Chapter 7
Schema-Rich Heterogeneous Network Mining

Abstract Traditional heterogeneous information network usually has simple network schema, where there are a small number of types of nodes and links and meta paths are easily enumerated. However, in many real applications, some heterogeneous information networks have a huge number of types of nodes and links, and it is hard to depict their network schema. We call this kind of networks as schema-rich heterogeneous information network. For example, knowledge graph, constructed with $< object, relation, object >$ tuples, can be considered as a schema-rich heterogeneous network, where there are usually tens of thousands of types of nodes and links. In this chapter, we introduce two data mining tasks on schema-rich heterogeneous network: link prediction and entity set expansion. Through these two tasks, we illustrate the challenges and potential solutions on mining this kind of more complex and popular heterogeneous networks.

7.1 Link Prediction in Schema-Rich Heterogeneous Network

7.1.1 Overview

Link prediction is a fundamental data mining 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 heterogeneous information network (HIN). As a unique semantic characteristic of HIN, meta path [24], 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 [4, 21, 23, 28]. For example, Sun et al. [21] proposed PathPredict to solve the problem of co-author relationship prediction, Cao et al. [4] proposed an iterative framework to predict multiple types of links collectively in HIN, and Sun et al. [23] modeled

C. Shi and P.S. Yu, *Heterogeneous Information Network Analysis and Applications*, Data Analytics, DOI 10.1007/978-3-319-56212-4_7

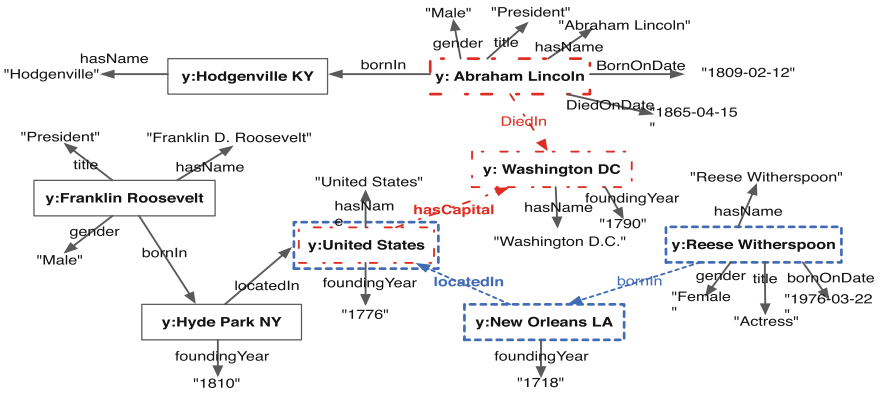


Fig. 7.1 A snapshot of the RDF structure extracted from DBpedia

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 [24]. For example, a bibliographic network with star schema is used in [21, 23, 28] 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 [19], where its resource description framework (RDF) [25] $< object, relation, object >$ naturally constructs an HIN. In such an 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 networks, it is hard to describe them with simple schema, so we call them schema-rich HIN. Figure 7.1 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, and 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. 7.1 is one extreme little part of the whole DBpedia network, and there will be huge number of meta paths that 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 is 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 is 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 chapter, 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 a 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.

7.1.2 The LiPaP Method

In this section, we firstly define the link prediction in schema-rich HIN problem and then present a novel link prediction method named LiPaP. This method includes two steps: Firstly, we design an algorithm called 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.

7.1.2.1 Problem Definition

Heterogeneous information network [10] is a kind of information network that includes different types of nodes and links. Traditional HIN usually has a simple network schema, such as bipartite [29] and star schema [17]. However, in some complex HINs, there are so many node types or link types that are 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 $\psi = \{(u_i, v_i)|1 \leq i \leq r\}$.

7.1.2.2 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

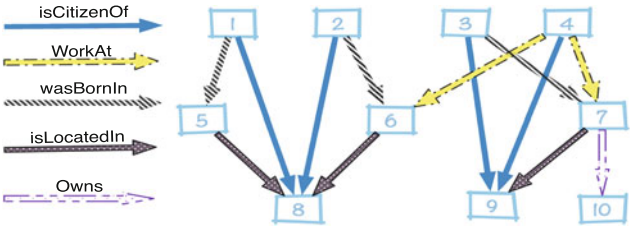


Fig. 7.2 Subgraph example of schema-rich HIN

meta paths smartly in schema-rich HIN. We would illustrate AMPG through a toy example in Fig. 7.2, 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 which to be found would not only connect more training pairs, but also show much closer relationship to present implicit features of the training set. For example, $\xrightarrow{isCitizenOf}$ is the meta path initially found by our method in Fig. 7.3, 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 pair 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) [11], which is to calculate the relevance between the given entity pairs in the meta paths. A meta path with the higher the similarity score S is more likely 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 shown in Fig. 7.3. 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:

$$S = \sum_{s} \frac{1}{T} \sum_{t} [\sigma(s, t|\prod) \bullet r(s)], \tag{7.1}$$

