] utilized meta-paths to predict organization chart or management hierarchy. And Sun et al. [23] modeled the distribution of relationship building time to predict when a certain relationship will be formed.

Some researchers also utilize probabilistic models to do link prediction tasks in HIN. For example, Yang et al. [29] developed a probabilistic method MRIP to predict links in multi-relational heterogeneous networks. Dong et al. [5] proposed a transfer-based ranking factor graph model that combines several social patterns with network structure information for link prediction and recommendation. Huang et al. [6] designed the joint manifold factorization (JMF) method to do trust prediction with the ancillary rating matrix via aggregating HINs.

The methods mentioned above mostly focus on link prediction in one single HIN. Recently, some works study the problems of link prediction across multiple aligned HINs [13,32]. Zhang et al. [32] proposed SCAN-PS method to solve the social link prediction problem for new users using the "anchors." Liu et al. [13] designed the aligned factor graph model for user–user link prediction problem by utilizing information from another similar social network.

However, the research developments of link prediction are all developed for simple HIN. When the HIN becomes bigger and more complicated, we should design different link prediction methods for it.

3 Preliminary and problem definition

In this section, we introduce some basic concepts used in this paper and give the problem definition.

Heterogeneous information network (HIN) [7] is a kind of information network defined as a directed network graph $G = (V, E)$, which consists of either different types of nodes V or different types of edges E. Specifically, an information network can be abstracted to a network schema $M = (R, L)$ where R is the set of the node types and L is the set of the edge types, and there is a node-type mapping function $\theta : V \to R$ and an edge-type mapping function $\varphi : E \to L$. When the number of node types $|R| > 1$ or the number of edge types $|L| > 1$, the network is a heterogeneous information network. For example, in bibliographic database, like DBLP [4], papers are connected together via authors, venues and terms, and they can be organized as a star schema HIN. Another example is the users and items in e-commerce website which constitutes a bipartite HIN [8].

In an HIN, there can be different paths connecting two entity nodes, and these paths are called as meta-path [25]. A meta-path $\prod$ that is defined as $\prod^{R_1,\cdots,R_{l+1}} = R_1 \xrightarrow{L_1} R_2 \xrightarrow{L_2} \cdots \xrightarrow{L_l} R_{l+1}$, which describes a path between two node types R_1 and R_{l+1}, going through a series of node types $R_1, \cdots, R_{l+1}$ and a series of link types $L_1, \cdots, L_l$. Taking the knowledge graph in Fig. 1 as an example, we can consider this Knowledge Graph as an HIN, which includes many different node types (e.g., person, city, country) and link types (e.g., bornIn and locatedIn). Every two node types can be connected by multiple meta-paths. For example, there are two meta-paths connecting Person and Country in Fig. 1: $Person \xrightarrow{bornin} City \xrightarrow{locatedIn} Country$ and $Person \xrightarrow{Diedin} City \xrightarrow{hasCapital^{-1}} Country$. Obviously, different meta-paths show different semantic meanings, so that nodes connected by different meta-paths have different similarity. Thus, we can calculate the similarity of entity node pairs based on different meta-paths, which represent different features.

To use meta-path feature properly, meta-path-based similarity measures are proposed to make meta-path feature quantization and quantify the similarity of nodes [12,16,24] in HIN. Most of the studies or applications of HIN are based on these similarity measures to be performed. Sun et al. [24] proposed PathSim to calculate the similarity of the same-typed entity nodes based on symmetric paths. Lao et al. [12] designed a path-constrained random walk (PCRW) algorithm to measure the entity relativity in a labeled directed graph. Shi et al. [16] proposed HeteSim to measure relevance of any entity pair under arbitrary meta-path. Although all of these measures can do similarity calculation, not every measure could be used for fast calculation in the process of finding

important meta-paths which are symmetric or asymmetric. PathSim could only compute the relevance based on symmetric paths, and it is not suitable for arbitrary meta-path. HeteSim could measure the relevance of any object pair under arbitrary meta-path, but it is complicated when the length of the relevance path is even. PCRW utilizes the transition probability to estimate the similarity iteratively based on any meta-path, and it can roughly distinguish the importance of meta-paths by the reachable probability. Therefore, PCRW is suitable for our method to find meta-paths iteratively, and then we choose it as basic similarity measure in our study. Besides, there may be other similarity measures which are also suitable for our method, and we would do further exploration of choosing similarity measures in experiment section. The definition of PCRW [12] is as follows:

For two given entities node a and b, having a meta-path $\prod = V_1 \xrightarrow{R_1} V_2 \xrightarrow{R_2} \cdots \xrightarrow{R_l} V_n$, the similarity value of two nodes is defined by the random walk beginning at a and ending at b along this path $\prod$, as follows:
firstly,

$$\sigma\left(a, a|\prod^{1\ldots1}\right) = 1, \tag{1}$$

if $\prod^{1\ldots i} = V_1 \xrightarrow{R_1} V_2 \xrightarrow{R_2} \cdots \xrightarrow{R_{i-1}} V_i$, then

$$\sigma\left(a, t_i|\prod^{1\ldots i}\right) = \frac{R(x,t_i)}{|R(x,.)|}\sum_{x\in I(V_{i-1})}\sigma\left(a, x|\prod^{1\ldots i-1}\right), \tag{2}$$

where $I(V_{i-1})$ is the set of the entities which can be arrived by random walking across path $\prod^{1\ldots i-1}$ from node a and x is a entity node of $I(V_{i-1})$. $R(x,t_i)$ shows that x can pass the link R_{i-1} to t_i or not, being 1 if can, otherwise 0. $|R(x,.)|$ is the number of nodes which x can reach passing by R_{i-1}.

Traditional HIN usually has a simple network schema, such as bipartite [31] and star schema [17]. In those simple HIN, the meta-paths for similarity calculation can be easily enumerated. But in reality, there are a series of complex HINs, which have so many node types or link types that it is hard to describe their network schema. We call the HIN with many types of nodes and links as schema-rich HIN. In this type of HIN, it is difficult to enumerate or predefine meta-paths and current studies hardly work. Thus, data mining in schema-rich HIN will face new challenges. Specifically, we define a new task as follows:

Link prediction in schema-rich HIN Given a schema-rich HIN G and a training set of k entity pairs $\phi = \{(s_i,t_i)|1\le i\le k\}$, search a set of e meta-paths $\Upsilon = \{\prod_i |1\le i\le e\}$ which can exactly describe the pairs. With these meta-paths, we design a model $\eta(s,t|\Upsilon)$ to do link prediction on the test set of r entity pairs $\psi = \{(u_i,v_i)|1\le i\le r\}$.

