Learning Semantic Hierarchies: A Continuous Vector Space Approach

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu

***Abstract*—Semantic hierarchy construction aims to build structures of concepts linked by hypernym–hyponym (“is-a”) relations. A major challenge for this task is the automatic discovery of such relations. This paper proposes a novel and effective method for the construction of semantic hierarchies based on continuous vector representation of words, named word embeddings, which can be used to measure the semantic relationship between words. We identify whether a candidate word pair has hypernym–hyponym relation by using the word-embedding-based semantic projections between words and their hypernyms. Our result, an F-score of 73.74%, outperforms the state-of-the-art methods on a manually labeled test dataset. Moreover, combining our method with a previous manually built hierarchy extension method can further improve F-score to 80.29%.**

***Index Terms*—Piecewise linear projections, semantic hierarchy, word embedding.**

I. Introduction

SEMANTIC hierarchies are natural ways to organize knowledge. They are the main components of ontologies or semantic thesauri [1], [2]. In the WordNet hierarchy, senses are organized according to the “is-a” relations. For example, “`dog`” and “`canine`” are connected by a directed edge. Here, “`canine`” is called a hypernym of “`dog`.” Conversely, “`dog`” is a hyponym of “`canine`.” As key sources of knowledge, semantic thesauri and ontologies can support many natural language processing applications. However, these semantic resources are limited in its scope and domain, and their manual construction is knowledge intensive and time consuming. Therefore, many researchers have attempted to automatically extract semantic relations or to construct taxonomies.

Manuscript received July 31, 2014; revised November 03, 2014; accepted November 16, 2014. Date of current version February 26, 2015. This work was supported by the National “863” Project under Grant 2015AA011808 and the National Natural Science Foundation of China (NSFC) under Grants 61133012 and 61273321. R. Fu and J. Guo contributed equally to the paper. The guest editor coordinating the review of this manuscript and approving it for publication was Prof. Haizhou Li. *(Corresponding author: T. Liu.)*

R. Fu was with the School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China. He is now with iFLytek Co, Ltd, Hefei 230088, China (e-mail: rjfu@ir.hit.edu.cn).

J. Guo, B. Qin, W. Che, and T. Liu are with the School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China (e-mail: jguo@ir.hit.edu.cn; bqin@ir.hit.edu.cn car@ir.hit.edu.cn; tliu@ir.hit.edu.cn).

H. Wang is with Baidu, Inc., Beijing 100085, China (e-mail: wanghaifeng@baidu.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASLP.2014.2377580

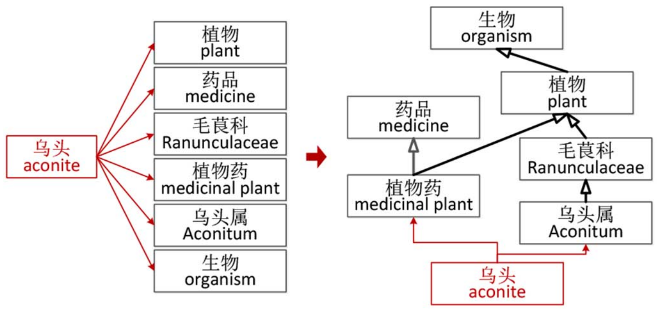


Fig. 1. An example of semantic hierarchy construction.

A major challenge for this task is the automatic discovery of hypernym-hyponym relations. In our previous work [3], we propose a distant supervision method to extract hypernyms for entities from multiple sources. The output of the model is a list of hypernyms for a given entity (left panel, Fig. 1). However, there usually also exists hypernym–hyponym relations among these hypernyms. For instance, “ (`plant`)” and “ (`Ranunculaceae`)” are both hypernyms of the entity “ (`aconit`),” and “; (`plant`)” is also a hypernym of “ (`Ranunculaceae`).” Given a list of hypernyms of an entity, our goal in the present work is to construct a semantic hierarchy of these hypernyms (right panel, Fig. 1).[1]

Some previous works extend and refine manually-built semantic hierarchies by using other resources (e.g., Wikipedia) [2]. However, the coverage is limited by the scope of the resources. Several other works relied heavily on lexical patterns, which would suffer from deficiency because such patterns can only cover a small proportion of complex linguistic circumstances [4], [5]. Besides, distributional similarity methods [6], [7] are based on the assumption that a term can only be used in contexts where its hypernyms can be used and that a term might be used in any contexts where its hyponyms are used. However, it is not always rational. Our previous method based on web mining [3] works well for hypernym extraction of entity names, but it is unsuitable for semantic hierarchy construction which involves many words with broad semantics. Moreover, all of these methods do not use the word semantics effectively.

This paper proposes a novel approach for semantic hierarchy construction based on a special kind of word representations, named word embeddings. Word embeddings, also known as distributed word representations, typically represent words with dense, low-dimensional and continuous-valued vectors (more details will be introduced in Section III-B). Word embeddings have been empirically shown to preserve linguistic regularities,

[1]In this study, we focus on Chinese semantic hierarchy construction. The proposed method can be easily adapted to other languages.

such as the semantic relationship between words [8]. For example, $v(\texttt{king}) - v(\texttt{queen}) \approx v(\texttt{man}) - v(\texttt{woman})$, where $v(\omega)$ is the embedding of the word ω. We observe that a similar property also applies to the hypernym–hyponym relationship (Section III-C), which is the main inspiration of the present study.

However, we further observe that hypernym–hyponym relations are more complicated than a single offset can represent. To address this challenge, we propose a more sophisticated and general method—learning a linear projection which maps words to their hypernyms (Section III-C1). Furthermore, we propose a piecewise linear projection method based on relation clustering to better model hypernym–hyponym relations (Section III-C2). Subsequently, we identify whether an unknown word pair is a hypernym–hyponym relation using the projections (Section III-D). To the best of our knowledge, we are the first to apply word embeddings to this task.

For evaluation, we manually annotate a dataset containing 418 Chinese entities and their hypernym hierarchies, which is the first dataset for this task as far as we know. The experimental results show that our method achieves an F-score of 73.74% which significantly outperforms the previous state-of-the-art methods. Moreover, combining our method with the manually-built hierarchy extension method proposed by [2] can further improve F-score to 80.29%.

Some preliminary results have been reported at our previous paper [9]. This paper is an extended version. In this paper, we expand more details of our method, more background of word embeddings, detailed step descriptions of construction of evaluation data, more experiments and analysis.

