A Heterogeneous Information Network Method for Entity Set Expansion in Knowledge Graph

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

[1] Beijing Key Lab of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications,Beijing, China 100876
[2] Institute for Infocomm Research, A*STAR, Singapore, Singapore
devil_baba@126.com,shichuan@bupt.edu.cn,zyy0716_source@163.com, username.djy@gmail.com,xlli@i2r.a-star.edu.sg,wubin@bupt.edu.cn

Abstract. Entity Set Expansion (ESE) is an important data mining task, e.g. query suggestion. It aims to expand an entity seed set to obtain more entities which have traits in common. Traditionally, text and Web information are widely used for ESE. Recently, some ESE methods employ Knowledge Graph (KG) to extend entities. However, these methods usually fail to sufficiently and efficiently utilize the rich semantics contained in KG. In this paper, we use the Heterogeneous Information Network (HIN) to represent KG, which would effectively capture hidden semantic relations between seed entities. However, the complex KG introduces new challenges for HIN analysis, such as generation of meta paths between entities and addressing ambiguity caused by multiple types of objects. To solve these problems, we propose a novel Concatenated Meta Path based Entity Set Expansion method (CoMeSE). With the delicate design of the concatenated meta path generation and multi-type-constrained meta path, CoMeSE can quickly and accurately detect important path features in KG. In addition, heuristic learning and PU learning are employed to learn the weights of extracted meta paths. Extensive experiments on real dataset show that the CoMeSE accurately and quickly expands the given small entity set.

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

1 Introduction

Entity Set Expansion (ESE) is mainly about, given a small set of seed entities, finding out other entities belonging to it and expanding the set to a more complete one. For example, given a few seeds like "New York", "Los Angeles" and "Chicago", ESE will discover the relation among them and obtain other city instances in America, such as "Houston". ESE has been widely used in many applications, e.g., dictionary construction [2] and query suggestion [1].

Plenty of ESE works have been done, and most of them discover distribution information or context pattern of seeds in text or Web resources to infer the

intrinsic relation for ESE [8]. Recently, Knowledge Graph (KG), a kind of structured data source, has been more and more important for knowledge mining. So some ESE works use KG as a supplement for text to improve performance [7].

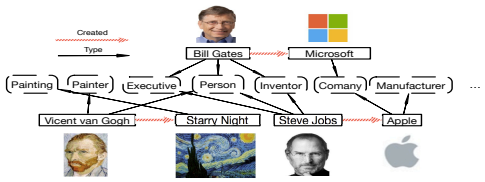


Fig. 1. A snapshot in knowledge graph.

However, few researches utilize KG as individual data source for ESE. Owing to rich semantics and structural representation of KG, it is feasible to employ KG to extend entities. KG, constructed by triples $< Subject, Property, Object >$, can be considered as a Heterogeneous Information Network (HIN) that contains different types of objects and relations [9]. In HIN, meta path [9], a sequence of relations connecting two objects, is widely used for semantic capture. For example, in Fig. 1, the fact triples can form a network and the path in it can show semantics. Therefore, we can consider KG as an HIN and employ meta path features to solve ESE problem. However, this idea faces challenges as follows.

- It is impossible to enumerate meta paths in KG. In traditional HIN with only a few types of objects and relations, it is easy to enumerate useful meta paths. However, it is not the case for KG because of its complexity. For example, DBpedia has 3 billion facts. It is impossible to find meta paths by manual.
- In KG, objects connected by a relation may affiliate to multiple types, which will cause ambiguity. In traditional HIN, objects have a unique type, which makes meta path have definite semantics. But in KG, objects may affiliate to multiple types which will lead to uncertain semantics. For example, in Fig. 1, the objects connecting to the $\xrightarrow{created}$ relation affiliate to the types of executive, painter and etc. The $\xrightarrow{created}$ between different pairs has different meaning. The relation between (Bill Gates, Microsoft) means establishing, while for (Vincent van Gogh, Starry Night), it means painting.
- It is not easy to combine path features for ESE, though we extract path features among entities. ESE problem usually has few seeds, so it is difficult to use traditional supervised method to build a ranking or classification model.