4 The method description

In order to solve the link prediction problem defined above, we propose a novel link prediction method named *Li*nk *P*rediction with *a*utomatic meta *P*aths (LiPaPs). This method includes two steps: firstly, we design an algorithm called automatic meta-path generation (AMPG) to automatically discover useful meta-paths which can indicate the latent relation features of the training pairs. Secondly, we use a supervised method to integrate meta-paths to form a model for further specific prediction.

4.1 Automatic meta-path generation

In order to extract the appropriate and relevant meta-paths as model features for link prediction, we develop the automatic meta-path generation (AMPG) algorithm, which can generate useful meta-paths smartly in schema-rich HIN. In order to explain algorithm clearly, we would illustrate the process of AMPG through a toy example in Fig. 2, where the training pairs are {(1,8), (2,8), (3,9), (4,9)}.

The main goal of AMPG is, given the training set of entity pairs, to find all the useful and relevant meta-paths connecting them by the descending order of relevance approximatively. These paths to be found would not only connect more training pairs, but also show much closer relationship to present implicit features of the training set. For example, $\xrightarrow{isCitizenOf}$ is the meta-path initially found by our method in Fig. 3, and it is not only the shortest relation but also the one connecting most training pairs. Besides, the meta-paths to be found are still most relevant in the candidate paths. Basically, we start to search from the source nodes step by step to find out the useful meta-paths greedily. At each step, we select the meta-path that is most relevant and maybe reaching more target nodes. Then we check whether the path connects the training pairs or not. If so, we pick out the meta-path, otherwise make a move forward until the unchecked meta-paths are irrelevant

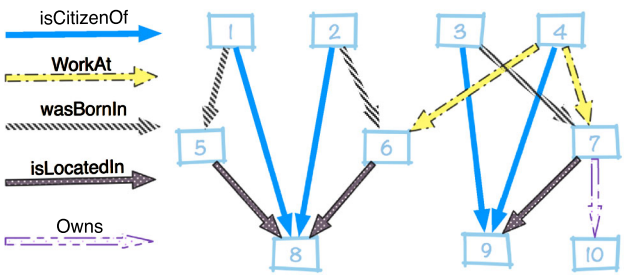

Fig. 2 Subgraph example of schema-rich HIN

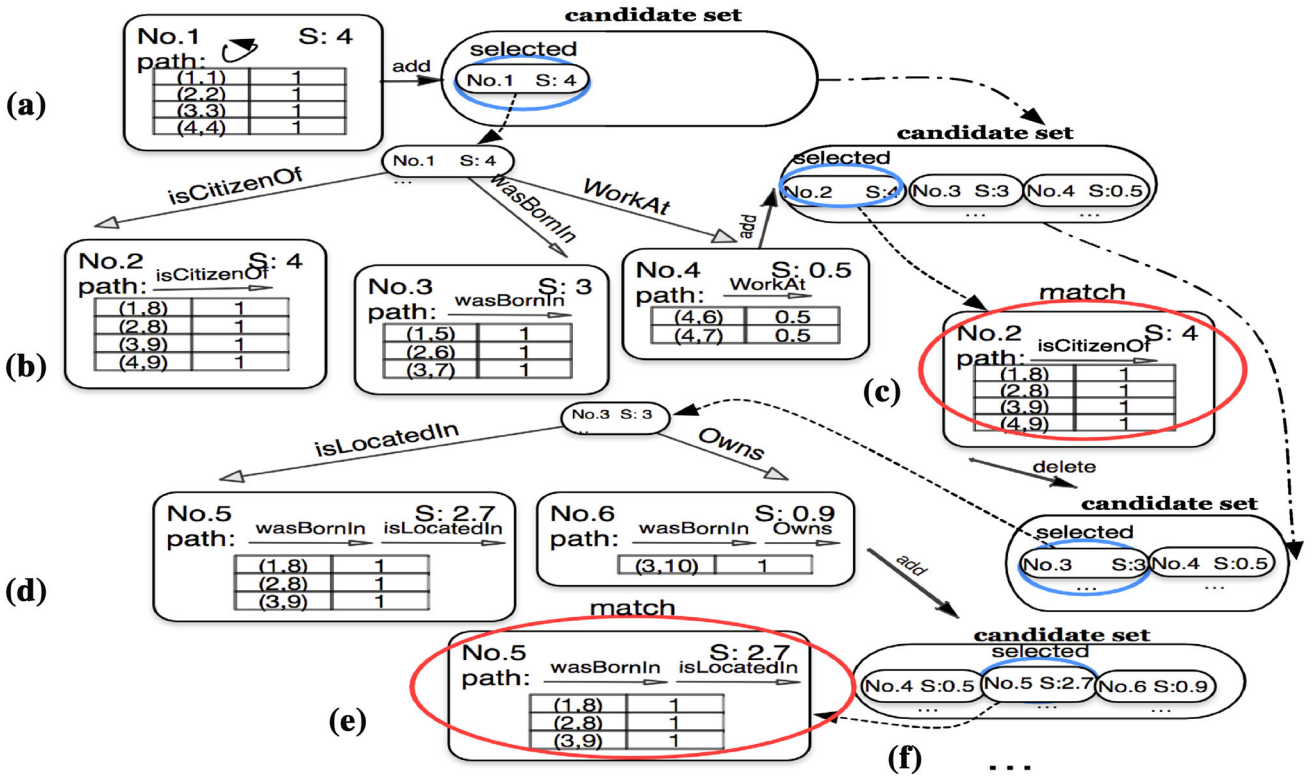

Fig. 3 An example of meta-path automatic generation

enough. It guarantees that the generated meta-paths all well described the relationship between each training pair and the selected paths are not too many to add noisy paths.

The AMPG method is a greedy algorithm that heuristically chooses the optimal paths at each step. For judging the priority of meta-paths for selection, AMPG utilizes a preference score S as a selection criterion. The higher preference score S is more likely the meta-path to be chosen. The S is based on a similarity measure path-constrained random walk (PCRW) [12], which has been introduced above. Choosing PCRW as basic similarity measure is because that its computing process is iterative and suitable for our method to find meta-path iteratively.

