Entity Set Expansion with Meta Path in Knowledge Graph

Yuyan Zheng[1], Chuan Shi[1,2(✉)], Xiaohuan Cao[1], Xiaoli Li[3], and Bin Wu[1]

[1] Beijing Key Lab of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing 100876, China
{shichuan,wubin}@bupt.edu.cn, zyy0716_source@163.com, devil_baba@126.com
[2] Beijing Advanced Innovation Center for Imaging Technology, Capital Normal University, Beijing 100048, China
[3] Institute for Infocomm Research, A*STAR, Singapore, Singapore
xlli@i2r.a-star.edu.sg

Abstract. Entity set expansion (ESE) is the problem that expands a small set of seed entities into a more complete set, entities of which have common traits. As a popular data mining task, ESE has been widely used in many applications, such as dictionary construction and query suggestion. Contemporary ESE mainly utilizes text and Web information. That is, the intrinsic relation among entities is inferred from their occurrences in text or Web. With the surge of knowledge graph in recent years, it is possible to extend entities according to their occurrences in knowledge graph. In this paper, we consider the knowledge graph as a heterogeneous information network (HIN) that contains different types of objects and links, and propose a novel method, called MP_ESE, to extend entities in the HIN. The MP_ESE employs meta paths, a relation sequence connecting entities, in HIN to capture the implicit common traits of seed entities, and an automatic meta path generation method, called SMPG, is provided to exploit the potential relations among entities. With these generated and weighted meta paths, the MP_ESE can effectively extend entities. Experiments on real datasets validate the effectiveness of MP_ESE.

Keywords: Heterogeneous information network · Entity set expansion · Knowledge graph · Meta path

1 Introduction

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 [4], query refinement [6] and query suggestion [2].

J. Kim et al. (Eds.): PAKDD 2017, Part I, LNAI 10234, pp. 317–329, 2017.
DOI: 10.1007/978-3-319-57454-7_25

Numerous methods have been proposed for ESE and most of them are based on the text or Web environment [5,9,14,19,20]. These methods utilize distribution information or context pattern of seeds to expand entities. For instance, Wang and Cohen [19] 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 text or Web based methods, some researchers also began to leverage knowledge graph as auxiliary for the performance improvement of ESE. For example, Qi et al. [12] use Wikipedia semantic knowledge to choose better seeds for ESE. However, seldom work only utilizes knowledge graph as individual data source for ESE.

In this paper, we firstly study the entity set expansion with knowledge graph. Since knowledge graph is usually constituted by $<Subject, Property, Object>$ tuples, we can consider it as a heterogeneous information network (HIN) [15], which contains different types of objects and relations. Based on this HIN, we design a novel *M*eta *P*ath based *E*ntity *S*et *E*xpansion approach (called MP_ESE). Specifically, the MP_ESE employs the meta path [18], a relation sequence connecting entities, to capture the implicit common feature of seed entities, and designs an automatic 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.

2 Related Work

In recent years, there has been a significant amount of work on ESE and ESE has received considerable attention from both research [11,19,20] and industry circles (e.g., Google Sets). According to the difference of data sources utilized by ESE, these methods are based on text, Web environment and others.

For those text data source based ESE methods, they utilize the distribution information of the surrounding words of entities to expand certain class [5,9,14]. For those Web environment based ESE methods, proper patterns of seeds are extracted and then these patterns are used to extract new candidate entities. This kind of methods can also be used in text data source. Recently, some researchers began to take advantage of external semantic information to improve performance of set expansion for text or Web data source. Qi et al. [12] introduce the semantic knowledge by leveraging Wikipedia and reduce the seed ambiguity. Sadamitsu et al. [13] use topic information to alleviate semantic drift. Jindal and Roth [7] specify some negative examples to confine the expansion category.

More recently, HIN and knowledge graph have also been applied for related work. Yu et al. [21] propose a meta-path-based ranking model ensemble to represent semantic meaning for entity query. Different from our work, it has solved

a similar but different problem (i.e., entity query), and the meta paths in their method need to be provided by domain expert users. QBEES [10] is designed for entity similarity search based on aspects of the entities, and Chen et al. [3] design a system for entity exploration and debugging. They both utilize the knowledge graph, but they do not employ the HIN method.