It is not a trivial task to solve these challenges. A very recent attempt by Zheng et al. [14] illustrates limited performance improvement on the problems but cost huge time and space. In this paper, we propose a novel *Co*ncatenated *M*eta Path based *E*ntity *S*et *E*xpansion method (CoMeSE) for ESE problem in KG. The CoMeSE includes three steps. Firstly, in order to reduce time and reuse

visited paths, CoMeSE designs a novel random walk based Concatenated Meta Path Generation (RWCP) method to quickly and accurately discover useful meta paths by concatenating meta paths that have been visited in KG. Secondly, in order to solve the ambiguity problem caused by multiple types of objects, we propose a multi-type-constrained meta path concept to subtly capture semantics in KG, and further design a novel similarity measure based on it. Thirdly, for solving the problem of very limited positive samples, besides a heuristic weight strategy, we employ a PU learning method (Learning from Positive and Unlabled Example) to effectively learn the weights of meta paths. Plenty of experiments on real dataset have been done to validate the effectiveness and efficiency of CoMeSE. The experiments show that, compared to the state of the arts, CoMeSE can quickly and accurately extend entities because of its delicate designs.

2 Preliminary

In this section, we describe some main concepts and preliminary knowledge.

Heterogeneous Information Network [4] is a kind of information network defined as a directed graph $G = (\mathcal{O}, \mathcal{R})$, which consists of either different types of objects $\mathcal{O}$ or different types of relations $\mathcal{R}$. In an HIN, there can be different paths connecting two objects and these paths are called **meta path** [12]. A meta path $\mathcal{P}$ is defined as $\mathcal{P}^{R_1 \circ \cdots \circ R_n} = \mathcal{T}(o_1) \xrightarrow{R_1} \cdots \xrightarrow{R_n} \mathcal{T}(o_{n+1})$, where o_i presents the object at position i in $\mathcal{P}$, $\mathcal{T}(o_i)$ is the type of o_i, and R_i is a type of relation. Note that $\mathcal{T}(o_i)$ corresponds to a unique entity type.

Knowledge Graph [10] is a knowledge base system with semantic properties and derived from text data of knowledge sources. KG is conducted by triples $< Subject, Property, Object >$. In this paper, we model KG as an HIN, "Subject" and "Object" as nodes, "Property" as links. This HIN is not like general HIN with simple schema but with thousands of node and link types. Besides, in KG, one entity is subordinate to multiple types and the meanings of links may introduce ambiguity. So traditional meta path would capture exact semantics badly. In order to solve the ambiguity problem, we propose a novel concept of Multi-Type-Constrained Meta Path to more subtly capture semantic relations.

Definition 1. ***Multi-Type-Constrained Meta Path*** *(MuTyPath) is a special meta path where each object position is constrained by a set of entity types. A MuTyPath* $\tilde{\mathcal{P}}$ *is represented as* $\tilde{\mathcal{P}}^{R_1 \circ \cdots \circ R_n} = \mathcal{TS}(o_1) \xrightarrow{R_1} \cdots \xrightarrow{R_n} \mathcal{TS}(o_{n+1})$*, where* $\mathcal{TS}(o_i)$ *represents the type set of object* o_i *at position* i *in* $\tilde{\mathcal{P}}$*. Different from* $\mathcal{T}(o_i)$ *in meta path,* $\mathcal{TS}(o_i)$ *can correspond to multiple entity types. When the cardinality of every* $\mathcal{TS}(o_i)$ *is 1, the MuTyPath is equal to meta path.*

Let's give an example in Fig. 1 to show the difference between MuTyPath and meta path. As the fact that Steve Jobs established Apple, meta path can not normally show their relation because entities have multiple types, like Jobs belonging to Executive, Person, and Inventor. Or meta path can only describe this fact

as "$Obj \xrightarrow{Created} Obj$" through ignoring the node types, which may cause semantic ambiguity. However, we can use MuTyPath "$\{Person, Inventor, Excusive\} \xrightarrow{Created} \{Company, Manufacturer\}$" to describe the fact more exactly.

3 The Method Description

In order to expand entity set in KG efficiently and accurately, we propose an algorithm named ***Co*ncatenated *M*eta Path based *E*ntity *S*et *E*xpansion** (CoMeSE) to capture semantic relations between seeds. Firstly, for reducing space and time, we design an efficient concatenated meta path generation method. Secondly, to handle ambiguity of meta path, we extract multi-type-constrained meta paths and design a novel similarity measure MuTySim. Thirdly, due to the lack of negative cases, we design a heuristic weight strategy and PU learning method to assign the importance of extracted paths for ESE model.

