Incorporating Term Definitions for Taxonomic Relation Identification

Yongpan Sheng[1], Tianxing Wu[2], and Xin Wang[3(✉)]

[1] School of Computer Science and Engineering,
University of Electronic Science and Technology of China, Chengdu, China
shengyp2011@gmail.com
[2] School of Computer Science and Engineering, Nanyang Technological University,
Singapore, Singapore
wutianxing@ntu.edu.sg
[3] College of Intelligence and Computing,Tianjin University, Tianjin, China
wangx@tju.edu.cn

Abstract. Taxonomic relations (also called "is-A" relations) are key components in taxonomies, semantic hierarchies and knowledge graphs. Previous works on identifying taxonomic relations are mostly based on linguistic and distributional approaches. However, these approaches are limited by the availability of a large enough corpus that can cover all terms of interest and provide sufficient contextual information to represent their meanings. Therefore, the generalization abilities of the approaches are far from satisfactory. In this paper, we propose a novel neural network model to enhance the semantic representations of term pairs by encoding their respective definitions for the purpose of taxonomic relation identification. This has two main benefits: (i) Definitional sentences represent specified corpus-independent meanings of terms, hence definition-driven approaches have a great generalization capability to identify unseen terms and taxonomic relations which are not expressed in domain specificity of the training data; (ii) Global contextual information from a large corpus and definitions in the sense level can provide richer interpretation of terms from a broader knowledge base perspective, and benefit the accurate prediction for the taxonomic relations of term pairs. The experimental results show that our model outperforms several competitive baseline methods in terms of F-score on both specific and open domain datasets.

Keywords: Taxonomic relation identification · Definition-driven approach

1 Introduction

Taxonomic relation (also called "is-A" relation) identification is a task to determine whether a specific pair of terms[1] holds the taxonomic relation or not. Concretely, given a pair of terms (x, y), if y holds a semantically border category that includes x, we call y a hypernym of x and x a hyponym of y [26]. For instance, "*scientist*" is a hypernym of "*Einstein*", "*actor*" is a hypernym of "*Mel Gibson*", "*Paris*" is a hyponym of "*exciting city*". The accurate prediction of these taxonomic relations benefits for a variety of downstream applications, such as serving as building blocks for semantic structure

[1] This paper uses "terms" to refer to any words or phrases.

X. Wang et al. (Eds.): JIST 2019, LNCS 12032, pp. 1–17, 2020.
https://doi.org/10.1007/978-3-030-41407-8_1

construction including taxonomies, semantic hierarchies and knowledge graphs, and facilitating question answering systems [7].

Previous approaches for this task can be generally classified into two categories: linguistic and distributional approaches. Linguistic approaches rely on lexical-syntactic patterns (e.g., A typical pattern is *"A such as B"*) to capture textual expressions associated with taxonomic relations, and match them with given documents or multiple Web sources to identify taxonomic relations between terms [20,28]. These patterns can be created manually [28] or learnt automatically [20,26]. Despite their higher precision in several applications, such identified patterns are too specific to cover the wide range of complex linguistic circumstances, and the ambiguity of natural language compounded by data sparsity makes linguistic matching methods less robust. In contrast, distributional methods focus on embedding the two terms of interest into context-aware vector representations, and then predict their taxonomic relation based on these representations. The usage of term embeddings [16] allows machines to make predictions from the entire corpus.

These approaches, however, make use of linguistic features as well as contextual information of terms alone acquiring in a large corpus, often with unsatisfactory results. In many real-word settings, we can also explore the value of external evidence, primarily textual definition evidence related to terms, which can be extracted from structured knowledge resources, to express corpus-independent meanings of terms for the purpose of enhancing the generalization ability of the system to unseen terms, and even to the scenario of low-resource languages. Humans, too, often benefit from richer information deriving from the definitions of terms when trying to determine the taxonomic relation between the terms even for terms they have not been exposed in the training data.

Based on the above considerations, we propose a neural network model[2], which can enhance the representations of term pairs by encoding their separative definitions, instead of just focusing on the meaning of each term. Moreover, we formulate the taxonomic relation detection problem as (x_{hypo}, d_x, y_{hyper}, d_y, 1/0), where x_{hypo} and d_x denote hyponym and its definition, respectively. While y_{hyper} and d_y denote hypernym and its definition, respectively. The binary value "1/0" indicates y_{hyper} is x_{hypo}'s hypernym or not. Technically, we first model the representation for each pair in $\{(x_{hypo}, y_{hyper}), (x_{hypo}, d_y), (d_x, y_{hyper}), (d_x, d_y)\}$ and given as the input to a top-performing baseline system (Sect. 3.1) to obtain four separate representations, which are concatenated by heuristic matching strategies in the form of a joint vector representation. The overall representation is fed into a softmax classifier to determine whether the taxonomic relation holds or not (Sect. 3.2).

The experimental results show that our approach achieves the state-of-the-art performance on both general and specific domain datasets. In addition, another advantage of our proposed model is that it has the capacity to generalize the taxonomic relation properties of pairs that even are not being expressed in domain specificity of the training data. The key contributions of our work are highlighted as follows:

– We propose a novel neural network model which accounts for the taxonomic relation detection problem primarily by the knowledge of the term definitions. Definitions

[2] The dataset and the code for our model are available at https://github.com/shengyp/Taxonomic-relation/.

can provide complementary knowledge to the context from corpus, so that our proposed model enables better tolerance for unseen terms, rare terms, and terms with biased sense distribution.
– Our model enables combine distributional model with definition encoding, rather than a simple concatenation of the two subsystems. This benefits to generate more indicative features across distributional contexts and definitions for the accurate prediction of taxonomic relations of term pairs.
– The experiment results on both general and domain-specific datasets corroborate the effectiveness and robustness of our model over several competitive models in F-score metrics.

2 Related Work

Taxonomic relation identification is one of the most topics in NLP research. Many approaches have been explored for this task can be divided into two branches, including linguistic and distributional approaches.

In linguistic approaches, the range of pre-defined rules or lexical-syntactic patterns are leveraged to extract taxonomic relations from text corpus. Patterns are either chosen manually [9] or learnt automatically via bootstrapping [26]. While such approaches can result in taxonomic relations with relatively high accuracies. Unfortunately, using patterns as features may result in the sparsity of the feature space [19]. More approaches require the co-occurrence of the two terms in the same sentence, which strongly hinders the recall of these methods. Higher recall can be achieved contributes to distributional methods.

