Discovering Meta-Paths in Large Heterogeneous Information Networks

Changping Meng*
Purdue University
West Lafayette, US
meng40@purdue.edu

Reynold Cheng
University of Hong Kong
Pokfulam Road, Hong Kong
ckcheng@cs.hku.hk

Silviu Maniu*
Noah's Ark Lab, Huawei
Science Park, Hong Kong
silviu.maniu@huawei.com

Pierre Senellart
Télécom ParisTech; CNRS LTCI
& NUS; CNRS IPAL
pierre@senellart.com

Wangda Zhang
University of Hong Kong
Pokfulam Road, Hong Kong
wdzhang2@cs.hku.hk

ABSTRACT

The Heterogeneous Information Network (HIN) is a graph data model in which nodes and edges are annotated with class and relationship labels. Large and complex datasets, such as Yago or DBLP, can be modeled as HINs. Recent work has studied how to make use of these rich information sources. In particular, *meta-paths*, which represent sequences of node classes and edge types between two nodes in a HIN, have been proposed for such tasks as information retrieval, decision making, and product recommendation. Current methods assume meta-paths are found by domain experts. However, in a large and complex HIN, retrieving meta-paths manually can be tedious and difficult. We thus study how to discover meta-paths automatically. Specifically, users are asked to provide example pairs of nodes that exhibit high proximity. We then investigate how to generate meta-paths that can best explain the relationship between these node pairs. Since this problem is computationally intractable, we propose a greedy algorithm to select the most relevant meta-paths. We also present a data structure to enable efficient execution of this algorithm. We further incorporate hierarchical relationships among node classes in our solutions. Extensive experiments on real-world HIN show that our approach captures important meta-paths in an efficient and scalable manner.

1. INTRODUCTION

Heterogeneous Information Networks (HINs), such as *Yago* [17] and *DBpedia* [2], has attracted plenty of attention in recent years. These graph data sources, which contains a gigantic number of inter-related facts, enables discovery of interesting knowledge. Yago, for instance, stores over 10 million entities and 120 million facts [17]. Figure 1 illustrates a knowledge base, a kind of HIN, which captures the relationships among entities (or graph nodes). Each node and edge is associated with a "label", indicative of the node class and edge type, respectively. For example, *Barack Obama* (node 1) has class labels USPresident and Writer. One of his offsprings is

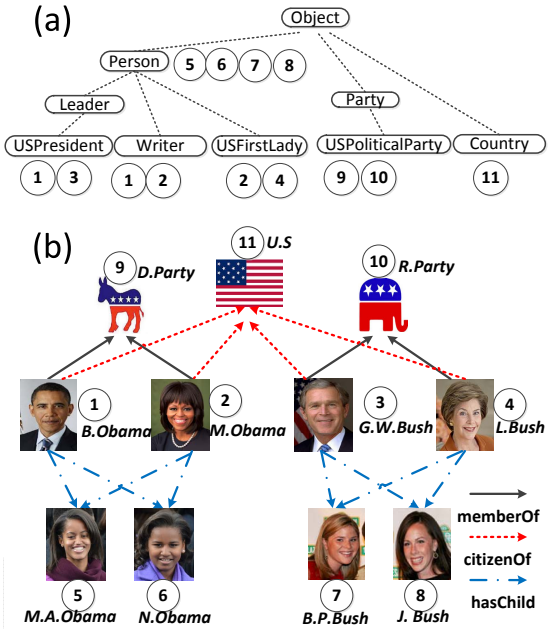

Figure 1: HIN with class hierarchy

Malia Obama (node 5), as indicated by the hasChild edge. A *class hierarchy* describes the relationships among node class labels, as shown in Figure 1(a).

The problem of understanding the vast amount of information modeled in HINs has received a lot of interest. In particular, the concept of *meta-paths* has been proposed [10, 19, 20] to explain how nodes are related. A meta-path is a sequence of node classes and edge types between two given nodes. In Figure 1, *Barack Obama* and *Michelle Obama* are connected by the following three different meta-paths:[1]

$$\mathsf{USPresident} \xrightarrow{\mathsf{hasChild}} \mathsf{Person} \xrightarrow{\mathsf{hasChild}^{-1}} \mathsf{USFirstLady},$$

$$\mathsf{USPresident} \xrightarrow{\mathsf{memberOf}} \mathsf{USPoliticalParty} \xrightarrow{\mathsf{memberOf}^{-1}} \mathsf{USFirstLady},$$

$$\mathsf{USPresident} \xrightarrow{\mathsf{citizenOf}} \mathsf{Country} \xrightarrow{\mathsf{citizenOf}^{-1}} \mathsf{USFirstLady}.$$

*Work mainly done while the authors were affiliated with University of Hong Kong.

[1] We use $\mathsf{hasChild}^{-1}$ to denote the opposite direction of the edge labeled hasChild.

WWW 2015, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3469-3/15/05.
http://dx.doi.org/10.1145/2736277.2741123 .

Meta-paths can be used to predict the closeness, or *similarity*, among graph nodes. This is especially useful when an edge between two nodes does not exist. As in Figure 1, assume, for the sake of argument, that there is no edge between *Barack Obama* and *Michelle Obama*. However, the two nodes are closely related, as illustrated by the three meta-paths above. Notice that a meta-path can have a path length of more than one. Thus, it can be used to explain sophisticated relationships among nodes that cannot be depicted by a single edge. Recently, researchers have studied how to use meta-paths between two heterogeneous graph nodes to quantify their similarity. They have proposed *meta-path–based similarity measures* (e.g., *path count* (PC) [19], *path constraint random walk* (PCRW) [10], and *HeteSim* (HS) [16]). These metrics on meta-paths between two nodes can be used for link prediction, product recommendation, and decision making. For example, in bibliography networks (e.g., *DBLP* [11]), these measures can be used to predict relationships among authors and research topics. In a recommendation system that stores salesperson, customer, and product information, these similarity values can be used to suggest products to be promoted to customers, and advise salespersons about potential customers [10, 19, 20].

Prior works. Although meta-paths are useful, the important issue of how to generate them has not been well studied. Most existing work assumes that meta-paths are provided by domain experts. While this assumption may be true for a relatively simple HIN (e.g., DBLP [11]), we question its validity for larger ones such as Yago and DBpedia. For these cases, millions of nodes and edges are annotated with thousands of labels, and node class labels exhibit complex hierarchical relationships (as in Figure 1). Moreover, long meta-paths can be extremely difficult to discover, especially if this is done without the aid of machines. The only meta-path generation solution is proposed in [10], and yields meta-paths within a fixed length l. However, it is not clear how l should be set. As we found in our experiments, the performance and effectiveness of their approach is very sensitive to l. In [5], a method called AMIE was proposed to mine association rules from HIN. While association rules are similar to meta-paths, the output of AMIE is "global", i.e., the individual preferences of a user are not considered. On the contrary, our method can be *customized*; a user can provide her opinions about the meta-paths to be generated, through *example node pairs*, as we discuss next. Moreover, AMIE does not use in any special manner the hierarchy of node classes, which we found to be very important in understanding relationships.

Our goal is to develop a systematic and user-friendly solution for efficiently discovering important meta-paths in large knowledge bases. In our approach, users provide example pairs of nodes that exhibit high proximity (e.g., *George W. Bush* and *Laura Bush*). They do not need to specify a maximum length l. Meta-paths that best explain the similarity between these node pairs are then automatically generated. A simple way to do this is to enumerate all the possible meta-paths and the possible meta-path subsets between the given node pairs, and select those that have the highest similarity. Unfortunately, this solution is impractical. As pointed out in [10], the number of possible meta-paths for a given heterogeneous graph grows exponentially with the length of a meta-path. Even if all these meta-paths are known, selecting the most important subsets among them is an NP-complete problem [10]. There is thus a need for a better approach that avoids enumerating all possible meta-paths.

Our solutions. We develop the *Forward Stagewise Path Generation* algorithm (or FSPG), which derives meta-paths that best predict the similarity between the given node pairs. Inspired by machine learning algorithms used for feature selection [4], FSPG contains greedy strategies that generate the most relevant subset of meta-paths under a given regression model. Another advantage of FSPG is that it supports existing meta-path-based similarity measures – PC and PCRW. This is because FSPG adopts our proposed metric, called *Biased Path Constrained Random Walk*, or *BPCRW*, which generalizes PC and PCRW. We will explain how parameters of BPCRW can be tuned to get the best out of these two measures. To facilitate the efficient execution of FSPG, we investigate a data structure called the GreedyTree. This data structure is easily constructed and efficiently supports the online generation of relevant meta-paths. We further explain how to extend our solution to incorporate class hierarchy information in our solutions.