3.1 Random Walk based Concatenated Meta Path Generation Method

Meta path is a kind of effective feature to capture semantic relationship among nodes. In traditional HIN with simple schema, meta path is usually predefined, while it is hard in KG owing to its massive types of objects and relations. Thus, we propose an algorithm named **R**andom **W**alk based **C**oncatenated Meta **P**ath generation method (RWCP) to quickly and automatically generate meta paths.

A naive method to generate meta path in KG is, using a walker to wander with one-directional random walk to find meta paths between seed pairs. MP_ESE [14] adopts this idea. However, it has two disadvantages. Firstly, it has a huge space and time cost. If m is the average number of neighbors of a node, discovering n-length paths should visit m^n nodes. Secondly, many paths are duplicately visited and the visiting information is not reused, which makes it inefficient. For saving time and space, we can use a bi-directional random walk where two walkers wander from two sides respectively and meet at an intersection node while wandering. The searching space would reduce to $m^{n/2}$. For reusing visited path, we record the wandering information of walkers, and the walkers can continue to wander or decide to concatenate existing paths.

Specifically, given the seed set, RWCP randomly walks with a half or less of the maximum path length to obtain a set of paths for seeds, and then concatenates different paths with visiting information to get useful meta paths. In fact, RWCP is a repeating process of meta path concatenation and extension. For explaining RWCP clearly, we give some basic structural definitions as follows.

Definition 2. ***Recorder*** *is a structure to record visiting information. It includes: the meta path passed by, a series of entity pairs generated along the path and corresponding similarity values, entity lists between the entity pairs along the path, an arriving entity set and the score $\mathcal{S}co$ of meta path defined later.*

Definition 3. *The score of meta path,* ***Sco****, is designed to indicate the importance of the path to seed set. The value of* $\mathcal{S}co$ *means the priority to handle.*

$$\mathcal{S}co(\mathcal{P}) = \sum_{s} \frac{1}{K} \sum_{t} \sigma(s, t|\mathcal{P}), \tag{1}$$

where s *and* t *are source and arriving node respectively on meta path* $\mathcal{P}$*,* K *as the number of arriving nodes from* s*, and* $\sigma(s, t|\mathcal{P})$ *is the similarity value based on* $\mathcal{P}$*, which is calculated by* MuTySim for meta path *introduced in Sect. 3.2. Moreover,* ***AvgS*** *is an average value of* $\mathcal{S}co$ *of two meta paths which are pending to be concatenated. The* $\mathcal{A}vg\mathcal{S}$ *shows priority of Recorder pair for path concatenation.*

Besides, some assistant sets are needed. We use Recorder Set ($\mathcal{RS}$) to store Recorders, Extension Backlog ($\mathcal{EB}$) to record serial numbers and $\mathcal{S}co$ of Recorders which would be extended, and Concatenation Backlog ($\mathcal{CB}$) to record serial number pairs and $\mathcal{A}vg\mathcal{S}$ of the Recorder pairs which would be concatenated.

Definition 4. η *is a threshold determining whether adding the new generated Recorder into* $\mathcal{RS}$*, for excluding unimportant paths.*

$$\eta = \epsilon \cdot (|\mathcal{SS}| + |\mathcal{PS}|), \tag{2}$$

where ϵ *is a limited coefficient, and* $|\mathcal{SS}|$ *is the cardinality of seed set* $\mathcal{SS}$*,* $|\mathcal{PS}|$ *is the number of meta paths chosen. So* η *is dynamically increasing according to the number of chosen meta paths to converge faster to terminate the algorithm.*

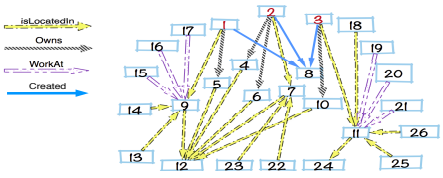


Fig. 2. Subgraph example for RWCP.

Thus, as introduced above, we mainly use $\mathcal{S}co$ and $\mathcal{A}vg\mathcal{S}$ together to determine action in RWCP. In detail, comparing $\mathcal{A}vg\mathcal{S}$ and $\mathcal{S}co$ values, if there are $\mathcal{A}vg\mathcal{S}$ greater than all $\mathcal{S}co$ values in $\mathcal{EB}$, we will do path concatenation for these Recorder pairs. When all $\mathcal{A}vg\mathcal{S}$ in $\mathcal{CB}$ are smaller than the largest $\mathcal{S}co$, we extend the Recorder with largest $\mathcal{S}co$ to generate new Recorder pairs. Path concatenation and Recorder extension take place alternately. Fig. 2 is a simple network and seed set is {1,2,3} which means the seed pairs are {(1,2),(1,3),(2,1),(2,3),(3,1),(3,2)}. Fig. 3 describes how RWCP algorithm works in the sample shown in Fig. 2.