In distributional approaches, by studying the relations of distributional representations (word embeddings) derived from contexts, between hypernyms and their respective hypernyms, the taxonomic relations can be identified by learning semantic prediction models, especially for several unsupervised measures [10,22]. Such approaches draw primarily on the distributional hypothesis [8], which states that terms appear in similar context may share semantic relationship. The main advantage of distributional approaches is that they can discover relations not directly expressed in the text. However, such approaches depend on the choice of feature from domain specificity of the training data, e.g., an IT corpus hardly mentions “apple” as a fruit. Furthermore, rare terms are poorly expressed by their sparse global context and, more generally, these methods would not generalize to the low-resource language setting.

Our proposed model shares the same inspiration with distributional methods. More importantly, it is beyond the framework of distributional models acquiring context-aware term meaning in a large corpus, and instead explore a novel information resources - definitive sentences, to enhance the robustness of the system.

3 The Proposed Method

In this section, we first briefly describe a top-performing neural network architecture. This could be viewed as a baseline system to encode a pair of texts. Then, we elaborate on the adaptation we make towards the architecture so that the resulting system can better serve the taxonomic relation identification problem.

3.1 The Baseline System

For the design of the baseline system, we follow the idea of Siamese Network [21], as shown in Fig. 1. Concretely, the two identical sentence encoders share the same set of weights during training, and generate two neural representations. We observe from the figure that the system mainly consists of three layers from bottom to top: (i) Sentence input layer; (ii) Sentence encoding layer; (iii) Sentence output layer. We will explain the last two layers in detail in the following subsection. And the sentence input layer will be introduced in Sect. 3.2.

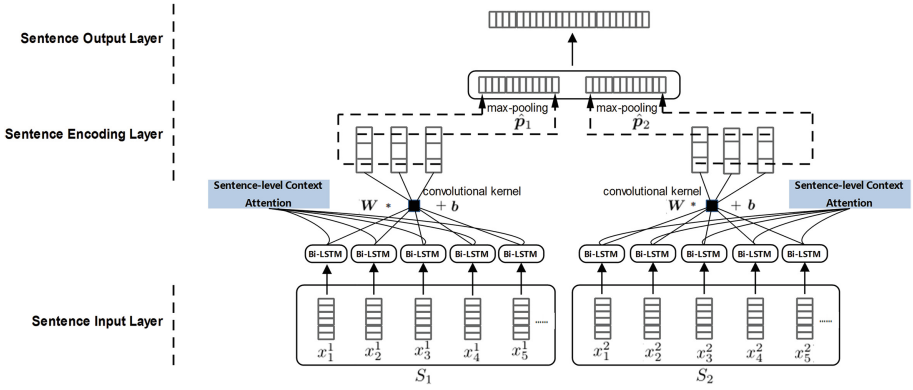

Fig. 1. Architecture of the baseline system

Sentence Encoding Layer. Take the sentence encoder on any one side as an example. Given a sentence S_i (i = 1 or 2) with length L, our goal is to find a neural representation of S_i. We first map L words of S_i to a sequence of word embedding vectors $\{x_j^i\}$ ($j = 1, ..., L$, and the dimension of word embedding denotes as d_m), based on the pre-trained embeddings that will be described in the following Sect. 4.1. Then we employ a Bi-LSTM which is composed of a forward LSTM and backward LSTM component, to process $\{x_j^i\}$ in the forward left-to-right and the backward right-to-left directions, respectively. In each direction, the reading of $\{x_j^i\}$ is modelled as a recurrent process with a single hidden state. Given an initial value, the state changes its value recurrently, and each time-step consumes an incoming word.

Take the forward LSTM component as an example. Denoting the initial hidden state as $\overrightarrow{\boldsymbol{h}}_0$, the recurrent state transition values can be calculated by $\{\overrightarrow{\boldsymbol{h}}_1, \overrightarrow{\boldsymbol{h}}_2, ..., \overrightarrow{\boldsymbol{h}}_L\}$ when it reads the input $\{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_L\}$. At time t, the current hidden state vector $\overrightarrow{\boldsymbol{h}}_t$ is computed based on the previous hidden $\overrightarrow{\boldsymbol{h}}_{t-1}$, the previous cell $\boldsymbol{c}_{t-1}$ and the current input word embedding $\boldsymbol{x}_t$. The detail computations of the forward LSTM are defined as follows [6]:

$$\hat{\boldsymbol{i}}_t = \delta(\boldsymbol{W}_{\boldsymbol{xi}}\boldsymbol{x}_t + \boldsymbol{W}_{\boldsymbol{hi}}\overrightarrow{\boldsymbol{h}}_{t-1} + \boldsymbol{b}_{\boldsymbol{i}}),$$

$$\hat{\boldsymbol{f}}_t = \delta(\boldsymbol{W}_{\boldsymbol{xf}}\boldsymbol{x}_t + \boldsymbol{W}_{\boldsymbol{hf}}\overrightarrow{\boldsymbol{h}}_{t-1} + \boldsymbol{b}_{\boldsymbol{f}}),$$

$$\begin{aligned}
\boldsymbol{o}_t &= \delta(\boldsymbol{W_{xo}}\boldsymbol{x}_t + \boldsymbol{W_{ho}}\overrightarrow{\boldsymbol{h}}_{t-1} + \boldsymbol{b_o}),\\
\boldsymbol{u}_t &= \delta(\boldsymbol{W_{xu}}\boldsymbol{x}_t + \boldsymbol{W_{hu}}\overrightarrow{\boldsymbol{h}}_{t-1} + \boldsymbol{b_u}),\\
\boldsymbol{i}_t, \boldsymbol{f}_t &= softmax(\hat{\boldsymbol{i}}_t, \hat{\boldsymbol{f}}_t),\\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{u}_t,\\
\overrightarrow{\boldsymbol{h}}_t &= \boldsymbol{o}_t \odot tanh(\boldsymbol{c}_t),
\end{aligned} \tag{1}$$

where $\boldsymbol{i}_t$, $\boldsymbol{f}_t$, $\boldsymbol{o}_t$ and $\boldsymbol{u}_t$ are an input gate, a forget gate, an output gate and an actual input at t time, respectively. The activation function of the LSTM δ is set to $tanh$. $\boldsymbol{W}_{(.)}$ represent trained weight matrices, $\boldsymbol{c}_t$ is a vector representation of state in recurrent cell at time step t, and $\boldsymbol{b}_x(x \in \{i, o, f, u\})$ is a bias vector. $\odot$ denotes the Hadamard product.