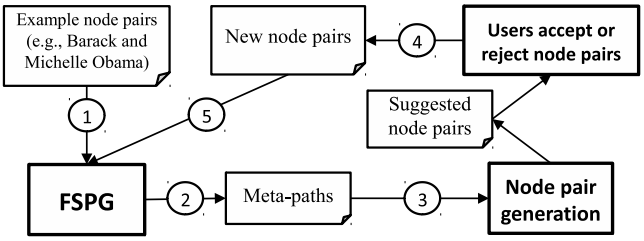

Figure 2: System framework

We have performed extensive experiments on two real knowledge bases, namely DBLP and Yago. Our results show FSPG allows for a significant improvement in the accuracy of link prediction – between 10% and 15%. Moreover, FSPG detects new meta-paths not provided by experts, and this leads to better results than only using the meta-paths provided by experts. We show that FSPG computes meta-paths in an efficient and scalable manner, and achieves an improvement in time of up to two orders of magnitude. We further demonstrate that class hierarchy information significantly increases accuracy of the similarity model, with negligible overhead. We show that FSPG discovers more association rules than AMIE. We further perform a case study, where human users provide opinions on generated meta-paths; FSPG did well in this experiment.

Figure 2 illustrates our high-level approach for discovering meta-paths. In Step 1, users provide example pairs to FSPG. This algorithm generates meta-paths (Step 2), which are then used to generate new node pairs (Step 3), e.g., via similarity search and join operations [18, 27]. From these node pairs, users select those that are indeed similar (Step 4). The accepted node pairs are then input to FSPG to iteratively refine the meta-paths generated (Step 5). An advantage of this framework is flexibility: users can review and change the meta-paths generated in Step 2; they can also provide new example pairs later after the meta-paths are generated. In this paper, we focus on the design and implementation of FSPG (Step 2).

The rest of this paper is organized as follows. We introduce the formal settings and similarity measures in Section 2. We present FSPG and GreedyTree in Section 3, and evaluate them experimentally in Section 4. We review the related work in Section 5, and conclude in Section 6.

2. META-PATH BASED SIMILARITY

We now discuss the formal model of HINs and our problem settings. We then review existing meta-path-based similarity functions, and present a method to generalize them.

2.1 Problem Definition

We model the heterogeneous information network as in [20]:

DEFINITION 1 (**HETEROGENEOUS INFORMATION NETWORK**). *A Heterogeneous Information Network is a directed graph* $G =$

(V, E, Φ, Ψ)*, where* V *is the set of nodes (or entities) of the graph;* $E \subseteq V \times V$ *is the set of edges connecting the nodes in* V*; and* Φ *and* Ψ *are functions for labeling nodes and edges. We have* $\Phi : V \to 2^A$*, where* A *is the set of node classes, and* $\Psi : E \to 2^R$*, where* R *is the set of edge types.*

Many graph datasets can be modeled as HIN such as: ontologies (Yago [17], DBPedia [2]), and bibliographic networks (DBLP [11]). Here, we mainly work with knowledge bases (KB) as an instance of HINs.

EXAMPLE 1. *Let us revisit the HIN in Figure 1, where a triple of the form:* $(Barack\ Obama, \mathsf{hasChild}, Natasha\ Obama)$ *can be represented as an edge having type:* $\Psi(Barack\ Obama, Natasha\ Obama) = \{\mathsf{hasChild}\}$*. Each entity can be mapped to multiple types:* $\Phi(Barack\ Obama) = \{\mathsf{USPresident}, \mathsf{Writer}, \mathsf{Person}\}$.

Moreover, in most HINs on the Web – Yago and DBpedia being prime examples – the classes of the entities are organized in a *hierarchical* manner. For instance, USPresident is a subclass of Leader, and Leader is a subclass of Person. All the classes have a single root, e.g., Object. They are usually organized in KBs as nodes in the graph, linked with edges of type subclassOf. Figure 1 shows an example of hierarchical organization of node classes. This hierarchical organization can affect how one chooses the classes of nodes in a KB, as we shall see later.

Given a path in G between two entities, there is a corresponding *meta-path* that can be derived:

DEFINITION 2 (**META-PATH** [20]). *Given a HIN* $G = (V, E, \Phi, \Psi)$*, a meta-path* $\Pi^{1\ldots n}$ *is a sequence of node class sets* $\mathsf{C}_1, \ldots, \mathsf{C}_n$*, connected by link types* $\mathsf{e}_1, \ldots, \mathsf{e}_{n-1}$*:* $\Pi^{1\ldots n} = \mathsf{C}_1 \xrightarrow{\mathsf{e}_1} \ldots \mathsf{C}_i \xrightarrow{\mathsf{e}_i} \ldots \mathsf{C}_n$.

We say that a path $P = v_1 \xrightarrow{\mathsf{e}_1} \ldots v_i \xrightarrow{\mathsf{e}_i} \ldots v_n$ satisfies a meta-path $\Pi = \mathsf{C}_1 \xrightarrow{\mathsf{e}_1} \ldots \mathsf{C}_i \xrightarrow{\mathsf{e}_i} \ldots \mathsf{C}_n$ if $\forall i \in \{1, \ldots, n\}$, $\mathsf{C}_i \cap \Phi(v_i) \neq \emptyset$, and $\forall i \in \{1, \ldots, n-1\}$, $\{v_i, v_{i+1}\} \in E$ and $\mathsf{e}_i \in \Psi(v_i, v_{i+1})$. In other words, each entity on the path has a class that exists in the corresponding class set on the meta-path, and each edge on the path has a corresponding type.

EXAMPLE 2. *In Figure 1,* $\mathsf{USPresident} \xrightarrow{\mathsf{hasChild}} \mathsf{Person} \xrightarrow{\mathsf{hasChild}^{-1}} \mathsf{USFirstLady}$ *is satisfied by the path Barack Obama* $\xrightarrow{\mathsf{hasChild}}$ *Natasha Obama* $\xrightarrow{\mathsf{hasChild}^{-1}}$ *MichelleObama. Another meta-path that is satisfied by a path between these two nodes is* $\mathsf{USPresident} \xrightarrow{\mathsf{memberOf}} \mathsf{USPoliticalParty} \xrightarrow{\mathsf{memberOf}^{-1}} \mathsf{USFirstLady}$*, since Barack and Michelle Obama are both members of the Democratic Party.*

Two entities can have paths that satisfy different meta-paths between them. The challenge then is finding them and including them in a model that can describe accurately the relationship between the nodes. Moreover, we wish to do this when *multiple* pairs are given:

PROBLEM 1 (**RELEVANT META-PATHS**). *Given a HIN* $G = (V, E, \Phi, \Psi)$ *and a set of* e *example node pairs* $\Lambda = \{(s_i, t_i) \mid 1 \leqslant i \leqslant e\}$*, find a set of* p *meta-paths* $\Theta = \{\Pi_i \mid 1 \leqslant i \leqslant p\}$ *which accurately predicts* Λ *for a given similarity scoring function* $\sigma(s, t \mid \Theta)$.

For instance, in Yago, the user can provide several pairs of entities of types Person and Country. A citizenOf link between entities of these types shows that a person is a citizen of a country. When this link is missing, the examples might implicitly suggest a relationship of residency in a country, as explained by a combination of meta-paths $\mathsf{Person} \xrightarrow{\mathsf{livedIn}} \mathsf{Country}$ and $\mathsf{Person} \xrightarrow{\mathsf{worksAt}} \mathsf{Institution} \xrightarrow{\mathsf{locatedIn}} \mathsf{Country}$.

The challenge is then: (i) to find *all* and *only* the relevant meta-paths, and (ii) to train a model able to predict the examples pairs. Each path between the entities in the examples pairs that is satisfied by a meta-path in the model contributes to the *similarity* between the example entities. Next, we study meta-path based similarity measures.

2.2 Similarity Function σ

We now explain how to use a combination of meta-paths to derive a similarity measure between two entities in a HIN. The *similarity score* between two entities s and t is a real number, computed by an aggregate function F of the similarity scores for each meta-path of a set Θ: $\sigma(s, t \mid \Theta) = \mathrm{F}(\{\sigma(s, t \mid \Pi_i) \mid 1 \leqslant i \leqslant p\})$, where $\sigma(s, t \mid \Pi_i)$ is a similarity score measuring how "strong" the connection between two entities s and t *given meta-path* Π_i, $\Theta = \{\Pi_1, \ldots, \Pi_p\}$, and F is an aggregate function over the meta-path scores.

Let us start by explaining how to compute $\sigma(s, t \mid \Pi)$, and discuss the function F later. We first review the state of the art of meta-path–based similarity definitions on HINs, and then introduce our general formulation of these similarity measures.

Path Count (PC) [19]. The simplest form of similarity between two nodes is the count of the different paths which satisfy a given meta-path Π. The intuition behind this measure is that the larger the number of paths satisfy a given configuration is, the more similar the entities are.