First of all, RWCP builds the initial Recorder $\mathcal{R}ec_1$ and puts it into $\mathcal{RS}$, (serial number SN_1, score $\mathcal{S}co_1$) into $\mathcal{EB}$ in Step A. Then, RWCP is in a loop

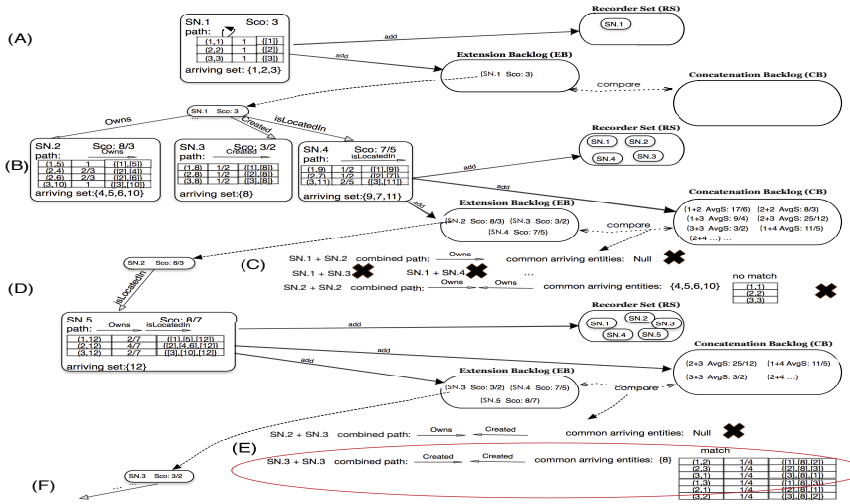


Fig. 3. A sample of RWCP process.

process of path concatenation and extension. When $\mathcal{CB}$ is empty and $\mathcal{EB}$ contains Recorder, like in Step B, or all $\mathcal{A}vg\mathcal{S}$ in $\mathcal{CB}$ are smaller than the largest $\mathcal{S}co$ in $\mathcal{EB}$, like in Step D, RWCP pops the Recorder with the max $\mathcal{S}co$ from $\mathcal{EB}$ to extend outwards. When extending, like in Step B, the arriving nodes in $\mathcal{R}ec_1$ move one step forward and RWCP generates new Recorders with the serial numbers $SN.2-4$ based on different paths passing by, and $(SN.1)$ is removed from $\mathcal{EB}$. After that, RWCP judges if $\mathcal{S}co$ of each new generated Recorder is larger than the minimum threshold, η. If yes, put it into $\mathcal{RS}$ and $\mathcal{EB}$. For each Recorder in $\mathcal{RS}$, pair it with the new Recorder and put this pair with its $\mathcal{A}vg\mathcal{S}$ into $\mathcal{CB}$. If the largest $\mathcal{A}vg\mathcal{S}$ of Recorder pairs is larger than all $\mathcal{S}co$ in $\mathcal{EB}$, like in the Step C and E, RWCP will get the Recorder pair with largest $\mathcal{A}vg\mathcal{S}$ out of $\mathcal{CB}$ and concatenate paths. For example, in Step E, handling the Recorder pair $(SN.3, SN.3)$, RWCP concatenates two meta paths of them as "$\xrightarrow{Created}\xleftarrow{Created}$", and judges whether two arriving entity sets have nodes in common or not. If yes, generate new entity pairs based on the common node set $\{8\}$. If the new entity pairs match to seed pairs, the concatenated path would be chosen as a path feature $\mathcal{P}$. Finally, if $\mathcal{CB}$ and $\mathcal{EB}$ are all empty, RWCP would be terminated.

3.2 Multi-Type-Constrained Meta Path Extraction and Similarity Calculation

Traditional meta path may not well capture subtle semantics in KG, because the type of entities at ends of a relation may be non-unique and entity may be subordinate to multiple types. To avoid ambiguity problem, we design MuTyPath in Def. 1 to make the explored relationships more accurate and a measure named ***Mu**lti-**Ty**pe-Constrained Meta Path-based **Sim**ilarity measure* (MuTySim) to

compute similarity of entity pairs in MuTyPath. The similarity value vectors based on MuTyPaths can be used as features for ESE model.