The backward LSTM component follows the same recurrent state transition process as described in Eq. (1). Starting from an initial state $\boldsymbol{h}_{n+1}$, which is a model parameter, it reads the input $\{\boldsymbol{x}_n, \boldsymbol{x}_{n-1}, ..., \boldsymbol{x}_0\}$, changing its value to $\{\overleftarrow{\boldsymbol{h}}_n, \overleftarrow{\boldsymbol{h}}_{n-1}, ..., \overleftarrow{\boldsymbol{h}}_0\}$, respectively. A separate set of parameters $\boldsymbol{W}_{(.)}$ and $\boldsymbol{b}_x$ are used for the backward component.

Finally, the Bi-LSTM concatenates the vector value of $\overrightarrow{\boldsymbol{h}}_t$ and $\overleftarrow{\boldsymbol{h}}_t$ to represent the encoding information of $\boldsymbol{x}_t$ at t time, which is denoted as $\boldsymbol{h}_t = [\overrightarrow{\boldsymbol{h}}_t; \overleftarrow{\boldsymbol{h}}_t]$.

An additive attention mechanism [2] is exploited to the resulting hidden states corresponding to $\{\boldsymbol{h}_t\}$ $(t = 1, ..., L)$. That is a weighted calculation for learning more accurate and focused sentence representations, based on the following formulas:

$$\boldsymbol{g}_t = \sum_i \boldsymbol{\alpha}_i^t \boldsymbol{h}_i, \tag{2}$$

where $\boldsymbol{h}_i$ is the column vector denoting the hidden state of x_i, $\boldsymbol{\epsilon}_i$ can be regarded as the intermediate attention representation of x_i in the sentence and can be obtained from a linear transformation of $\boldsymbol{h}_i$. $\boldsymbol{\alpha}_i$ denotes the attention weight of x_i (i.e., namely *attention vector* $\boldsymbol{\alpha}$) and is computed by the combination of weighted values in $\boldsymbol{\epsilon}_i$.

$$\boldsymbol{\alpha}_i^t = \frac{e^{\boldsymbol{u}^T \boldsymbol{\epsilon}_i}}{\Sigma_j e^{\boldsymbol{u}^T \boldsymbol{\epsilon}_j}}, \tag{3}$$

$$\boldsymbol{\epsilon}_i = tanh(\boldsymbol{W}_a \boldsymbol{h}_i + \boldsymbol{b}_a), \tag{4}$$

where $\boldsymbol{W}_a$ is a trained weight matrices, $\boldsymbol{u}^T$ denotes transpose of a trained parameter $\boldsymbol{u}$. $\boldsymbol{b}_a$ is a bias vector.

We concatenate contextual information vector for each time step as follows:

$$\boldsymbol{g} = [\boldsymbol{g}_1, \boldsymbol{g}_2, ..., \boldsymbol{g}_L], \tag{5}$$

In our task, the indicative features related to the taxonomic relation are likely to appear in any area of the sentence under different contexts. Hence, we should encode the sentence further by utilizing all local features and form better neural representations globally. When using a neural network, the convolution approach is a natural means of extracting local features with a sliding window of length l over the sentence [4].

Here, typically the size of the sliding window l is 3. Then, it combines all fine-grained features via a max-pooling operation to obtain a fixed-sized vector for the output of the convolution operation.

Here, convolution is defined as a matrix multiplication between a sequence of vectors $\mathbf{g}$ which is formed by Eq. (5), a convolution matrix $\mathbf{W} \in \mathbb{R}^{d_m \times (d_m \times l)}$ and a bias vector $\mathbf{b}$ with a sliding window [30]. Let us define the vector $\boldsymbol{q}_i \in \mathbb{R}^{d_m \times l}$ as the concatenation of a sequence of input representations $\boldsymbol{g}$ in the i-th window, we have:

$$\boldsymbol{q}_i = \boldsymbol{g}_{i:i+l-1}; (1 \leq i \leq L - l + 1). \tag{6}$$

Hence, the output of a single convolutional kernel $\mathbf{p}_i$ (i.e., i-th window) can be expressed as follows:

$$\boldsymbol{p}_i = f(\boldsymbol{W}\boldsymbol{q}_i + \boldsymbol{b}), \tag{7}$$

where f is the activation function. A convolutional layer can comprise d_c convolutional kernels which could result in a output matrix $\boldsymbol{P} = [\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_{d_c}] \in \mathbb{R}^{d_c \times (d_m - l + 1)}$. Then, a max-pooling operation is applied to this matrix to obtain maximum valued features as follows:

$$\hat{\boldsymbol{p}} = max(\boldsymbol{P}) = \sum_i max(\boldsymbol{p}_i). \tag{8}$$

Sentence Output Layer. In our settings, the neural representation of each sentence S_i (i = 1, 2) (term can be treated as short sentence) is generated using Eq. (8), and can be shortly denoted as $\hat{\boldsymbol{p}}_i$ (i = 1, 2). Finally, we obtain the overall representation for the sentence pair by concatenating $\hat{\boldsymbol{p}}_1$ and $\hat{\boldsymbol{p}}_2$.

3.2 Our Proposed Model

Based on the baseline system, we make the adaptation of this architecture for taxonomic relation identification problem and the overall architecture of our proposed model is shown in Fig. 2.

The Sentence Input Strategy. For a distributional method such as Shwartz *et al.* [23] directly concatenates the term path representation vector and term embedding vector as the input of an off-the-shelf classifier. However, to the best of our knowledge, a simple combination of term distributional representation and definition encoding could not integrate with the advantages of these two subsystems. Moreover, the analysis on our datasets hints that our model can obtain more indicative features by modeling the term pair such as (term, definition), which may cross the distributional models and definition encoding. For instance, the definition of term "*dolphin*" in WordNet is "*large slender food and game fish widely distributed in warm seas*". Intuitively, when the system meets the term pair ("*dolphin tuna*", "*percoid fish*"), it should be easy to make the decision on the taxonomic relation since "*fish*" appears in the definitive sentence.