3 Preliminary

In this section, we describe some key concepts and present some preliminary knowledge in this paper.

Knowledge graph (KG) [16] 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. 1. Yago [17] and DBpedia [1] are two prime examples of KG. The types of entities or relations in KG are often organized as concept hierarchy structure, which describes the sub-class relationship among entity types or relations. Figure 1(b) is a snapshot of Yago and we can see that actor is sub-class of person shown by the dashed line in Fig. 1(b). All the types share a common root called thing.

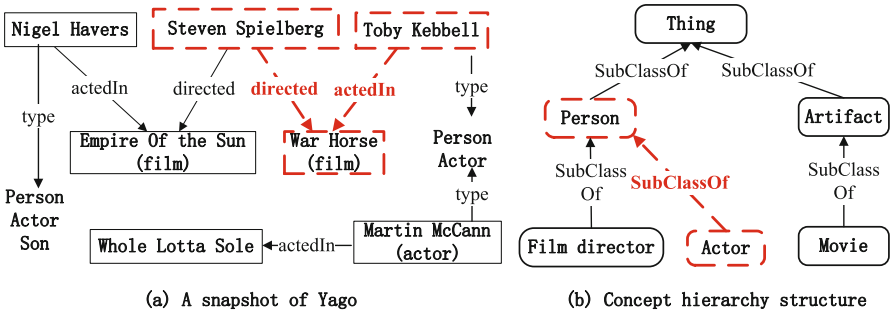

Fig. 1. A snapshot of Yago with concept hierarchy structure.

Heterogeneous information network (HIN) [18] is defined as a directed graph $G = (V, E)$ with an object type mapping function $\varphi : V \rightarrow \mathcal{A}$ and a link type mapping function $\psi : E \rightarrow \mathcal{R}$, where V, E, $\mathcal{A}$ and $\mathcal{R}$ denotes object set, link set, object type set and relation type set, respectively, and the number of object types $|\mathcal{A}| > 1$ or the number of relation types $|\mathcal{R}| > 1$. In HIN, meta path [18] is widely used to capture the rich semantic meaning and is denoted in the form of $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \ldots \xrightarrow{R_l} A_{l+1}$, which is a sequence of object types and link types between objects.

Since KG contain different types of objects (i.e., subject and object) and links (i.e., property), KG is a natural HIN. In Fig. 1, *actedIn* and *directed* are two kinds of links types, actor and film director are different object types.

$Person \xrightarrow{actedIn} Movie \xrightarrow{directed^{-1}} Person$ is a meta path shown by the dashed line in Fig. 1(a), $directed^{-1}$ is the opposite direction of the edge *directed*.

In addition, Toby Kebbell and Martin McCann belong to 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.

4 The Proposed Method

In order to solve the problem of ESE with knowledge graph, we propose a novel approach called *M*eta *P*ath based *E*ntity *S*et *E*xpansion (MP_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 Seed-based Meta Path Generation (SMPG) to automatically discover important meta paths between seeds. Finally, we get a ranking model through combining the meta paths with a heuristic strategy.

4.1 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 in Sect. 3, to get a proper set of candidate entities from knowledge graph. Specifically, it includes the following four steps as shown in Fig. 2. 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.

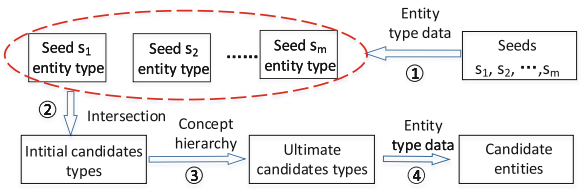

Fig. 2. The procedure of candidate entities extraction.

In order to clearly illustrate the process of candidate entities extraction, we take Fig. 1 as an example and choose Toby Kebbell and Nigel Havers as the seeds. 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 too general, which makes the number of candidate entities large. Therefore, we filter some candidates types using concept hierarchy structure as shown in Fig. 1(b). We choose the most specific 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.