Path-Constrained Random Walk (PCRW) [10]. For a meta-path $\Pi = \mathsf{C}_1 \xrightarrow{\mathsf{e}_1} \ldots \mathsf{C}_i \xrightarrow{\mathsf{e}_i} \ldots \mathsf{C}_n$, the similarity score between two entities s and t is defined by the random walk starting at s and following only paths satisfying Π. It is defined recursively as follows, with a slight modification to allow for choosing node classes:

$$\sigma(v_i, t \mid \Pi^{i\ldots n}) = \frac{1}{|\rho_{\mathsf{e}_i}(v_i, \mathsf{C}_{i+1})|} \sum_{x \in \rho_{\mathsf{e}_i}(v_i, \mathsf{C}_{i+1})} \sigma(x, t \mid \Pi^{i+1\ldots n}),$$

$$\sigma(t, t \mid \Pi^{n\ldots n}) = 1,$$

where $\rho_{\mathsf{e}_i}(v_i, \mathsf{C}_{i+1})$ is the set of nodes x which have classes in C_{i+1} and are linked by an edge of type e_i from v_i, and $\Pi^{i\ldots n}$ is the meta-path Π truncated to the sub-path between C_i and C_n. The evaluation starts at $\sigma(s, t \mid \Pi) = \sigma(s, t \mid \Pi^{1\ldots n})$. Essentially, PCRW indicates the probability that a walker constrained on a particular meta-path reaches the target node.

PC emphasizes the absolute number of paths satisfying a meta-path while PCRW weighs the paths based on the neighbourhoods of nodes along them. Both measures can be generalized in a single measure, the *Biased Path Constrained Random Walk (BPCRW)*, defined in a manner similar to PCRW as follows:

$$\sigma(v_i, t \mid \Pi^{i\ldots n}) = \frac{1}{|\rho_{\mathsf{e}_i}(u_i, \mathsf{C}_{i+1})|^{\alpha}} \sum_{x \in \rho_{\mathsf{e}_i}(v_i, \mathsf{C}_{i+1})} \sigma(x, t | \Pi^{i+1\ldots n}), \tag{2.1}$$

$$\sigma(t, t \mid \Pi^{n\ldots n}) = 1.$$

When $\alpha = 0$, BPCRW is the same as PC; when $\alpha = 1$, the same as PCRW. If $\alpha \in (0, 1)$, α balances the number of meta-paths to be counted and the contribution of neighbors. Particularly, when α is large, we care more about the number of similar neighbors for a node; when α is small, we place more emphasis on the number of paths between two nodes. We will evaluate the effect of α on BPCRW experimentally in Section 4.

Remarks. Our solutions do not consider two recently proposed metrics, *PathSim* and *HeteSim*. The *PathSim* measure [20] obtains similar nodes (or the so-called "peer objects") for single symmetric

meta-paths. It is not clear how *PathSim* can handle non-symmetric meta-paths, which are common in KBs. The *HeteSim* metric [15,16] measures the relevance of a single meta-path. Since it does not have the form of BPCRW (Equation 2.1), it is not supported by our FSPG solution. We experimentally compare BPCRW and *HeteSim* for a given single meta-path in Section 4. An interesting future work is to extend FSPG to support *HeteSim*.

2.3 Similarity Aggregate Function F

Following the previous work on meta-path–based link prediction [10, 19], we use a linear regression model for F, which trains a linear model of the form:

$$\sigma(s,t \mid \Theta) = \sum_{1 \leqslant j \leqslant p} w_j \cdot \sigma\left(s,t \mid \Pi_j\right) + w_0, \tag{2.2}$$

where $w_0, \ldots, w_p$ are real-valued coefficients. The function F forms the basis of our meta-path selection algorithm.

3. GENERATING META-PATHS

We now study efficient algorithms for solving Problem 1. As a matter of fact, an optimal solution for this problem is likely to be intractable [24]. Even if the paths are known in advance, selecting only the most relevant paths is difficult: it has been proved in [1, 24] that the problem is NP-hard, and it cannot be approximated within a constant factor unless P=NP. Therefore, we study efficient algorithms that do well in the average case. Since relevant paths may not be known initially, we propose a greedy framework that starts from a regression model with no meta-paths. We then insert meta-paths to the model, until a desired accuracy is attained. This method is inspired by the design of *forward selection algorithms* [9] in machine learning. As we will explain, forward selection enables meta-paths to be generated as needed.

Algorithm Framework. Our solution consists of two phases. The first phase (Section 3.1) generates meta-paths with only edge types. To support this step efficiently, we develop an appropriate data structure (Section 3.2). In the second phase (Section 3.3), we augment the meta-paths generated in Phase 1 with node class information, by considering the node class hierarchy. Let us now study these in detail.

3.1 Phase 1: Link-Only Path Generation

This phase generates meta-paths with only edge types, by using *Forward Stagewise Path Generation* (or FSPG). We greedily select the next most relevant meta-path, and add it as a feature to the regression model, until the model can fit the example pairs. Specifically, we build a regression model for the input example pairs; it uses meta-paths (with only link types) as features and assigns a weight for each feature. We train this model based on a modified version of the Least-Angle Regression [4] (LARS). This method tests the addition of each feature against the model constructed, repeatedly adding the feature that improves the model the most, until no more features can be added. While LARS needs to know the whole feature space at the beginning of regression which means you need to enumerate all possible meta paths, we propose a heuristic to avoid this.

To support LARS and to avoid generating irrelevant meta-paths, we propose a greedy method that iteratively returns the meta-path with the largest correlation with the partially constructed model. We initialize the model with the first meta-path found by our greedy method. Then, at each stage, we continue testing the next meta-path, by computing its correlation to the currently constructed model. As in LARS, the criterion for choosing a feature is correlation – each time we choose the feature that has the largest correlation with the expected output at this step, so that we can approach the expected output fastest. At each stage, we insert the meta-path that has the largest correlation value with the current residual (explained below), based on a standard cosine function:

$$\cos(\mathbf{m},\mathbf{r}) = \frac{\mathbf{m}\cdot\mathbf{r}}{\|\mathbf{m}\| \times \|\mathbf{r}\|} \tag{3.1}$$

where $\mathbf{m}$ is the *resulting vector* of a certain meta-path Π on the training examples (each entry of $\mathbf{m}$ is a BPCRW score $\sigma(s,t \mid \Pi)$ for input pair (s,t)), and $\mathbf{r}$ is the vector of *residual values*, denoting the difference between the values of the ground truth examples (input pairs) and the model results $\mathbf{m}$. Initially, $\mathbf{r}$ is equal to the ground truth $\{1,\ldots,1,-1,\ldots,-1\}$ of length $|\Lambda|$ with values 1 for positive example pairs, and -1 for negative example pairs.

We now explain the process of generating training data. We require the user to provide *positive* examples only. One does not need to state *negative* examples, which can be difficult. However, if only positive examples are used, the resulting meta-paths may characterize how the node pairs are connected by the *largest* number of meta-paths, instead of how they are *uniquely* connected. Consider two positive examples: (*B. Obama*,*M. Obama*), and (*G. W. Bush*,*L. Bush*), which can be predicted by the three meta-paths: 1) they have the same children, 2) they belong to the same political party, and 3) they are both US citizens. Although all these meta-paths describe the relationship of the two node pairs, meta-path 1) is more rare and important than meta-path 3). To reflect this, our algorithm generates negative node pairs that are pertinent to the positive examples.

One simple way to generate negative examples is to randomly select nodes pairs from the KB. Unfortunately, these examples may be totally irrelevant. For instance, a random pair (*J. Cameron, Avatar*) is unrelated to the positive examples above. In our solution, the number of negative examples is the same as that of the positive ones. We randomly extract entities that share the same *Lowest Common Ancestor* (or *LCA*) label in the class hierarchy as the entities present in the positive pairs. In Figure 1, the LCA of the entities used in our examples, namely *B. Obama* and *G. W. Bush*, is USPresident. We then randomly select nodes that have the LCA USPresident (e.g., *B. Clinton* and *R. Reagan* in Figure 3). Similarly, since the LCA of *M. Obama* and *L. Bush* is USFirstLady (Figure 1), two candidates for negative examples are *B. Ford* and *B. Bush* (Figure 3). These selected nodes are then combined randomly to form negative examples: *(B. Clinton, B. Ford)* and *(R. Reagan, B. Bush)*. Although this process might give us some false negative pairs, this chance is shown to be low in our experiments.

