Representation Learning of Taxonomies for Taxonomy Matching

Hailun Lin[1(✉)], Yong Liu[1], Peng Zhang[1], and Jianwu Wang[2]

[1] Institute of Information Engineering, Chinese Academy of Science, Beijing, China
linhailun@iie.ac.cn

[2] Department of Information Systems, University of Maryland, Baltimore County, Baltimore, USA

Abstract. Taxonomy matching aims to discover categories alignments between two taxonomies, which is an important operation of knowledge sharing task to benefit many applications. The existing methods for taxonomy matching mostly depend on string lexical features and domain-specific information. In this paper, we consider the method of representation learning of taxonomies, which projects categories and relationships into low-dimensional vector spaces. We propose a method to takes advantages of category hierarchies and siblings, which exploits a low-dimensional semantic space to modeling categories relations by translating operations in the semantic space. We take advantage of maximum weight matching problem on bipartite graphs to model taxonomy matching problem, which runs in polynomial time to generate optimal categories alignments for two taxonomies in a global manner. Experimental results on OAEI benchmark datasets show that our method significantly outperforms the baseline methods in taxonomy matching.

Keywords: Taxonomy matching · Representation learning · Category embedding · Relation embedding · Maximum weight matching

1 Introduction

Taxonomy is used to annotate entity semantic information in knowledge base, which contains a hierarchy of categories. Categories in a taxonomy can be described as multi-relational data and represented as triples (h_c, r, t_c), where h_c denotes the head category, h_t denotes the tail category, r expresses the direct relationship between h_c and t_c. r has three kinds of value: *subclass*, *superclass* and *sibling*. Specifically, if the value of r is *subclass*, it indicates that h_t is the parent of h_c; if the value of r is *superclass*, it indicates that h_t is the child of h_c; if the value of r is *sibling*, it indicates that h_t would be the sibling of h_c. As is known to all, different taxonomies sometimes contain both overlapping and complementing data. In order to implement knowledge sharing from two

J. M. F. Rodrigues et al. (Eds.): ICCS 2019, LNCS 11536, pp. 383–397, 2019.
https://doi.org/10.1007/978-3-030-22734-0_28

taxonomies, we study the problem of taxonomy matching to discover categories alignments between them.

For taxonomy matching, the key step is to calculate the category pair relevance between two taxonomies. After all the category pairs relevant scores have been calculated, we can obtain the most relevant category pairs between two taxonomies. In recent years, taxonomy matching has received a lot of research interests, and many approaches have been proposed (e.g., [4,9,14–16]). These works mostly depend on string lexical features or domain-specific information to predict the relevance score between categories. Although there are many studies on taxonomy matching, most of those approaches have been demonstrated to achieve good performance only on fairly domain-specific taxonomies [1,13].

Therefore, we present a representation learning based taxonomy matching approach, which exploits a unified semantic model where we can learn to place categories, supercategories, and siblings as points in a hypothetical common semantic space, i.e., a continuous low-dimensional vector space. The category representation vector can significantly promote taxonomy matching. This method follows the assumption in TransE [3] (designed for learning knowledge graph representations), modeling categories relationships by translating operations between two categories in the semantic space.

Our methods works in three stages: it firstly embeds taxonomies including both categories and relations into a continuous low-dimensional vector space. Secondly, it creates a weighted bipartite graph to model the candidate relevant category pairs between two taxonomies. Thirdly, it performs a maximum weight matching algorithm to generate an optimal matching for two taxonomies in a global manner. Key aspects of our method are: (1) it automatically learns category and relation feature representations in semantic space to calculate the relevance between categories, without external data resources except taxonomies themselves; (2) it proposes category matching in a global manner, by finding matching with maximum weight in a bipartite graph. In general, the main contribution of this paper is three-fold:

- We show a multi-relational data modeling formulation for taxonomy matching that learns a unified semantic space for categories, supercategories and siblings, while drawing relations between them.
- We present a maximum weight matching algorithm for matching taxonomies, which can obtain a global optimal matchings between two taxonomies.
- We show from the experiments that the multi-relational data modeling with the maximum weight matching algorithm helps improve taxonomy matching accuracy.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 formulates the problem of taxonomy matching and proposes TransC framework. Section 4 introduces our model for taxonomy matching, and discusses its implementation. Section 5 introduces the experimental results. Finally, the paper is concluded in Sect. 6.