Extraction of Multi-Type-Constrained Meta Path Applying RWCP algorithm, we get the meta paths with relations only and a series of visited entity lists for seed pairs along the paths. These entity lists are the path instances. With the lists, we can change meta paths to MuTyPaths for more precise semantics.

We design an extraction strategy for MuTyPath. Given an n-length meta path $\mathcal{P}$ and a list of instances $\{p_1, \cdots, p_m\}$, every position o_i of $\mathcal{P}$ has an object set $\{a_{1i}, \cdots, a_{mi}\}$. We check the types of each entity a_{ji} one by one and judge whether current entity type set $\mathcal{TS}(a_{ji})$ has intersection with existing common type sets $\mathcal{TS}_k^{'}(o_i)$(initial common type set $\mathcal{TS}_1^{'}(o_i) = \mathcal{TS}(a_{1i})$). If yes, update with $\mathcal{TS}_k^{'}(o_i) = \mathcal{TS}_k^{'}(o_i) \bigcap \mathcal{TS}(a_{ji})$. Otherwise, create another common type set to store its types. Then we will have one or more common type sets $\mathcal{CTS}_i = \{\mathcal{TS}_1^{'}, \cdots\}$ at every position o_i. After that, we use Cartesian Product, $\mathcal{CTS}_1 \times \cdots \times \mathcal{CTS}_n$, to get multiple common type set combinations, and combine them with the original meta path to finally form one or more MuTyPaths.

Multi-Type-Constrained Meta Path-based Similarity Measure Based on the common type set of seeds obtained from selected MuTyPaths, we can get candidates sharing the same common types. To find out the relationship between seeds and candidates, we should re-calculate the similarity of each seed-candidate pair and seed-seed pair along MuTyPaths. Here we propose a novel ***Mu**lti-**Ty**pe-Constrained Meta Path-based **Sim**ilarity measure* (MuTySim).

MuTySim has the following advantages. Firstly, MuTySim supports meeting at any node along the path for RWCP. Current similarity measures have fixed random walk direction and measurement. Secondly, MuTySim considers both conditions of two ends between a link, while existing measures do not. Moreover, MuTySim considers multi-type constraint of MuTyPath.

Given a MuTyPath $\tilde{\mathcal{P}}^{o_1} = \mathcal{TS}(o_1)$, where $\mathcal{TS}(o_1)$ is the common type set at position o_1 constrained by MuTyPath, the similarity of object s and itself is:

$$\sigma(s, s|\tilde{\mathcal{P}}^{o_1}) = 1 - \alpha \cdot \frac{|\mathcal{TS}(o_1) - \mathcal{TS}(s)|)}{|\mathcal{TS}(o_1)|}, \tag{3}$$

where $\mathcal{TS}(s)$ represents the type set of s and α is the impact factor of type set importance degree over similarity with range of [0,1]. The larger α is, the more constraints of types will be, and the clearer the semantic will be.

Given two objects s and t, and a MuTyPath $\tilde{\mathcal{P}}^{o_1 \cdots o_n} = \mathcal{TS}(o_1) \xrightarrow{R_1} \cdots \xrightarrow{R_{n-1}} \mathcal{TS}(o_n)$, considering both conditions of two ends of a relation and type set constraint, the MuTySim similarity of two objects along the path $\tilde{\mathcal{P}}$ is defined as:

$$\sigma(s, t|\tilde{\mathcal{P}}^{o_1 \cdots o_n}) = \sigma(t, t|\tilde{\mathcal{P}}^{o_n}) \sum_{x \in I(t|R_{n-1})} \frac{2|O(x|R_{n-1}) \bigcap I(t|R_{n-1})|}{|O(x|R_{n-1})| + |I(t|R_{n-1})|} \cdot \sigma(s, x|\tilde{\mathcal{P}}^{o_1 \cdots o_{n-1}}), \tag{4}$$

where $O(x|R_{n-1})$ is the out-neighbors of object x based on relation R_{n-1}, and $I(t|R_{n-1})$ is the in-neighbors of t based on R_{n-1}.

