Link Prediction in Schema-Rich Heterogeneous Information Network

Xiaohuan Cao[1], Yuyan Zheng[1], Chuan Shi[1](✉), Jingzhi Li[2], and Bin Wu[1]

[1] Beijing Key Lab of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing 100876, China
devil_baba@126.com, zyy0716_source@163.com, {shichuan,wubin}@bupt.edu.cn
[2] Department of Mathematics, Southern University of Science and Technology, Shenzhen 518055, China
lijz@sustc.edu.cn

Abstract. Recent years have witnessed the boom of heterogeneous information network (HIN), which contains different types of nodes and relations. Many data mining tasks have been explored in this kind of network. Among them, link prediction is an important task to predict the potential links among nodes, which are required in many applications. The contemporary link prediction usually are based on simple HIN whose schema are bipartite or star-schema. In these HINs, the meta paths are predefined or can be enumerated. However, in many real networked data, it is hard to describe their network structure with simple schema. For example, the knowledge base with RDF format include tens of thousands types of objects and links. On this kind of schema-rich HIN, it is impossible to enumerate meta paths. In this paper, we study the link prediction in schema-rich HIN and propose a novel *Li*nk *P*rediction with *a*utomatic meta *P*aths method (LiPaP). The LiPaP designs an algorithm called Automatic Meta Path Generation (AMPG) to automatically extract meta paths from schema-rich HIN and a supervised method with likelihood function to learn weights of the extracted meta paths. Experiments on real knowledge database, Yago, validate that LiPaP is an effective, steady and efficient method.

Keywords: Heterogeneous Information Network · Link prediction · Similarity measure · Meta path

1 Introduction

Nowadays, the study of Heterogeneous Information Network (HIN) become more and more popular in data mining area [5], 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 [14], and classification [7]. Among those researches in HIN, link prediction is a fundamental problem that attempts to estimate the likelihood of the existence of a link between two nodes, based on observed links and the attributes of nodes. Link prediction is the base of many data mining tasks, such as data clearness and recommendation.

Some works have been done to predict link existence in HIN. Because of the unique semantic characteristic of HIN, meta path [14], a sequence of relations connecting two nodes, is widely used for link prediction. Utilizing the meta path, these works usually employ a two-step process to solve 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 [3,12,13,15]. For example, Sun et al. [12] propose PathPredict to solve the problem of co-author relationship prediction, Cao et al. [3] propose an iterative framework to predict multiple types of links collectively in HIN, and Sun et al. [13] model the distribution of relationship building time to predict when a certain relationship will be formed. These works usually 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 manually enumerate some meaningful and short meta paths [14]. For example, a biboligraphic network with star schema is used in [12,13,15] and only several meta paths are enumerated.

However, in many real networked data, the network structures are more complex, and meta paths cannot be enumerated. Knowledge graph is the base of the contemporary search engine [10], where its resource description framework (RDF) [1] $< object, relation, object >$ naturally constructs a HIN. In such a HIN, the types of nodes and relations are huge. For example, DBpedia [2], a kind of knowledge graph, has recorded more than 38 million entities and 3 billion facts. In this kind of network, it is hard to describe them with simple schema, so 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 to connect two object types. For example, for Person and Country types, there are two meta paths: $Person \xrightarrow{bornin} City \xrightarrow{locatedIn} Country$ and $Person \xrightarrow{Diedin} City \xrightarrow{hasCapital^{-1}} Country$. Note that Fig. 1 is one extreme little part of the whole DBpedia network, and there will be huge number of

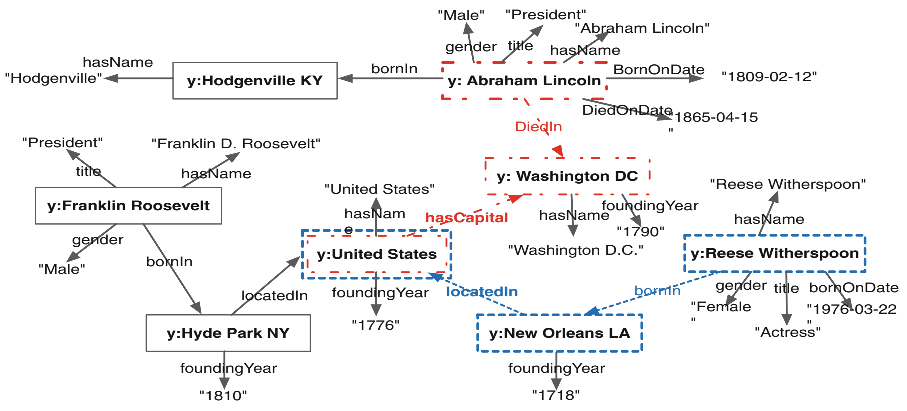


Fig. 1. A snapshot of the RDF structure extracted from DBpedia.

meta paths can connect Person and Country in a real network. So that the meta paths in this kind of schema-rich HIN are too many to enumerate and it's hard to analyze them.