4.2 Seed-Based Meta Path Generation

In order to automatically discover meta paths between seeds, we design the Seed-based Meta Path Generation algorithm (SMPG). 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.

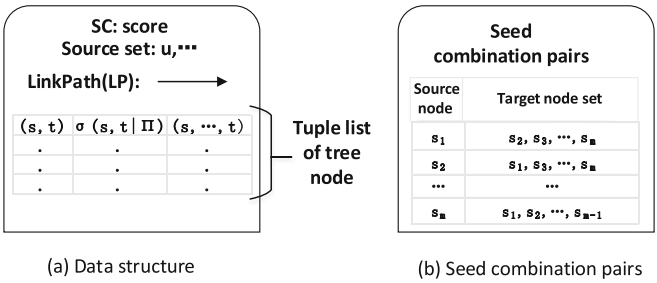

Fig. 3. Notation of data structure and seed combination pairs.

The process of meta path generation is traversing the KG in deed, and thus a novel tree structure is introduced in SMPG. SMPG works by expanding the tree structure and Fig. 3(a) 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 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 source set 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.

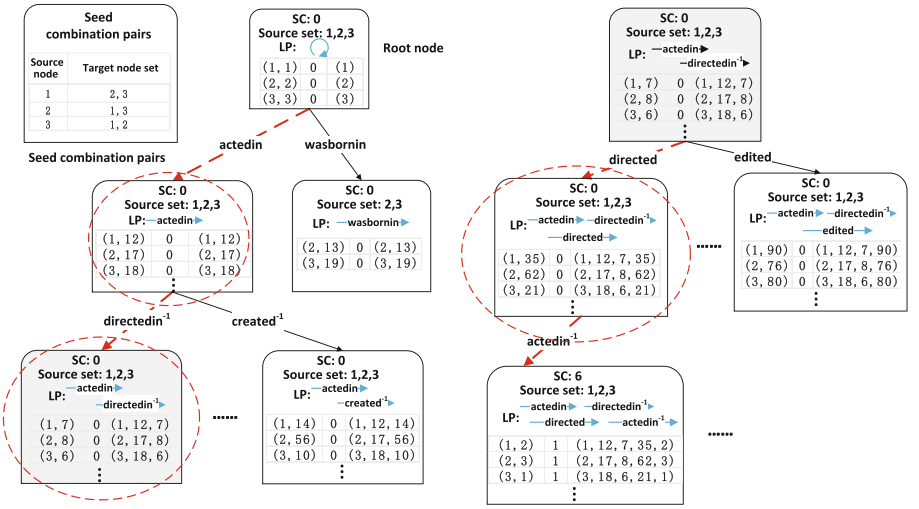

Fig. 4. Seed-based meta path generation method.

Specifically, in SMPG, we use a source set in the tree node to record the source nodes of all entity pairs in 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. 3(b) 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 algorithm with an example shown in Fig. 4, 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. 4. 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 4 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. Except seeds 1, 2 and 3, the other marked entities such as 35, 62 denote those being

visited in-between entities of the path. After several expansions, a length-4 path $Actor \xrightarrow{actedIn} Movie \xrightarrow{directed^{-1}} Person \xrightarrow{directed} Movie \xrightarrow{actedIn^{-1}} Actor$ is found shown by the dash line in Fig. 4. And we continue to repeat the process until the condition is satisfied or the tree can not be further expanded.