Algorithm 1 details the procedure of obtaining the set of meta-paths with their weights, for an input vector of positive example pairs Λ. We denote all selected meta-paths and their scores in a feature matrix: $X = (\mathbf{m_0},\ldots,\mathbf{m_k})$. In this matrix, each column $\mathbf{m_k}$ is the BPCRW scores of some meta path for all examples in Λ, and k is the number of meta paths generated. For each meta-path feature, we return its weight in another vector $\mathbf{w}$. After initializing the residual and weight vectors in Step 1, we obtain the first meta-path by invoking GreedyTree in Step 2. This method, which we will explain, returns the meta-path with the largest correlation value. Steps 4–11, we iterate with the meta-path with the next largest correlation.

Note that LARS iterates until every feature has been processed. In our case, the number of features (meta-paths) is unknown, and it is difficult to generate them efficiently. Hence, we modify LARS, and incrementally build the regression model by using a new stopping condition: we iterate until the residual $\mathbf{r}$ is negligible (i.e., $|\mathbf{r}| \leqslant \varepsilon$ in Step 4). In our implementation, traversing the entire space of meta-paths just to find that no path has been found is practically

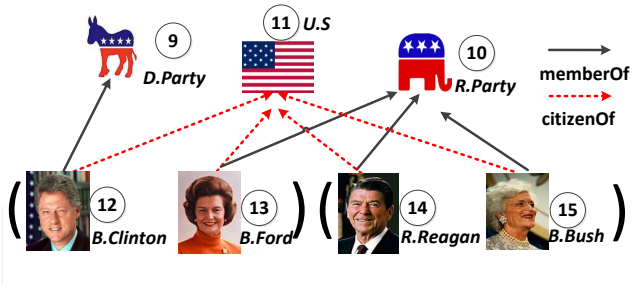

Figure 3: Selected negative example pairs

impossible. So, we also establish a *maximal number of iterations* – 50 in our experiments – that forces the algorithm to stop even if convergence has not been achieved. This is enough, since most models contain a lot of small meta-paths; having more than 50 features start to add noise to the model. In our experiments, this threshold is rarely reached; FSPG terminates within 20 iterations.

In each iteration, we obtain the next meta-path (Step 6) having the largest correlation with $\mathbf{r}$, and put it to matrix X (Step 7). Next, we compute the correlation of newly added meta-path feature with the current residual (Steps 8), obtaining direction vector $\mathbf{u}$ and stepsize γ:

$$\mathbf{u} = X(X^TX)^{-1}(\mathbf{1}^T(X^TX)^{-1}\mathbf{1})^{1/2}\mathbf{1} \tag{3.2}$$
$$\gamma = \min_{0\leqslant j<k}{}^{+}\left\{\frac{corr-\cos(\mathbf{m_j},\mathbf{r})}{(\mathbf{1}^T(X^TX\mathbf{1})^{1/2})-\mathbf{m_j}^T\mathbf{u}}, \frac{corr+\cos(\mathbf{m_j},\mathbf{r})}{(\mathbf{1}^T(X^TX\mathbf{1})^{1/2}+\mathbf{m_j}^T\mathbf{u}}\right\}$$

where $\mathbf{1}$ is a vector of 1's of length k and $\min^+$ is the minimal positive value. While these equations appear in LARS [4], we made two changes. First, we use the existing feature matrix X to heuristically represent the entire feature matrix. Second, when computing γ, we also only consider the existing meta-path features vectors, or $(\mathbf{m_j})_{0\leqslant j<k}$ in the kth iteration.

Algorithm 1: FSPG(G,Λ)

Input: network G, example pairs Λ
Output: meta-paths $\Pi_{0,\ldots,k}$, path weight vector $\mathbf{w}$

1 $\mathbf{r}\leftarrow\{1,\ldots,-1\}$; $w\leftarrow 0$; $k\leftarrow 0$;
2 $\Pi_0, \mathbf{m_0}\leftarrow$ GreedyTree$(G,\Lambda,\mathbf{r})$;
3 $k\leftarrow 0$;
4 **while** $|\mathbf{r}|>\varepsilon$ **do**
5 $\quad k\leftarrow k+1$;
6 $\quad \Pi_k, \mathbf{m_k}\leftarrow$ GreedyTree$(G,\Lambda,\mathbf{r})$;
7 $\quad X\leftarrow X\cup\mathbf{m_k}$;
8 $\quad corr\leftarrow\cos(\mathbf{m_k},\mathbf{r})$;
9 $\quad$ compute $\mathbf{u}$, γ using Eq. (3.2) ;
10 $\quad \mathbf{r}\leftarrow\mathbf{r}-X\mathbf{u}\gamma$;
11 $\quad \mathbf{w}\leftarrow\mathbf{w}+\mathbf{u}\gamma$;
12 **return** $\Pi_{0,\ldots,k}, \mathbf{w}$

3.2 The GreedyTree Structure

Next, we study how to find the meta-path $\mathbf{m_k}$ with the largest correlation, by applying the GreedyTree algorithm. GreedyTree has three major advantages. First, it can reduce the search space by applying search heuristics. Second, when starting from multiple example nodes, it minimizes the redundancy in computing the common path sequences that may appear, by using the tree structure to store such paths. Third, the tree structure can be re-used for multiple expansions in the run of the algorithm.

Let us illustrate the algorithm by using the previous example. The positive example pairs are $\{(1,2),(3,4)\}$ in Figure 1 and negative example pairs are $\{(12,13),(14,15)\}$ in Figure 3. The algorithm

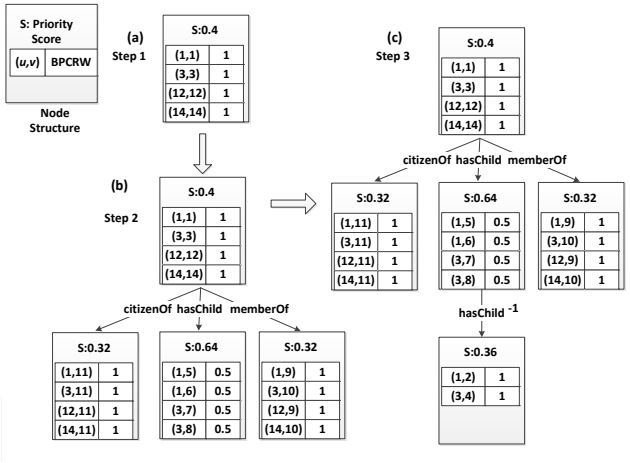

Figure 4: GreedyTree

works by expanding a tree structure, where each tree edge is annotated with an edge type, and each tree node stores a list of node pairs with their BPCRW values and a *priority score* S. Here, the list stores tuples in the form $\langle(u,v),\sigma(u,v\mid\Pi)\rangle$ where (u,v) represents a path in the graph by its starting and current graph nodes, respectively; Π is the edge-only meta-path starting from u to v. The edges of the tree are edge types in the graph. $\sigma(u,v\mid\Pi)$ is the BPCRW score for this meta-path. We compute the value of S as a heuristic on the correlation value, as follows:

$$S=\frac{\sum_{u+}\sigma(u,v\mid\Pi)\cdot\mathbf{r}(u,*)}{\sqrt{\sum_u\sigma(u,v\mid\Pi)^2}\times|\mathbf{r}|}\cdot\beta^L, \tag{3.3}$$

where $\mathbf{r}$ is the current residual vector, $u+$ signifies the starting node u in (u,v) which belongs to positive examples (for instance, $u+\in\{1,3\}$ in Figure 4), and $\mathbf{r}(u,*)$ is the largest value in the residual vector for example pairs starting from u (initially $\mathbf{r}(u,*)=1$). β is a decay factor ranging from 0 to 1, and L is the length of current meta path. Unlike the homogeneous graph, there is no guarantee that the current node is on the correct route to reach the target nodes of the example pairs. We add this decay factor β, to avoid the meta path expanding to an infinite length. As shown in Figure 4(a) with $\beta=0.8$, initially the root of the GreedyTree contains all the starting nodes in the input example pairs, and has $S=0.4$. After one step of moving in the graph in Figure 4, it passes through three types of edges – citizenOf, hasChild, and memberOf – and generates three new tree nodes in the GreedyTree (Figure 4(b)). It also computes the priority score and all the BPCRW values. Then, it greedily selects the tree node with the largest priority to expand – in this example, we continue to expand the node through hasChild^{-1} link, since it has the largest priority score $S=0.64$.

After this expansion, we see in Figure 4(c) that $(1,2)$ and $(3,4)$ are already an input pair. Thus, we find a link-only meta-path Π: $?\xrightarrow{\text{hasChild}}?\xrightarrow{\text{hasChild}^{-1}}?$ and we compute the actual value of the correlation, as well as the resulting vector m on example pairs. If this value of S is larger than the priority scores of the other two tree leaf nodes (the upper bounds of their own correlations, i.e., 0.32), then Π is the best meta-path currently found and is then returned along with its $\mathbf{m}$ vector. If other priority scores are larger, then we continue to expand the tree node with the largest score, and repeat this procedure until this condition is satisfied or the tree cannot be expanded.