II. Background

A. Semantic Hierarchies

As main components of ontologies, semantic hierarchies have been studied by many researchers. Some have established concept hierarchies based on manually-built semantic resources such as WordNet [1]. Such hierarchies have good structures and high accuracy, but their coverage is limited to fine-grained concepts (e.g., "`Ranunculaceae`" is not included in WordNet.). We have made similar observation that about a half of hypernym–hyponym relations are absent in a Chinese semantic thesaurus. Therefore, a broader range of resources is needed to supplement the manually built resources. In the construction of the famous ontology YAGO, [2] link the categories in Wikipedia onto WordNet. However, the coverage is still limited by the scope of Wikipedia.

Several other methods are based on lexical patterns. They use manually or automatically constructed lexical patterns to mine hypernym–hyponym relations from text corpora. A hierarchy can then be built based on these pairwise relations. The pioneer work by [4] has found out that linking two noun phrases (NPs) via certain lexical constructions often implies hypernym relations. For example, NP_1 is a hypernym of NP_2 in the lexical pattern "such NP_1 as NP_2." [5] propose to automatically extract large numbers of lexico-syntactic patterns and subsequently detect hypernym relations from a large newswire corpus. Their method relies on accurate syntactic parsers, and the quality of the automatically extracted patterns is difficult to guarantee. Generally speaking, these pattern-based methods often suffer from low recall or precision because of the coverage or the quality of the patterns.

The distributional methods assume that the contexts of hypernyms are broader than the ones of their hyponyms. For distributional similarity computing, each word is represented as a semantic vector composed of the pointwise mutual information (PMI) with its contexts. [6] design a directional distributional measure to infer hypernym–hyponym relations based on the standard IR Average Precision evaluation measure. [7] propose another measure focusing on the contexts that hypernyms do not share with their hyponyms. However, broader semantics may not always infer broader contexts. For example, for terms "`Obama`" and "`American people`", it is hard to say whose contexts are broader.

Our previous work [3] applies a web mining method to discover the hypernyms of Chinese entities from multiple sources. They assume that the hypernyms of an entity co-occur with it frequently. It works well for named entities. But for class names (e.g., singers in Hong Kong, tropical fruits) with wider range of meanings, this assumption may fail.

In this paper, we aim to identify hypernym–hyponym relations using word embeddings, which have been shown to preserve good properties for capturing semantic relationship between words.

B. Word Embeddings

Different from the conventional *one-hot* word representation,[2] words can also be embedded into a low-dimensional (e.g. 300) and continuous vector space using either context-predicting models, such as neural network language models [8], [10]–[12], or spectral methods such as canonical correlation analysis [13].

Such kind of word representation is called word embedding, which is first proposed by [10]. In general, word embeddings are designed to capture *attributional similarities* [14] between words in the vocabulary: words that appear in similar contexts will be distributed close to each other in the embedding space.

In recent years, word embeddings have gained more and more interest in natural language processing for several reasons, one of which is that word embeddings preserve rich and useful linguistic regularities. [8] first demonstrated that many syntactic/semantic relations of words can be recovered by means of vector arithmetic in the embedding space learned by recurrent neural network language models. For example, let's again denote the embedding of a word ω as $v(\omega)$, then we have $v(\texttt{Russia}) - v(\texttt{Moscow}) \approx v(\texttt{China}) - v(\texttt{Beijing})$, which indicates a semantic relational similarity. On the other hand, we also have $v(\texttt{apples}) - v(\texttt{apple}) \approx v(\texttt{cars}) - v(\texttt{car})$, which indicates a syntactic relational similarity. It was later shown that relational similarities can also be recovered by other context-predicting architectures [12], [15]. [16] further studied this problem and

[2]A feature vector of the same size of the vocabulary, and only one dimension is on.

showed that the sparse and explicit word vectors learned by distributional semantic approach also capture the relational similarities well.

Therefore, by using simple vector arithmetic, one could apply the relation and solve analogy questions of the form "a is to a^* as b is to _ " by finding the nearest word to the vector $a - a^* + b$. We can also answer questions like "does a and b satisfy a specific relation R?" if we already have some seeded word pairs of relation R. Here, a specific relation is represented by the vector offset of a word pair (a, a^*).

In this study, we will show how to effectively exploit word embeddings to model the hypernym-hyponym relations. We show that the hypernym-hyponym relations are much more complicated than simple vector offsets can capture. They should be decomposed to more fine-grained relations and then be modeled separately. Due to the complexity of hypernym-hyponym relation, solving the analogy question mentioned above is not straightforward, since we cannot accurately sample a word pair (a, a^*) that is exactly representative of the specific hypernym-hyponym relation we are interested in. To address this problem, we propose a learning-based framework for determining hypernym-hyponym relationship in this study (Section III).

III. Method

In this section, we first define the task formally. Then we elaborate on our proposed method composed of three major steps, namely, word embedding training, projection learning, and hypernym–hyponym relation identification.

A. Task Definition

Given a list of hypernyms of an entity, our goal is to construct a semantic hierarchy on it (Fig. 1). We represent the hierarchy as a directed graph G, in which the nodes denote the words, and the edges denote the hypernym–hyponym relations. Hypernym-hyponym relations are *asymmetric* and *transitive* when words are unambiguous:

- $\forall x, y \in L : x \xrightarrow{H} y \Rightarrow \neg(y \xrightarrow{H} x)$
- $\forall x, y, z \in L : (x \xrightarrow{H} z \wedge (z \xrightarrow{H} y) \Rightarrow x \xrightarrow{H} y$

Here, L denotes the list of hypernyms. x, y and z denote the hypernyms in L. We use $\xrightarrow{H}$ to represent a hypernym–hyponym relation in this paper. Actually, x, y and z are unambiguous as the hypernyms of a certain entity. Therefore, G should be a directed acyclic graph (DAG).

B. Word Embedding Training

Various models for learning word embeddings have been proposed, including neural net language models [8], [10], [11] and spectral models [13]. More recently, [12] propose two log-linear models, namely the *Skip-gram* and *CBOW* model, to efficiently induce word embeddings. These two models can be trained very efficiently on a large-scale corpus because of their low time complexity. Additionally, their experiment results have shown that the *Skip-gram* model performs best in identifying semantic relationship among words. Therefore, we employ the *Skip-gram* model for estimating word embeddings in this study.