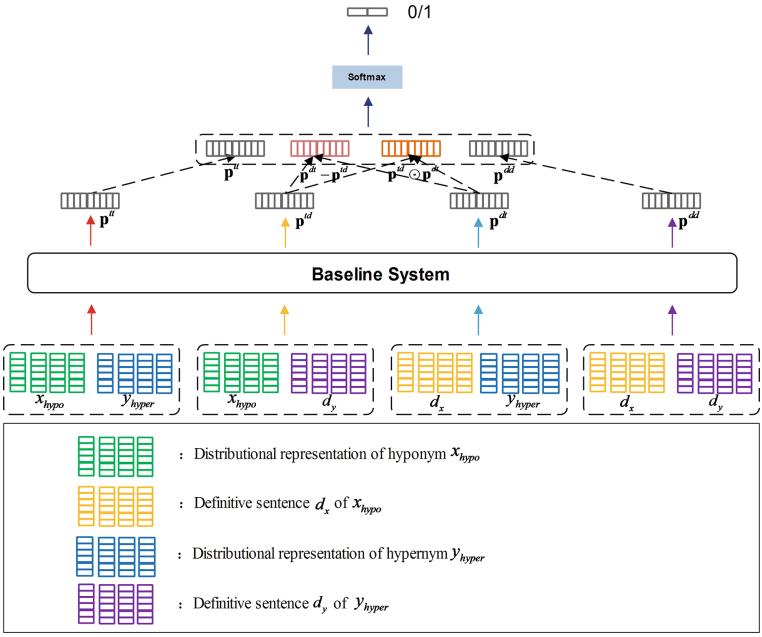

Fig. 2. Architecture of our proposed model.

Inspired by the above observations, we investigate four strategies to define the input representations on the baseline system as described in the baseline system, including: (x_{hypo}, y_{hyper}), (x_{hypo}, d_y), (d_x, y_{hyper}) and (d_x, d_y). Moreover, we obtain four separate representations based on the above combinations as the output: $\mathbf{p}^{tt}$ from (x_{hypo}, y_{hyper}), $\mathbf{p}^{td}$ from (x_{hypo}, d_y), $\mathbf{p}^{dt}$ from (d_x, y_{hyper}), and $\mathbf{p}^{dd}$ from (d_x, d_y). In the following, we further give the explanations of these combinations with more details:

The baseline system over (x_{hypo}, y_{hyper}), which considers two vector representations as the input, and outputs a joint vector representation. This combination intends to model embeddings from a hyponym to its hypernym via a network with weights. It actually is common with a pioneer work in this field [5], which employs uniform linear projection and piecewise linear projection to map the embeddings of a hyponym to its hypernym.

The baseline system over (x_{hypo}, d_y) and (d_x, y_{hyper}) also outputs a joint vector representation. These combinations benefit to generate indicative features across distributional context and definition for discriminating taxonomic relations from other semantic relations, e.g., as we described in the beginning of this subsection, the hyponym or prominent context words of the hyponym are expected to appear in its hypernym's definition, and vice versa. Based on our high-quality training data, this can provide direct clues for inferring taxonomic relation between the terms.

The baseline system over (d_x, d_y) may provide an alternative evidence for interpreting the terms. Inherent structure information in definitive sentences is often important to understand term meanings for the purpose of accurately detecting taxonomic relations. The assumption in the background is that: if two sentences represent the features

of fine-grained alignments in the structure in some level, their corresponding terms may hold the taxonomic relation. For example, the definition of term "*apple*" and "*Malus*" in WordNet are "*An apple is a sweet, edible fruit produced by an apple tree (Malus domestica)*", "*apple trees; found throughout temperate zones of the northern hemisphere.*" respectively. Actually, "*Malus*" is a hypernym of "*apple*".

Heuristic Matching. Inspired in part by the ideas of natural language inference provided by [14,18], we combine the output from the baseline system into a joint vector representation via different strategies:

$$\boldsymbol{p} = [\mathbf{p}^{tt}; \mathbf{p}^{dd}; \mathbf{p}^{td} - \mathbf{p}^{dt}; \mathbf{p}^{td} \odot \mathbf{p}^{dt}], \tag{9}$$

where semicolons refer to the concatenation of multiple column vectors, $\odot$ denotes the element-wise multiplication of two vectors, $\boldsymbol{p} \in \mathbb{R}^{4d_h}$ is the output of this layer, and d_h is the number of hidden units of the LSTMs.

Softmax Output. Since our task refers to predict whether there exists or not a taxonomic relation for the given term pair, it is modeled as a binary classification problem. Thus, the feature vector $\boldsymbol{p}$ in Eq. (9) is fed into a softmax classifier for computing the confidence of each output result:

$$\boldsymbol{o} = \boldsymbol{W}_1(\boldsymbol{p} \circ \boldsymbol{r}) + \boldsymbol{b}. \tag{10}$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{n_1 \times 4d_h}$ is the transformation matrix, and $\boldsymbol{o} \in \mathbb{R}^{n_1}$ is the final output of the network. n_1 is equal to 2.

Loss Function and Training. Given an input instance $x_i = (x_{hypo}, d_x, y_{hyper}, d_y, 1/0)$, our model with parameter θ outputs the vector $\boldsymbol{o}$ computed in Eq. (10), which is a 2-dimensional vector, where the i-th value o_i of $\boldsymbol{o}$ is the probability score for determining the taxonomic relation of term pair (x_{hypo}, y_{hyper}) is holds or not, and the sum of values in $\boldsymbol{o}$ is to 1. To obtain the conditional probability $p(i|x_i, \theta)$, we apply a softmax operation as:

$$p(i|x_i, \theta) = \frac{e^{o_i}}{\Sigma_k e^{o_k}}, \tag{11}$$

Given all of our training instances $\boldsymbol{T} = \{x_i \mid i = 1, 2, ..., N\}$, we can then define the negative log-likelihood loss function:

$$J(\theta) = -\sum_{i=1}^{N} log\ p(y_i|x_i, \theta), \tag{12}$$

where θ denotes the parameters in our model. We train the model through a simple optimization technique called stochastic gradient descent (SGD) over shuffled mini-batches with the Adadelta rule. Regularization is implemented by a dropout [11] and L_2 norm.