To be specific, the challenges of link prediction in schema-rich HIN are mainly from two aspects. (1) The meta path cannot be enumerated. 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's impossible to enumerate 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. So that we need to learn weight for each meta path, where the weight represents the importance of paths for link prediction.

In this paper, we study the link prediction in schema-rich HIN and propose the *Li*nk *P*rediction with *a*utomatic meta *P*aths method (LiPaP). The LiPaP designs a novel algorithm, called Automatic Meta Path Generation (AMPG), to automatically extract meta paths from schema-rich HIN. And then we design an supervised method with likelihood function to learn the weights of meta paths. On a real knowledge base Yago, we do extensive experiments to validate the performances of LiPaP. 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.

2 Preliminary and Problem Definition

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

The **Heterogeneous Information Network (HIN)** [5] 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, a 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, 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 [6].

In a HIN, there can be different paths connecting two entity nodes and these paths are called as **meta path** [14]. 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}, is 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 base in Fig. 1 as an example, we can consider the knowledge base as an HIN, which includes many different node types (e.g., person, city, country) and link types (e.g., bornIn and locatedIn). Two node types can be connected by multiple meta paths. For example, there are two meta paths connnecting Person and Country: $Person \xrightarrow{bornin} City \xrightarrow{locatedIn} Country$ and $Person \xrightarrow{Diedin} City \xrightarrow{hasCapital^{-1}} Country$.

Traditional HIN usually has a simple network schema, such as bipartite [16] and star schema [9]. However, in some complex HINs, there are 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 simple HIN, the meta paths can be easily enumerated, but it is difficult to do the same in the schema-rich HIN. 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 entity node pairs $\phi = \{(s_i, t_i)|1 \leq i \leq k\}$, search a set of meta paths $\Upsilon = \{\prod_i |1 \leq i \leq e\}$ 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 $\psi = \{(u_i, v_i)|1 \leq i \leq r\}$.

3 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 method (LiPaP). This method includes two steps: Firstly, we design an algorithm called **Automatic Meta Path Generation (AMPG)** to discover useful meta paths with training pairs automatically. Secondly, we use a supervised method to integrate meta paths to form a model for further prediction.

3.1 Automatic Meta Path Generation

In order to extract the appropriate and relevant meta paths as model features for link prediction, we would like to show the AMPG algorithm, which can generate useful meta paths smartly in schema-rich HIN. We would illustrate 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. These paths to be found would not only connect more training pairs, but also show much closer relationship to

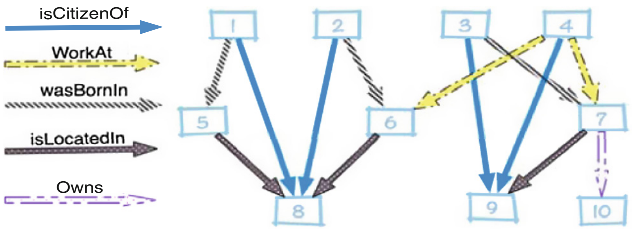


Fig. 2. Subgraph example of schema-rich HIN.

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 enough. It guarantees that the generated meta paths all well describe the relationship between each training pairs and the selected paths are not too many to add noise 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 similarity score S as a selection criterion based on a similarity measurement Path-Constrained Random Walk (PCRW) [8], which is to calculate the relevance between the given entity pairs in the meta paths. The higher similarity score S is, more likely the meta path is to be chosen.

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 reached and their PCRW values and the similarity score S of the current structure, as Fig. 3 shown. Besides, we create a candidate set to record the structure to be handled.

The similarity score S of the structure mentioned above is for judging the priority of the structure. 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 every step. The definition of similarity score S is as follows:

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

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 selected paths. It means if one of 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 fewer matches will get high priority to be traversed greedily.

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

$$l = \epsilon \bullet |A|, \tag{2}$$

where ϵ is a limited coefficient, $|A|$ is the number of entity pairs in the structure. If S is no less than l, add this structure into the candidate set, otherwise delete it.