When two objects s and t are connected by the concatenated MuTyPath $\tilde{\mathcal{P}}^{o_1 \cdots o_n}$ and meet at any position along the MuTyPath, the MuTySim similarity values of two objects are equal:

$$\sigma(s, t|\tilde{\mathcal{P}}^{o_1 \cdots o_n}) = \sum_{x \in C_j} \sigma(s, x|\tilde{\mathcal{P}}^{o_1 \cdots o_j}) \cdot \sigma(x, t|\tilde{\mathcal{P}}^{o_j \cdots o_n}), \tag{5}$$

where o_j is the meeting position and C_j is the object set at position o_j.

In particular, when α equals to 0, the entity type will not be considered and MuTySim can measure similarity based on traditional meta path, so that it can be seen as *MuTySim for meta path* which is useful for random walking in opposite way and meeting at any position. Therefore, applying *MuTySim for meta path* in RWCP can discover meta path feature more accurately.

3.3 Weight Learning of Meta Paths

MuTyPaths should be combined effectively based on their importances to constitute ESE model. ESE is in fact to build a ranking model that calculates the probability of a candidate in the expansion set, and take the top $\mathcal{K}$ entities as the expansion result. The formula of the ranking model can be defined as follows:

$$CSSim(c, \mathcal{SS}) = \sum_{s \in \mathcal{SS}} CSim(c, s), \tag{6}$$

where c is the candidate node, and $\mathcal{SS}$ is the seed set, and $CSim(c, s)$ represents the matching probability of candidate node c and seed s.

Whether the candidate matches the seed can be seen as a classification problem. We can regard the MuTySim similarity value vector of entity pair on selected MuTyPaths as a feature for classification. Besides, positive data are the seed pairs, while the pairs of candidates and seeds are all unlabeled data. However, there are no effective methods for the automatic selection of the negative data. Without the negative data, we can not use traditional supervised learning method to do classification. To solve the problem, we come up with two weight learning solutions: the heuristic method and PU learning method.

Weight Learning with Heuristic Method It is easy to understand that the meta path connecting more seed pairs will be more important, and the path with larger similarity value indicates closer relationship. Depending on the importance degree of each MuTyPath connecting the seed set, we calculate the corresponding weight for each path based on the similarity information of seed pairs and linearly combine the weight with similarity value together to form the matching probability equation for the candidates.

$$CSim(c, s) = \sum_{\tilde{\mathcal{P}}_i \in \tilde{\mathcal{PS}}} \varpi_i \cdot \sigma(c, s|\tilde{\mathcal{P}}_i), \tag{7}$$

$$\varpi_i = \frac{f(\tilde{\mathcal{P}}_i) \cdot \sum_{s_m, s_n \in \mathcal{SS}, m \neq n} \sigma(s_m, s_n | \tilde{\mathcal{P}}_i)}{\sum_{\tilde{\mathcal{P}}_t \in \tilde{\mathcal{PS}}} f(\tilde{\mathcal{P}}_t) \cdot \sum_{s_m, s_n \in \mathcal{SS}, m \neq n} \sigma(s_m, s_n | \tilde{\mathcal{P}}_t)}, \tag{8}$$

where $\tilde{\mathcal{PS}}$ is the MuTyPath set generated in Set. 3.2 and ϖ_i is the weight for path $\tilde{\mathcal{P}}_t$. $f(\tilde{\mathcal{P}}_i) = \xi^{|zeros(\tilde{\mathcal{P}}_i)|}$ is a penalty function for having seed pairs not connected by $\tilde{\mathcal{P}}_i$. So $|zeros(\tilde{\mathcal{P}}_i)|$ is the number of similarity values of seed pairs as 0 based on path $\tilde{\mathcal{P}}_i$ and ξ is the penalty constant for it as 1/2. $\sigma(s_m, s_n | \tilde{\mathcal{P}}_i)$ is the MuTySim value in seed pair (s_m, s_n) based on $\tilde{\mathcal{P}}_i$.

Weight Learning with PU Learning PU learning is used to train classifier with positive and unlabeled training data, which is suitable for ESE. In our method, the seed pairs can be seen as positive data while the candidate-seed pairs as unlabeled data. We adopt a novel PU learning method proposed by Elkan et al [3], which could train a traditional classifier to distinguish the positive and unlabeled examples and get a better result than existing PU learning methods. The main idea is to detect the reliable negative samples and then use the positive and negative cases to do classification training. This method is very flexible to choose any traditional classifier for PU learning, so that we can use suitable classifier to form the matching model for candidate nodes based on the exact situation.