4 Experiments and Analysis

In this section, we first provide an overview of datasets and the pre-trained method for word embeddings in Sect. 4.1. We then describe the experimental settings in Sect. 4.2 and investigate the performance results of different methods on both specific and open domain datasets in Sect. 4.3. Finally, the error analysis is discussed in Sect. 4.4.

Table 1. Dataset used in the experiments

Dataset	#Train		#Test		#Validation	
Splits	Random splits	Lexical splits	Random splits	Lexical splits	Random splits	Lexical splits
BLESS [3]	12459	757	2376	675	404	103
Conceptual Graph (See footnote 4)	58484	29475	19610	7808	4079	2095
WebIsA-Animal (See footnote 5)	5614	3784	1942	1021	407	249
WebIsA-Plant (See footnote 5)	5534	2933	1610	861	305	169

4.1 Dataset

Random and Lexical Dataset Splits. As pointed out by Levy *et al*. [13], mostly supervised distributional lexical inference methods tend to learn a dependent semantics of a single term, instead of learning the relation between two terms, this can be expressed as "lexical memorization" phenomenon. To address this, Levy *et al*. [13] made a suggestion of splitting the train and test sets such that each of them will contain distinct term pairs for presenting the model from overfitting during training.

To investigate such behaviors, we also follow the solution for a lexical split of our dataset. In this case, we maintain roughly a ratio of 14:5:1 for training set, test set and validation set partitioned randomly. Moreover, we maintain roughly a ratio of 8:1 for positive instances and negative instances in random or lexical splits in our datasets. The overall statistics of the datasets used in the experiments are summarized in Table 1. We present briefly the summary of each dataset below.

- **BLESS** dataset [3][3]. It consists of 200 distinct, unambiguous concepts. Each of which is involved with other terms, called *relata*, in some relations. We extract from BLESS 12,994 pairs of terms for the following four types of relations: taxonomic relation, "meronymy" (a.k.a. part-of relation), "coordinate" (i.e., two terms having the same hypernym), and random relations. From these term pairs, we set taxonomic relations as positive instances, while other relations form the negative instances.
- **Conceptual Graph**. This is a popular taxonomic benchmark dataset derived from Microsoft Concept Graph project[4]. It contains more than 5 million unique concepts, 12 unique million entities and 85 million taxonomic relations. We randomly pick the term pairs with possible (direct and indirect) taxonomic relation, along with high frequencies as the positive instances, while the term pairs are as negative instances when their relation frequencies are relatively lower.
- **WebIsA-Animal, Plant** dataset. WebIsA[5] is a publicly large-scale database containing more than 400 million taxonomic relation pairs. In this work, we select

[3] https://sites.google.com/site/geometricalmodels/shared-evaluation.
[4] https://concept.research.microsoft.com/Home/Download.
[5] http://webdatacommons.org/isadb/.

1.1M subset pertaining to two specified domains (i.e., the classes like "Animal" and "Plant"). The positive instances are created by extracting all possible (direct and indirect) taxonomic relation from the taxonomies. The negative instances are generated by randomly pairing two terms do not have any taxonomic relation.

Term Definition Collection. In this work, we pick two types of structured knowledge sources, including WordNet and the complete English Wikipedia, for extracting definitive descriptions for term pairs. We chose WordNet as the source of textual definitions partly because WordNet has been used for related tasks before, e.g., Snow *et al.* [26] constructed a larger taxonomic relation dataset based on WordNet, and partly because a number of accurate definitions of terms in sense level are available in WordNet. However, some literatures, e.g., Shwartz *et al.* [24], claimed that only limited coverage for almost all knowledge resources particularly to several rare or recently pairs, e.g., ("*Bullet tuna*", "*fish*"), ("*lead acid battery*", "*automobile*"). As the English Wikipedia can be viewed as the complementary data source to WordNet. Concretely, for each term pair in the training set, we first try to extract their respective definition from WordNet based on the term in strings[6]. For a few pairs which contain terms not covered by WordNet. We then switch to Wikipedia in which the term can be involved, and select the top-2 sentences in the first subgraph in the introductory sections, as its definitive description. If we are failed in two knowledge resources, we set definitions the same as the term in strings. As a result, we obtain the training instances in the form of (x_{hypo}, d_x, y_{hyper}, d_y, 1/0) for our experiments, where each term is accompanied by its definition.

Pre-trained Word Embeddings. To cover abundant words in the terms and definitions and provide the better support for initial input of our model. We use two large-scale textual corpus to train word embeddings. The first contains 570M entity content pages consisting of approximately 40 million sentences extracted from the English wikipedia[7]. The second is larger in size and derived from extended abstracts of DBpedia[8], then we employ the NLPIR system[9] for segmentation and Skip-gram method [15] for training. Note that the out-of-vocabulary words in the training set are randomly initialized with sampling values that meet the uniformly distributional representation in the range of $(-0.05, 0.05)$. As to the word or phrase with more than one word, we treat it as a whole to learning word embedding, instead of using character-level embeddings.

Pre-trained embeddings as preliminary inputs used in our model are not being updated in training, mainly for two reasons: (i) Reduce the number of needed parameters in the training stage; (ii) Improve the generalization capability of the model as it ensures that the words in training and the new words in the test set lie in the same space.

[6] As WordNet sorts sense definitions by sense frequency [17], we only choose the top-1 sense definition to denote a term.

[7] https://dumps.wikimedia.org/wikidatawiki/20180320/.

[8] http://tagesnetzwerk.de.

[9] https://github.com/NLPIR-team/NLPIR.