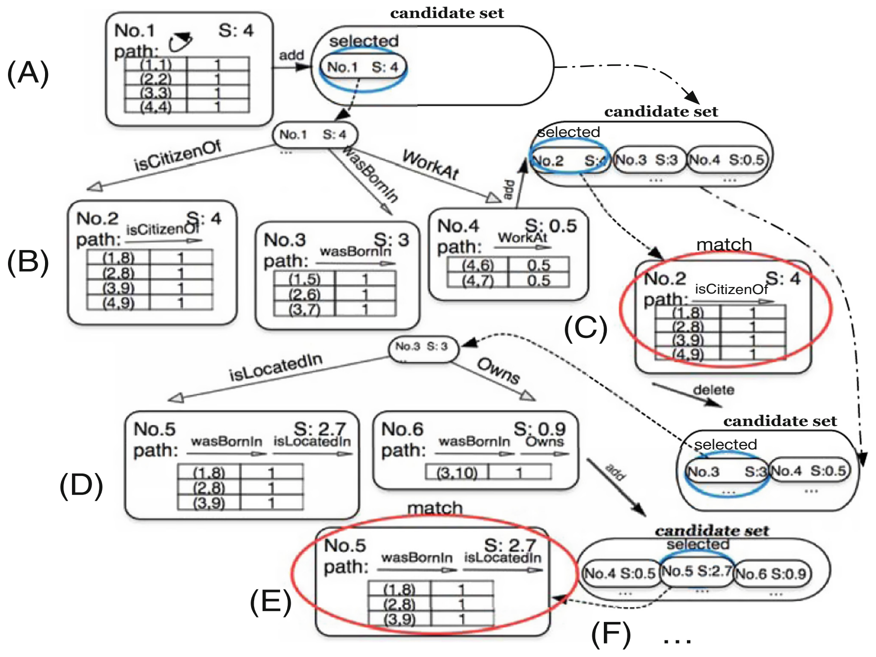


Fig. 3. An example of meta-path automatic generation.

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 inserts it into the candidate set as Fig. 3(A) shown. The entity pair is composed of the source node and itself and no meta path is generated at this step. Our algorithm will read candidate set iteratively and choose the structure with highest S at each step. For each selected structure, it will be checked if any training pairs are matched. If not, we move one step in HIN, as Fig. 3(B) shown. 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 and No.4. Then, we check the new structures whether fit the conditions of expanding further and insert them into the candidate set. Remove the used structure No.1 and read next structure. Otherwise, as Fig. 3(C) shown, four pairs are matched, so a new relevant meta path $\xrightarrow{isCitizenOf}$ is generated and its similarity value vector is recorded. Remove the used structure No.2 and continue to read next. The algorithm terminates when the candidate set is empty.