Algorithm 1. Seed-based Meta Path Generation Algorithm

```
   Input: Knowledge graph G, seed set S = {s_1, s_2, ..., s_m}.
   Output: The set of meta paths P, seed pairs SP that each meta path connects.
 1 Create the root node of the tree T;
 2 sl ⇐ link types set;//the link needs connect 2 seeds or more
 3 while T can be expanded do
 4     N ⇐ tree nodes with the maximum number of source set in T;
 5     n ⇐ tree node with the minimum number of tuples in N;
 6     for each tuple tp ∈ n do
 7         get pair tp.(s, t);
 8         for each neighbor e of tp.t in G do
 9             l ⇐ link from tp.t to e;
10             if e not being visited and l ∈ sl then
11                 if l not in n.child then
12                     add l to n.child;
13                     create a new child tree node with key n.key + l;
14                 if (s, e) ∈ seed combination pairs then
15                     σ(s,e|∏) ⇐ 1;
16                     add seed pair (s, e) to the tree node with key n.key + l;
17                 else
18                     σ(s,e|∏) ⇐ 0;
19                 add e to the visited set (s, ..., t);
20                 insert tuple ⟨(s, e),σ(s,e|∏),(s, e, ··· , t)⟩ into tree node with key n.key + l;
21                 add tp.s to the source set of tree node with key n.key + l;
22                 update the SC of tree node with key n.key + l;
23     for each node en in T do
24         if en.SC > threshold ν then
25             add meta path ∏ of en into P;
26             add the corresponding seed pairs that ∏ connects into SP;
27 return P, SP
```

We present the detailed steps of SMPG in Algorithm 1. Firstly, we create the root node of the tree in Step 1 and give some predefined constants in Step 2. Then we expand the tree and find the important meta paths in Steps 3–26. At each expansion, we choose the tree node with the maximum number of seeds as well as the minimum number of tuples in Steps 4, 5. Step 10 judges whether the link is in the set of the given link type, whether the neighbor node isn't visited before. If so, we make an expansion and examine whether the entity pair is in seed combination pairs in Step 14. If so, Step 15 records the connected entity pair. And we insert the new tuple into the corresponding tree node in Step 20. Meanwhile Step 21 adds the source node of the entity pair to the source set of the tree node and Step 22 updates SC. Steps 23–26 get the expected meta paths and the corresponding seed pairs.

4.3 Combination of Meta Path

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 paths connect. 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:

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

where l is the number of meta paths P.

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

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

where c_i denotes the ith candidate entity, n is the number of candidates. $S = \{s_1, s_2, \cdots, s_m\}$ is the set of seeds. $r\{(c_i, s_j)|p_k\}$ 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. 2, and then rank all candidate entities.

5 Experiments

5.1 Dataset

As a typical KG, Yago is a huge semantic knowledge graph derived from Wikipedia, WordNet and GeoNames [17]. Currently, it has knowledge about more than 10 million entities and contains more than 120 million facts. We adopt "yagoFacts", "yagoSimpleTypes" and "yagoTaxonomy" parts of this dataset to conduct experiments, which contain 35 relationships, more than 1.3 million entities of 3455 instance classes. Table 1 is the description of the relevant data.

Table 1. 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

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.

5.2 Criteria

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 $p@90$. $AP = \frac{\sum_{i=1}^{k} p@i \times rel_i}{\#\text{ of correct instances}}$, where rel_i equals 1 if the result at rank i is correct instance and 0 otherwise.

5.3 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 three baselines. (1) 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. (2) Nearest-Neighbor. Inspired by QBEES [10], we consider 1-hop link and 1-hop entity at the same time, called Nearest-Neighbor. (3) PCRW. Based on the path constrained random walk [8], 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 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. 5. From Fig. 5, 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 can not further distinguish the character of the fine grained class but MP_ESE can distinguish 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 2 depicts the top 3 meta paths returned by SMPG for Actor*. We observe that these meta paths reveal some common character 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