We preserve the expanded tree structure for subsequent iterations of Algorithm 1. Before resuming tree expansions, we need to first

update the priority scores of the leaf nodes, since the residual vector $\mathbf{r}$ has been changed by the addition of a new meta-path feature. By a linear scan of the node pair list in a leaf node, we can efficiently compute the summation in Equation (3.3), and update the value of S. Then, we continue to expand the tree starting from the node with highest S.

Algorithm 2: `GreedyTree`$(G, \Lambda, \mathbf{r})$

Input: KB G, example pairs Λ, residual vector $\mathbf{r}$
Output: meta-path Π having largest correlation with $\mathbf{r}$, and its result vector $\mathbf{m}$
Data: pattern tree T

1 update priority scores of leaf nodes in T with $\mathbf{r}$;
2 **while** *T can be expanded* **do**
3 $M \leftarrow$ node with largest score S in T;
4 $\mathbf{m} \leftarrow \{0, \ldots, 0\}$;
5 **foreach** *tuple p in M* **do**
6 **if** $p.(u,v) \in \Lambda$ **then**
7 $\mathbf{m}(u,v) \leftarrow \sigma(u,v \mid \Pi)$;
8 **if** $\mathbf{m}$ *has non-zero entry* **then**
9 $M.S \leftarrow \cos(\mathbf{m}, \mathbf{r})$ (Eq. (3.1));
10 **if** $M.S \geq \max_{T.leaf} S$ **then**
11 $\Pi \leftarrow$ the meta-path from root to M;
12 break;
13 **else**
14 **foreach** *tuple p in M* **do**
15 **foreach** *out-neighbor w of $p.v$ on graph G* **do**
16 $e \leftarrow$ link type from $p.v$ to w;
17 **if** *M has no child N linked by e* **then**
18 create new child tree node N of M ;
19 $\Pi \leftarrow$ the meta-path from root to N;
20 insert tuple $\langle (p.u, w), \sigma(p.u, w \mid \Pi) \rangle$ to N;
21 update $N.S$ according to Eq. (3.3);
22 **return** $\Pi, \mathbf{m}$

We present the detailed steps of `GreedyTree` in Algorithm 2. First, we check and update the priority scores of leaf nodes changed by new residual vector $\mathbf{r}$ (Step 1). Then, we expand the tree by moving to out-neighbor nodes on the graph until a meta-path can be found or the graph is completely traversed (Steps 2–21). We target the tree node with largest priority (Step 3) and examine whether its tuples are example pairs (Step 6). If so, we store BPCRW scores in $\mathbf{m}$ (Step 7), compute the actual correlation value as $M.S$ (Step 9), and determine if we have found a proper meta-path (Steps 10–12). If no example pairs are encountered (Step 13), then we extend each node pair by moving to an out-neighbor (Step 15). We insert this new pair with its BPCRW score to a child node (Steps 17–20), and update its priority score (Step 21).

3.3 Phase 2: Node Class Generation

Existing methods (e.g., [10, 20]) often assume that each KB node has only one class. In complex and real KBs, nodes can have multiple classes. For instance, Barack Obama is not only a president, but also a lawyer and writer. As introduced, these node classes are organized in a hierarchy. For example, in Yago (see Figure 1), Leader and Writer are subclasses of Person. This increases the number of possible paths for satisfying a meta-path. In the previous section, the meta-paths generated do not specify node classes. Here we present ways to assign node classes.

One option is to simply disregard the node classes. So the meta path contains only edge types generated in Phase 1, without any node class constraints. However, the link-only meta-path is generally too common in a large KB, thus introducing more false positive results. For instance, ? $\xrightarrow{\text{liveIn}}$? is a much more common meta-path than Scientist $\xrightarrow{\text{liveIn}}$ CapitalCity. Since link-only meta-paths are less specific to input pairs, they will eventually impair the result quality.

A better method is to choose the classes which are the Lowest Common Ancestor (LCA) in the type hierarchy. For instance, in Figure 1(a), the LCA of USPresident and Writer is Person. Another example is that the LCA of Harvard and Yale is IvyLeague. This way, if given some pairs of persons who graduated from Harvard and Yale, we can generate meta-paths showing they both graduated from IvyLeague. Thus, for every KB node satisfying a certain meta-path node, we generate LCAs for all its possible classes, and use them in the model. The LCAs of KB nodes can simply be recorded in each node of the GreedyTree. Thus, while we expand the GreedyTree, we just need to find the LCA of the current node and its parent node. This approach has the advantage of preserving the same weights as the ones trained in Algorithm 1, at the cost of only a bottom-up traversal in the class hierarchy.

Finally, we can combine the score of the classes on the hierarchy in a tf-idf-like manner, for a given label φ: $score(\varphi) = \frac{tf(\varphi)}{\log of(\varphi)}$, where $tf(\varphi)$ is the count, or frequency, of the label φ in the positive examples, and $of(\varphi)$ the overall count of φ in the entire KB.

For instance, in the first node of the GreedyTree in Figure 4, $tf(\text{USPresident})$ is 2 since it contains nodes 1 and 3 – as the labels of nodes can be easily obtained from the type hierarchy in Figure 1(a). On the other hand, $of(\text{USPresident})$ is 42, since there are 42 nodes which have label USPresident in the entire KB, and then $score(\text{USPresident}) = 1.23$, which is much higher than other labels for these example nodes. In terms of performance, *of* can be easily pre-computed by counting all the labels in the HIN, as a pre-processing step.

We show in the next section that this selection mechanism improves considerably the accuracy of the similarity model.

4. EXPERIMENTS

Datasets and setup. In this section, we perform experiments on two representative datasets for online HIN: DBLP and Yago.

DBLP [11] is a bibliographic information network which is frequently used in the study of heterogeneous networks. Our dataset [20, 25]is a subset containing scientific papers in four areas: databases, data mining, artificial intelligence, and information retrieval. The dataset has four classes of nodes: Paper, Author, Topic, and Venue. It also has four edge types: authorOf, publishedIn, containsTopic, and cites. It contains 14,376 papers, 14,475 authors, 8,920 topics, and 20 venues. There are 170,794 links in total.

Yago is a large-scale knowledge base derived from Wikipedia, WordNet, and GeoNames [17]. In our experiments, we use the "CORE Facts" part of this dataset, which contains 4 million facts (network links) of 125 types, made from 2.1 million entities. These entities have 365,000 node classes, organized in a hierarchy tree. A fact is a triple of the form: (Entity, relationship, Entity), e.g., (*Barack Obama*, hasChild, *Malia Obama*).

We validate our algorithm's efficiency and effectiveness mainly by performing link prediction tasks (additional experiments on rule mining and a user study are also presented at the end of this section). We stress that our models are not only usable for link prediction, but for a variety of other tasks, such as search and similarity joins. We chose link prediction for the evaluation because it provides for a measurable and objective way to evaluate the similarity models and algorithms.

For a certain type of link in Yago, for instance citizenOf, we remove all such links and try to predict them with the model that our `FSPG` algorithm learns. We randomly select a number of pairs of objects as training data, and validate the model using a test set of an equal number of pairs. In our experiments, we set the value of ε in Algorithm 1 to 0.01. We have found that even low values of

Table 1: 5 most relevant meta-paths for Yago - citizenOf

meta-path	w
Person $\xrightarrow{\text{bornIn}}$ City $\xrightarrow{\text{locatedIn}}$ Country	5.477
Person $\xrightarrow{\text{livesIn}}$ Country	0.361
Person $\xrightarrow{\text{graduateOf}}$ University $\xrightarrow{\text{locatedIn}}$ Country	0.023
Person $\xrightarrow{\text{diedIn}}$ City $\xrightarrow{\text{locatedIn}}$ Country	0.245
Person $\xrightarrow{\text{bornIn}}$ City $\xrightarrow{\text{happenedIn}^{-1}}$ Event $\xrightarrow{\text{happenedIn}}$ Country	0.198

this parameter allow for a reasonable number of features selected, usually lower than 20. If ε is set to a higher value, the models will be smaller but also less accurate. We compared FSPG with PCRW to models which generate paths of finite length in $\{1,2,3,4\}$, as used in [10]. The PCRW models use the logistic regression model to combine these meta-paths, in order to be coherent with previous work in the area and to allow fair comparisons. For node class selection (Section 3.3), we use the LCA of entities' classes. Finally, unless otherwise specified, the similarity function was BPCRW with a value for α of 0.5. We found this value approximates both PC and PCRW well, and also that it helps to keep the vector of meta-path similarity values normalized. We use β as 0.6 in GreeyTree to avoid the meta path expanding infinitely.

Effectiveness. We present in Figure 5 the results for link prediction for three types of links: citizenOf and advisorOf for Yago, and authorOf for DBLP. For each of these links, we generated 100 training and 100 test pairs, as described above. Each figure shows the Receiver Operating Characteristic (ROC) curve, where the larger area signifies a larger accuracy in prediction.