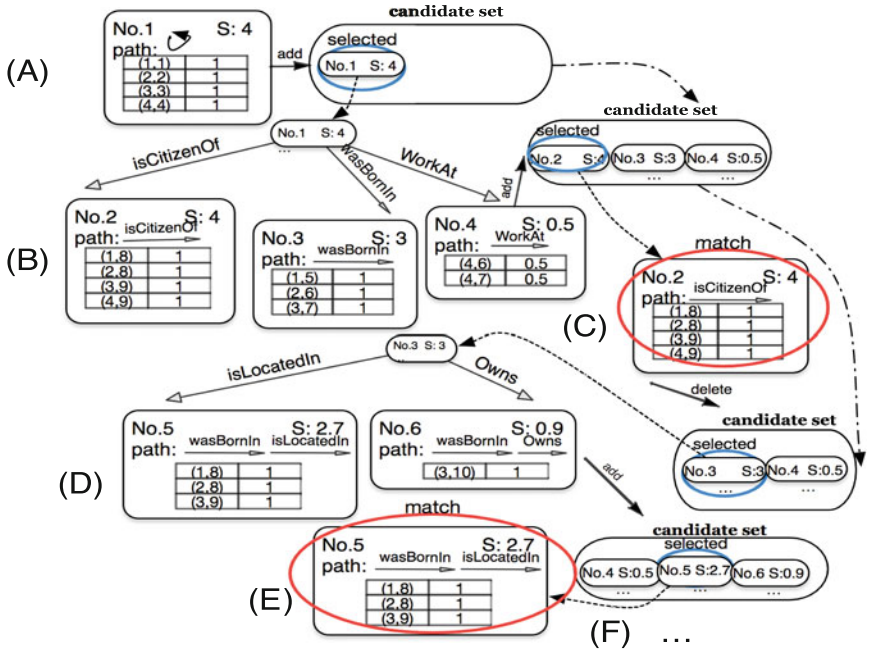


Fig. 7.3 An example of meta-path automatic generation

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.

$$l = \varepsilon \bullet |A|, \tag{7.2}$$

where ε is a limited coefficient and $|A|$ 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.

Furthermore, we explain AMPG with a case study shown in Fig. 7.3. The training pairs are (1, 8), (2, 8), (3, 9), and (4, 9) and sources nodes are 1, 2, 3, and 4. The case starts with creating an initial structure No.1 and inserts it into the candidate set as shown in Fig. 7.3a. 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 shown in Fig. 7.3b. We can pass by three edge types $\xrightarrow{isCitizenOf}$, $\xrightarrow{wasBornIn}$, and $\xrightarrow{WorkAt}$. For each passed edge type, we create new structures such as 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 shown in Fig. 7.3c, 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 found in [6].

7.1.2.3 Integration of Meta Paths

Each meta path found by AMPG is important but has different importances for further link prediction. It is 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, a positive value is the similarity value vector of each positive pair on all selected meta paths, while a 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:

$$\max h = \sum_{x^+ \in q^+} \frac{ln(t(\varpi, x^+))}{|q^+|} + \sum_{x^- \in q^-} \frac{ln(1 - t(\varpi, x^-))}{|q^-|} - \frac{||\varpi||^2}{2}, \tag{7.3}$$

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^+ is positive sample, and x^- is negative sample. 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.

$$\eta(s, t|\Upsilon) = (1 + e^{-(\sum_{x \in \Upsilon} \varpi_x \bullet \sigma(s,t|\prod_x) + \varpi_0)})^{-1}, \tag{7.4}$$

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.

7.1.3 Experiments

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.

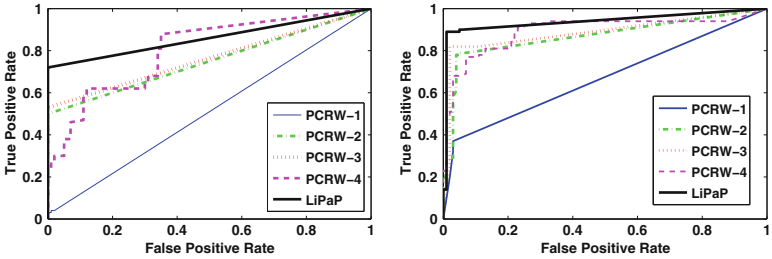
7.1.3.1 Experiment Settings

In our experiments, we use Yago to conduct relevant experiments and it is a large-scale knowledge graph, which is derived from Wikipedia, WordNet, and GeoNames [20]. 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., $< NewYork, locatedin, UnitedStates >$.