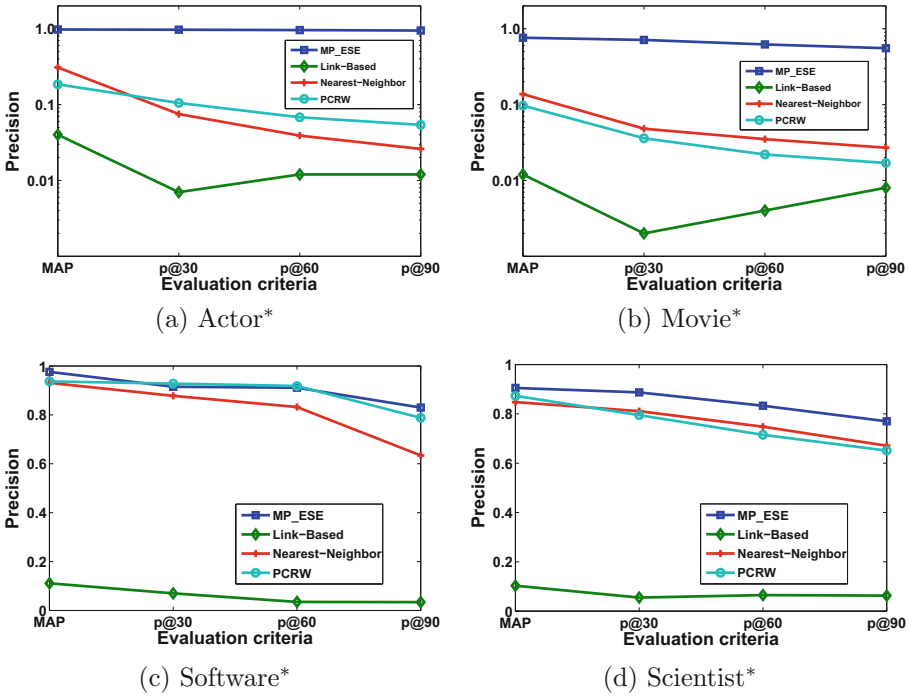

Fig. 5. The result of entity set expansion.

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 2. Most relevant 3 meta paths for Actor*

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

5.4 Impact of Seed Size

To evaluate the impact of seed size on the performance, we conduct relevant experiments in the range of seed size from 2 to 6 for Actor*. For each seed size, we randomly select the corresponding seeds from the instance set to conduct an experiment. We run our algorithm 30 times and record the maximum, the minimum and the average results, which are demonstrated in Table 3.

Table 3. The impact of seed size on Actor*.

Seed size	p@30			p@60			p@90			MAP		
	Max	Min	Mean	Max	Min	Mean	Max	Min	Mean	Max	Min	Mean
2	1.0	0.067	0.858	1.0	0.117	0.834	1.0	0.111	0.811	1.0	0.110	0.900
3	1.0	0.433	0.970	1.0	0.333	0.959	1.0	0.256	0.945	1.0	0.427	0.976
4	1.0	0.733	0.977	1.0	0.517	0.957	1.0	0.4	0.938	1.0	0.852	0.988
5	1.0	0.967	0.998	1.0	0.95	0.996	1.0	0.767	0.988	1.0	0.967	0.997
6	1.0	1.0	1.0	1.0	0.933	0.996	1.0	0.867	0.991	1.0	0.987	0.999

From Table 3, we can see that the performance has an improvement with the increasing of seed size. The performance with 2 seeds is the lowest, since 2 seeds do not contain plenty of information and may have several class labels. The performance with 3 seeds has been good enough. When the number of seeds is larger than 3, the improvement is tiny but the running time is much. Therefore, we employ 3 seeds in other experiments. In all, the proper seed size should be determined. Besides, there is a big difference between the maximum and minimum precisions because of the random seed sets.

5.5 Influence of Weight

To demonstrate the influence of the weight on the performance, We conduct experiments with different seed combinations of size 3 many times and record the average results. Table 4 reports the results with different weights on the four tasks introduced above. We can observe that the heuristic weight has better performance than average and random weights on the whole, which suggests that the importance of meta paths is different and some paths can better reflect the implicit character of seeds than others. For Software* task, weight has a tiny effect on performance, because the total number of meta paths is 5 and there exist several class labels.

Table 4. The impact of different weights.