2 Related Work

The problem of taxonomy matching has its roots in the problems of identifying duplicate entities, which is also known as record linkage, duplicate detection, or coreference resolution. There are a lot of research work has been proposed for taxonomy matching (e.g., [2,5,7,9,12,14,16]). In this paper, we try to analyze the studies on this problem from the perspective of their measures for calculating category relevance [5].

Specifically, we simply classify the studies into the following six categories. (1) Lexicon-based measure: They perform taxonomy matching task according to the mention forms (i.e., words representation) of the categories in taxonomies. (2) Semantic-based measure: They adopt semantic dictionaries to complete taxonomy matching according to the meaning of the words. (3) Structure-based measure: They adopt category hierarchical information, including supercategories and subcategories. (4) Instance-based measures: They use the instances of categories. (5) Context-based measure: The adopt the descriptive text of categories. (6) Hybrid-based measure: They adopt various combination of different types of information, i.e., lexicon, semantic, structure, instance and context.

PARIS [16] adopted instance based measure for taxonomy matching. It considered that the category structure in one taxonomy may be more fined-grained than the category in the other taxonomy, so it aimed to find subclass matching relationships between two taxonomies. RiMOM [12] exploited a dynamic multistrategy for finding categories alignments, which automatically combined the measures based on two estimated factors, i.e., the lexicon similarity factor and the structure similarity factor. RiMOM adopted similarity flooding technique on a relationship graph between two taxonomies to enhance the structural information contributing to taxonomy matching. ServOMap [14], designed for biomedical ontologies, took advantage of lexicon and context based measure to calculate the relevance scores of categories. It exploited an inverted index used in information retrieval to reduce the number of candidate categories to consider. LogMap [7], also designed for biomedical ontologies, combined lexicon and semantic as well as structure measures to aligning categories between taxonomies.

In addition, Chen et al. [5] proposed FFCA technique, a combination of the fuzzy theory and formal concept analysis (FCA), to match taxonomies with the same domain. FFCA enriched each category from the source taxonomy by information obtained form WordNet. Demidova et al. presented a Markov Logic Network (MLN) based semi-supervised method for matching task [6]. They mentioned serval heuristic rules based on first-order logic to capture the similar semantic elements. Lee et al. [11] presented a Monte Carlo algorithm for finding greedy cuts to entity resolution problem. They adopted a combination of properties and instances based measures. SiGMa [9] adopted a simple greedy matching algorithm with a combination of lexicon and structure measures, which finds aligned categories in a greedy local search manner. The greedy based method can be seen as an efficient method to the task of large-scale taxonomy matching. However, due to its greedy nature, it can not correct previous mistakes in

making decisions. Therefore, the greedy based method could not guarantee obtaining a global optimal matching for two taxonomies.

Based on the above analysis, the studies on taxonomy matching either focus on specific domains, or aim at providing a general way across various domains. Furthermore, we can see that most of existing studies employ the combinational strategies. Extensive experiments also show that the combination method outperforms the single strategy based method [9,12]. However, they are mostly capturing the linguistic features and structural features to predict the relevance score between categories, there is no single dominant taxonomy matcher that performs the best, regardless of its application domain [1].

3 Taxonomy Matching

In this section, we will study the problem of automatically taxonomy matching. For this purpose, we will firstly give some notations and formulate the problem of taxonomy matching in Sect. 3.1. Subsequently, the overall framework of TransC will be introduced in Sect. 3.2.

3.1 Notations and Problem Formulation

Suchanel et al. defined a taxonomy as a set of a formal collection of knowledge, including categories, relations, and the instances with their assertions [16]. In this paper, we describe a taxonomy as multi-relational data with numerous triple facts $T = \{(h_c, r, t_c)\}$. Given a triple $(h_c, r, t_c) \in T$, where $h_c, t_c \in C$ denote categories and $r \in R$ denotes the relationship between h_c, t_c. C is categories set and R is relation set, where $R = \{subclass, superclass, sibling\}$. Each category $c \in C$ contains an attributes set A_c. Each category and relation embedding in the hypothetical common semantic space takes value in $\mathbb{R}^k$.

Structure-Based Embeddings: $\mathbf{h_c^s}$ and $\mathbf{t_c^s}$ are the embeddings of category h_c, t_c, which are learned from the hierarchical structure of taxonomies. These embeddings are learned from translation-based method TransE [3].

Attribute-Based Embeddings: $\mathbf{h_c^a}$ and $\mathbf{t_c^a}$ are the attribute-based embeddings of category h_c, t_c, which are learned based on category attributes. In the following, we will elaborate on an encoder to learn attribute-based embeddings for taxonomy categories.