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), and as the x coordinate. TPR is the ratio of the number of true positive decisions and actual positive cases while FPR is the ratio of the number of false positive decisions and actual negative cases. The area under the curve is referred to the AUC. The larger the area is, the larger the accuracy in prediction is.

7.1.3.2 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 [11]), 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 entity pairs of them are used as the training set; the others are used as the test set. In LiPaP, we set ε in Eq. 7.2 as 0.005 and the max path length is also limited to 4.



(a) IsLocatedIn (b) IsCitizenOf

Fig. 7.4 Prediction accuracy of different methods on two link prediction tasks

Table 7.1 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

The results of two link prediction tasks are shown in Fig. 7.4. 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 7.1 shows the top four generated meta paths and the corresponding training weights for the $\xrightarrow{isCitizenOf}$ task. It is obvious that four 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 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.

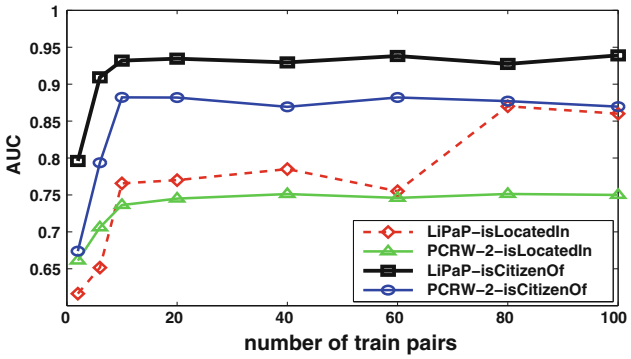


Fig. 7.5 Influence of different sizes of training set

7.1.3.3 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 sizes 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. 7.5, 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.

7.1.3.4 Impact of Weight Learning

To illustrate the benefit of weight learning, we redone the experiments on the $\xrightarrow{isCitizenOf}$ task mentioned above. We run LiPaP with the weight learning or random weights, and with average weights. Figure 7.6 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 features generated by AMPG are all relevant and important, the most important feature also has not got 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.

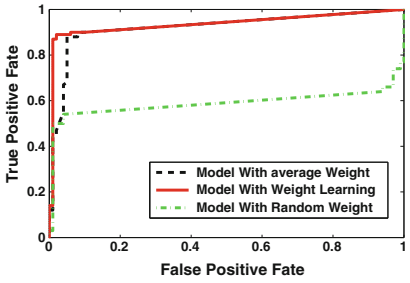


Fig. 7.6 Effectiveness of weight learning

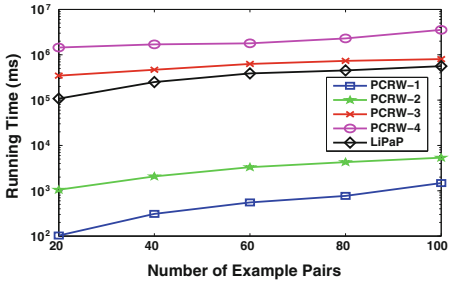


Fig. 7.7 Running times of different methods

7.1.3.5 Efficiency Experiments

In this section, we choose five different sizes of training set, i.e., {20, 40, 60, 80, 100}, to validate the efficiency of finding meta paths of different methods. Figure 7.7 demonstrates the running time on different models for the $\xrightarrow{isLocatedIn}$ task. It is obvious that the running times of these models approximate an linear increase with the increase in the size of the 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.

7.2 Entity Set Expansion with Meta Path in Knowledge Graph

7.2.1 Overview

Entity Set Expansion (ESE) refers to the problem of expanding a small set with a few seed entities into a more complete set, entities of which belong to a particular class. For example, given a few seeds like "China," "America," and "Russia" of country class, ESE will leverage data sources (e.g., text or Web information) to obtain other

country instances, such as Japan and Korea. ESE has been used in many applications, e.g., dictionary construction [7], query refinement [9], and query suggestion [5].

Amounts of methods have been proposed for ESE and most of them are based on the text or Web environment [8, 12, 16, 26, 27]. These methods utilize distribution information or context pattern of seeds to expand entities. For instance, Wang and Cohen [26] propose a novel approach that can be applied to semi-structured documents written in any markup language and in any human language. Recently, knowledge graph has become a popular tool to store and retrieve fact information with graph structure, such as Wikipedia and Yago. Among those texts or Web based methods, some ones also began to leverage knowledge graph as auxiliary for the performance improvement of ESE. For example, Qi et al. [15] use Wikipedia semantic knowledge to choose better seeds for ESE. However, seldom work only utilizes knowledge graph as an individual data source for ESE.