Specifically, in AMPG, we use a data structure to record the situation of each step. The structure records a meta-path passed by, a set of entity pairs of source nodes and their reaching nodes, and their PCRW values and the preference score S of the current structure, as Fig. 3 shown. Besides, we create a candidate set to record the structure to be handled.

The preference score S of the structure mentioned above is for judging the priority of structure handled. S measures the similarity of the whole arrival pairs in the structure. The highest S means the most relevant relationship and the most promising meta-paths, so we get the structure with the highest S at each step. The definition of preference score S is as follows:

$$S = \sum_{s} \frac{1}{T} \sum_{t} \left[\sigma \left(s, t | \prod \right) \bullet r(s) \right], \tag{3}$$

where s and t are source and reaching entity node, respectively, on meta-path $\prod$, T is the number of reaching entity nodes, and $\sigma(s, t|\prod)$ is the PCRW value. $r(s) = 1 - \alpha \bullet N$ is the contribution of s to the current structure for training pairs selection balance, where α is the decreasing coefficient of the contribution as 0.1 because of the good performance on it, and N is the number of the target nodes that s has reached through other meta-paths which are found already. It means if one of the source nodes in $\sum_s$ has more target nodes matched before, N will be larger and S will be reduced due to the smaller $r(s)$, so that the structure with other source nodes which have no or fewer matches will get high priority to be traversed greedily and has as many opportunities as possible to find a meta-path connecting them. Thus, the found meta-paths would describe the relationships of the whole training pairs comprehensively.

In order to get rid of unimportant or low pair-matched meta-paths, we set a threshold value l to judge the structures whether being put to the candidate set or not.

$$l = \epsilon \bullet |A|, \tag{4}$$

where ϵ is a limited coefficient, $|A|$ is the number of entity pairs in the input set. If S is no less than l, add this structure into the candidate set, otherwise delete it. The purpose of l is to let AMPG only further search the structures with high enough similarity and probability to match. Because $|A|$ is the size of training pairs, l also could be regarded as the minimum S to be handled, and ϵ can be seen as the minimum averages of similarity values of training pairs. In a structure, a large number of source nodes implies a high possibility of matching training pairs, and a high average similarity value of the source nodes signifies the close relation about the meta-path. Only when the two factors big enough to make S larger than the limitation l, the structure can be further handled. That is, the irrelevant or low possibility of pair-matched meta-paths could be abandoned.

Furthermore, we explain AMPG with a case study shown in Fig. 3. The training pairs are (1, 8), (2, 8), (3, 9), (4, 9), and sources nodes are 1, 2, 3, 4. The case starts with creating an initial structure No.1 and inserting it into the candidate set as Fig. 3a shows. The entity pair in No.1 is composed of the source node and itself, and no meta-path is generated at initial step. Our algorithm will read candidate set iteratively and take out the structure with the highest S at each step. For each selected structure, it will be checked firstly if any entity pairs in structure match training pairs. If not, we move outward one hop in HIN, as Fig. 3b shows. We can pass by three edge types $\xrightarrow{isCitizenOf}$, $\xrightarrow{wasBornIn}$ and $\xrightarrow{WorkAt}$. For each passed edge type, we create new structures like No.2 to No.4 which are constructed by the way introduced above. Then, we check the new structures whether fit the conditions of expanding further which are presented above, and insert them into the candidate set. Remove the used structure and read next structure. Otherwise, as Fig. 3c shows, four pairs of No.2 are matched, so a new relevant meta-path $\xrightarrow{isCitizenOf}$ of No.2 is selected and its similarity value vector is recorded. Remove the used structure No.2 and continue to handle the next structure with the highest S cyclically. The algorithm would terminate when the candidate set is empty.

The process of AMPG is described in Algorithm 1 in detail. Steps 1–2 are the variable initialization step. Steps 3–26 show the main process of searching meta-paths by greedy S in a loop. In every searching movement, we pop the structure with the largest S to handle until the candidate set is empty. Finally, the algorithm will generate a set of meta-paths with the related similarity matrix of training pairs.

Algorithm 1: AMPG(G, ϕ)

Input: G: schema-rich HIN; ϕ: set of entity training pairs;
Output: Υ: set of selected meta-paths; M: similarity matrix of ϕ corresponding to Υ.

1 $N \Leftarrow \{0,0,\ldots,0\}$; //length: $|\phi|$; element is times of each training pair matched to calculate S
2 Create the starting structure and insert to candidate set T
3 **while** T *is not empty* **do**
4 $m \Leftarrow \{0,0,\ldots,0\}$; //length: $|\phi|$; record if meta-path has pairs matched in this expanding
5 $W \Leftarrow$ popping the structure with the largest score S from T .
6 **for** *each pair* $(q, p) \in W$ **do**
7 **if** $(q, p) \in \phi$ **then**
8 $m(q, p) \Leftarrow \sigma(q, p|\prod)$;
9 $N(q, p) \Leftarrow N(q, p) + 1$;
10 **if** m *has nonzero element* **then**
11 add the meta-path $\prod$ of W into Υ;
12 $M \Leftarrow M \bigcup m$;
13 break;
14 **else**
15 create a empty temp Map E inserted with (next passed link, related structure);
16 **for** *each pair* $(q, p) \in W$ **do**
17 **for** *each neighbor* s *without passed in HIN* G **do**
18 $u^d \Leftarrow$ edge type u with direct d from p to s //forward: d=1; reverse:d=-1
19 **if** E *does not have the key* u^d *or the related structure* **then**
20 create a new structure N from W adding into E.
21 $\prod \Leftarrow$ the meta-path of N
22 insert the tuple$((q, s), \sigma(q, s|\prod))$ to N
23 **for** *each structure* $K \in E$ **do**
24 $K.S \Leftarrow$ calculated by Equation (3)
25 **if** $K.S >$ *threshold value* l **then**
26 add K into T
27 return Υ, M

4.2 Integration of meta-path

Each meta-path found by AMPG is important but has different importance for further link prediction. For example, $< BarackObama, America >$ can be used as training pair in both $\xrightarrow{isLeaderOf}$ and $\xrightarrow{IsCitizenOf}$ tasks and path P_l: $Person \xrightarrow{livein} Country$ could be found in both tasks with the same similarity value. Actually, P_l well describes the meaning of $\xrightarrow{IsCitizenOf}$, but it is not useful for the task $\xrightarrow{isLeaderOf}$. Therefore, the same path feature in different tasks may have different weights, and it is necessary to develop a solution of measuring the importance of each meta-path and integrating them properly into a particular link prediction model.