The figure shows that fixed-length PCRW suffers from several issues. When the maximum length is too small (1 or 2), meta-paths cannot connect example pairs, and as such the model is not better than a random guess and the model will have low recall. When the maximum length is too big, the model introduces too many meta-paths. For length 3, there are 135 meta-paths, and over 2,000 for length 4. In DBLP, length 4 (measured by link hop) meta-paths – equivalent to the method presented in [20] – for authorship prediction work poorly when training pairs are randomly chosen, due to the low number of paths connecting them.

FSPG is clearly better in predicting the links, and it generates only a limited number of meta-paths. For instance, the citizenOf link in Yago has a model of only 15 meta-paths. Moreover, these meta-paths are highly relevant and serve as good explanation of the similarity links. Table 1 shows the most relevant 5 meta-paths for the citizenOf similarity model. Unsurprisingly, the meta-path illustrating the fact that a person has been born in a city of a country is the best predictor of citizenship, but other, longer, paths are also highly relevant. Compared with PCRW with maximum length 2, it has higher recall because it also detected longer important meta-paths, for instance, Person $\xrightarrow{\text{bornIn}}$ City $\xrightarrow{\text{happenedIn}^{-1}}$ Event $\xrightarrow{\text{happenedIn}}$ Country. We found that in Yago some facts are missing. For instance, a fact denoted by the direct link Paris $\xrightarrow{\text{locatedIn}}$ France is missing. [2] In this case, our algorithm can be useful to predict it via longer paths, such as City $\xrightarrow{\text{happenedIn}^{-1}}$ Event $\xrightarrow{\text{happenedIn}}$ Country.

Whereas considering direct links is widely done when querying ontologies, multi-hop meta-paths can in some cases improve query result accuracy. For instance, in citizenOf prediction (Figure 5), only one type of direct links between people and country exists, namely, liveIn. As a result, most information in predicting the citizenship is lost when limiting to direct connections, causing low recall and AUC score. For advisorOf prediction, there is even no direct link between the example pairs. This means resorting to multi-hop meta-paths is necessary.

Efficiency. Figure 6 show the running time of FSPG compared to models with fixed length, and when varying the number of example pairs given as input. It can be observed that generally, the algorithm running time increases sub-linearly in the number of example pairs. The increase is due to the PCRW random walks which need to be performed concurrently for each example pair, but the number of meta-paths in the model does not increase at the same rate.

In Yago, the algorithm performs better than models of paths longer than 2 by a factor of up to 2 orders of magnitude. However, the models of short path length have limited predictive power, despite their better running time. In comparison, our algorithm is capable of finding even longer paths, for an increase in accuracy without much sacrifice in running time.

In DBLP, the running time of FSPG is comparable to length 5 for smaller test set sizes, and comparable to length 4 for larger example set sizes. This is not unexpected, considering that the DBLP KB is very small in size – both as number of types and as graph size – and fixed length models will have reasonably sized number of features. However, even if the running time is worse, we already saw that FSPG on DBLP has generally better accuracy, making it very useful even in very small KBs.

Input set size and precision@k. We turn now to evaluating the influence of the input set size on the accuracy of the models. We compare in terms of Area Under the Curve (AUC) two methods, our FSPG algorithm and PCRW of length 2. We chose length 2 because it was consistently the best performer out of all fixed length models in Yago. The evaluation was done on the same Yago node classes, advisorOf and citizenOf. In Figure 7 and Figure 8, "a-FSPG" represents the advisorOf prediction using FSPG while "c-FSPG" represents for citizenOf prediction using FSPG. The same abbreviation rule also applies to "a-PCRW" and "c-PCRW".

Figure 7 shows the results. It can be seen that the set size does not greatly influence the accuracy of the model, which suggests that even in more realistic scenarios in which users or experts give very few examples the models will be reliable. It can also be seen that FSPG keeps its advantage over PCRW of fixed length, regardless of the input set size. Please note that, for each positive input set size, we have generated an equal number of negative example using the methodology described before.

Figure 8 shows the difference between FSPG and PCRW of path length 2, with an input set size of 100 negative and positive pairs, in the following scenario: we sort the results by their similarity and check the number of true positives at a given value of k in the sorted list. FPSG achieved considerably high precision of predicted pairs at low values of k, suggesting that this method is practically usable especially since similarity search results sets are usually low in numbers. For cases in which prediction is easy – e.g., citizenOf – PCRW becomes competitive only at high values of k. This does not happen in the case of advisorOf, as it is harder to predict.

Improving expert meta-paths. Our method can also be used to improve on expert meta-path models, and we show here how it behaves for authorship prediction in DBLP. In our experiments, we started from the meta-paths defined by experts in DBLP, as explained in [20]. Starting from the already existing model of meta-paths, we first build the corresponding GreedyTree, we compute the residual vector r and "resume" the execution of FSPG.

[2] Although other 2-hop links from Paris to France exist, they were not ranked as high as City $\xrightarrow{\text{happenedIn}^{-1}}$ Event $\xrightarrow{\text{happenedIn}}$ Country.