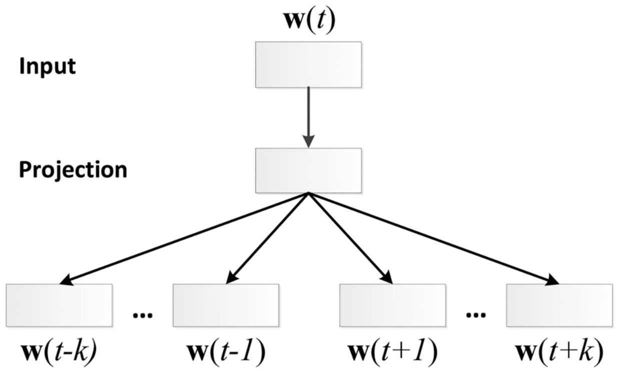


Fig. 2. The architecture of *Skip-gram* model [12].

The *Skip-gram* model adopts log-linear classifiers to predict context words given the current word w as input. First, w is projected to its embedding $v(w)$. Then, a log-linear classifier is employed, taking the embedding as input and predict the conditional probability distribution of w's context words within a certain range, e.g. k words in the left and k words in the right.

$$p(c|\omega;\theta) = \frac{exp(v(c)^\top v(\omega))}{\sum_{c' \in V} exp(v(c')^\top v(\omega))} \tag{1}$$

where c is one of the context words of w. The parameters θ are v_{c_i}, v_{w_i} for $w, c \in V$ and $i = 1, \ldots, d$. Then, the log-likelihood over the entire dataset can be computed as:

$$LL = \sum_{(w,c)\in D} \log p(c|w;\theta) \tag{2}$$

where D is the training datasest. The architecture of *Skip-gram* model can be illustrated as Fig. 2. After maximizing the log-likelihood, the embeddings are learned. We use the negative sampling [15] method for optimization,[3] and the asynchronous stochastic gradient descent algorithm (Asynchronous SGD) [18] for parallel weight updating.

C. Projection Learning

[8] observe that word embeddings preserve interesting linguistic regularities, capturing a considerable amount of syntactic/semantic relations. Looking at the well-known example: $v(\texttt{king}) - v(\texttt{queen}) \approx v(\texttt{man}) - v(\texttt{woman})$, it indicates that the embedding offsets indeed represent the shared semantic relation between the two word pairs.

We observe that the same property also applies to some hypernym–hyponym relations. As a preliminary experiment, we compute the embedding offsets between some randomly sampled hypernym–hyponym word pairs and measure their similarities. The results are shown in Table I.

The first two examples imply that a word can also be mapped to its hypernym by utilizing word embedding offsets. However, the offset from "`carpenter`" to "`laborer`" is distant from the one from "`gold fish`" to "`fish`," indicating that hypernym–hyponym relations should be more complicated than a single vector offset can represent. To verify this hypothesis, we compute the embedding offsets over all hypernym–hyponym

[3]In fact, negative sampling optimizes a different objective from the original *Skip-gram* log-likelihood. More details are analyzed in [17].

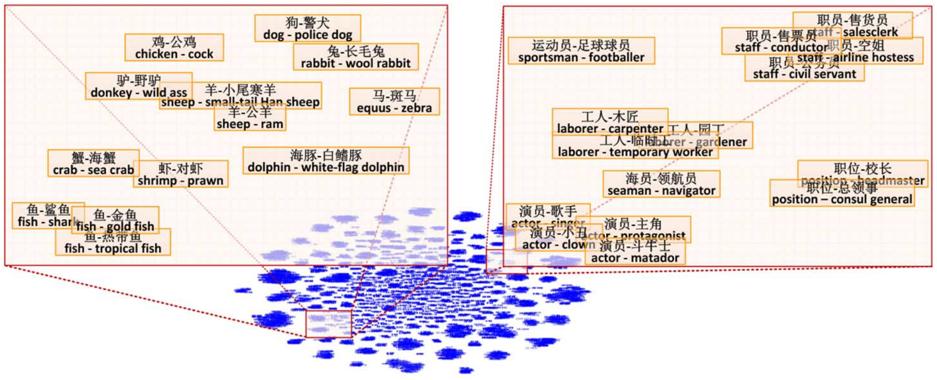


Fig. 3. Clusters of the vector offsets in training data. The figure shows that the vector offsets distribute in some clusters. The left cluster shows some hypernym–hyponym relations about animals. The right one shows some relations about people's occupations.

TABLE I
EMBEDDING OFFSETS ON A SAMPLE OF HYPERNYM-HYPONYM WORD PAIRS

No.	Examples
1	v(虾) $-$ v(对虾) $\approx$ v(鱼) $-$ v(金鱼) v(shrimp) $-$ v(prawn) $\approx$ v(fish) $-$ v(gold fish)
2	v(工人) $-$ v(木匠) $\approx$ v(演员) $-$ v(小丑) v(laborer) $-$ v(carpenter) $\approx$ v(actor) $-$ v(clown)
3	v(工人) $-$ v(木匠) $\not\approx$ v(鱼) $-$ v(金鱼) v(laborer) $-$ v(carpenter) $\not\approx$ v(fish) $-$ v(gold fish)

word pairs in our training data and visualize them.[4] Fig. 3 shows that the relations are well distributed in groups, which implies that hypernym–hyponym relations indeed can be decomposed into more fine-grained relations. Moreover, the relations about animals are spatially close, but separate from the relations about people's occupations.

To address this challenge, we propose to model the hypernym–hyponym relations by learning projection matrices which map the hypernyms to their hyponyms in continuous vector space.

A Uniform Linear Projection: Intuitively, we assume that all words can be projected to their hypernyms based on a uniform transition matrix. That is, given a word x and its hypernym y, there exists a matrix Φ so that $y = \Phi x$. For simplicity, we use the same symbols as the words to represent the embedding vectors. Obtaining a consistent exact Φ for the projection of all hypernym–hyponym pairs is difficult. Instead, we can learn an approximate Φ using Equation (3) on the training data, which minimizes the mean-squared error:

$$\Phi^* = \arg\min_{\Phi} \frac{1}{N} \sum_{(x,y)} \|\Phi x - y\|^2 \tag{3}$$

where N is the number of (x, y) word pairs in the training data. This is a typical linear regression problem. The only difference is that our predictions are multi-dimensional vectors instead of scalar values. We use SGD for optimization.

Piecewise Linear Projections: A uniform linear projection may still be under-representative for fitting all of the hypernym–hyponym word pairs, because the relations are rather diverse, as shown in Fig. 3. To better model the various kinds of hypernym–hyponym relations, we apply the idea of piecewise linear regression [19] in this study.