The link prediction can be considered as a classification problem. Besides, there also may be a few nonsignificant meta-paths which may connect any pair. So we need to use the positive and negative samples to train a model to predict whether the link exists between the given pairs or not. Positive samples are the training pairs, while negative samples are generated by replacing the target nodes of the training pairs with the same-typed nodes without the same relations. Thus,

positive value is the similarity value vector of each positive pair on all selected meta-paths, while negative value is the vector of negative pair.

For training model, we assume that the weight of each meta-path $\prod_i$ is $\varpi_i (i = 1, \cdots, N), \varpi_i \geq 0$, and $\sum_{i=1}^{N} \varpi_i = 1$. In order to train the appropriate path weights, we use the log-likelihood function to build our model. Using this classical function, we could let the weights of important paths large and the weights of nonsignificant paths small, so that the model would distinguish the positive and negative pairs well. The specific formula is as follows:

$$\max h = \sum_{x^+ \in q^+} \frac{ln(t(\varpi, x^+))}{|q^+|} + \sum_{x^- \in q^-} \frac{ln(1 - t(\varpi, x^-))}{|q^-|} - \frac{||\varpi||^2}{2}, \tag{5}$$

where $t(\varpi, x)$ is the sigmoid function (i.e., $t(\varpi, x) = \frac{e^{\varpi^T x}}{e^{\varpi^T x} + 1}$). x is similarity value vector of sample pair in all selected paths, x^+ positive sample and x^- the negative. q^+ is similarity matrix of positive pairs made of x^+. And q^- is similarity matrix of negative pairs made of x^-. $\frac{||\varpi||^2}{2}$ is the regularizer to avoid overfitting.

After learning weights of relevant meta-paths Υ, we use a logistic regression model to integrate meta-paths for link prediction.

$$\eta(s, t|\Upsilon) = (1 + e^{-(\sum_{x \in \Upsilon} \varpi_x \bullet \sigma(s,t|\prod_x) + \varpi_0)})^{-1}, \tag{6}$$

where (s, t) is the pair we should do link prediction, x is each selected meta-path feature, and $\sigma(s, t|\prod_x)$ is the similarity value between s and t based on the path x, while ϖ_x is the weight of x we learned above. And Υ is the set of selected meta-paths. If $\eta(s, t|\Upsilon)$ is larger than a specific value, we judge that they would be connected by the link predicted.

5 Experiment

In order to verify the superiority of our designed method of link prediction in schema-rich HIN, we conduct a series of relevant experiments and validate the effectiveness of LiPaP from four aspects.

5.1 Dataset

In our experiments, we use Yago to conduct relevant experiments, and it is a large-scale Knowledge Graph, which derived from Wikipedia, WordNet and GeoNames [21]. The dataset includes more than ten million entities and 120 million facts made from these entities. We only adopt "$COREFact''$ of this dataset, which contains 4484914 facts, 35 relationships and 1369931 entities of 3455 types. A fact is a triple: $< entity, relationship, entity >$, e.g.,$< NewYork, locatedin, UnitedStates >$.

5.2 Criteria

We use receiver operating characteristic curve known as ROC curve to evaluate the performance of different methods. ROC curve is defined as a plot of true positive rate (TPR) as the y coordinate versus false positive rate (FPR) as the x coordinate. TPR is the ratio of the number of true positive decisions and actually positive cases, while FPR is the ratio of the number of false positive decisions and actually negative cases. The area under the curve is referred to as the AUC. The larger the area is, the larger the accuracy in prediction is.

5.3 Effectiveness experiments

This section will validate the effectiveness of our prediction method LiPaP on accurately predicting links existing in entity pairs. Since there are no existing solutions for this problem, as a baseline (called PCRW [12]), we enumerate all meta-paths, and the same weight learning method with LiPaP employed. Because meta-paths with length more than 4 are most irrelevant, the PCRW enumerates the meta-paths with the length no more than 1, 2, 3 and 4, and the corresponding methods are called PCRW-1, PCRW-2, PCRW-3 and PCRW-4, respectively. Besides, we use PathSim method as another baseline [24] to measure the symmetric paths. And we limit the length no more than 2 and 4, called PathSim-2 and PathSim-4, respectively. Based on Yago dataset, we randomly and respectively select 200 entity pairs from three relations $\xrightarrow{isLocatedIn}$, $\xrightarrow{isCitizenOf}$ and $\xrightarrow{isLeaderOf}$. Note that, we assume that these three types of links are not available in the prediction task. In this experiment, 100 entity pairs of them are used as training set, and the others are used as test set. In LiPaP, we set ϵ in Equation (4) in the range between 0.005 and 0.01 and set optimal values. And the max path length is also limited to 4.

Firstly, we show that the meta-paths generated by AMPG are effective and the generated order is corresponding to relevance by case studies. In order to intuitively observe the effectiveness of meta-paths generated, Table 1 shows the top four generated meta-paths, the corresponding training weights, and the order of them for the $\xrightarrow{isCitizenOf}$ task. And Table 2 shows the results for $\xrightarrow{isLocatedOf}$ task.