In this chapter, we firstly study the entity set expansion with knowledge graph. Since knowledge graph is usually constituted by $< object, relation, object >$ tuples, we can consider it as a heterogeneous information network (HIN) [18], which contains different types of objects and relations. Based on this HIN, we design a *M*eta *P*ath based *E*ntity *S*et *E*xpansion approach (called MP_ESE). Specifically, the MP_ESE employs the meta path [22], a relation sequence connecting entities, to capture the implicit common feature of seed entities, and designs a Seed-based Meta Path Generation method, called SMPG, to exploit the potential relations among entities. In addition, a heuristic weight learning method is adopted to assign the importance of meta paths. With the help of weighted meta paths, MP_ESE can automatically extend entity set. Based on the Yago knowledge graph, we generate four different types of entity set expansion tasks. On almost all tasks, the proposed method outperforms other baselines.

7.2.2 The MP_ESE Method

In order to solve the problem of ESE with knowledge graph, we propose a novel approach calledMP_ESE. As we have said, KG is a natural HIN, we employ the widely used meta path in HIN to exploit the potential common feature of seeds. The MP_ESE includes the following three steps. Firstly, we design a strategy of extracting candidate entities. Secondly, we develop an algorithm, called SMPG, to automatically discover important meta paths between seeds. Finally, we get a ranking model through combining the meta paths with a heuristic strategy.

7.2.2.1 Knowledge Graph as a HIN

Knowledge graph (KG) [19] is a large and complex graph dataset, which consists of triples of the form $< Subject, Property, Object >$, such as $< StevenSpielberg, directed, War\ Horse(film) >$ shown in Fig. 7.8. Yago [20], DBpedia [1] and Freebase

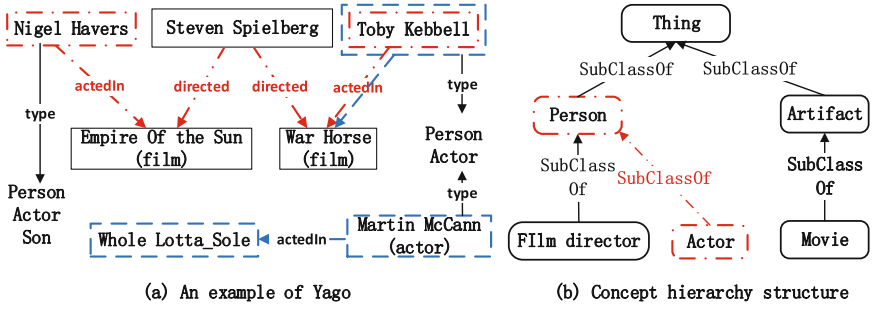


Fig. 7.8 An example of Yago with concept hierarchy structure

[3] are prime examples of KG. The types of entities or relations in KG are often organized as concept hierarchy structure, which describes the subclass relationship among entity types or relations. Figure 7.8b is a toy example and we can see that actor is subclass of person. All the types share a common root called thing.

Heterogeneous information network (HIN) [22] is a network including different types of nodes or links. In HIN, meta path [22], a sequence of relations between objects, is widely used to capture the rich semantic meaning. Since KG contains different types of objects (i.e., subject and object) and links (i.e., property), KG is a natural HIN. In Fig. 7.8, *actedIn* and *directed* are two kinds of link types and actor and film director are different object types. $Person \xrightarrow{actedIn} Movie \xrightarrow{directed^{-1}} Person$ is a meta path between Toby Kebbell and Steven Spielberg and $directed^{-1}$ is the opposite direction of the edge *directed*. In addition, Toby Kebbell and Martin McCann belong to the actor class. Toby Kebbell and Nigel Havers are not only the instances of actor class but also included in the actors who acted in movies Steven Spielberg directed. In order to distinguish the two kinds of sets, we call the latter as the fine-grained set and the former as the coarse grained set.

7.2.2.2 Candidate Entities Extraction

Because the number of entities in knowledge graph is extremely huge, it is unpractical and unreasonable to compute the similarity of each entity and seed. In order to reduce the number of candidate entities, we design a strategy, which leverages concept hierarchy structure introduced above, to get a proper set of candidate entities from knowledge graph. Specifically, it includes the following four steps as shown in Fig. 7.9. Step 1 obtains entity types of each seed. Step 2 generates the initial candidates types by the intersection operation. Step 3 filters the initial candidates types with the concept hierarchy structure. Step 4 extracts candidate entities of satisfying the ultimate candidates types.

In order to clearly illustrate the process of candidate entities extraction, we take Fig. 7.8 as an example and choose Toby Kebbell and Nigel Havers as the seeds.