Specifically, the input space is first segmented into several regions. That is, all word pairs (x, y) in the training data are first clustered into several groups, where word pairs in each group are expected to exhibit similar hypernym–hyponym relations. Each word pair (x, y) is represented with their vector offsets: $y - x$ for clustering. The reasons are twofold: (1) Mikolov's work has shown that the vector offsets imply a certain level of semantic relationship. (2) The vector offsets distribute in clusters well, and the word pairs which are close indeed represent similar relations, as shown in Fig. 3.

Then we learn a separate projection for each cluster, respectively (Equation (4)).

$$\Phi_k^* = \arg\min_{\Phi_k} \frac{1}{N_k} \sum_{(x,y)\in C_k} \|\Phi_k x - y\|^2 \tag{4}$$

where N_k is the amount of word pairs in the k th cluster C_k.

We use the k-means algorithm for clustering, where k is tuned on a development dataset.

Training Data: To learn the projection matrices, we extract training data from a Chinese semantic thesaurus, Tongyi Cilin (Extended) (CilinE for short) which contains 100,093 words [20].[5] CilinE is organized as a hierarchy of five levels, in which the words are linked by hypernym–hyponym relations (right panel, Fig. 4). Each word in CilinE has one or more sense codes (some words are polysemous) that indicate its position in the hierarchy.

The senses of words in the first level, such as " (`object`)" and " (`time`)," are very general. The fourth level only has sense codes without real words. Therefore, we extract words in the second, third and fifth levels to constitute hypernym–hyponym pairs (left panel, Fig. 4).

Note that mapping one hyponym to multiple hypernyms with the same projection (Φx is unique) is difficult. Therefore, the pairs with the same hyponym but different hypernyms are expected to be clustered into separate groups. Fig. 4 shows that the word "`dragonfly`" in the fifth level has two hypernyms: "`insect`" in the third level and "`animal`" in the second level. Hence the relations `dragonfly` $\overset{H}{\rightarrow}$ insect and `dragonfly` $\overset{H}{\rightarrow}$ `animal` should fall into different clusters.

[4]Principal Component Analysis (PCA) is applied for dimensionality reduction.

[5]www.ltp-cloud.com/download/

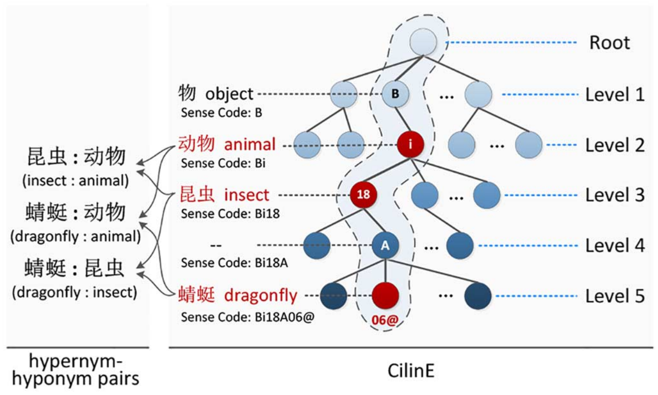


Fig. 4. Hierarchy of CilinE and an Example of Training Data Generation.

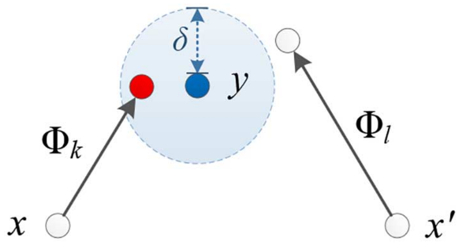


Fig. 5. In this example, $\Phi_k x$ is located in the circle with center y and radius δ. So y is considered as a hypernym of x. Conversely, y is not a hypernym of x'.

In our implementation, we apply this constraint by simply dividing the training data into two categories, namely, *direct* and *indirect*. Hypernym-hyponym word pair (x, y) is classified into the *direct* category, only if there doesn't exist another word z in the training data, which is a hypernym of x and a hyponym of y. Otherwise, (x, y) is classified into the *indirect* category. Then, data in these two categories are clustered separately.

D. Hypernym-Hyponym Relation Identification

Upon obtaining the clusters of training data and the corresponding projections, we can identify whether two words have a hypernym–hyponym relation. Given two words x and y, we find cluster C_k whose center is closest to the offset $y - x$, and obtain the corresponding projection Φ_k. For y to be considered a hypernym of x, one of the two conditions below must hold.

Condition 1: The projection Φ_k puts $\Phi_k x$ close enough to y (Fig. 5). Formally, the euclidean distance between $\Phi_k x$ and y: $d(\Phi_k x, y)$ must be less than a threshold δ.

$$d(\Phi_k x, y) = \|\Phi_k x - y\|^2 < \delta \tag{5}$$

Condition 2: There exists another word z satisfying $x \overset{H}{\to} z$ and $z \overset{H}{\to} y$. In this case, we use the transitivity of hypernym–hyponym relations.

Besides, the final hierarchy should be a DAG as discussed in Section III-A. However, the projection method cannot guarantee that theoretically, because the projections are learned from pairwise hypernym–hyponym relations without the whole hierarchy structure. All pairwise hypernym–hyponym relation identification methods would suffer from this problem actually. It is

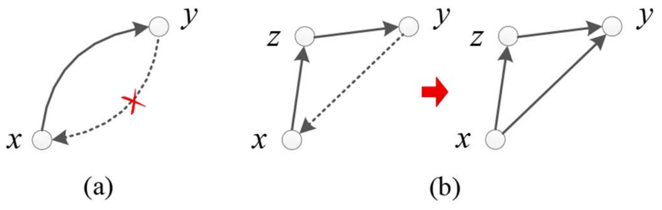


Fig. 6. (a) If $d(\Phi_j y, x) > d(\Phi_k x, y)$, we remove the path from y to x; (b) if $d(\Phi_j y, x) > d(\Phi_k x, z)$ and $d(\Phi_j y, x) > d(\Phi_i z, y)$, we reverse the path from y to x.

an interesting problem how to construct a globally optimal semantic hierarchy conforming to the form of a DAG. But this is not the focus of this paper. So if some conflicts occur, that is, a relation circle exists, we remove or reverse the weakest path heuristically (Fig. 6). If a circle has only two nodes, we remove the weakest path. If a circle has more than two nodes, we reverse the weakest path to form an *indirect* hypernym–hyponym relation.