In Table 1, we list top four $\xrightarrow{isCitizenOf}$ related meta-paths generated by AMPG. In our general understanding, the citizen relationship means that a person is born in or lives in a country. The top three paths we found in the list exactly

Table 1 Most relevant four meta-paths for $isCitizenOf$

Meta-path	Weight	Order
Person $\xrightarrow{wasBornIn}$ City $\xrightarrow{islocatedIn}$ Country	0.1425	2
Person $\xrightarrow{livesIn}$ Country	0.0819	1
Person $\xrightarrow{livesIn}$ City $\xrightarrow{islocatedIn}$ Country	0.0744	3
Person $\xrightarrow{wasBornIn}$ City $\xleftarrow{isLeaderOf}$ Person $\xrightarrow{graduatedFrom}$ University $\xrightarrow{islocatedIn}$ Country	0.0609	10

Table 2 Most relevant four meta-paths for $isLocatedIn$

Meta-path	Weight	Order
City $\xleftarrow{livesIn}$ Person $\xrightarrow{liveIn}$ Country	0.2009	2
City $\xleftarrow{hasCapital}$ County	0.1506	1
City $\xleftarrow{happenedIn}$ Event $\xrightarrow{happenIn}$ County	0.0900	3
City $\xleftarrow{liveIn}$ Person $\xrightarrow{bornIn}$ City $\xrightarrow{hasCapital}$ County	0.0460	4

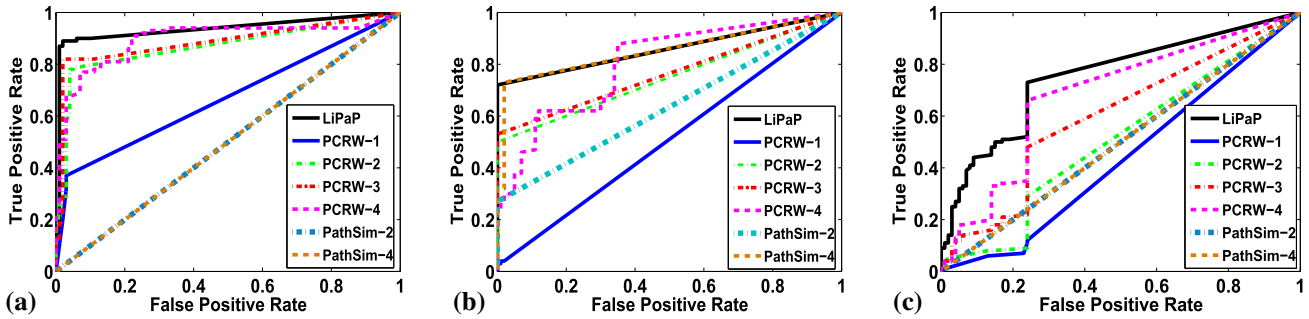

Fig. 4 Prediction accuracy of different methods on three link prediction tasks. **a** IsCitizenOf. **b** IsLocatedIn. **c** IsLeaderOf

describe the same meaning, while the last one with length 4 seems not so related to it, but actually it has certain logistic relation to the target link. However, these long and important meta-paths will be missed if the maximum length of meta-path was limited too short, as PCRW does. While our method can automatically find these paths. Besides, the top three paths in the list are the first three ones generated by AMPG in order. Although the 4th path is the 10th generated, the weights of 4th–9th generated paths are close to its. Namely, AMPG is able to find relevant meta-paths in order approximatively. Although it may not enable the order of generating paths to be the totally importance order, it can achieve an approximate effect. That's because AMPG chooses the most possible relevant paths to expand. So we need further integration method to assign appropriate importance to them.

Table 2 also shows the top four important meta-paths on $\xrightarrow{isLocatedIn}$, which perfectly describe "isLocatedIn." We take the first meta-path for example; it shows that a person lives in a city and a country at the same time which indicates the city must be located in the country. And the order of weight is also corresponding to the generating order. What is more, we find out some missing facts on "isLocatedIn," for instance, "Jerusalem $\xleftarrow{happenedIn}$ State of Palestine." Because of this, our algorithm would be useful to predict and complement the missing links in Yago.

Secondly, we entirely display the effectiveness results of the three link prediction tasks, as shown in Fig. 4. It is clear that LiPaP has much higher accuracy than all the PCRW and PathSim methods in every tasks, which implies that LiPaP can not only effectively generate useful meta-paths but also make a good model to perform excellently in link prediction. Moreover, the PCRW generally has better performance when the path length is longer, since it can exploit more useful meta-paths. However, it will cost more to search more meta-paths, most of which are irrelevant. And these irrelevant paths are noisy which may make the model worse. For example, PCRW-3 generates more than 80, 70 and 40 paths, and PCRW-4 finds more than 600, 290 and 190 paths with lots of irrelevant paths for the $\xrightarrow{isCitizenOf}$, $\xrightarrow{isLocatedIn}$ and $\xrightarrow{isLeaderOf}$ tasks, respectively. On the contrary, LiPaP only generates 30, 24 and 30 meta-paths in these tasks and has better accuracy. In addition, PathSim can only get not-bad

results in the $\xrightarrow{isLocatedIn}$ task, but in other tasks PathSim has the worst performance. PathSim only exploits symmetric paths to calculate the similarity of two entities so that it is effective for same-typed entity nodes, while it can not find out asymmetric meta-paths to represent relation between entities; thus, it may lose many important features. In the $\xrightarrow{isLocatedIn}$ task, the classes of entity pairs are City and Country which could be seen as the same type of location, so PathsSim would get a good result. However, in other two tasks, the important meta-path features are all asymmetric so that PathSim performs badly while LiPaP has better results. Thus, LiPaP not only effectively utilizes a few meta-paths to construct a good link prediction model but also works well on the tasks with different types of entities.

5.4 Influence of the size of training set

In this section, we evaluate and study the influence of the size of training set on the prediction performance. We do these relevant experiments on the $\xrightarrow{isLocatedIn}$ and $\xrightarrow{IsCitizenOf}$ tasks. The sizes of training set are set as $\{2, 6, 10, 20, 40, 60, 80, 100\}$. Besides our LiPaP, we choose PCRW-2 as baseline, since most of its generated paths are relevant to get achieve a not-bad result compared to other PCRW methods, and the cost of space and time is not too much to waste. As illustrated in Fig. 5, when the number of training pairs is smaller than 10, the performance of both methods improves rapidly with the size of pairs growing. However, when the size is more than 10, the size of training set has little effect on the performance of both methods. We think the reason lies in that too small training set is not various enough to discover all useful meta-paths, while too large training set may introduce much noise. In a conclusion, when the size of training set is from 10 to 20 in this dataset, it is good enough to discover all useful meta-paths and avoid much noise. Furthermore, it could also save space and time to learn model and make the performance of our method better.