4.2 Experimental Settings

Compared Methods. In a series of previous works [12,26,27], several pattern-based, text-based inference methods have been applied for taxonomic relation identification. Their experiments showed that these methods achieve the F-score lower than 65% in most cases, which are not suggested to be strong baselines to compare with our approach. To make the convincing conclusion, in the experiments, we use the following competitive baseline methods for comparison:

- **Word2Vec + SVM.** This model first obtain the two term embeddings by applying the Skip-gram method [15] on the same corpus used for training pre-trained word embeddings as ours, and then combine their vectors to train an off-the-shelf SVM classifier for the taxonomic relation identification. Note that, in the Skip-gram model, if a term with more than one word, its embedding is calculated as the average of all words in the term.
- **DDM + SVM** [29]**.** This learns term embeddings via a dynamic distance-margin model, and then a SVM classifier is trained on concatenation of term pair vectors for the taxonomic relation detection.
- **DWNN + SVM** [1]**.** This is a extended copy of **DDM** [29], it not only utilizes the information of hypernyms and hyponyms, but also considers the contextual information between them via a dynamic weighting neural network when learning term embeddings.
- **Best unsupervised (dependency-based context)** [25]**.** This is the best unsupervised method, which implements similarity measurement over weighted dependency based context vectors.
- **Ours**$_{SubInput}$**.** This is the variant of our method. The collection $\{(x_{hypo}, y_{hyper}), (x_{hypo}, d_y), (d_x, y_{hyper}), (d_x, d_y)\}$ provided for each input vector pair is changed to its sub-collection $\{(x_{hypo}, y_{hyper}), (d_x, d_y)\}$. We use the sub-script *SubInput* to denote this setting.
- **Ours**$_{Concat}$**.** This is the variant of our method. To form the joint representation to distributional features, we directly concatenate the output of the four separate representations from term pairs as follows: $\mathbf{p}^{tt}$ from (x_{hypo}, y_{hyper}), $\mathbf{p}^{td}$ from (x_{hypo}, d_y), $\mathbf{p}^{dt}$ from (d_x, y_{hyper}), and $\mathbf{p}^{dd}$ from (d_x, d_y), rather than relying on the heuristic matching. We use the sub-script *Concat* to denote this setting.

Evaluation Metrics. We observed that diverse relatively straightforward, generic domain terms and entities especially in the open domain datasets, are usually used for hyponyms, may correspond to the same hypernyms, e.g., ("*volleyball*", "*sport*"), ("*hockey*", "*sport*"), ("*fishing*", "*sport*"), ("*overfishing*", "*sport*"). We therefore ranked these term pairs by the frequency of hypernym, and report Precision at rank 200 (P@200), Recall at rank 200 (R@200), and F-score at rank 200 (F@200). On the one hand, it benefits to further evaluate the effectiveness of our approach by encoding term definitions, because they are usually near-hyponyms in these term pairs, but the respective corresponding taxonomic relation may not be true. On the other hand, it is well-suited for validating the capability of discriminating coordinate relations from other

relations in the datasets for different methods. As for the Lexical Splits, we report the Mean Average Precision (P), Mean Average Recall (R), and Mean Average F-score (F).

Parameter Tuning. We conducted extensive experiments to determine the optimal configuration of parameters for our model. There are two types of parameters in the model: the first type includes weights and biases in the model layers, which can be initiated randomly and learned afterwards from each iteration; the second type includes the parameters that should be configured manually. In particular, we select six most common hyper-parameters for our model, namely the dimension of word embedding d_m in the input representation layer, the number of hidden units of all the LSTMs d_h, convolutional filters length Fl, number of filters Nr in the CNN component of the baseline system, the learning rate lr, and the ratio of dropout ρ. Since the weight **W** and bias **b** in each neural layer can be learned automatically via the evolution of model network, we focus on turning the hyper-parameters d_m, d_h, Fl, Nr, lr, and ρ. In practice, we train our models with a batch size of 128 for at most 100 epochs for each experiment and performed parameter selection strategy[10] on the validation set to tune parameters of the model for better convergence. Finally, we set $d_m = 300$, $d_h = 300$, $Fl = 3$, $Nr = 2$, $lr = 10^{-4}$, $\rho = 0.1$, and early-stopping on validation accuracy. Our model is implemented using the TensorFlow[11] machine learning framework.

4.3 Performance on Specific and Open Domain Datasets

For domain-specific datasets (i.e., WebIsA-Animal (See footnote 5) and WebIsA-Plant (See footnote 5)), the experimental results are given in Table 2. On a random split, we can see that **Ours** method achieves significantly improvements on the average of F-score by 8.1% and 14.8% compared to the **Word2Vec** and **DDM** methods. This indicates that **Ours** method is effective to further improving the performance of our task. Moreover, although **DWNN** is similar to **DDM**, the average F-measure is improved by 10.2% compared to **DDM** as it considers both the meanings of hypernym and hyponym as well as the contextual information between them in the process of the term embedding learning, while **DDM** ignores the effects of contextual information for taxonomic relation identification. **Best unsupervised** method achieves the average F-score 73.3% and is not effective for our task due to the heavily depending on the availability of a large enough corpus that can provide dependency-based context (i.e., considering a syntactic distributional space for terms by computing their neighbors in the dependency parse tree).

Similarity, for open domain datasets (BLESS [3] and Conceptual Graph (See footnote 4)), the experimental results are given in Table 3. On a random split, our method improves the average F-score by 11.6% compared to **Word2Vec**, by 3.5% compared to **DDM**, and by 2.3% compared to **DWNN** methods. The **Word2Vec** embeddings are mainly depend on co-occurrence based similarity, which is not effective for the classifier to accurately identify the whole of the taxonomic relations. **DDM** only learns term embeddings from the distributional contexts from the corpus, **DWNN** takes more

[10] We employ grid search for a range of hyper-parameters, and picked the combination of ones that yield the highest F-score on the validation set.

[11] https://www.tensorflow.org/.

Table 2. Performance comparison of different methods on domain-specific datasets (including WebIsA-Animal and WebIsA-Plant). We report Precision at rank 200 (P@200), Recall at rank 200 (R@200), F-score at rank 200 (F@200) for Random Splits, and Mean Average Precision (P), Mean Average Recall (R), Mean Average F-score (F) for Lexical Splits. The best performance in the F-score column is boldfaced (the higher, the better).