The detail process of AMPG is described in Algorithm 1. Step 1–2 is the variable initialization step. Step 3–26 shows 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: $G$: shema-rich HIN; $\phi$: set of entity training pairs;
   Output: $\Upsilon$: set of selected meta paths; $M$: similarity matrix of $\phi$ corresponding to $\Upsilon$.
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 $\Upsilon$;
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             $K.S \Leftarrow$ cauculated by Equation (1)
25             if $K.S >$ threshold value $l$ then
26                 add $K$ into $T$
27 return $\Upsilon$, $M$
```

3.2 Integration of Meta Path

Each meta path found by AMPG is important but has different importances for further link prediction. It's necessary to find a solution of measuring the importance for each meta path and integrating them into a link prediction model.

The link prediction can be considered as a classification problem. So we 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. The specific formula is as follows:

$$\max h = \sum_{x^+ \in q^+} \frac{ln(t(\varpi, x^+))}{|q^+|} + \sum_{x^- \in q^-} \frac{ln(1 - t(\varpi, x^-))}{|q^-|} - \frac{||\varpi||^2}{2}, \tag{3}$$

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^- 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.

$$\eta(s,t|\Upsilon) = (1 + e^{-(\sum_{x\in\Upsilon} \varpi_x \bullet \sigma(s,t|\prod_x) + \varpi_0)})^{-1}, \tag{4}$$

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

4 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.

4.1 Dataset

In our experiments, we use Yago to conduct relevant experiments and it is a large-scale knowledge graph, which derives from Wikipedia, WordNet and GeoNames [11]. 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., $< NewYork, locatedin, UnitedStates >$.

4.2 Criteria

We use receiver operating characteristic curve known as ROC curve to evaluate the performance of different methods. It 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.

4.3 Effectiveness Experiments

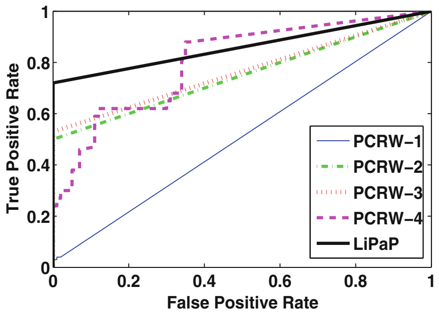
This section will validate the effectiveness of our prediction method LiPaP on accurately predicting links existing in entity pairs. Since there are no existing solutions for this problem, as a baseline (called PCRW [8]), we enumerate all meta paths, and the same weight learning method with LiPaP is 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. Based on Yago dataset, we randomly and respectively select 200 entity pairs from two relations $\xrightarrow{isLocatedIn}$ and $\xrightarrow{isCitizenOf}$. Note that, we assume that these two types of links are not available in the prediction task. In this experiment, 100 entities pairs of them are used as the training set, the other are used as the test set. In LiPaP, we set ϵ in Eq. (2) as 0.005 and the max path length is also limited to 4.

The results of two link prediction tasks are shown in Fig. 4. It is clear that LiPaP has better performances than all PCRW methods, which implies that LiPaP can effectively generate useful meta paths. Moreover, the PCRW generally has better performance when the path length is longer, since it can exploit more useful meta paths. However, it will take more cost to search more meta paths, most of which are irrelevant. For example, PCRW-3 generates more than 80 paths and PCRW-4 finds more than 600 paths with lots of irrelevant paths. On the contrary, LiPaP only generates 30 meta paths for the $\xrightarrow{isCitizenOf}$ task.

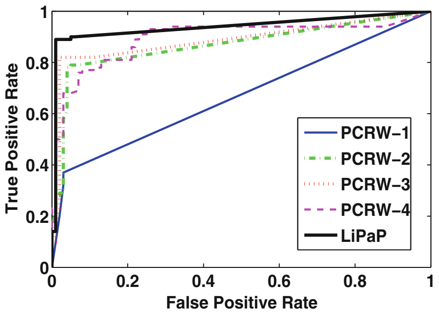
In order to intuitively observe the effectiveness of meta paths found, Table 1 shows the top 4 generated meta paths and the corresponding training weights for the $\xrightarrow{isCitizenOf}$ task. It is obvious that 4 meta paths are all relevant to the link $\xrightarrow{isCitizenOf}$. The most relevant one is the first meta path which shows the fact that a person is born in a city and the city is located in a country. It describes the citizen relationship in fact. The last one with length 4 seems not

Table 1. Most relevant 4 meta paths for *isCitizenOf*

Meta path	Weight
Person $\xrightarrow{wasBornIn}$ City $\xrightarrow{islocatedIn}$ County	0.1425
Person $\xrightarrow{livesIn}$ County	0.0819
Person $\xrightarrow{livesIn}$ City $\xrightarrow{islocatedIn}$ County	0.0744
Person $\xrightarrow{wasBornIn}$ City $\xleftarrow{isLeaderOf}$ Person $\xrightarrow{graduatedFrom}$ university $\xrightarrow{islocatedIn}$ County	0.0609



(a) IsLocatedIn



(b) IsCitizenOf

Fig. 4. Prediction accuracy of different methods on two link prediction tasks.