IV. Experimental Setup

A. Settings for Training

In this work, we learn word embeddings from a Chinese encyclopedia corpus named Baidubaike[6], which contains about 30 million sentences (about 780 million words). The Chinese segmentation and part of speech tagging is provided by the open-source Chinese language processing platform LTP[7][20]. Then, we employ the *Skip-gram* method (Section III-B) to train word embeddings. Finally we obtain the embedding vectors of 0.56 million words.

The training data for projection learning is collected from CilinE (Section III-C3). We obtain 15,247 word pairs of hypernym–hyponym relations (9,288 for *direct* relations and 5,959 for *indirect* relations).

B. Construction of Evaluation Data

For evaluation, we collect the hypernyms for 418 entities, which are selected randomly from Baidubaike, following [3]. In that paper, we propose a method for finding hypernyms of Chinese open-domain entities from multiple sources. First, we collect candidate hypernyms from multiple sources for a given entity. Then, a statistical model is built for hypernym ranking based on a set of effective features.

Candidate Hypernym Collection: We collect candidate hypernyms with wide coverage from search results, encyclopedia category tags and the head word for a given entity. We search the entity using a search engine and count the co-occurrence frequency between the target entity and other words in the returned snippets and titles. We select top 10 frequent nouns (or noun phrases) as the main candidates.

Furthermore, the user-generated encyclopedia category tags are important clues if the entity exists in a encyclopedia. Thus we add these tags into the candidates. In this work, we consider

[6]Baidubaike (baike.baidu.com) is one of the largest Chinese encyclopedias containing more than 9.26 million entries as of July, 2014.

[7]www.ltp-cloud.com

TABLE II
THE FEATURES FOR HYPERNYM RANKING

Feature	Comment	Value Range
Prior	the prior probability of a candidate being a potential hypernym	[0, 1]
Is_Tag	whether a candidate is a category tag in the encyclopedia page of the entity if it exists	0 or 1
Is_Head	whether a candidate is the head word of the entity	0 or 1
In_Titles	some binary features based on the frequency of occurrence of a candidate in the document titles in the search results	0 or 1
Synonyms	the ratio of the synonyms of the candidate in the candidate list of the entity	[0, 1]
Radicals	the ratio of the radicals of characters in a candidate matched with the last character of the entity	[0, 1]
Source_Num	the number of sources where the candidate is extracted	1, 2, 3, or 4
Lexicon	the hypernym candidate itself and its head word	0 or 1

two Chinese encyclopedias, Baidubaike and another Chinese encyclopedia Hudongbaike[8], as hypernym sources.

In addition, the head words of entities are also their hypernyms sometimes. For example, the head word of " (Emperor Penguin)" indicates that it's a kind of " (penguins)". Thus we put head words into the hypernym candidates. In Chinese, head words are often laid after their modifiers. Therefore, we try to segment a given entity. If it can be segmented and the last word is a noun, we take the last word as the head word.

Hypernym Ranking: We propose a set of features (Table II) to build a Logistic Regression (LR) model to rank the candidate hypernyms on the training data collected automatically. The features include hypernym prior, source information (i.e. Is_Tag, Is_Head, and Source_Num), lexicon, and other additional information. Our previous work [3] shows that these features are effective for hypernym ranking models.

Hypernym Prior: Intuitively, different words have different probabilities as hypernyms of some other words. Some are more probable as hypernyms, such as *animal*, *plant* and *fruit*. Some other words such as *sun*, *nature* and *alias*, are not usually used as hypernyms. Thus we use a prior probability to express this phenomenon. The assumption is that if the more frequent that a noun appears as category tags, the more likely it is a hypernym. We extract category tags from 2.4 million pages in Baidubaike, and compute the prior probabilities $prior(w)$ for a word w being a potential hypernym using Equation (6). $count_{CT}(w)$ denotes the times a word appeared as a category tag in the encyclopedia pages.

$$prior(w) = \frac{count_{CT}(w)}{\sum_{w'} count_{CT}(w')} \tag{6}$$

In Titles: When we enter a query into a search engine, the engine returns a search result list, which contains document titles and their snippet text. The distributions of hypernyms and non-hypernyms in titles are compared with that in snippets respectively in our training data. We discover that the average frequency of occurrence of hypernyms in titles is 15.60 while this number of non-hypernyms is only 5.18, while the difference in snippets is very small (Table III). Thus the frequency of candidates in titles can be used as features. In this work the frequency is divided into three cases: greater than 15.60, less than 5.18, and between 5.18 and 15.60. Three binary features are used to represent these cases.

TABLE III
DISTRIBUTIONS OF CANDIDATE HYPERNYMS IN TITLES AND SNIPPETS

	Avg. Frequency in	
	titles	snippets
Hypernym	15.60	33.69
Non-Hypernym	5.18	30.61

Synonyms: If there exist synonyms of a candidate hypernym in the candidate list, the candidate is probably correct answer. For example, when " (medicine)" and " (medicine)" both appear in the candidate list of an entity, the entity is probably a kind of medicine. We get synonyms of a candidate from a Chinese semantic thesaurus–Tongyi Cilin (Extended) (CilinE for short)[9] and compute the score as a feature using Equation (7).

$$ratio_{syn}(h, l_e) = \frac{count_{syn}(h, l_e)}{len(l_e)} \tag{7}$$

Given a hypernym candidate h of an entity e and the list of all candidates l_e, we compute the ratio of the synonyms of h in l_e. $count_{syn}(h, l_e)$ denotes the count of the synonyms of h in l_e. $len(l_e)$ is the total count of candidates.

Radicals: Chinese characters are a form of ideogram. By far, the bulk of Chinese characters were created by linking together a character with a related meaning and another character to indicate its pronunciation. The character with a related meaning is called radical. Sometimes, it is a important clue to indicate the semantic class of the whole character. For example, the radical " means insects, so it hints "" (dragonfly)" is a kind of insects. Similarly " hints " (lymphoma)" is a kind of diseases. Thus we use radicals as a feature the value of which is computed by using Equation (8).

$$radical(e, h) = \frac{count_{RM}(e, h)}{len(h)} \tag{8}$$

Here $radical(e, h)$ denotes the ratio of characters radical-matched with the last character of the entity e in the hypernym h. $count_{RM}(e, h)$ denotes the count of the radical-matched characters in h. $len(h)$ denotes the total count of the characters in h.

[8]www.baike.com

[9]CilinE contains synonym and hypernym relations among 77 thousand words, which is manually organized as a hierarchy of five levels.