Class	Heuristic weight				Average weight				Random weight			
	p@30	p@60	p@90	MAP	p@30	p@60	p@90	MAP	p@30	p@60	p@90	MAP
Actor*	0.970	0.959	0.945	0.976	0.948	0.934	0.917	0.958	0.941	0.915	0.889	0.949
Software*	0.915	0.911	0.830	0.926	0.911	0.908	0.824	0.925	0.918	0.903	0.812	0.930
Movie*	0.711	0.620	0.554	0.760	0.639	0.530	0.461	0.710	0.480	0.414	0.369	0.554
Scientist*	0.887	0.833	0.770	0.905	0.822	0.743	0.665	0.871	0.546	0.486	0.429	0.627

6 Conclusions

In this paper, we study the problem of entity set expansion in knowledge graph. We model knowledge graph as a heterogeneous information network and propose a *M*eta *P*ath based *E*ntity *S*et *E*xpansion approach called MP_ESE, which employs the meta path to exploit the implicit common feature of seeds. In order to automatically find the important meta paths between seeds, MP_ESE designs a novel algorithm called SMPG. And then we design a heuristic strategy to assign the importance of meta paths. MP_ESE utilizes the weighted meta paths to expand entities. Experiments on Yago validate the effectiveness of MP_ESE.

Acknowledgements. This work is supported in part by the National Natural Science Foundation of China (No. 61375058), National Key Basic Research and Department (973) Program of China (No. 2013CB329606), and the Co-construction Project of Beijing Municipal Commission of Education.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC/ISWC - 2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007). doi:10.1007/978-3-540-76298-0_52
2. 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. ACM (2008)
3. Chen, J., Chen, Y., Du, X., Zhang, X., Zhou, X.: SEED: a system for entity exploration and debugging in large-scale knowledge graphs. In: ICDM, pp. 1350–1353. IEEE (2016)
4. 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. ACM (2004)
5. He, Y., Xin, D.: Seisa: set expansion by iterative similarity aggregation. In: WWW, pp. 427–436. ACM (2011)
6. Hu, J., Wang, G., Lochovsky, F., Sun, J.T., Chen, Z.: Understanding user's query intent with Wikipedia. In: WWW, pp. 471–480. ACM (2009)
7. Jindal, P., Roth, D.: Learning from negative examples in set-expansion. In: ICDM, pp. 1110–1115. IEEE (2011)
8. Lao, N., Cohen, W.W.: Relational retrieval using a combination of path-constrained random walks. Mach. Learn. **81**(1), 53–67 (2010)
9. Li, X.L., Zhang, L., Liu, B., Ng, S.K.: Distributional similarity vs. PU learning for entity set expansion. In: ACL, pp. 359–364. ACL (2010)
10. Metzger, S., Schenkel, R., Sydow, M.: Qbees: query by entity examples. In: CIKM, pp. 1829–1832. ACM (2013)
11. Pasca, M.: Weakly-supervised discovery of named entities using web search queries. In: CIKM, pp. 683–690. ACM (2007)
12. Qi, Z., Liu, K., Zhao, J.: Choosing better seeds for entity set expansion by leveraging Wikipedia semantic knowledge. In: Liu, C.-L., Zhang, C., Wang, L. (eds.) CCPR 2012. CCIS, vol. 321, pp. 655–662. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33506-8_80

13. Sadamitsu, K., Saito, K., Imamura, K., Kikui, G.: Entity set expansion using topic information. In: ACL: HLT: short papers-Volume 2, pp. 726–731. ACL (2011)
14. Sarmento, L., Jijkuon, V., de Rijke, M., Oliveira, E.: More like these: growing entity classes from seeds. In: CIKM, pp. 959–962. ACM (2007)
15. Shi, C., Li, Y., Zhang, J., Sun, Y., Yu, P.S.: A survey of heterogeneous information network analysis. arXiv preprint arXiv:1511.04854 (2015)
16. Singhal, A.: Introducing the knowledge graph: things, not strings. Official Google Blog (2012)
17. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: WWW, pp. 697–706. ACM (2007)
18. 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)
19. Wang, R.C., Cohen, W.W.: Language-independent set expansion of named entities using the web. In: ICDM, pp. 342–350. IEEE (2007)
20. Wang, R.C., Cohen, W.W.: Iterative set expansion of named entities using the web. In: ICDM, pp. 1091–1096. IEEE (2008)
21. Yu, X., Sun, Y., Norick, B., Mao, T., Han, J.: User guided entity similarity search using meta-path selection in heterogeneous information networks. In: CIKM, pp. 2025–2029. ACM (2012)