Datasets	WebIsA-Animal						WebIsA-Plant					
	Random splits (@200)			Lexical splits			Random splits (@200)			Lexical splits		
Method	P	R	F	P	R	F	P	R	F	P	R	F
Previous methods												
Word2Vec + SVM	0.796	0.706	0.748	0.785	0.674	0.725	0.817	0.730	0.771	0.708	0.623	0.663
DDM + SVM	0.706	0.620	0.660	0.655	0.521	0.580	0.759	0.695	0.726	0.712	0.517	0.599
DWNN + SVM	0.893	0.714	0.794	0.820	0.550	0.658	0.916	0.705	0.797	0.875	0.689	0.771
Best unsupervised	0.897	0.625	0.737	0.730	0.510	0.600	0.827	0.650	0.728	0.702	0.609	0.652
Our method and its variants												
Ours$_{SubInput}$	0.693	0.747	0.719	0.617	0.404	0.488	0.677	0.722	0.699	0.618	0.689	0.652
Ours$_{Concat}$	0.877	0.707	0.783	0.734	0.637	0.682	0.895	0.699	0.785	0.752	0.645	0.694
Ours	0.914	0.755	**0.827**	0.892	0.697	**0.783**	0.920	0.799	**0.855**	0.881	0.692	**0.775**

consideration on the contextual information between the hypernym and its hyponym, rather than their respective meanings. While our method can learn more indicative features related to the taxonomic relations under the guidance of both the distributional contexts and encoded definitions. Besides, we observe an interesting scenario is that the Precision of **DDM** improves significantly in the open domain datasets compared to the specific domain datasets. One possible explanation is that the method learns term embeddings using pre-extracted taxonomic relations from Probase, and if a relation dose not in Probase, there is high possibility that it becomes a negative instance and be recognized as a non-taxonomic relation by the classifier. Therefore, the training data extracted from Probase plays an import role in this method. For open domain datasets (BLESS [3] and Conceptual Graph (See footnote 4)), there are approximately 75%–85% of taxonomic relations in these datasets found in Probase, while only approximately 25%–45% of relations in domain-specific datasets (WebIsA-Animal (See footnote 5) and WebIsA-Plant (See footnote 5)) can be found in Probase. Therefore, **DDM** achieves better performance in the open domain datasets than the specific ones. Our approach, in contrast, mainly depends on the valuable evidence - corpus-independent textual definitions, thus, it has better generalization capability and could achieve higher F-score in domain-specific datasets.

4.4 Error Analysis

We analyse two categories of error cases in the experiments, including inaccurate definitions and other relations[12].

[12] In our settings, meronymy, coordinate relations, and term pairs with reversed positions of the hypernym and hyponym (shortly denoted as reversed error), are considered as other relations.

Table 3. Performance comparison of different methods on open domain datasets (including BLESS and Conceptual Graph). We report Precision at rank 200 (P@200), Recall at rank 200 (R@200), F-score at rank 200 (F@200) for Random Splits, and Mean Average Precision (P), Mean Average Recall (R), Mean Average F-score (F) for Lexical Splits. The best performance in the F-score column is boldfaced (the higher, the better).

Datasets	BLESS						Conceptual Graph					
	Random splits (@200)			Lexical splits			Random splits (@200)			Lexical splits		
Method	P	R	F	P	R	F	P	R	F	P	R	F
Previous methods												
Word2Vec + SVM	0.719	0.693	0.706	0.702	0.648	0.674	0.750	0.629	0.684	0.699	0.608	0.650
DDM + SVM	0.839	0.744	0.789	0.785	0.684	0.731	0.889	0.669	0.763	0.764	0.675	0.717
DWNN + SVM	0.914	0.677	0.778	0.792	0.545	0.646	0.930	0.697	0.797	0.885	0.640	0.743
Best unsupervised	0.654	0.590	0.620	0.675	0.541	0.601	0.731	0.557	0.632	0.702	0.607	0.651
Our method and its variants												
Ours$_{SubInput}$	0.675	0.659	0.670	0.621	0.600	0.610	0.683	0.623	0.652	0.608	0.572	0.589
Ours$_{Concat}$	0.864	0.719	0.785	0.803	0.677	0.735	0.874	0.760	0.813	0.760	0.679	0.717
Ours	0.871	0.723	**0.790**	0.811	0.694	**0.748**	0.899	0.775	**0.832**	0.854	0.728	**0.786**

Inaccurate Definition. The error statistics show that this kind of error account for approximately 78%, 83% and 86% total errors in BLESS [3], Conceptual Graph (See footnote 4), and WebIsA (See footnote 5), respectively. For example, our model obtains the definition "*fruit with red or yellow or green skin and sweet to tart crisp whitish flesh*" for the term "*Apple*" in the pair ("*Apple*", "*IT company*"), however, a correct detection may require another definition "*Apple Inc. is an American multinational technology company headquartered in Cupertino*", which comes from the article's abstract of the English Wikipedia[13]. This is a common issue due to the ambiguity of entity mentions. To alleviate this problem, we will explore more advanced entity liking techniques, or extract more accurately one from all highly related definitions by combining current context, along with the efficient ranking algorithm.

As to the roughly expression or misleading information errors appearing in definitions. We shall illustrate two term pairs for further analysis. The definition of "*volleyball*" in the first term pair ("*volleyball*", "*game*") is: "*a game in which two teams hit an inflated ball over a high net using their hands*", and the initial prediction result of our model is the value of 1. If we removed the phrase "*a game*" in the definition, and then the model outputs the value of 0 as the prediction result. Similarly, in the second term pair ("*mexico*", "*latin american country*"), the initial prediction result of our model is the value of 1, and the definition of "*mexico*" is: "*a republic country in the southern of latin America*". If we removed the phrase "*of latin America*" in the definition, and then our model outputs the value of 0 as the prediction result. These demonstrate that our model is more sensitive for the prominent context words which depict the taxonomic relation in the definitions. Meanwhile, the definitions indeed provide more rich

[13] https://en.wikipedia.org/wiki/Apple_Inc.

knowledge for understanding the term pairs. But when we cannot obtain the enough information in the definitions of the terms, our model is not enough intelligent to avoid either inaccurate or misleading errors without human-crafted knowledge.

Apart from the above situation, due to parts of term and entities pairs are rare ones, e.g., ("*coma*", "*knowledge*"), ("*bacterium*", "*microorganism*"), ("*chromium*", "*metal*"). As a consequence, it is difficult for the model to make a correct decision alone when only encoding their term meanings as the definitions in our model.