to be close, but actually has certain logistic relation with the link $\xrightarrow{isCitizenOf}$. However, these long and important meta paths can be missed if the maximum length of meta path was limited too short, as PCRW does. While our method can automatically find these paths and assign them a high importance.

4.4 Influence of the Size of Training Set

In this section, we evaluate the influence of the size of training set on the prediction performances. The size of training set are set with $\{2, 6, 10, 20, 40, 60, 80, 100\}$. Besides our LiPaP, we choose PCRW-2 as baseline, since it can generate most of useful meta paths and achieve good performances compared to other PCRW methods. As illustrated in Fig. 5, when the number of training pairs is smaller than 10, the performances of both methods improve 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 performances of both methods. We think the reason lies in that too small training set cannot discover all useful meta paths, while large training set may introduce much noise. 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 can save space and time to learn model and make the performance of our method better.

4.5 Impact of Weight Learning

To illustrate the benefit of weight learning, we redone the experiments on the $\xrightarrow{isCitizenOf}$ task mentioned in Sect. 4.3. We run LiPaP with the weight learning or random weights, and with average weights. Figure 6 shows the performances of these methods. It is obvious that the weight learning can improve prediction performances. The model with random weight performs worst, owing to giving the more relevant paths low weights. The model with weight just has a little better performance than the model with average weight, because the meta path

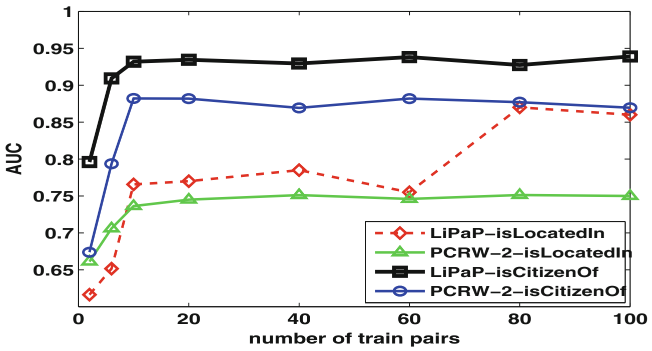


Fig. 5. Influence of different sizes of training set.

features generated by AMPG are all relevant and important, the most important feature also has not get a very low weight in the model with average weight. So the performance of the model with average weight is also not poor in spite of being inferior to the model 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.

4.6 Efficiency

In this section, we choose 5 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 running time on different models for the $\xrightarrow{isLocatedIn}$ task. It is obvious that the running time of these models approximate linearly increase with the increase of 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 performances. 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.

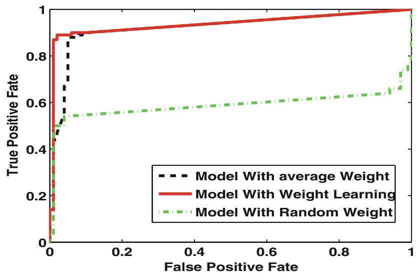


Fig. 6. Effectiveness of weight learning.

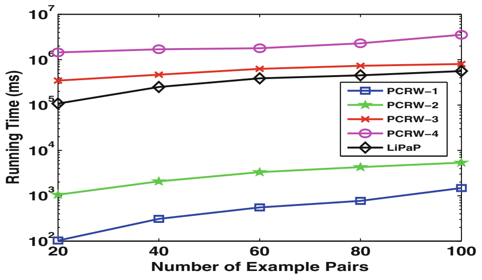


Fig. 7. Running times of different methods.

5 Conclusions

In this paper, we introduce a novel link prediction method in schema-rich HIN named *L*ink *P*rediction with *a*utomatic meta *P*aths (LiPaP), which proposes an algorithm called AMPG to automatically extract meta paths based on given training pairs and designs an 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.

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

References

1. Rdf current status. http://www.w3.org/standards/techs/rdf#w3c_all
2. 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.: Sci. Serv. Agents World Wide Web **7**(3), 154–165 (2009)
3. Cao, B., Kong, X., Yu, P.S.: Collective prediction of multiple types of links in heterogeneous information networks. In: ICDM, pp. 50–59 (2014)
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. Jaiwei, H.: Mining heterogeneous information networks: the next frontier. In: SIGKDD, pp. 2–3 (2012)
6. Jamali, M., Lakshmanan, L.: HeteroMF: recommendation in heterogeneous information networks using context dependent factor models. In: WWW, pp. 643–654 (2013)
7. 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)
8. Lao, N., Cohen, W.W.: Relational retrieval using a combination of path-constrained random walks. Mach. Learn. **81**(1), 53–67 (2010)
9. Shi, C., Kong, X., Yu, P.S., Xie, S., Wu, B.: Relevance search in heterogeneous networks. In: EDBT, pp. 180–191 (2012)
10. Singhal, A.: Introducing the knowledge graph: things, not strings. Official Google Blog (2012)
11. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: WWW, pp. 697–706 (2007)
12. 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)
13. 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)
14. 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)
15. Yu, X., Gu, Q., Zhou, M., Han, J.: Citation prediction in heterogeneous bibliographic networks. In: SDM, pp. 1119–1130 (2012)
16. Zha, H., He, X., Ding, C.H.Q., Gu, M., Simon, H.D.: Bipartite graph partitioning and data clustering (2001). CoRR cs.IR/0108018