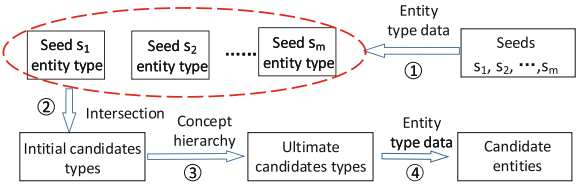


Fig. 7.9 The procedure of candidate entities extraction

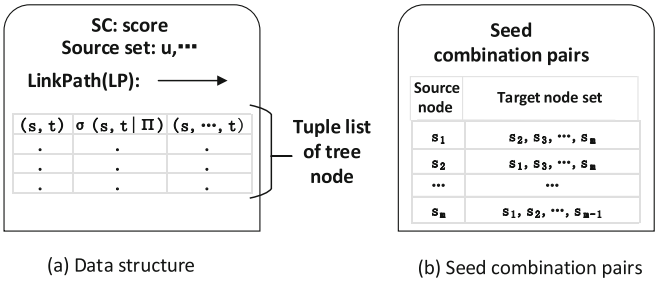


Fig. 7.10 Notation of data structure and seed combination pairs

Their entity types set is {person, actor} and {son, person, actor}, respectively. And the intersection of them is {person, actor} called the initial candidates types. These candidates types may be noisy, which makes the number of candidate entities large. Therefore, we filter some candidates types using concept hierarchy structure as shown in Fig. 7.8b. We choose the class closest to the bottom as the ultimate candidates types. Here, we choose actor class. According to the ultimate types, we extract the candidate entities from Yago.

7.2.2.3 Seed-Based Meta Path Generation

In order to automatically discover meta paths between seeds, we design the Seed-based Meta Path Generation (SMPG) algorithm. The basic idea is that SMPG begins to search the KG from all seeds and finds important meta paths that connect certain number of seed pairs, and the meta paths can reveal the implicit common character of seeds.

The process of meta path generation is traversing the KG indeed, and thus a tree structure is introduced in SMPG. SMPG works by expanding the tree structure and Fig. 7.10a shows the data structure of each tree node, which stores a tuple list of entity pairs with similarity value and the set of being visited entities. The tuple form of the list is $\langle(s, t), \sigma(s,t|\prod), (s, \cdots, t)\rangle$, where (s, t) denotes the source node and target node of the current path $\prod$. Each tree edge denotes the link type between entities. The root node of the tree contains all entity pairs composed of each seed and itself. SMPG starts to expand from the root node step by step to discover important meta

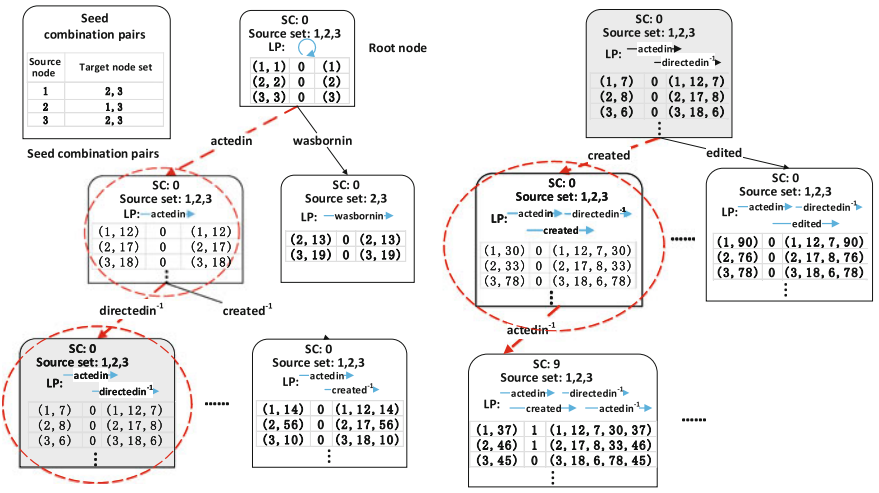


Fig. 7.11 Seed-based meta path generation method

paths. At each step, we check whether the score *SC* of the tree node is larger than the predefined threshold value ν, which guarantees that the meta path is important enough to reveal the character of seeds. If so, we pick out the corresponding meta path, otherwise make a move forward until the tree can not be further expanded. When moving forward, we choose the tree node with the maximum number of different source nodes as well as the minimum number of tuples to expand, which indicates that the path of the tree node covers more seeds and has a better discriminability.

Specifically, in SMPG, we use a source set in the tree node to record the source nodes of all entity pairs in the tuple list. In order to prevent the circle, we record the nodes having been visited along the path $\prod$ in $(s, \cdots, t)$ of the tuple $\langle(s, t), \sigma(s,t|\prod), (s, \cdots, t)\rangle$. Here, $\sigma(s,t|\prod)$ is the similarity that represents whether node t is in the target node set of source node s, it is 1 if so and 0 otherwise. The target node set of each source node can be found in seed combination pairs as shown in Fig. 7.10b and each seed can be combined with the other seeds. $\sigma(s,t|\prod)$ also means that whether the meta path connects the seed pair. And seed pairs that each meta path connects are also recorded. In addition, *LP* is the passing link path and the score *SC* of the tree node is the sum of all tuples similarity, which measures the importance of the tree node or path.