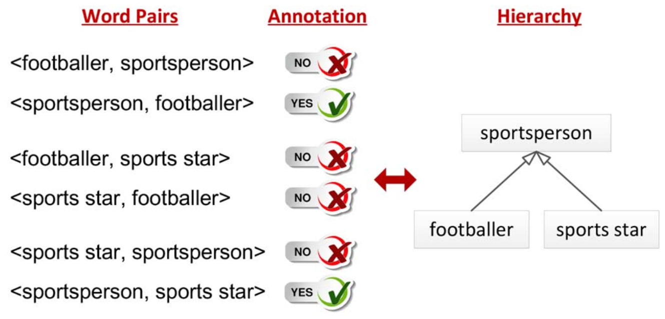

Word Pairs
Annotation
Hierarchy
<footballer, sportsperson> NO
<sportsperson, footballer> YES
<footballer, sports star> NO
<sports star, footballer> NO
<sports star, sportsperson> NO
<sportsperson, sports star> YES
sportsperson
footballer
sports star

Fig. 7. An example for annotation of evaluation data.

Hierarchy Annotation: We then ask two annotators to manually label the semantic hierarchies of the correct hypernyms. For example, the extracted correct hypernyms of "Lionel Messi" include "footballer", "sportsperson" and "sports star". We annotated all of the six directed pairs as Fig. 7 shows.

The final data set contains 655 unique hypernyms and 1,391 hypernym–hyponym relations among them. We randomly split the labeled data into 1/5 for development and 4/5 for testing (Table IV). The hierarchies are represented as relations of pairwise words. We measure the inter-annotator agreement using the kappa coefficient [21]. The kappa value is 0.96, which indicates a good strength of agreement.

C. Evaluation Metrics

We use the widely adopted precision (P), recall (R), and F-score (F) as our metrics to evaluate the performances of the methods:

$$P = \frac{\#\text{ of correct hypernym} - \text{hyponym relations identified}}{\#\text{ of hypernym} - \text{hyponym relations identified}}$$

$$R = \frac{\#\text{ of correct hypernym} - \text{hyponym relations identified}}{\#\text{ of gold hypernym} - \text{hyponym relations}}$$

$$F = \frac{2 \times P \times R}{P + R}$$

Since hypernym–hyponym relations and its reverse (hyponym–hypernym) have one-to-one correspondence, their performances are equal. For simplicity, we only report the performance of the former in the experiments.

V. Results and Analysis

A. Varying the Amount of Clusters

We first evaluate the effect of different number of clusters based on the development data. We vary the numbers of the clusters both for the *direct* and *indirect* training word pairs.

As shown in Fig. 8, the performance of clustering is better than non-clustering (when the cluster number is 1), thus providing evidences that learning piecewise projections based on clustering is reasonable. We finally set the numbers of the clusters of *direct* and *indirect* to 20 and 5, respectively, where the best performances are achieved on the development data.

B. Comparison with Previous Work

In this section, we compare the proposed method with previous methods, including manually-built hierarchy extension, pairwise relation extraction based on patterns, word distributions, and web mining (Section II). Results are shown in Table V.

TABLE IV
The Evaluation Data. * Since Hypernym–Hyponym Relations and Hyponym–Hypernym Relations have One-to-One Correspondence, Their Numbers are the Same

Relation	# of word pairs	
	Dev.	Test
hypernym–hyponym	312	1,079
hyponym–hypernym*	312	1,079
unrelated	1,044	3,250
Total	1,668	5,408

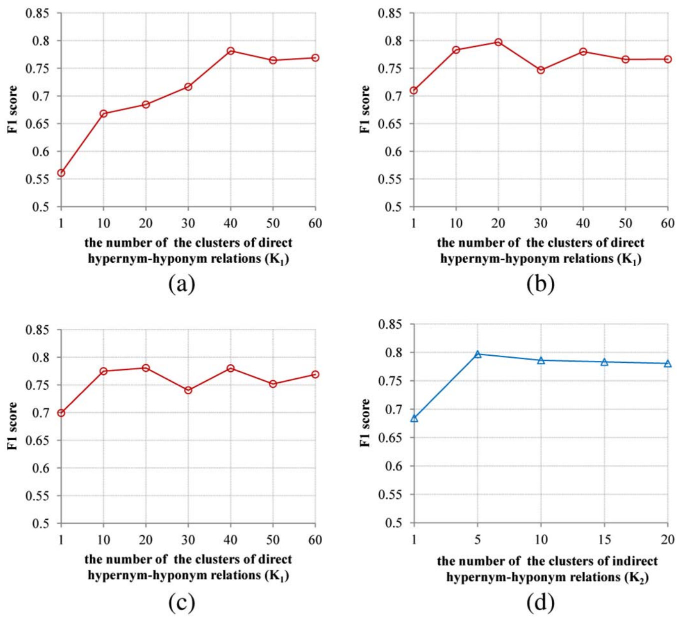


Fig. 8. Performance on development data w.r.t. cluster size. (a) The number of the clusters of indirect hypernym–hyponym relations $K_2 = 1$ (b) The number of the clusters of indirect hypernym–hyponym relations $K_2 = 5$ (c) The number of the clusters of indirect hypernym–hyponym relations $K2 = 20$ (d) The number of the clusters of indirect hypernym–hyponym relations $K1 = 20$.

TABLE V
Comparison of the Proposed Method with Existing Methods in the Test Set

	P(%)	R(%)	F(%)
$M_{Wiki+CilinE}$	92.41	60.61	73.20
$M_{Pattern}$	97.47	21.41	35.11
M_{Snow}	60.88	25.67	36.11
$M_{balApinc}$	54.96	53.38	54.16
M_{invCL}	49.63	62.84	55.46
M_{Fu}	87.40	48.19	62.13
M_{Emb}	80.54	67.99	**73.74**
$M_{Emb+CilinE}$	80.59	72.42	76.29
$M_{Emb+Wiki+CilinE}$	79.78	80.81	**80.29**

Overall Comparison: $M_{Wiki+CilinE}$ refers to the manually-built hierarchy extension method of [2]. In our experiment, we use the category taxonomy of Chinese Wikipedia[10] to extend

[10]dumps.wikimedia.org/zhwiki/20131205/

TABLE VI
CHINESE HEARST-STYLE LEXICAL PATTERNS. THE CONTENTS IN SQUARE BRACKETS ARE OMISSIBLE

Pattern	Translation
w 是[一个|一种] h	w is a [a kind of] h
w [、] 等 h	w[,] and other h
h [，] 叫[做] w	h[,] called w
h [，] [像]如 w	h[,] such as w
h [，] 特别是 w	h[,] especially w

CilinE. Table V shows that this method achieves a high precision but also a low recall, mainly because of the limited scope of Wikipedia.