4 Experiment

In order to verify the superiority of CoMeSE for entity set expansion in KG, we validate the effectiveness of CoMeSE with a series of experiments.

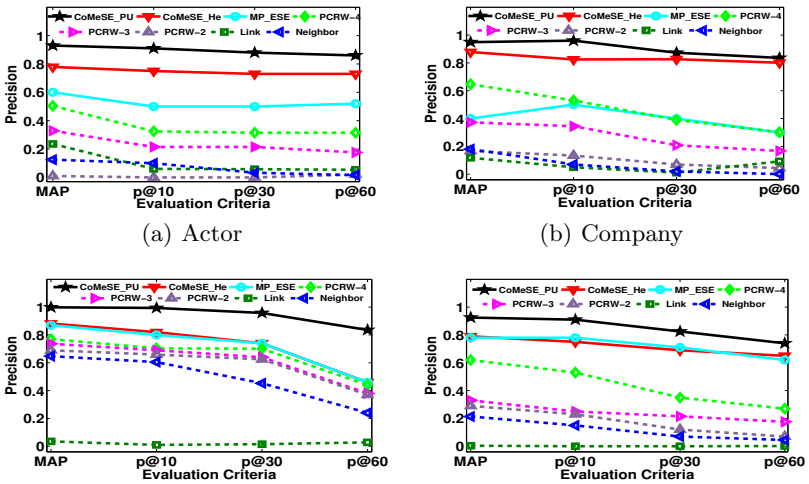
4.1 Experiment Settings

Dataset We use KG Yago [11] to conduct relevant experiments. In experiments, we adopt "*COREFact*" and "*yagoSimpleTypes*" parts of this dataset, which contain 4.4 million facts, 35 relationships and 1.3 million entities of 3455 types.

Four ESE tasks are chosen to evaluate the performance of CoMeSE. These tasks are as follows: (1) in the *Actor* task, the seeds are actors who won Emmy Award, and their spouses are also actors; (2) in the *Company* task, the seeds are companies which own a channel in America; (3) in the *Writer* task, the seeds are writers which are graduated from the universities in New York; (4) in the *Movie* task, the seeds are movies, and their director won National Film Award. The real numbers of instances in these tasks are 193, 76, 60, and 653, respectively.

Criteria We use precision-at-k (p@k) and Mean Average Precision (MAP) to evaluate performance. Here, they are p@10, p@30, and p@60. And MAP is the mean of the Average Precision (AP) of the p@10, p@30, and p@60.

Compared Methods We denote the CoMeSE with heuristic and PU learning method, as "$CoMeSE\_He$" and "$CoMeSE\_PU$", respectively. And we use four baselines as follows: (1) Link. According to the pattern-based methods [8], it only considers 1-hop link of an entity, denoted as Link. (2) Neighbor. Inspired by QBEES [6], it considers 1-hop link and 1-hop entity as features, called Neighbor. (3) PCRW. With path constrained random walk based similarity measure PCRW[5], it employs different max path lengths to connect objects. The PCRW within length-2, 3, 4 paths are denoted as PCRW-2, PCRW-3, PCRW-4, respectively. (4) MP_ESE [14]. The KG-based ESE method finds meta paths by an one-directed generated method and uses a heuristic weight learning method.



(a) Actor (b) Company

(c) Writer (d) Movie

Fig. 4. The results of entity set expansion.

In CoMeSE, we set ϵ as 10^{-6} in Eq. 2, α as 1 in Eq. 3 for calculating similarity based on MuTyPath, based on parameter study. The max length of path is set to be 4 since meta paths with length more than 4 are almost irrelevant [13].

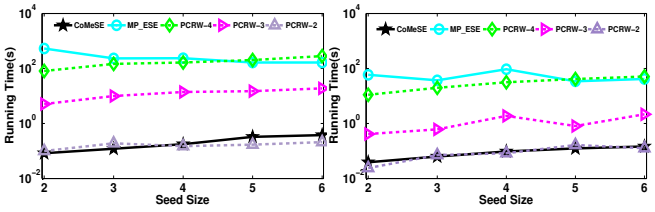
4.2 Effectiveness Experiments

In this section, we validate the effectiveness of CoMeSE in 4 tasks introduced above. For each task, we randomly take three seeds from the instance set to conduct an experiment. We run 20 times and report the average results.