Let us elaborate the process with an example shown in Fig. 7.11, where the set of seeds is {Toby Kebbell, Nigel Havers, Harrison Ford} marked as {1, 2, 3}. The set of seed combination pairs is {[1, (2, 3)], [2, (1, 3)], [3, (1, 2)]} shown in Fig. 7.11. The root node of the tree contains all entity pairs composed of each seed and itself, and has $SC = 0$. The first expansion passes through two types of links: *actedIn* and *wasBornIn*, and gets two new tree nodes. For each new tree node, SMPG records each tuple, *P* and *SC* as well as source set. At the moment, all paths do not connect any seed pairs, so we choose the tree node with the maximum number of source set as well as the minimum number of tuples to expand. Here, we choose the tree

node with link *actedIn* to expand and then get five new tree nodes. Figure 7.11 only demonstrates two of them. After the second expansion, there is not still path connecting seed pairs. Then we continue to choose the tree node with the maximum number of source set and the minimum number of tuples to expand, and we update the corresponding values. After several expansions, a length-4 path $Actor \xrightarrow{actedIn} Movie \xrightarrow{directed^{-1}} Person \xrightarrow{created} Movie \xrightarrow{actedIn^{-1}} Actor$ is found shown by the dash line in Fig. 7.11. And we continue to repeat the process until the condition is satisfied or the tree can not be further expanded.

7.2.2.4 Expanding Entities with Meta Paths

SMPG discovers the important meta paths P, but the importance of each meta path is different for the further entity set expansion, and it is related to the number of seed pairs that meta path connects. Intuitively, the more seed pairs the meta path connects, the more important it is. Thus, we consider the ratio of SP_k and $m * (m-1)$ to be the weight w'_k of meta path $p_k (p_k \in P)$, where SP_k is the number of seed pairs that meta path P_k connects, $m * (m-1)$ denotes the total number of seed pairs, and m is the number of seeds. In order to normalize w'_k, we define the final weight as follows:

$$w_k = \frac{w'_k}{\sum_{k=1}^{l} w'_k} \tag{7.5}$$

where l is the number of meta paths P.

With the w_k, we can combine meta paths to get the following ranking model:

$$R(c_i, S) = \frac{1}{m} \sum_{j=1}^{m} \sum_{k=1}^{l} w_k \cdot r\{(c_i, s_j)|p_k\} \qquad s_j \in S,\ i \in \{1, 2, \cdots, n\} \tag{7.6}$$

where c_i denotes the ith candidate entity and n is the number of candidates. $S = \{s_1, s_2, \cdots, s_m\}$ is the set of seeds and l is the number of meta paths. $r\{(c_i, s_j)|p_k\}$ denotes whether the path p_k connects c_i and s_j; it is 1 if connected and 0 otherwise.

We can compute relevance between each candidate entity and each seed using the ranking model in Eq. 7.6, and then rank all candidate entities.

7.2.3 Experiments

7.2.3.1 Experiment Settings

As a typical KG, Yago [20] has knowledge about more than ten million entities and contains more than 120 million facts. We adopt "yagoFacts," "yagoSimpleTypes,"

Table 7.2 Description of the data

Data	Template of triples	# triples
yagoFacts	< entity relatinship entity >	4,484,914
yagoSimpleTypes	< entity rdf:type wordnet_type >	5,437,179
yagoTaxonomy	< wordnet_type rdfs:subclassof wordnet_type >	69,826

and "yagoTaxonomy" parts of this dataset to conduct experiments, which contain 35 relationships, more than 1.3 million entities of 3455 instance classes. Table 7.2 is the description of the relevant data.

We choose four representative expansion tasks to evaluate the performance of MP_ESE. The classes used in these tasks are summarized as follows: Actors of the movies Steven Spielberg directed, softwares of the companies located in Mountain View of California, movies whose director won National Film Award, and scientists of the universities located in Cambridge of Massachusetts. Four classes are written as Actor*, Software*, Movie*, and Scientist*, the real number of instances in these four classes are 112, 98, 653, and 202, respectively.

We employ two popular criteria of precision-at-k ($p@k$) and mean average precision (MAP) to evaluate the performance of our approach. $p@k$ is the percentage of top k results that belong to correct instances. Here, they are $p@30$, $p@60$, and $p@90$. MAP is the mean of the average precision (AP) of the $p@30$, $p@60$, and $p@90$. $AP = \frac{\sum_{i=1}^{k} \mathrm{p@}i \times rel_i}{\#\text{ of correct instances}}$, where rel_i equals 1 if the result at rank i is correct instance and 0 otherwise.