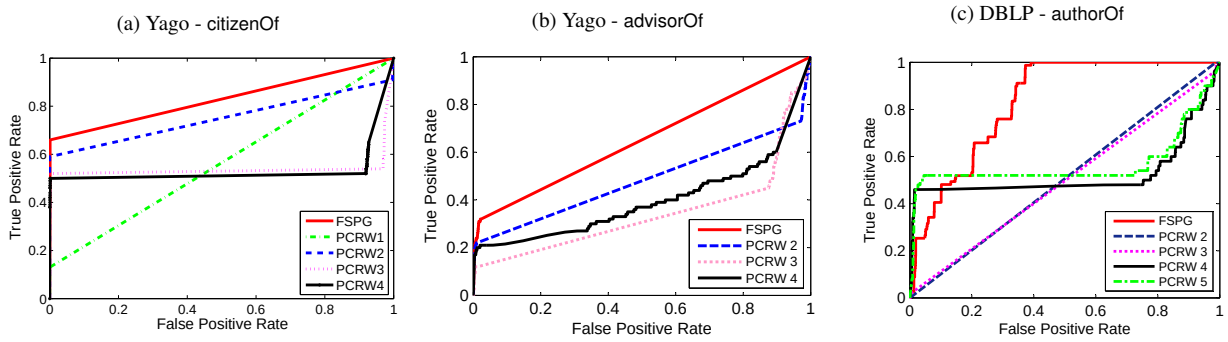

Figure 5: ROC for link prediction

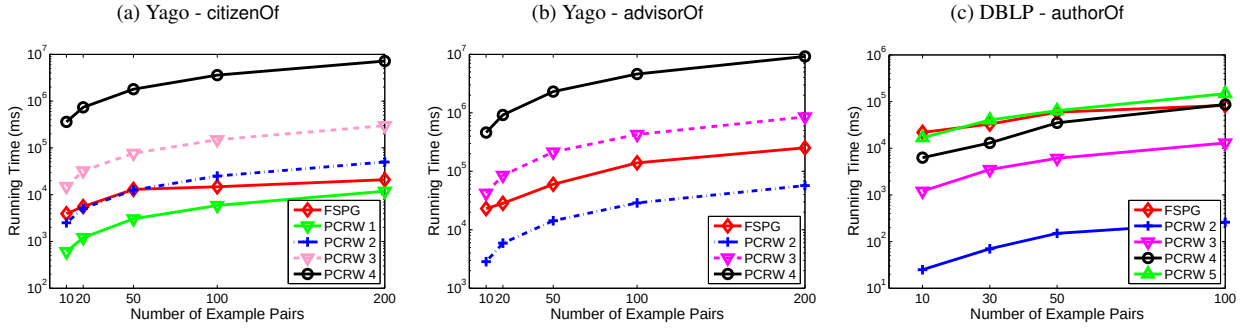

Figure 6: Running time of `FSPG`

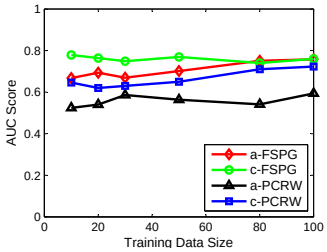

Figure 7: AUC for varying number of example pairs

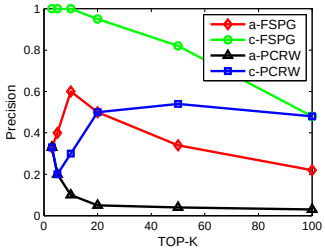

Figure 8: Precision@k

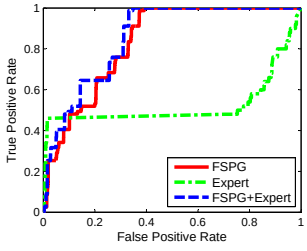

Figure 9: ROC of `Expert`

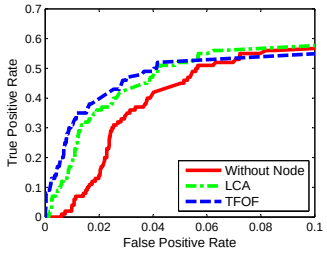

Figure 10: ROC of label selection

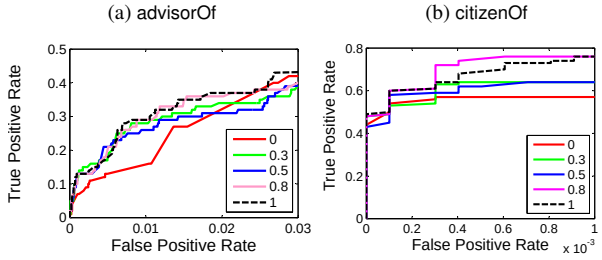

Figure 11: ROC of BPCRW for various α values

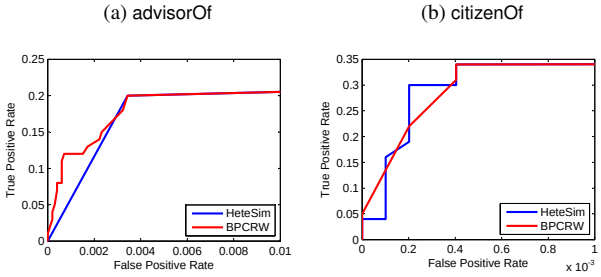

Figure 12: ROC of HeteSim on a single meta-path

We show the resulting ROC curves in Figure 9. It can be seen that starting from expert meta-paths slightly improves the accuracy of FSPG, but greatly improves the accuracy of the expert-only model. This is because the algorithm is able to generate relatively long meta-paths, which are missed by experts, as partially shown in Table 2. This result shows that the algorithm can readily be employed as an expert tool, to suggest meta-paths which might be missed by experts.

Class label selection. Figure 10 shows the accuracy of our model, depending on which class selection from Section 3.1 is chosen. Each training set was generated as before, with corresponding negative pairs. It can be seen that not restricting the class nodes on meta-paths significantly impacts the accuracy. The *tfof* method of generating class labels is better for high precision queries, but LCA is better than it for higher recall rates.

Influence of α. To illustrate the difference between the choice of α values, we illustrate in Figure 11 their power in predicting links of Yago, given a value of α for the BPCRW, for two types of links: advisorOf and citizenOf. The choice of α has a considerable impact on the link effectiveness. It also shows that – even in the same knowledge base – there is no single choice of α which always "wins" over the other similarity models, and it highly depends on the example pairs being given by users. For instance, the best value for advisorOf seems to be 0.8, while for citizenOf it is 1 because they have the largest AUC score.

Due to the low running times of FSPG the best value of α can simply be generated by a loop over reasonable values of α increments, e.g., every 0.1, and choosing the model which has the best fit with the training pairs.

HeteSim. Figure 12 compares BPCRW and *HeteSim* [15, 16] in terms of relationship prediction. As explained in Section 2.2, *HeteSim* is not supported in FSPG, and we study *HeteSim* on one single meta-path only. We select Person $\xrightarrow{\text{bornIn}}$ City $\xrightarrow{\text{locatedIn}}$ Country for citizenship prediction, since it is the most important meta-path (Table 1). For advisor prediction, we use Person $\xrightarrow{\text{graduateFrom}}$ University $\xrightarrow{\text{workAt}^{-1}}$ Scientist, since it is also the most important one as found from another experiment. We observe that BPCRW is comparable to *HeteSim*. Moreover, the prediction quality is much worse using a single meta-path than considering multiple meta-paths. In Figure 5, for citizenship prediction, using a single meta-path yields 34% recall, while using multiple meta-path generates 65% recall. This shows why it is important to use multiple meta-paths, as adopted by our approach.

AMIE. The FSPG method can also be used for mining association rules in heterogeneous graphs, since the meta-paths associated to a given node pair can be interpreted as an association rule for that pair. To evaluate this use case, we compare FSPG with AMIE, which is used to generate association rules in knowledge bases.

We generate the association rules for the citizenOf, advisorOf and ivyLeagueAlumnus links [3] , and show the ROC curve results in Figure 13. We first evaluate the association rules by performing link prediction, i.e., we use the association rules generated by AMIE and combine them through logistic regression – just as with the paths generated with FSPG. Our method shows generally better ROC curves for citizenOf and advisorOf. For citizenOf, AMIE produces 8 association rules, while FSPG can produce 3 more association rules such as Object $\xrightarrow{wasBorn}$ Object $\xrightarrow{locatedIn}$ Object $\xrightarrow{locatedIn}$ Object. For advisorOf, AMIE produces 4 association rules and FSPG adds 4 more, such as Object $\xrightarrow{worksAt}$ Object $\xrightarrow{graduatedFrom}$ Object. Table 3 shows other association rules missed by AMIE and generated by FSPG for advisorOf.

[3]This link does not exist in YAGO. We simulate it by creating a link between every pairs of graduates from Ivy League universities.

Table 3: Missed association rules in AMIE for advisorOf

meta-path
Person $\xrightarrow{\text{influence}^{-1}}$ Person
Person $\xrightarrow{\text{diedIn}}$ City $\xrightarrow{\text{diedIn}^{-1}}$ Person
Person $\xrightarrow{\text{workAt}}$ University $\xrightarrow{\text{graduatedFrom}}$ Person
Person $\xrightarrow{\text{liveIn}}$ Country $\xrightarrow{\text{isCitizenOf}^{-1}}$ Person

However, the most important advantage for FSPG lies in its ability to generate rules for the personalized case, where few pairs are given as input. We illustrate this on the link ivyLeagueAlumnus – which does not exist in YAGO. We input these pairs connected by this virtual link ivyLeagueAlumnus. AMIE will look for global rules, and thus generates general rules which fail to distinguish Ivy League alumni from the alumni of any other university. In contrast, our method achieved a ROC of 0.989, and was able to generate more specific association rules such as

Object $\xrightarrow{graduateFrom}$ IvyLeague $\xrightarrow{graduateFrom^{-1}}$ Object or

Object $\xrightarrow{isAffiliatedTo}$ Organization $\xrightarrow{isAffiliatedTo^{-1}}$ Object.

User study. Finally, we conduct a user study to determine whether meta-paths learned by FSPG are of interest to users.

We use the data from the Time Magazine's "100 Most Important People of the 20th Century" [21]. Ten volunteers, who are students and research assistants of the CS department of HKU, are asked to select pairs of similar persons. The volunteers have selected 10 pairs of people which are similar, e.g., pairs such as (Franklin D. Roosevelt, Theodore Roosevelt) and (Mao Zedong, Ho Chi Minh). Based on the input pairs, we use FSPG to generate the meta-paths. After the model is trained, we shuffle the most relevant meta-paths with randomly selected, but reasonable, meta-paths and ask the same volunteers to rank them into three categories: *Relevant*, *Somewhat Relevant*, *Not Relevant*. We assign each answer a point value of 1 for *Relevant*, 0.5 for *Somewhat Relevant* and 0 for *Not Relevant*. We then average the ranking over all question for the FSPG-returned meta-paths and the most well-ranked random meta-path for each question. The FSPG generated meta-path score average was $0.471(\sigma = 0.229)$ and for the random paths it was $0.285(\sigma = 0.100)$. Statistical significance was estimated using the one-tailed Student's t-test, yielding a p-value of 0.09. This value is satisfactory, taking into account the fact that the users were not always familiar with the historical figures.

At a qualitative level, the result also showed that our generated meta-paths rank consistently higher than the random meta paths. For instance, the most important meta path between Franklin D. Roosevelt and Theodore Roosevelt is A $\xrightarrow{isPoliticianOf}$ country $\xrightarrow{isPoliticianOf^{-1}}$ *B*, and the most important meta path between Mao Zedong and Ho Chi Minh is A $\xrightarrow{influence^{-1}}$ person $\xrightarrow{influence}$ *B*.