$M_{Pattern}$ refers to the pattern-based method of [4]. We extract hypernym–hyponym relations in the Baidubaike corpus, which is also used to train word embeddings (Section IV-A). We use the Chinese Hearst-style patterns (Table VI) proposed by [3], in which w represents a word, and h represents one of its hypernyms. The result shows that only a small part of the hypernyms can be extracted based on these patterns because only a few hypernym relations are expressed in these fixed patterns, and many are expressed in highly flexible manners.

In the same corpus, we apply the method M_{Snow} originally proposed by [5]. The same training data for projections learning from CilinE (Section III-C3) is used as seed hypernym–hyponym pairs. Lexico-syntactic patterns are extracted from the Baidubaike corpus by using the seeds. We then develop a logistic regression classifier based on the patterns to recognize hypernym–hyponym relations. This method relies on an accurate syntactic parser, and the quality of the automatically extracted patterns is difficult to guarantee.

We re-implement two previous distributional methods $M_{balApinc}$[6] and M_{invCL}[7] in the Baidubaike corpus. Each word is represented as a feature vector in which each dimension is the PMI value of the word and its context words. We compute a score for each word pair and apply a threshold to identify whether it is a hypernym–hyponym relation.

M_{Fu} refers to our previous web mining method [3]. This method mines hypernyms of a given word w from multiple sources and returns a ranked list of the hypernyms. We select the hypernyms with scores over a threshold of each word in the test set for evaluation. This method assumes that frequent co-occurrence of a noun or noun phrase n in multiple sources with w indicate possibility of n being a hypernym of w. The results presented in [3] show that the method works well when w is an entity, but not when w is a word with a common semantic concept. The main reason may be that there are relatively more introductory pages about entities than those about common words in the Web.

M_{Emb} is the proposed method based on word embeddings. Table III shows that the proposed method achieves a better recall and F-score than all of the previous methods do. It can significantly ($p < 0.01$)[11] improve the F-score over the state-of-the-art method $M_{Wiki+CilinE}$.

M_{Emb} and CilinE can also be combined. The combination strategy is to simply merge all positive results from the two methods together, and then to infer new relations based on the transitivity of hypernym–hyponym relations. The F-score is further improved from 73.74% to 76.29%. Note that, the combined method achieves a 4.43% recall improvement over M_{Emb}, but the precision is almost unchanged. The reason is that the inference based on the relations identified automatically may lead to error propagation. For example, the relation $x \overset{H}{\to} y$ is incorrectly identified by M_{Emb}. When the relation $y \overset{H}{\to} z$ from CilinE is added, it will cause a new incorrect relation $x \overset{H}{\to} z$.

[11]We use the method proposed in [22] to test significance.

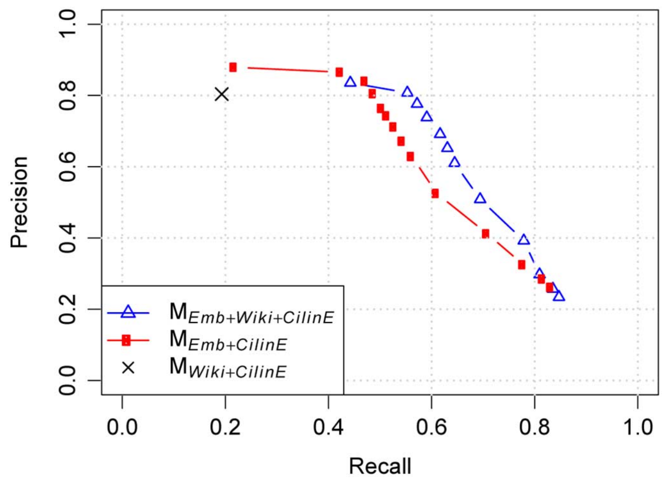


Fig. 9. Precision-Recall curves on the out-of-CilinE data in the test set.

Combining M_{Emb} with $M_{Wiki+CilinE}$ achieves a 7% F-score improvement over the best baseline $M_{Wiki+CilinE}$. Therefore, the proposed method is complementary to the manually-built hierarchy extension method [2].

Comparison on the Out-of-CilinE Data: We are greatly interested in the practical performance of the proposed method on the hypernym–hyponym relations outside of CilinE. We say a word pair is outside of CilinE, as long as there is one word in the pair not existing in CilinE. In our test data, about 62% word pairs are outside of CilinE. Table VII shows the performances of the best baseline method and our method on the out-of-CilinE data. The method exploiting the taxonomy in Wikipedia, $M_{Wiki+CilinE}$, achieves the highest precision but has a low recall. By contrast, our method can discover more hypernym–hyponym relations with some loss of precision, thereby achieving a more than 29% F-score improvement. The combination of these two methods achieves a further 4.5% F-score improvement over $M_{Emb+CilinE}$. Generally speaking, the proposed method greatly improves the recall but damages the precision.

Actually, we can get different precisions and recalls by adjusting the threshold δ (Equation (5)). Fig. 9 shows that $M_{Emb+CilinE}$ achieves a higher precision than $M_{Wiki+CilinE}$ when their recalls are the same. When they achieve the same precision, the recall of $M_{Emb+CilinE}$ is higher.

C. Effect of the Word Embedding Dimensionality

To analyze the effect of the word embedding dimensionality on the final performance, we train word embeddings with dimensions ranging from 100 to 500 and learn projections respec-

TABLE VII
PERFORMANCE ON THE OUT-OF-CILINE DATA IN THE TEST SET

	P(%)	**R(%)**	**F(%)**
$\mathrm{M}_{Wiki+CilinE}$	80.39	19.29	31.12
$\mathrm{M}_{Emb+CilinE}$	71.16	52.80	60.62
$\mathrm{M}_{Emb+Wiki+CilinE}$	69.13	61.65	**65.17**

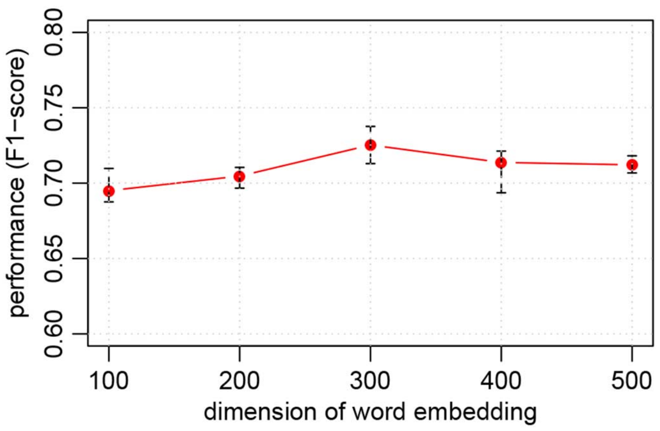


Fig. 10. The performance w.r.t. different dimensions of word embeddings on the test data. We show the *mean*, *max*, *min* values for each dimension under different clustering settings, i.e., the number of clusters for the *direct* and *indirect* training word pairs.

tively based on different clustering settings. The performances are shown in Fig. 10.