We note that, in a taxonomy, the categories set C is global, which means that some categories maybe identical across different taxonomies. Moreover, in addition to the equivalent ($\equiv$) relationship between two categories c and $c^{'}$, the relationship between c and $c^{'}$ could be subcategory relationship $c \subseteq c^{'}$ or supercategory relationship $c \supseteq c^{'}$. As the subcategory relationship or supercategory relationship can be inferred by equivalent relationship between categories, we aim to find out whether one category c of one taxonomy is equivalent to another category $c^{'}$ of another taxonomy. Since the set C is global in a taxonomy, we consider the one-to-one matching of categories between two taxonomies.

Definition 1. Given two taxonomies $T = \{(h_c, r, t_c)\}$ and $T^{'} = \{(h_c^{'}, r^{'}, t_c^{'})\}$, the goal of the taxonomy matching is to obtain a one-to-one (1-1) equivalent matching M from the categories set C of T to the categories set $C^{'}$ of $T^{'}$, which contains all semantically equivalent categories between two taxonomies.

3.2 The TransC Framework

Based on the problem definition, we propose a method called TransC, to address the task of taxonomy matching using two modules as follows:

Taxonomy Representation Learning. To supplement lexical representation in measuring category relevance scores, this module exploits a unified semantic model where we can learn to place categories, supercategories, and siblings as points (or vectors) in a hypothetical common semantic space.

Taxonomy Matching Generation. Based on taxonomy representations, this module exploits a weighted bipartite graph to model the candidate relevant category pairs between two taxonomies, and performs a maximum weight matching algorithm to generate an optimal matching for two taxonomies.

In the following sections, we will introduce those modules in details.

4 Methodology

In this section, we introduce our method that obtains the categories alignments for taxonomies. In what follows, we first introduce how to learn the representations of taxonomies, and then elaborate on the process for finding an optimal matching for two taxonomies based on the representations.

4.1 Taxonomy Representation Learning

To exploit both triple facts $(h_c, r, t_c) \in T$ and category attributes, we follow a representation learning method DKRL for knowledge graphs [17], and propose structure-based category embeddings and attribute-based category embeddings. These embeddings adopt energy-based model, which learns category representation vectors in low-dimensional vector space. The structure-based category embeddings are used to capture information in triple facts of taxonomies, and the attribute-based category embeddings are used to capture meta information in category attributes. We use the same embedding vector space to learn the two types of category representations. The energy function of our method is then defined as follows:

$$F(h_c, r, t_c) = F_S(h_c, r, t_c) + F_A(h_c, r, t_c), \tag{1}$$

where $F_S(h_c, r, t_c) = \|\mathbf{h_c^s} + \mathbf{r} - \mathbf{t_c^s}\|$ is the part of structure-based category embedding energy function, $F_A(h_c, r, t_c)$ is the part of attribute-based category embedding energy function. In this paper, we define $F_A(h_c, r, t_c)$ as follows:

$$F_A(h_c, r, t_c) = F_{AA}(h_c, r, t_c) + F_{AS}(h_c, r, t_c) + F_{SA}(h_c, r, t_c) \tag{2}$$

where $F_{AA}(h_c, r, t_c) = \|\mathbf{h_c^a} + \mathbf{r} - \mathbf{t_c^a}\|$ in which $\mathbf{h_c^a}$ and $\mathbf{t_c^a}$ are the attribute-based embeddings of category h_c, t_c. In this paper, we also define $F_{AS}(h_c, r, t_c) = \|\mathbf{h_c^a} + \mathbf{r} - \mathbf{t_c^s}\|$ and $F_{SA}(h_c, r, t_c) = \|\mathbf{h_c^s} + \mathbf{r} - \mathbf{t_c^a}\|$.

In the following subsection, we present a continuous bag-of-words encoder to build attribute-based category representation.

Continuous Bag-of-Words Encoder. For each category, a set of attributes are used to denote the meta information of the category. We assume that if categories are similar, their attributes should be similar. In the encoder, we take the words in the attributes for each category as input. Firstly, we sum up the words representation vectors to obtain the attribute representation vector. Secondly, we sum up the representation vectors of attributes to obtain the category representation vector:

$$\mathbf{c^a} = \mathbf{a}_1 + \cdots + \mathbf{a}_k, \tag{3}$$

where $\mathbf{a}_i$ is the i-th attribute representation vector belonging to the attributes set A_c of category c; $\mathbf{a}_i = \mathbf{x}_1 + \cdots + \mathbf{x}_m$, where $\mathbf{x}_j$ is the j-th word representation vector belonging to the words set of attribute $a \in A_c$. In this paper, $\mathbf{x}_j$ can be obtained by Word2Vec. $\mathbf{c^a}$ will be used to minimize F_A. The encoder framework is illustrated in Fig. 1.