7.2.3.2 Effectiveness Experiments

In this section, we will validate the effectiveness of MP_ESE on entity set expansion. Since there are no direct solutions for ESE on KG, we design the following three baselines:

- Link-Based. According to the pattern-based methods in text or Web environment, we only consider 1-hop link of an entity, denoted as Link-Based.
- Nearest-Neighbor. Inspired by QBEES [13, 14], we consider 1-hop link and 1-hop entity at the same time, called Nearest-Neighbor.
- PCRW. Based on the path-constrained random walk [11], we only compare with length-2 path, denoted as PCRW. The reason is that the longer path needs more running time.

For each class introduced above, we randomly take three seeds from the instance set to conduct an experiment. We run algorithms 30 times and record the average results. In MP_ESE, we set the predefined threshold value ν to be $m * (m-1)/2 + 1$, which can guarantee that the path connects half number of seeds or more, m is the number of seeds. And the max length of path is set to be 4 since meta paths with

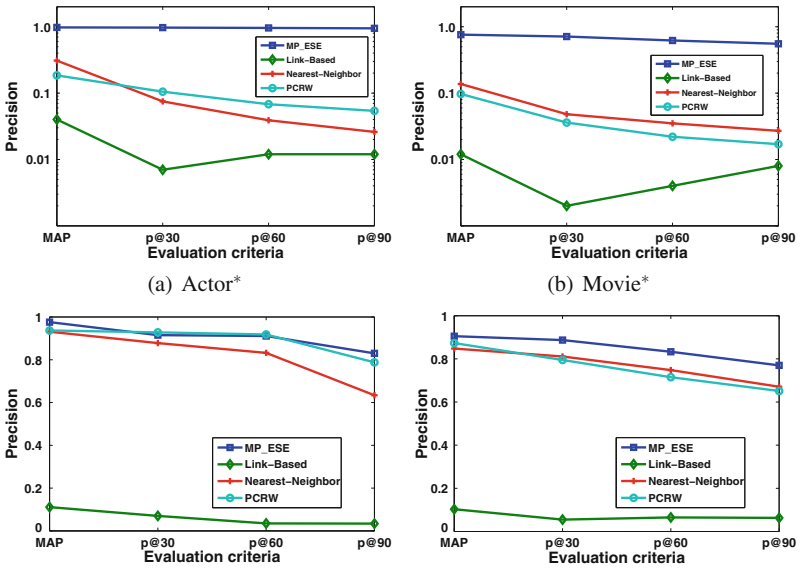


Fig. 7.12 The result of entity set expansion

length more than 4 are almost irrelevant. The optimal parameters are set for other baselines.

The overall results of entity set expansion are given in Fig. 7.12. From Fig. 7.12, we can see that our MP_ESE approach achieves better performances than other methods on almost all conditions, especially on the Actor* and Movie* tasks. All baselines have very bad performances on Actor* and Movie*. We think the reason is that the 1-hop link or 1-hop entity cannot further distinguish the character of the fine-grained class but MP_ESE can distinguish them well. On the Software* task, MP_ESE and PCRW have close performance. The reason is that Software* is an overlapping class and has another class label depicted by length-2 path $Software \xrightarrow{created^{-1}} Company \xrightarrow{created} Software$. Due to the fact that it has few semantic meaning, Link-Based has very bad performance. In all, MP_ESE has the best performances because it employs the important meta paths between seeds and can capture the subtle semantic meaning.

In order to intuitively observe the effectiveness of discovered meta paths, Table 7.3 depicts the top three meta paths returned by SMPG for Actor*. We observe that these meta paths reveal some common traits of Actor*. The first meta path indicates that actors act in movies directed by the same director, which shows that SMPG can effectively mine the most important semantic meaning of Actor*. The second and the third meta paths imply that some actors act in movies edited or composed by the same person. Through leveraging the important meta paths discovered by SMPG, we can find other entities belonging to the same class with seeds.

Table 7.3 Most relevant 3 meta paths for Actor*

Meta path	w
$\text{Person} \xrightarrow{actedIn} \text{Movie} \xrightarrow{directed^{-1}} \text{Person} \xrightarrow{directed} \text{Movie} \xrightarrow{actedIn^{-1}} \text{Person}$	0.2180
$\text{Person} \xrightarrow{actedIn} \text{Movie} \xrightarrow{writeMusicFor^{-1}} \text{Person} \xrightarrow{writeMusicFor} \text{Movie} \xrightarrow{actedIn^{-1}} \text{Person}$	0.1495
$\text{Person} \xrightarrow{actedIn} \text{Movie} \xrightarrow{edited^{-1}} \text{Person} \xrightarrow{edited} \text{Movie} \xrightarrow{actedIn^{-1}} \text{Person}$	0.1476