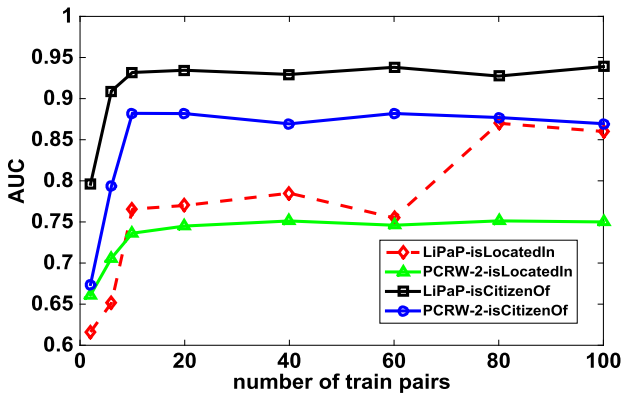

Fig. 5 Influence of different sizes of training set

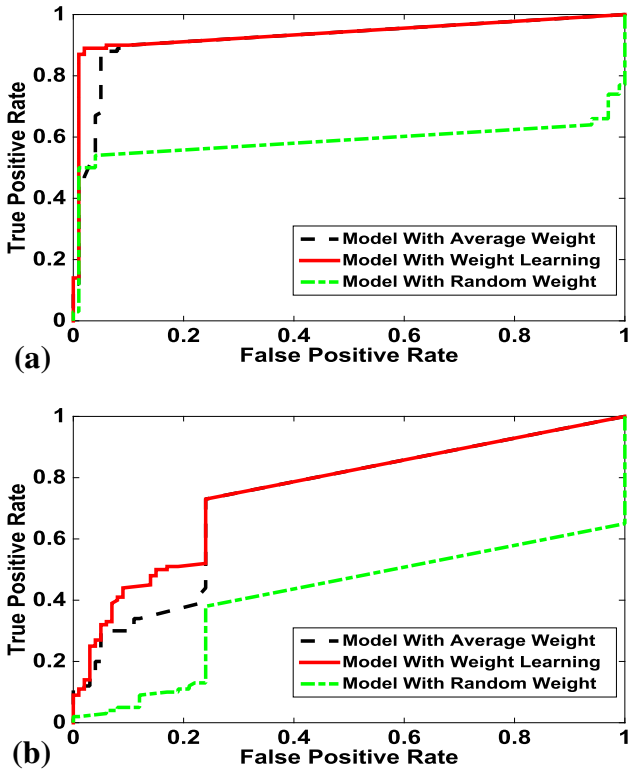

Fig. 6 Effectiveness of weight learning. **a** IsCitizenOf. **b** IsLeaderOf

5.5 Impact of weight learning

To illustrate the benefit of weight learning, we redone the experiments on the $\xrightarrow{isCitizenOf}$ and $\xrightarrow{isLeaderOf}$ tasks mentioned in Sect. 5.3. We run LiPaP with the weight learning, random weights and average weights. Figure 6 shows the performance of these methods. It is obvious that the weight learning can make prediction performance best in both tasks. The models with random weight perform worst, owing to giving the more relevant paths low weights and irrelevant paths high weights. The models with weight learning just have a little better performance than the models with average weight, because the meta-path features generated by AMPG are all relevant and important, and the most important feature also has not got a very low weight in the models with average weight. So the performance of the models with average weight is also not poor in spite of being inferior to the models with weight. Therefore, the weight learning can adjust the importance of different meta-paths so as to integrate them well and make the model better.

5.6 Efficiency

In this section, we choose five different sizes of training set, i.e., {20, 40, 60, 80, 100}, to validate the efficiency of finding meta-paths of different methods. Figure 7 demonstrates the

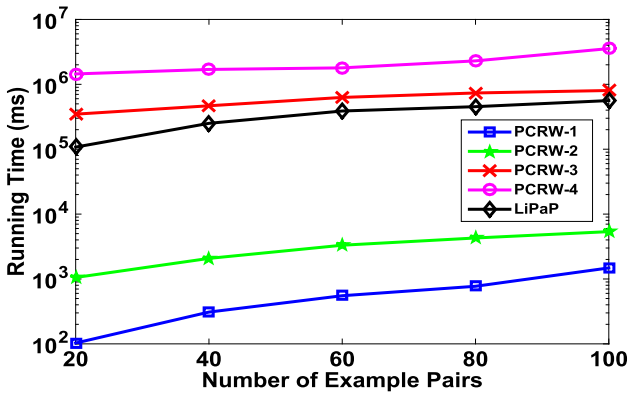

Fig. 7 Running times of different methods

running time on different models for the $\xrightarrow{isLocatedIn}$ task. It is obvious that the running times of these models approximate linearly increases with the increase in the size of training set. In spite of the small running time, the short meta-paths found by PCRW-1 and PCRW-2 restrict their prediction performance. Our LiPaP has smaller running time than PCRW-3 and PCRW-4, since it only finds a small number of important meta-paths. In this way, LiPaP has a better balance on effectiveness and efficiency.

5.7 Study on parameter ϵ

In this section, we would like to illustrate the influence of limited coefficient ϵ with different values (Equation(4)) in LiPaP. We also randomly and respectively select 200 entity pairs from the relations $\xrightarrow{isLocatedIn}$, $\xrightarrow{isCitizenOf}$ and $\xrightarrow{isLeaderOf}$, in which 100 pairs are for training set, and the others are for test set. For these link prediction tasks, given a value of ϵ, we record the number of generated meta-paths and calculate the AUC value to figure out the power in prediction result as shown in Fig. 8.

In overall view from Fig. 8, it is obvious that the number of generated meta-paths grows exponentially with the ϵ descending in (a). And we can see that the lower ϵ is, the higher accuracy is in (b). That is because that the lower ϵ value implies the less strict limitation of handling structures, so that it has more probability to find more paths. Besides, it has shown above that LiPaP finds meta-paths in descending order of relevance approximately. Though the newly generated path by decreasing ϵ value is less importance than others, which makes features rich and varied; thus, it makes the AUC value higher. However, when ϵ value is low to some extend, the accuracy will not increase with the number of meta-paths growing. The reason why it occurs is that the newly generated paths become too unimportant and noisy for prediction model. Thus, we need to try out a suitable value of ϵ to make the model work best, and it may be in the range between 0.001 and 0.01 according to the results.