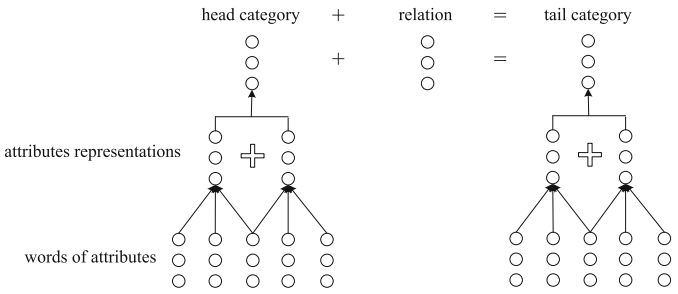

Fig. 1. The CBOW encoder

Training. Now we introduce how to learning category embeddings with our proposed models. Given a training set $S \subseteq T$ of triple facts (h_c, r, t_c), we adopt a margin-based ranking criterion as objective:

$$\mathcal{L} = \sum_{(h_c, r, t_c) \in S} \sum_{(h_c^{'}, r^{'}, t_c^{'}) \in T^{'}} \max(0, \gamma + F(h_c, r, t_c) - F(h_c^{'}, r^{'}, t_c^{'})) \tag{4}$$

where γ denotes a margin hyperparameter, $\gamma > 0$; $F(h_c, r, t_c)$ denote the dissimilarity score function between $\mathbf{h_c} + \mathbf{r}$ and $\mathbf{t_c}$, which we take to be either the L_1-norm or the L_2-norm; $S^{'}$ is the negative sampling set, generated according to Eq. 5. As we define two representation types for categories, h_c and t_c in the Eq. 4 could be either of these two types representations.

$$S^{'} = \{(h_c^{'}, r, t_c) | h_c^{'} \in C\} \cup \{(h_c, r, t_c^{'}) | t_c^{'} \in C\} \tag{5}$$

More specifically, during constructing corrupted triples, we follow the method described in [17], which sets different probabilities for replacing the head or tail entity for corrupting the golden triple. The loss function (Eq. 4) favors lower values for similar triples than for dissimilar triples. The input of the CBOW Encoders is the category attributes words, and its output is category representation vectors. The categories and relations are initialized by the random procedure proposed in [3]. We take stochastic gradient descent in minibatch mode to optimize the objective function.

4.2 Taxonomy Matching Generation

In this section, we will describe the process of our method to obtain a most suitable one-to-one equivalent categories matchings with highest confidence for a pair of taxonomies based on the categories embeddings.

Bipartite Graph Creation. In order to efficiently encode the complicated relationships between the categories C of T and the categories $C^{'}$ of $T^{'}$, we choose the bipartite graph model as our representation model, because that the bipartite graph can encode categories from taxonomies as the vertices, and encode the candidate matching relationships between these vertices explicitly.

Before we begin to construct a bipartite graph to model the candidate matching relationships of categories between two taxonomies, we firstly introduce the score function which measure the suitability of a matching between categories. Given a pair of categories $c, c^{'}$, their corresponding embeddings are $\mathbf{c}, \mathbf{c}^{'}$, the relevance score between $c, c^{'}$ is defined as:

$$w(c, c^{'}) = \cos(\mathbf{c}, \mathbf{c}^{'}) = \frac{\mathbf{c} \cdot \mathbf{c}^{'}}{\|\mathbf{c}\| \|\mathbf{c}^{'}\|} \tag{6}$$

We build a weighted bipartite graph $G = (V, E, W)$ exploiting the score function, where V denotes the vertices set, consisting of $|C|$ left vertices and $|C^{'}|$ right vertices; E denotes the edges set, including all the candidate links between categories from C and $C^{'}$; W: $E \rightarrow \mathbf{R}$ is the weight function. The graph G is defined as an undirected graph and its generation algorithm can be described in Algorithm 1, where V, E, W is initialized as a zero set, respectively.

Specifically, Algorithm 1 works in two steps as follows:

- Candidate matching categories selection. In this step, for each category c from taxonomy T, we pair it with each category $c^{'}$ contained in $T^{'}$. All categories $c^{'} \in C^{'}$ likely matching to the category c are selected (see Lines 2–6).
- Vertex connection. In this step, we assign the matching edge to the bipartite graph. For each category vertex c from taxonomy T in the graph, we add an edge between it and each of its candidate matching category $c^{'}$ from taxonomy $T^{'}$; the weight w of the edge $(c, c^{'})$ is set according to Eq. (6) (see Lines 7–12).