7.3 Conclusions

In this chapter, we extend the traditional heterogenous network to the schema-rich heterogeneous network where there are a huge number of types of nodes and links, such as knowledge graph. In this kind of networks, it is difficult to depict the network schema and impossible to enumerate the potential meta paths. We study two data mining tasks in schema-rich heterogeneous networks. In the link prediction task, we design the LiPaP to predict potential links among nodes, and we also propose the MP_ESE to automatically extend entity set with knowledge graph. In these methods, it is critical to efficiently and effectively discover meta paths and learning their weights. Since the knowledge graph is widely used in text analysis and search engine, when we consider the knowledge graph as heterogenous network, it will tremendously extend the study of heterogeneous network. Simultaneously, it also provides a new way for knowledge graph mining.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: A Nucleus for a Web of Open Data. Springer, Berlin (2007)
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. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: SIGMOD, pp. 1247–1250 (2008)
4. Cao, B., Kong, X., Yu, P.S.: Collective prediction of multiple types of links in heterogeneous information networks. In: ICDM, pp. 50–59 (2014)
5. Cao, H., Jiang, D., Pei, J., He, Q., Liao, Z., Chen, E., Li, H.: Context-aware query suggestion by mining click-through and session data. In: KDD, pp. 875–883 (2008)
6. Cao, X., Zheng, Y., Shi, C., Li, J., Wu, B.: Link prediction in schema-rich heterogeneous information network. In: PAKDD, pp. 449–460 (2016)
7. Cohen, W.W., Sarawagi, S.: Exploiting dictionaries in named entity extraction: combining semi-markov extraction processes and data integration methods. In: KDD, pp. 89–98 (2004)
8. He, Y., Xin, D.: Seisa: set expansion by iterative similarity aggregation. In: WWW, pp. 427–436 (2011)
9. Hu, J., Wang, G., Lochovsky, F., Sun, J.t., Chen, Z.: Understanding user's query intent with wikipedia. In: WWW, pp. 471–480 (2009)

10. Jaiwei, H.: Mining heterogeneous information networks: the next frontier. In: SIGKDD, pp. 2–3 (2012)
11. Lao, N., Cohen, W.W.: Relational retrieval using a combination of path-constrained random walks. Mach. Learn. **81**(1), 53–67 (2010)
12. Li, X.L., Zhang, L., Liu, B., Ng, S.K.: Distributional similarity vs. pu learning for entity set expansion. In: ACL, pp. 359–364 (2010)
13. Metzger, S., Schenkel, R., Sydow, M.: Qbees: query by entity examples. In: CIKM, pp. 1829–1832 (2013)
14. Metzger, S., Schenkel, R., Sydow, M.: Aspect-based similar entity search in semantic knowledge graphs with diversity-awareness and relaxation. In: IJCWI, pp. 60–69 (2014)
15. Qi, Z., Liu, K., Zhao, J.: Choosing better seeds for entity set expansion by leveraging wikipedia semantic knowledge. In: CCPR, pp. 655–662 (2012)
16. Sarmento, L., Jijkuon, V., de Rijke, M., Oliveira, E.: More like these: growing entity classes from seeds. In: CIKM, pp. 959–962 (2007)
17. Shi, C., Kong, X., Yu, P.S., Xie, S., Wu, B.: Relevance search in heterogeneous networks. In: EDBT, pp. 180–191 (2012)
18. Shi, C., Li, Y., Zhang, J., Sun, Y., Yu, P.S.: A survey of heterogeneous information network analysis. Comput. Sci. **134**(12), 87–99 (2015)
19. Singhal, A.: Introducing the knowledge graph: things, not strings. Official google blog (2012)
20. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: WWW, pp. 697–706 (2007)
21. 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)
22. Sun, Y., Han, J., Yan, X., Yu, P.S., Wu, T.: Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. VLDB **4**(11), 992–1003 (2011)
23. Sun, Y., Han, J., Aggarwal, C.C., Chawla, N.V.: When will it happen?: relationship prediction in heterogeneous information networks. In: WSDM, pp. 663–672 (2012)
24. Sun, Y., 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)
25. W3C: Rdf current status. http://www.w3.org/standards/techs/rdf#w3c_all
26. Wang, R.C., Cohen, W.W.: Language-independent set expansion of named entities using the web. In: ICDM, pp. 342–350 (2007)
27. Wang, R.C., Cohen, W.W.: Iterative set expansion of named entities using the web. In: ICDM, pp. 1091–1096 (2008)
28. Yu, X., Gu, Q., Zhou, M., Han, J.: Citation prediction in heterogeneous bibliographic networks. In: SDM, pp. 1119–1130 (2012)
29. Zha, H., He, X., Ding, C.H.Q., Gu, M., Simon, H.D.: Bipartite graph partitioning and data clustering. CoRR cs.IR/0108018 (2001)