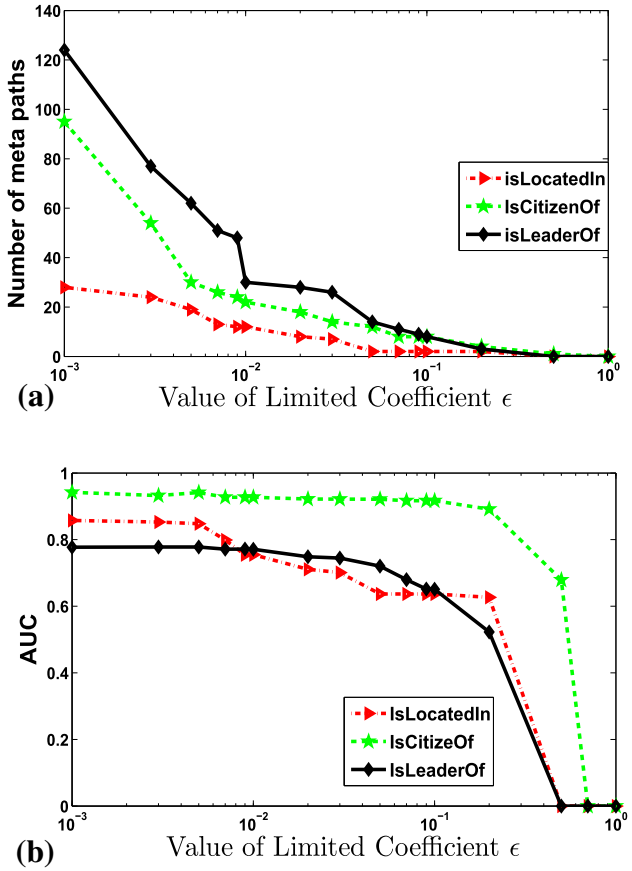

Fig. 8 Influence of various ϵ for link prediction. **a** Impact on the number of meta-path. **b** Impact on accuracy

5.8 Study on choices of similarity measure

In this section, we would figure out whether there are other similarity measures suitable for LiPaP. Considering the process in an iterative way and the similarity calculation in every expansion step, we want to find another iterative and easy-to-compute similarity measure for HIN to replace the PCRW method. The widely used similarity measures are PathCount [22], PathSim, PCRW, HeteSim, and so on. However, PathSim and HeteSim are not so suitable for our method, and the reasons have been explained above. PathCount is to count the number of relevance path which could be used simply and iteratively. So we only choose PathCount to do this experiment. We also use the same data set to redo the three tasks and set the size of selected meta-paths to be the same shown in Sect. 5.3 in order to compare the result of LiPaP with PCRW. From Fig. 9, we know that the performance of LiPaP with PathCount is not better than that of LiPaP with PCRW. The reason why LiPaP with PathCount does not have a good performance is that the meta-paths which are chosen preferentially are usually with a large number of path instances and with long length, but not in the order of relevance. The S constructed based on PathCount could not imply the importance

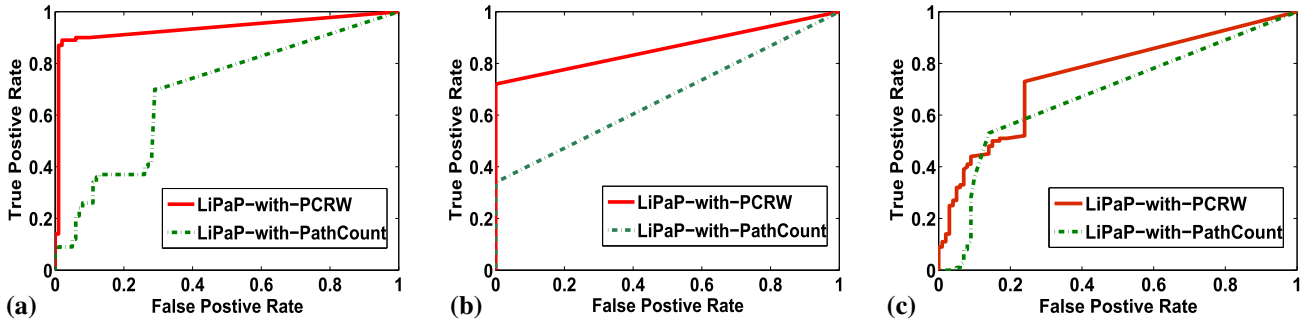

Fig. 9 Prediction accuracy with different similarity measure. **a** IsCitizenOf. **b** IsLocatedIn. **c** IsLeaderOf

of meta-paths so that it does not select meta-paths based on its importance for task. That is to say, PathCount could not distinguish the importance of different paths. For the same path, the bigger PathCount value would mean the closer relation based on the same semantics. However, for two different paths, like the $P1$: $Person \xrightarrow{bornin} City$ and $P2$: $Person \xrightarrow{workwith} Person \xrightarrow{livein} City$, we would think $P1$ is closer than $P2$ in semantics, but the PathCount value of $P1$ is less than $P2$. Nevertheless, PCRW can describe the relevance of meta-path by arriving probability, and PCRW would measure that the relation of $P1$ is closer than $P2$. The higher value is, the closer relation will be. So a high value of S constructed based on PCRW may mean a big change to find a significant meta-path. Thus, PCRW is more suitable for LiPaP than PathCount.

6 Conclusion and discussion

In this paper, we introduce a novel link prediction method in schema-rich HIN named *Li*nk *P*rediction with *a*utomatic meta *P*aths (LiPaPs), which proposes an algorithm called AMPG to automatically extract meta-paths based on given training pairs and designs a supervised method to learn weights of the extracted meta-paths to form a link prediction model. Experiments on real knowledge database, Yago, validate the effectiveness, efficiency and feasibility of LiPaP. And we also find some missing facts in Yago using LiPaP, which indicates the method would be useful to solve a problem of link completion. Besides, AMPG is useful for many other applications in schema-rich HIN, such as entity similarity search, node clustering and classification. For example, in entity similarity search task, we could use AMPG to find meta-paths to represent the similarity of entities, and utilize those meta-paths to construct similarity search model to find other similar entities.

Acknowledgements This work is supported in part by National Key Basic Research and Department (973) Program of China (No. 2013CB329606), the National Natural Science Foundation of China (Nos. 71231002, 61375058, 11571161, 11371115), the CCF-Tencent Open Fund, the Co-construction Project of Beijing Municipal Commission of Education and Shenzhen Sci.-Tech Fund No. JCYJ20140509143748226.