After those two steps, a weighted bipartite graph has been generated (see Line 13).

Algorithm 1. The algorithm for bipartite graph creation

Input: $T = \{(h_c, r, t_c)\}, T^{'} = \{(h_c^{'}, r^{'}, t_c^{'})\}, G = (V, E, W)$ and the embeddings of all the categories of T and $T^{'}$.
Output: $G = (V, E, W)$.

```
Initialize graph G = (V, E, W): V = ∅, E = ∅, W = ∅.
for all c ∈ C in T do
  for all c' ∈ C' in T' do
    Compute the likelihood w that c is equivalent matching to c' based on the
    embeddings of those two categories via Equation (6).
    if w > 0 then
      Add class c and c' to V.
      Add weight function W(c, c') = w to W.
      Add edge (c, c') to E.
      Assign weight to edge (c, c') with w.
    end if
  end for
end for
return G = (V, E, W).
```

Maximum Weight Matching in Bipartite Graph. In this section, we will introduce how to find an optimal one-to-one equivalent categories matching M for a pair of taxonomies. The goal of taxonomy matching is to find a most suitable one-to-one equivalent categories matching M for a pair of taxonomies with highest confidence, and the goal of maximum weight matching is to find a set of vertex-disjoint edges with maximum weight. Therefore, the taxonomy matching problem can be converted to a maximum weight matching problem.

Specifically, given a weighted bipartite graph $G = (V, E, W)$, we can use integer linear program (ILP) to model the matching problem:

$$\begin{aligned} &\max \sum_{e \in E} w(e)x(e) \\ &\text{s.t.} \\ &\quad \sum_{e=(v,v')} x(e) \leq 1 \qquad \forall v \in V \\ &\quad 0 \leq x(e) \leq 1 \qquad \forall e \in E \\ &\quad x(e) \text{ is an integer} \quad \forall e \in E, \end{aligned} \tag{7}$$

where x is the matching's incidence vector; w represents the likelihood of the categories matching as specified in Eq. 6. The dual of the Problem (7) is vertex cover problem. The dual problem is defined as:

$$\begin{aligned} &\min \sum_{v \in V} y(v) \\ &\text{s.t.} \\ &\quad y(e) \geq w(e) \quad \forall e \in E \\ &\quad y(v) \geq 0 \qquad \forall v \in V, \end{aligned} \tag{8}$$

where we define $y(v, v^{'}) \overset{def}{=} y(v)+y(v^{'})$. According to complementary relaxation condition, the matching M and y are optimal iff $\forall e \in M, y(e) = w(e)$ and for all free vertices $v, y(v) = 0$.

The essential idea of acquiring maximum weight matching is to repeatedly find an augmenting path in the bipartite graph and augment over it, until there are no augmenting paths left. The maximum weight matching problem has been extensively studied (e.g., [8,10]). The most efficient general algorithm for this problem is Hungarian algorithm [8]. Since the weights in the bipartite graph we constructed are real numbers, so we adjust the Hungarian algorithm improved by Lawler [10], to our taxonomy matching problem. In what follows, we describe our algorithm in details.

Maximum Weight Matching Algorithm. Given a weighted bipartite graph $G = (V, E, W)$ constructed from taxonomies T and $T^{'}$, we repeatedly find augmenting paths and augment over it, to find a set of vertex-disjoint edges with maximum weight. Our algorithm consists of four steps. It firstly initializes the dual variables in Problem (8). Secondly, it finds an augmenting path and augments over it. Thirdly, it computes the dual variable augmentation value and updates the dual variables in the fourth step. Repeating steps 2–4, we can finally obtain a matching with maximum weight according to the bipartite graph. In this paper, we model taxonomy matching problem as maximum weight matching problem. Therefore, we can guarantee to generate an optimal matching with highest confidence for two taxonomies in a global manner.

Here, we use $V_L = C$ and $V_R = C^{'}$ to represent the left vertices set and right vertices set in G, respectively. Let LF and RF denote the left and right free vertices (not matched vertex) in G, respectively, which is initialized as $LF = V_L$ and $RF = V_R$. Let $y(v)$ denote the dual variable value for each vertex $v \in V$, δ denote the dual variable augmentation value, whose initial value is $\delta_0 = \max\{W(e)|e \in E\}$. Let τ denote the current iteration times. Let M denote the set of vertex-disjoint edges with maximum weight, which is initialized as a zero set, i.e., $M = \varnothing$. The maximum weight matching algorithm can thus be described in Algorithm 2.