Other Relations. In this case, the majority of errors stem from confusing meronymy and taxonomic relations. For example, in fact, the term pair ("*paws*", "*cat*") is of the meronymy relation, rather than the taxonomic relation. We found that such a problem has been also reported in [23], and one possible solution for reducing this error is: adding more negative instances of this kind to the datasets.

The remained errors in the case are with respect to the type of the reversed error. In order to reduce most errors of this type, based on the previous literature study [29], one possible solution is: integrating the learning of term embeddings with the distance measure as the feature (e.g., 1-norm distance) into the model.

5 Conclusion

In this paper, we presented a neural network model, which can enhance the representations of term pairs by incorporating their separative accurately textual definitions, for identifying the taxonomic relation of pairs. In our experiments, we showed that our model outperforms several competitive baseline methods and achieves more than 82% F-score on two domain-specific datasets. Moreover, our model, once trained, performs competitively in various open domain datasets. This demonstrates the good generalization capacity of our model. Apart from this, we also conducted detailed analysis to give more insights on the error distribution.

In the future, our work can be extended by addressing the following issues: one is to consider how to integrate multiple types of knowledge (e.g., word meanings, definitions, knowledge graph paths, and images) to enhance the representations of term pairs and further improve the performance of this work. Since our model seems straightforwardly applicable for multi-class classification problem via some tuning, hence, the other work is to investigate whether this model would be used to the task of multiple semantic relations classification.

Acknowledgments. This work is supported by the National Natural Science Foundation of China (61572353, 61402323), the National High-tech R&D Program of China (863 Program) (2013AA013204), and the Natural Science Foundation of Tianjin (17JCYBJC15400).

References

1. Anh, T.L., Tay, Y., Hui, S.C., Ng, S.K.: Learning term embeddings for taxonomic relation identification using dynamic weighting neural network. In: EMNLP, pp. 403–413 (2016)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: ICLR (2015)

3. Baroni, M., Lenci, A.: How we blessed distributional semantic evaluation. In: Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics, pp. 1–10 (2011)
4. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12**, 2493–2537 (2011)
5. Fu, R., Guo, J., Qin, B., Che, W., Wang, H., Liu, T.: Learning semantic hierarchies via word embeddings. In: ACL (Volume 1: Long Papers), pp. 1199–1209 (2014)
6. Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Netw. **18**(5–6), 602–610 (2005)
7. Harabagiu, S.M., Maiorano, S.J., Paşca, M.A.: Open-domain textual question answering techniques. Nat. Lang. Eng. **9**(3), 231–267 (2003)
8. Harris, Z.S.: Distributional structure. Word **10**(2–3), 146–162 (1954)
9. Hearst, M.A.: Automatic acquisition of hyponyms from large text corpora. In: COLING, pp. 539–545. Association for Computational Linguistics (1992)
10. Kiela, D., Rimell, L., Vulić, I., Clark, S.: Exploiting image generality for lexical entailment detection. In: ACL-IJCNLP (Volume 2: Short Papers), pp. 119–124 (2015)
11. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: ICLR (2015)
12. Kotlerman, L., Dagan, I., Szpektor, I., Zhitomirsky-Geffet, M.: Directional distributional similarity for lexical inference. Nat. Lang. Eng. **16**(4), 359–389 (2010)
13. Levy, O., Remus, S., Biemann, C., Dagan, I.: Do supervised distributional methods really learn lexical inference relations? In: NAACL, pp. 970–976 (2015)
14. Liu, Y., Sun, C., Lin, L., Wang, X.: Learning natural language inference using bidirectional LSTM model and inner-attention (2016). https://arxiv.org/abs/1605.09090
15. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: ICLR (Workshop Poster) (2013)
16. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS, pp. 3111–3119 (2013)
17. Miller, G.A.: WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
18. Mou, L., et al.: Natural language inference by tree-based convolution and heuristic matching. In: ACL (2014)
19. Nakashole, N., Weikum, G., Suchanek, F.: Patty: a taxonomy of relational patterns with semantic types. In: EMNLP-CoNLL, pp. 1135–1145 (2012)
20. Navigli, R., Velardi, P., Faralli, S.: A graph-based algorithm for inducing lexical taxonomies from scratch. In: IJCAI, pp. 1872–1877 (2011)
21. Neculoiu, P., Versteegh, M., Rotaru, M.: Learning text similarity with Siamese recurrent networks. In: Proceedings of the 1st Workshop on Representation Learning for NLP, pp. 148–157 (2016)
22. Santus, E., Lenci, A., Lu, Q., Schulte im Walde, S.: Chasing hypernyms in vector spaces with entropy. In: EACL, pp. 38–42 (2014)
23. Shwartz, V., Goldberg, Y., Dagan, I.: Improving hypernymy detection with an integrated path-based and distributional method. In: ACL, pp. 2389–2398 (2016)
24. Shwartz, V., Levy, O., Dagan, I., Goldberger, J.: Learning to exploit structured resources for lexical inference. In: CoNLL, pp. 175–184 (2015)
25. Shwartz, V., Santus, E., Schlechtweg, D.: Hypernyms under Siege: linguistically-motivated artillery for hypernymy detection. In: EACL, pp. 65–75 (2017)
26. Snow, R., Jurafsky, D., Ng, A.Y.: Learning syntactic patterns for automatic hypernym discovery. In: NIPS, pp. 1297–1304 (2004)
27. Wong, M.K., Abidi, S.S.R., Jonsen, I.D.: A multi-phase correlation search framework for mining non-taxonomic relations from unstructured text. Knowl. Inf. Syst. **38**(3), 641–667 (2014)

28. Wu, W., Li, H., Wang, H., Zhu, K.Q.: Probase: a probabilistic taxonomy for text understanding. In: SIGMOD, pp. 481–492 (2012)
29. Yu, Z., Wang, H., Lin, X., Wang, M.: Learning term embeddings for hypernymy identification. In: IJCAI, pp. 1390–1397 (2015)
30. Zeng, D., Liu, K., Lai, S., Zhou, G., Zhao, J., et al.: Relation classification via convolutional deep neural network. In: COLING, pp. 2335–2344 (2014)