References

1. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: Dbpedia—a crystallization point for the web of data. Web Semant. **7**(3), 154–165 (2009)
2. Cao, B., Kong, X., Yu, P.S.: Collective prediction of multiple types of links in heterogeneous information networks. In: ICDM, pp. 50–59 (2014)
3. Cao, X., Zheng, Y., Shi, C., Li, J., Wu, B.: Link prediction in schema-rich heterogeneous information network. In: PAKDD, pp. 449–460 (2016)
4. Deng, H., Lyu, M.R., King, I.: A generalized co-hits algorithm and its application to bipartite graphs. In: KDD, pp. 239–248 (2009)
5. Dong, Y., Tang, J., Wu, S., Tian, J., Chawla, N.V., Rao, J., Cao, H.: Link prediction and recommendation across heterogeneous social networks. In: ICDM, pp. 181–190 (2012)
6. Huang, J., Nie, F., Huang, H., Tu, Y.: Trust prediction via aggregating heterogeneous social networks. In: CIKM, pp. 1774–1778 (2012)
7. Jaiwei, H.: Mining heterogeneous information networks: the next frontier. In: SIGKDD, pp. 2–3 (2012)
8. Jamali, M., Lakshmanan, L.: Heteromf: recommendation in heterogeneous information networks using context dependent factor models. In: WWW, pp. 643–654 (2013)
9. Ji, M., Sun, Y., Danilevsky, M., Han, J., Gao, J.: Graph regularized transductive classification on heterogeneous information networks. In: ECML/PKDD, pp. 570–586 (2010)
10. Kalaev, M., Bafna, V., Sharan, R.: Fast and accurate alignment of multiple protein networks. In: Vingron, M., Wong, L. (eds.) Research in Computational Molecular Biology, 12th Annual International Conference, Singapore, March 30-April 2, 2008. Lecture Notes in Computer Science, vol 4955, pp. 246–256. Springer (2008)
11. Kong, X., Yu, P.S., Ding, Y., Wild, D.J.: Meta path-based collective classification in heterogeneous information networks. In: CIKM, pp. 1567–1571 (2012)
12. Lao, N., Cohen, W.W.: Relational retrieval using a combination of path-constrained random walks. Mach. Learn. **81**(1), 53–67 (2010)
13. Liu, F., Xia, S.: Link prediction in aligned heterogeneous networks. In: PAKDD, pp. 33–44 (2015)
14. Ng, M.K., Li, X., Ye, Y.: Multirank: co-ranking for objects and relations in multi-relational data. In: KDD, pp. 1217–1225 (2011)
15. Rdf current status. http://www.w3.org/standards/techs/rdf#w3c_all

16. Shi, C., Kong, X., Huang, Y., Yu, P.S., Wu, B.: Hetesim: a general framework for relevance measure in heterogeneous networks. IEEE Trans. Knowl. Data Eng. **26**(10), 2479–2492 (2014)
17. Shi, C., Kong, X., Yu, P.S., Xie, S., Wu, B.: Relevance search in heterogeneous networks. In: EDBT, pp. 180–191 (2012)
18. Shi, C., Zhou, C., Kong, X., Yu, P.S., Liu, G., Wang, B.: Heterecom: a semantic-based recommendation system in heterogeneous networks. In: KDD, pp. 1552–1555 (2012)
19. Shih, Y., Parthasarathy, S.: Scalable global alignment for multiple biological networks. BMC Bioinform. **13**(S–3), S11 (2012)
20. Singhal, A.: Introducing the knowledge graph: things, not strings. Official Google Blog (2012)
21. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: WWW, pp. 697–706 (2007)
22. Sun, Y., Barber, R., Gupta, M., Aggarwal, C.C., Han, J.: Co-author relationship prediction in heterogeneous bibliographic networks. In: ASONAM, pp. 121–128 (2011)
23. Sun, Y., Han, J., Aggarwal, C.C., Chawla, N.V.: When will it happen? Relationship prediction in heterogeneous information networks. In: WSDM, pp. 663–672 (2012)
24. Sun, Y., Han, J., Yan, X., Yu, P., Wu, T.: Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. In: VLDB, pp. 992–1003 (2011)
25. Sun, Y., Norick, B., Han, J., Yan, X., Yu, P.S., Yu, X.: Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. In: KDD, pp. 1348–1356 (2012)
26. Sun, Y., Yu, Y., Han, J.: Ranking-based clustering of heterogeneous information networks with star network schema. In: KDD, pp. 797–806 (2009)
27. Unger, C., Bühmann, L., Lehmann, J., Ngomo, A.N., Gerber, D., Cimiano, P.: Template-based question answering over RDF data. In: WWW, pp. 639–648 (2012)
28. Yahya, M., Berberich, K., Elbassuoni, S., Ramanath, M., Tresp, V., Weikum, G.: Natural language questions for the web of data. In: EMNLP-CoNLL, pp. 379–390 (2012)
29. Yang, Y., Chawla, N.V., Sun, Y., Han, J.: Predicting links in multi-relational and heterogeneous networks. In: ICDM, pp. 755–764 (2012)
30. Yu, X., Gu, Q., Zhou, M., Han, J.: Citation prediction in heterogeneous bibliographic networks. In: SDM, pp. 1119–1130 (2012)
31. Zha, H., He, X., Ding, C.H.Q., Gu, M., Simon, H.D.: Bipartite graph partitioning and data clustering. CoRR cs.IR/0108018 (2001)
32. Zhang, J., Kong, X., Yu, P.S.: Predicting social links for new users across aligned heterogeneous social networks. In: ICDM, pp. 1289–1294 (2013)
33. Zhang, J., Yu, P.S., Lv, Y.: Organizational chart inference. In: KDD, pp. 1435–1444 (2015)
34. Zhou, D., Orshanskiy, S.A., Zha, H., Giles, C.L.: Co-ranking authors and documents in a heterogeneous network. In: ICDM, pp. 739–744 (2007)
35. Zou, L., Huang, R., Wang, H., Yu, J.X., He, W., Zhao, D.: Natural language question answering over RDF: a graph data driven approach. In: SIGMOD, pp. 313–324 (2014)