Algorithm 3 is to find an augmenting path by performing a bread-first-search (BFS) on a modified graph. Specifically, the algorithm firstly judges whether it has an augmenting path or not, using the left and right free vertices sets in G (see Lines 1–2). Secondly, the algorithm directs all edges in G (see Lines 4) and performs BFS algorithm to find an augmenting path (see Lines 5–6). Thirdly, augments the free vertices sets (see Lines 8). Finally, we acquire an augmenting path (see Lines 9–10).

Algorithm 2. The algorithm for maximum weight matching

Input: $G =< V, E, W >, V_L, V_R, LF, RF, \delta_0, M$.
Output: $M = \{(v, v^{'})|v \in V_L, v^{'} \in V_R\}$.

for all $v \in V$ **do**
  if v is a left vertex in G **then**
    $y(v) = \delta_0$.
  else
    $y(v) = 0$.
  end if
end for
Set $\tau = 0$.
repeat
  Find an augmenting path AP, using the method described in Algorithm 3.
  Augment the matchings M: $M = M \oplus AP$.
  $\tau = \tau + 1$.
  For all edge $(v, v^{'}) \in E$, get minimum left vertex dual variable value $l\_y(v) = \min\{y(v)\}$ and minimum right vertex dual variable minus value $r\_y(v^{'}) = \min\{y(v) + y(v^{'}) - W(v, v^{'})\}$.
  if $l\_y(v) < r\_y(v^{'})$ **then**
    halt.
  else
    Set dual variable augmentation value $\delta_\tau = l\_y(v)$.
  end if
  for all $v \in V$ **do**
    if v is a left vertex in G **then**
      $y(v) = y(v) - \delta_\tau$.
    else
      $y(v) = y(v) + \delta_\tau$.
    end if
  end for
until $(AP = \varnothing)$
return M.

5 Experiments

In this section, we test the performance of our method TransC on two benchmark datasets. We will compare the accuracy of our method with the baseline methods.

5.1 Experimental Settings

In this paper, we employ LogMap, AML, YAM-BIO, XMap and SiGMa as the baseline methods. We compared the accuracy of the final category matchings in terms of precision, recall and F1-measure on the number of categories correctly matched. We used two datasets from OAEI 2017[1].

[1] http://oaei.ontologymatching.org/2017/.

Algorithm 3. The augmenting path searching algorithm

Input: $G =< V, E, W >, M, \{y(v)|v \in V\}, V_L, V_R, LF, RF$.
Output: AP.
1: **if** $LF = \varnothing$ or $RF = \varnothing$ **then**
2: $AP = \varnothing$.
3: **else**
4: Direct unmatched edges from $V_L \rightarrow V_R$, matched edges $V_R \rightarrow V_L$.
5: Add vertices s, t and connect them to free vertices in LF and RF, respectively.
6: Run BFS algorithm on G to find an augmenting path AP containing only edges $\{(v, v^{'})|v \in LF, v^{'} \in RF\}$ for which $y(v) + y(v^{'}) = W(v, v^{'})$.
7: **end if**
8: Augment the free vertices set LF and RF based on the augmenting path AP, respectively.
9: $AP = AP \setminus \{s, t\}$.
10: **return** AP.

The first dataset was derived from the large BioMed track (denoted as DS_{bio}). The dataset contains three biomedical ontologies: FMA, SNOMED-CT and NCI. These ontologies are semantically rich and contain tens of thousands of categories. Large BioMed track contains three matching problem: FMA-NCI, FMA-SNOMED and NCI-SNOMED, and each matching problem contains two tasks involving "small" largebio dataset (denoted as DS_{bio_s}) and "whole" largebio dataset (denoted as DS_{bio_w}). The dataset DS_{bio} is used to find matchings between large ontologies with rich semantics.

The second dataset was derived from the conference track (denoted as DS_{conf}), The DS_{conf} dataset contains 16 different ontologies, which aims at finding all category matchings in ontology set describing the domain of organising conferences [13]. The DS_{conf} dataset contains 867 categories and 534 attributes in total.

For experiments of TransC, the parameters we used to measure the matching likelihood between two categories is experimentally set to $\lambda = 0.01$, $\gamma = 2$, $k = 50$, $d = L_1$, which yields the best accuracy, where λ is the learning rate for stochastic gradient descent (SGD), γ is the margin, k is the dimensions of category and relation embedding, d is the dissimilarity measure. We found reasonable values for the parameters by exploring its accuracy on the DS_{conf} dataset alignments, and then kept them fixed for all the experimental comparisons over the DS_{bio} and DS_{conf} datasets.