5. RELATED WORK

Knowledge bases. Yago [17] is a semantic knowledge base derived from Wikipedia, with more than 10 million entities (or nodes) of different types. Links among these entities represent more than 120 million facts about them. Constructed from Wikipedia and the Web in a similar way, DBpedia [2] is another widely used knowledge base. Since both entities and links have a variety of types, a simple homogeneous graph (with only one type of node

Table 2: Missed meta-paths by experts in DBLP authorship prediction

meta-path
$\text{Author} \xrightarrow{authorOf} \text{Paper} \xrightarrow{cites} \text{Paper} \xrightarrow{publishedIn} \text{Venue} \xrightarrow{publishedIn^{-1}} \text{Paper} \xrightarrow{cites} \text{Paper} \xrightarrow{authorOf^{-1}} \text{Author}$
$\text{Author} \xrightarrow{authorOf} \text{Paper} \xrightarrow{cites} \text{Paper} \xrightarrow{containsTopic} \text{Topic} \xrightarrow{containsTopic^{-1}} \text{Paper} \xrightarrow{cites} \text{Paper} \xrightarrow{authorOf^{-1}} \text{Author}$
$\text{Author} \xrightarrow{authorOf} \text{Paper} \xrightarrow{cites} \text{Paper} \xrightarrow{publishedIn} \text{Venue} \xrightarrow{publishedIn^{-1}} \text{Paper} \xrightarrow{containsTopic} \text{Topic} \xrightarrow{containsTopic^{-1}} \text{Paper} \xrightarrow{authorOf^{-1}} \text{Author}$

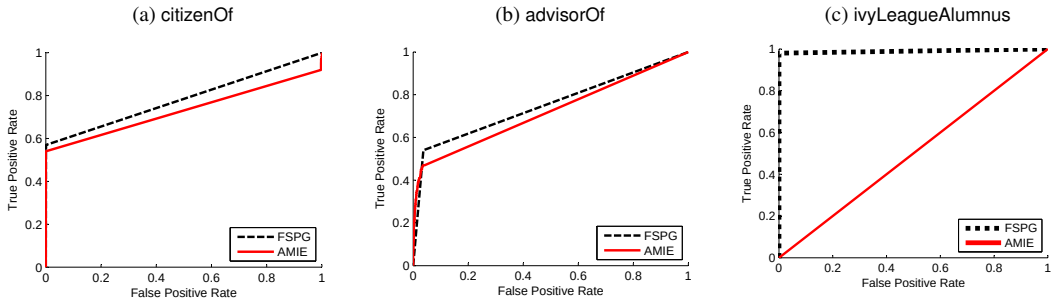

Figure 13: ROC of FSPG and AMIE

or edge) is unable to capture the rich information contained in a knowledge base.

Generating Meta-Paths. The issue of generating good meta-paths has not been satisfactorily addressed. A simple method is to first enumerate all possible meta-paths, by traversing the schema graph [20], and then use the paths as features to train a regression model that best fits the user-provided example pairs. However, the cost of generating all path patterns is prohibitive when the number of node types and edge types is large. Second, the high dimensionality of the resulting data means that, for each feature and example pair, we need to generate the similarity measures by performing the random walks described in Section 2.2. Moreover, the number of features is highly likely to introduce noise due to the curse of dimensionality [6]. On the contrary, our method only generates *relevant* meta-paths. The same authors in [20] also suggest to hire domain experts to define meta-paths. As we have explained, this may not be feasible for very large HIN. Another problem with this approach is that the paths defined in this way are *global*. In our solution, users can participate in the process of generating meta-paths by suggesting example pairs.

Another approach for generating meta-paths is proposed by [10]. Their solution enumerate all the meta-paths within a fixed length l. However, it is not clear how l should be set. More importantly, l can significantly affect the meta-paths generated: (i) if l is large, then many redundant meta-paths may be returned, leading to curse-of-dimensionality effects; and (ii) if l is small, important meta-paths with length larger than l might be missed. Our experiments have shown that the running time of the meta-path generation process grows exponentially with length l. Moreover, the accuracy can also drop with increase in l. Our solution does not require users to provide the value of l.

Link Prediction. Since the main focus of our work is to generate meta-paths, we only use link prediction to quantify our advantage compared with the existing meta-path generation methods, as demonstrated in Section 4. Compared with [14], our method predicts the relationship between different entities, rather than predicting the types of entities. [13] used the factorization of a three-way tensor to perform relational learning; it only considered simple node types and is not applicable to our experiment which has both complex node classes and edge types.

Node similarity. To measure the proximity between graph nodes, neighborhood-based metrics such as *common neighbors* and Jaccard's coefficient were proposed [12]. Other widely-used graph-theoretic measures that are based on random walks between nodes include *personalized PageRank* [3], *SimRank* [8], *hitting times* [27], and *random walk with restart* [22]. These measures do not consider the class labels of nodes and edges present in a HIN. Several similarity metrics have been designed for HIN: *path count* (PC) [19] and *path constraint random walk* (PCRW) [10]. We propose a general form of these two metrics, called the BPCRW. As discussed in Section 2.2, *PathSim* [20] and *HeteSim* [15, 16] are two other recently-proposed meta-path-based measures.

Query by example. We studied the problem of using example node pairs to generate meta-paths. These node pairs can also be used to facilitate query evaluation. In [7], Jayaram et al. examined how to find query graphs that can yield node pairs. It would be interesting to see how our meta-path–based similarity metrics can be used to enhance their solutions. In [23, 26], the problem of generating queries from user-provided example query results in a relational database was investigated.

6. CONCLUSIONS

We examined the problem of generating meta-paths from a given set of node pairs, using a general form of meta-path-based measures. We proposed the `FSPG` algorithm, and developed `GreedyTree` to facilitate its execution. Our experiments showed that `FSPG` generates important meta-paths efficiently. In the future, we will study the theoretical feasibility of meta-path selection for other non-linear similarity functions. We will also examine query algorithms for similarity search and join on large HINs, based on meta-path similarity measures.

Acknowledgments

Reynold Cheng, Silviu Maniu, Changping Meng, and Wangda Zhang were supported by RGC (Project HKU 711110) and HKU (Project 201311159095). We thank the reviewers for their comments.

7. REFERENCES

[1] E. Amaldi and V. Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theor. Comp. Sci.*, 209(1-2), 1998.

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

[3] S. Chakrabarti. Dynamic personalized PagRank in entity-relation graphs. In *WWW*, 2007.

[4] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2), 2004.

[5] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek. AMIE: Association rule mining under incomplete evidence in ontological knowledge bases. In *WWW*, 2013.

[6] G. F. Hughes. On the mean accuracy of statistical pattern recognizers. *IEEE Trans. Information Theory*, 14(1), 1968.

[7] N. Jayaram, A. Khan, C. Li, X. Yan, and R. Elmasri. Querying knowledge graphs by example entity tuples. *CoRR*, abs/1311.2100, 2013.

[8] G. Jeh and J. Widom. SimRank: a measure of structural-context similarity. In *KDD*, 2002.

[9] D. Koller. Toward optimal feature selection. In *ICML*, 1996.

[10] N. Lao and W. W. Cohen. Relational retrieval using a combination of path-constrained random walks. *Mach. Learn.*, 81(1), 2010.

[11] M. Ley. DBLP: some lessons learned. *PVLDB*, 2(2), 2009.

[12] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *J. Assoc. Inf. Sci. Technol.*, 58(7), 2007.

[13] M. Nickel, V. Tresp, and H.-P. Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, 2011.

[14] M. Nickel, V. Tresp, and H.-P. Kriegel. Factorizing YAGO: Scalable machine learning for linked data. In *WWW*, 2012.

[15] C. Shi, X. Kong, Y. Huang, P. S. Yu, and B. Wu. Hetesim: A general framework for relevance measure in heterogeneous networks. *TKDE*, 2014.

[16] C. Shi, X. Kong, P. Yu, S. Xie, and B. Wu. Relevance search in heterogeneous networks. *EDBT*, 2012.

[17] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledge. In *WWW*, New York, NY, USA, 2007.

[18] L. Sun, R. Cheng, X. Li, D. W. Cheung, and J. Han. On link-based similarity join. *PVLDB*, 4(11), 2011.

[19] Y. Sun, R. Barber, M. Gupt, C. Aggarwal, and J. Han. Co-author relationship prediction in heterogeneous bibliographic networks. In *ASONAM*, 2011.

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

[21] TIME. One Century, 100 Remarkable People. `http://content.time.com/time/specials/packages/0,28757,2020772,00.html`, 2008.

[22] H. Tong, C. Faloutsos, and J.-Y. Pan. Fast random walk with restart and its applications. *ICDM*, 2006.

[23] Q. T. Tran, C.-Y. Chan, and S. Parthasarathy. Query by output. In *SIGMOD*, 2009.

[24] K. S. Van Horn and T. R. Martinez. The minimum feature set problem. *Neural Networks*, 7(3), 1994.

[25] X. Yu, Q. Gu, M. Zhou, and J. Han. Citation prediction in heterogeneous bibliographic networks. In *SDM*, 2012.

[26] M. Zhang, H. Elmeleegy, C. M. Procopiuc, and D. Srivastava. Reverse engineering complex join queries. In *SIGMOD*, 2013.

[27] W. Zhang, R. Cheng, and B. Kao. Evaluating multi-way joins over discounted hitting time. *ICDE*, 2014.