We illustrate the experiment results in Fig. 4. Firstly, the meta path based methods, CoMeSE and MP_ESE, almost have higher accuracy than other methods in all tasks. That is because, in KG, path feature can effectively embody intrinsic relations among seeds. Secondly, the results of CoMeSE_He and CoMeSE_PU

are better than MP_ESE in almost all tasks. Traditional meta path fails to capture exact meanings owing to the uncertain types connected by relations, while MuTyPath used in CoMeSE can subtly distinguishes these paths. For example, in the *Company* task, the first and fourth found MuTyPaths in Table **??** would be both described as the same meta path "$\xrightarrow{Owns}\xrightarrow{islocatedIn}\xleftarrow{islocatedIn}\xleftarrow{Owns}$", but they have different semantics. Thirdly, CoMeSE_PU performs better than CoMeSE_He in all tasks. The reason is that PU learning judges path features more precisely than heuristic method. In all, CoMeSE performs the best.

4.3 Efficiency Study



(a) Actor (b) Movie

Fig. 5. Running times of different methods.

Here we validate the efficiency of finding meta paths under different seed size. We conduct experiments by varying the seed size from 2 to 6 on the *Actor* and *Movie* tasks. For each seed size, we randomly select the same-scale seeds to run 10 times. We show the average running time in Fig. 5. It is obvious that CoMeSE almost has the smallest running time in both tasks. PCRW-2 only explores 1-hop and 2-hops paths, so it has small running time. But short path exploration also gets bad performance, shown in Fig. 4. We think the bi-directional random walk strategy and the reuse of visited paths make the CoMeSE significant efficiency improvement. In addition, the running time of CoMeSE and PCRW methods near linearly increase with the increment of the seed size. It is reasonable, since these methods need to discover more paths to connect more seed pairs. Some strategies in MP_ESE make it less affected by seed size. In all, CoMeSE has obviously high efficiency of meta path discovery.

5 Conclusion

In this paper, we study the problem of entity set expansion in KG. We model KG as an HIN and propose a novel *Co*ncatenated *M*eta Path based *E*ntity *S*et *E*xpansion Method called CoMeSE, which proposes a random walk based concatenated meta path generation method to detect meta paths, a multi-type-constrained meta path algorithm to subtle capture path semantics, and uses two

path weight learning methods to determine the importance of paths. Extensive experiments on Yago validate the performance of CoMeSE.

Acknowledgement This work is supported in part by the National Natural Science Foundation of China (No. 61772082, 61375058), The National Key Research and Development Program of China (2017YFB0803304).

References

1. H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *KDD*, pages 875–883, 2008.
2. W. W. Cohen and S. Sarawagi. Exploiting dictionaries in named entity extraction: combining semi-markov extraction processes and data integration methods. In *KDD*, pages 89–98. ACM, 2004.
3. C. Elkan and K. Noto. Learning classifiers from only positive and unlabeled data. In *SIGKDD*, pages 213–220, 2008.
4. H. Jaiwei. Mining heterogeneous information networks: the next frontier. In *SIGKDD*, pages 2–3, 2012.
5. N. Lao and W. W. Cohen. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67, 2010.
6. S. Metzger, R. Schenkel, and M. Sydow. Qbees: query by entity examples. In *CIKM*, pages 1829–1832. ACM, 2013.
7. Z. Qi, K. Liu, and J. Zhao. Choosing better seeds for entity set expansion by leveraging wikipedia semantic knowledge. In *CCPR*, pages 655–662, 2012.
8. B. Shi, Z. Zhang, L. Sun, and X. Han. A probabilistic co-bootstrapping method for entity set expansion. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 2280–2290, 2014.
9. C. Shi, Y. Li, J. Zhang, Y. Sun, and P. S. Yu. A survey of heterogeneous information network analysis. *IEEE Trans. Knowl. Data Eng.*, 29(1):17–37, 2017.
10. A. Singhal. Introducing the knowledge graph: things, not strings. *Official google blog*, 2012.
11. F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*, pages 697–706, 2007.
12. Y. Sun, B. Norick, J. Han, X. Yan, P. S. Yu, and X. Yu. Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. In *KDD*, pages 1348–1356, 2012.
13. C. Wang, Y. Song, H. Li, M. Zhang, and J. Han. Knowsim: A document similarity measure on structured heterogeneous information networks. In *ICDM*, pages 1015–1020, 2015.
14. Y. Zheng, C. Shi, X. Cao, X. Li, and B. Wu. Entity set expansion with meta path in knowledge graph. In *PAKDD*, pages 317–329, 2017.