5.2 Experimental Analysis

In the following, we will test the performance of TransC and the baseline methods on taxonomies with different size and domains. Firstly, we tested the accuracy of all the methods on the DS_{bio} datasets, then tested the accuracy of all the methods on the DS_{conf} dataset.

In order to see how the accuracies of all the methods change with the increase of the size of ontologies, we test all the methods on the DS_{bio_s} dataset and

DS_{bio_w} dataset. Tables 1 and 2 show the results for the DS_{bio_s} dataset and DS_{bio_w} dataset, respectively. From the results, it can be seen that our method TransC achieves the best precision, recall and F1 measure on the datasets. In addition, we averaged the precision, recall and F1 measure of each method over the DS_{bio_s} and DS_{bio_w} datasets in Tables 1 and 2, respectively. The average results for these methods are shown in Table 3. From Table 3, we can notice that TransC achieves the best accuracy among all the methods. In summary, the experimental results show that TransC can obtain better accuracy over the baseline methods on the DS_{bio} dataset.

In the following, we test how the accuracies of all the methods change across ontologies from different domains. Firstly, we conducted experiments on the DS_{bio} dataset. Secondly, we conducted experiments on the DS_{conf} dataset. In order to test the performance of our method and the baseline methods on the DS_{bio} dataset, we averaged the accuracy on the DS_{bio_s} and DS_{bio_w} datasets (Table 3). The results is shown in Table 4. From Table 4, we can see that TransC achieves the best performance on the DS_{bio} dataset.

In the following, we conducted experiments on the DS_{conf} dataset. The results on the DS_{conf} dataset is presented in Table 5. From Table 5, it can be seen that TransC performs better than any of the baseline methods.

Table 1. Comparison over the DS_{bio_s} dataset

Method	Task								
	FMA-NCI			FMA-SNOMED			NCI-SNOMED		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
SiGMa	0.841	0.654	0.735	0.959	0.692	0.804	0.896	0.647	0.751
XMap	0.977	0.901	0.937	0.974	0.847	0.906	0.894	0.566	0.693
YAM-BIO	0.969	0.896	0.931	0.966	0.733	0.834	0.899	0.677	0.772
AML	0.958	0.910	0.930	0.923	0.762	0.835	0.871	0.746	0.804
LogMap	0.944	0.897	0.920	0.947	0.690	0.798	0.947	0.690	0.798
TransC	**0.981**	**0.928**	**0.954**	**0.979**	**0.854**	**0.912**	**0.961**	**0.749**	**0.842**

Table 2. Comparison over the DS_{bio_w} dataset

Method	Task								
	FMA-NCI			FMA-SNOMED			NCI-SNOMED		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
SiGMa	0.826	0.628	0.714	0.945	0.625	0.752	0.873	0.466	0.608
XMap	0.884	0.847	0.865	0.774	0.843	0.807	0.819	0.553	0.66
YAM-BIO	0.818	0.888	0.852	0.887	0.728	0.800	0.827	0.698	0.757
AML	0.838	0.872	0.855	0.882	0.687	0.772	0.904	0.668	0.768
LogMap	0.856	0.808	0.831	0.840	0.645	0.730	0.868	0.597	0.707
TransC	**0.896**	**0.892**	**0.894**	**0.948**	**0.849**	**0.896**	**0.911**	**0.708**	**0.797**

Table 3. Average comparison over the DS_{bio} dataset

Tasks	Approaches	Precision	Recall	F1
DS_{bio_s}	SiGMa	0.899	0.664	0.763
	XMap	0.948	0.770	0.851
	YAM-BIO	0.945	0.770	0.848
	AML	0.917	0.810	0.858
	LogMap	0.946	0.760	0.842
	TransC	**0.974**	**0.844**	**0.904**
DS_{bio_w}	SiGMa	0.881	0.573	0.691
	XMap	0.826	0.750	0.785
	YAM-BIO	0.844	0.770	0.806
	AML	0.875	0.740	0.803
	LogMap	0.855	0.680	0.759
	TransC	**0.918**	**0.816**	**0.864**

Table 4. Comparison over the DS_{bio} datasets

Approaches	Precision	Recall	F1
SiGMa	0.890	0.619	0.727
XMap	0.887	0.760	0.818
YAM-BIO	0.894	0.770	0.828
AML	0.896	0.770	0.831
LogMap	0.900	0.720	0.801
TransC	**0.946**	**0.830**	**0.884**

Overall, from Tables 4 and 5, we can see that TransC can obtain the best performance both on the DS_{bio} dataset and the DS_{conf} dataset. The results show that our method TransC can performs well on taxonomies from different domains. Based on the experimental results and analysis, we can see that TransC can performs well on taxonomies regardless of their scales and domains. This shows that TransC has good adaptability.

Table 5. Comparison over the DS_{conf} dataset

Approaches	Precision	Recall	F1
SiGMa	0.512	0.334	0.404
XMap	0.840	0.570	0.680
AML	0.840	0.660	0.740
LogMap	0.820	0.590	0.690
TransC	**0.862**	**0.690**	**0.766**

6 Conclusion

This paper presents a representation learning method for taxonomy matching. As our method models the taxonomy matching problem as an optimization problem on bipartite graphs, with the global nature of maximum weight matching, our method can obtain a global optimal matching for two taxonomies. Currently, our method mainly focuses on one-to-one equivalent category matchings between two taxonomies and runs in polynomial time. For future work, we plan to address the subclass and superclass matchings at the same time.

References

1. Achichi, M., Cheatham, M., Dragisic, Z., et al.: Results for the ontology alignment evaluation initiative 2017. In: Proceedings of the 12th International Workshop on Ontology Matching (2017)
2. Asprino, L., Presutti, V., Gangemi, A., Ciancarini, P.: Frame-based ontology alignment. In: Proceedings of 31st AAAI Conference on Artificial Intelligence, pp. 4905–4906 (2017)
3. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Proceedings of 27th Annual Conference on Neural Information Processing Systems, pp. 2787–2795 (2013)
4. Bouraoui, Z., Schockaert, S.: Learning conceptual space representations of interrelated concepts. In: Proceedings of 27th International Joint Conference on Artificial Intelligence, pp. 1760–1766 (2018)
5. Chen, R.C., Bau, C.T., Yeh, C.J.: Merging domain ontologies based on the wordnet system and fuzzy formal concept analysis techniques. Appl. Soft Comput. **11**(2), 1908–1923 (2011)
6. Demidova, E., Oelze, I., Nejdl, W.: Aligning freebase with the YAGO ontology. In: Proceedings of 22nd ACM International Conference on Conference on Information and Knowledge Management, pp. 579–588 (2013)
7. Jiménez-Ruiz, E., Grau, B.C.: LogMap: logic-based and scalable ontology matching. In: Proceedings of 10th International Semantic Web Conference, pp. 273–288 (2011)
8. Kuhn, H.W.: The hungarian method for the assignment problem. Naval Res. Logistics Q. **2**(1–2), 83–97 (1955)
9. Lacoste-Julien, S., Palla, K., Davies, A., Kasneci, G., Graepel, T., Ghahramani, Z.: SiGMa: simple greedy matching for aligning large knowledge bases. In: Proceedings of 19th SIGKDD, pp. 572–580 (2013)
10. Lawler, E.L.: Combinatorial Optimization: Networks and Matroids. Courier Dover Publications (1976)
11. Lee, T., Wang, Z., Wang, H., won Hwang, S.: Web scale taxonomy cleansing. Proc. VLDB Endowment **4**(12) (2011)
12. Li, J., Tang, J., Li, Y., Luo, Q.: RiMOM: a dynamic multistrategy ontology alignment framework. TKDE **21**(8), 1218–1232 (2009)
13. Lin, H., Wang, Y., Jia, Y., Xiong, J., Zhang, P., Cheng, X.: An ensemble matchers based rank aggregation method for taxonomy matching. In: Cheng, R., Cui, B., Zhang, Z., Cai, R., Xu, J. (eds.) APWeb 2015. LNCS, vol. 9313, pp. 190–202. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25255-1_16

14. Ba, M., Diallo, G.: Large-scale biomedical ontology matching with ServOMap. IRBM **34**(1), 56–59 (2013)
15. Ochieng, P., Kyanda, S.: A statistically-based ontology matching tool. Distrib. Parallel Databases **36**(1), 195–217 (2018)
16. Suchanek, F.M., Abiteboul, S., Senellart, P.: PARIS: probabilistic alignment of relations, instances, and schema. Proc. VLDB Endowment **5**(3), 157–168 (2011)
17. Xie, R., Liu, Z., Jia, J., Luan, H., Sun, M.: Representation learning of knowledge graphs with entity descriptions. In: Proceedings of 30th AAAI Conference on Artificial Intelligence, pp. 2659–2665 (2016)