Higher dimensions of word embeddings are expected to preserve stronger expression ability. However, we should note that the number of parameters in our projection learning model is n^2 for each cluster where n is the dimension of word embedding. This can be a very high-dimensional model as n increases and thus difficult to learn due to the curse of dimensionality. Hence, as we observed in Fig. 10, when we increase the word embedding dimension from 100 to 300, the performance can be improved. But when the dimension goes beyond 300, the performance decreases.

D. Error Analysis and Discussion

We analyze error cases after experiments. Some cases are shown in Fig. 11. We can see that there is only one general relation “ (`plant`)” $\xrightarrow{H}$ “ (`organism`)” existing in CilinE. Some fine-grained relations exist in Wikipedia, but the coverage is limited. Our method based on word embeddings can discover more hypernym–hyponym relations than the previous methods can. When we combine the methods together, we get the correct hierarchy.

Fig. 11 shows that our method loses the relation “ (`Aconitum`)” $\xrightarrow{H}$ “ (`Ranunculaceae`).” It is because they are very semantically similar (their cosine similarity is 0.9038). Their representations are so close to each other in the embedding space that we have not find projections suitable for these pairs. The error statistics show that when the cosine similarities of word pairs are greater than 0.8, the recall is only 9.5%. This kind of error accounted for about 10.9% among all the errors in our test set. But the amount of this kind of data (the similarity is greater than 0.8) is only 82 (0.5%) in the training data. The corresponding projections therefore are not learned fully. One possible solution may be adding more data of this kind to the training set.

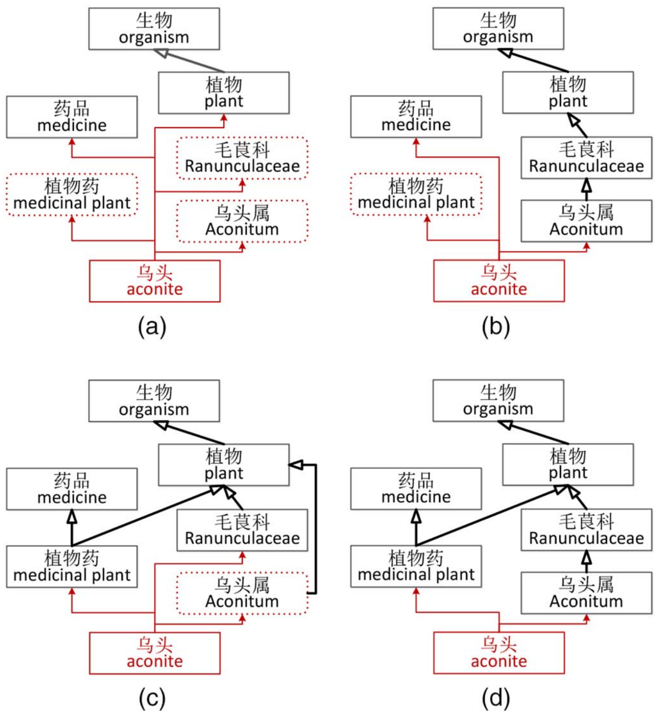


Fig. 11. An example for error analysis. The red paths refer to the relations between the named entity and its hypernyms extracted using the web mining method [3]. The black paths with hollow arrows denote the relations identified by the different methods. The boxes with dotted borders refer to the concepts which are not linked to correct positions. (a) CilinE (b) Wikipedia+CilinE (c) Embedding (d) Embedding+Wikipedia+CilinE.

VI. RELATED WORK

In addition to the works mentioned in Section II, we introduce another set of related studies in this section.

[23], [24], and [25] consider web data as a large corpus and use search engines to identify hypernyms based on the lexical patterns of [4]. However, the low quality of the sentences in the search results negatively influence the precision of hypernym extraction.

Following the method for discovering patterns automatically [5], [26] apply the same method to extract hypernyms of entities in order to improve the performance of a question answering system. [27] propose a method based on patterns to find hypernyms on arbitrary noun phrases. They use a support vector machine classifier to identify the correct hypernyms from the candidates that match the patterns. As our experiments show, pattern-based methods suffer from low recall because of the low coverage of patterns.

Besides [6] and [7], other researchers also propose directional distributional similarity methods [28]–[32]. However, their basic assumption that a hyponym can only be used in contexts where its hypernyms can be used and that a hypernym might be used in all of the contexts where its hyponyms are used may not always rational.

[33] provides a global optimization scheme for extending WordNet, which is different from the above-mentioned pairwise relationships identification methods.

Various deep learning methods have been applied to fields such as computer vision [34], [35], automatic speech recognition [36], [37], and natural language processing where they have been shown to produce state-of-the-art results on various tasks. In the field of natural language processing, the most widely researched tasks based on deep learning are word embeddings. Word embeddings have been successfully applied in many applications, such as in language modeling [38]–[40], sentiment analysis [41], paraphrase detection [42], chunking, and named entity recognition [43], [44]. These applications mainly utilize the representing power of word embeddings to alleviate the problem of data sparsity. [12] and [8] further observe that the semantic relationship of words can be induced by performing simple algebraic operations with word vectors. Their work indicates that word embeddings preserve some interesting linguistic regularities, which might provide support for many applications. In this paper, we improve on their work by learning multiple linear projections in the embedding space, to model hypernym–hyponym relationships within different clusters.

VII. Conclusion and Future Work

This paper proposes a novel method for semantic hierarchy construction based on word embeddings, which are trained using a large-scale corpus. Using the word embeddings, we learn the hypernym–hyponym relationship by estimating projection matrices which map words to their hypernyms. Further improvements are made using a cluster-based approach in order to model the more fine-grained relations. Then we propose a few simple criteria to identity whether a new word pair is a hypernym–hyponym relation. Based on the pairwise hypernym–hyponym relations, we build semantic hierarchies automatically.

In our experiments, the proposed method significantly outperforms state-of-the-art methods and achieves the best F1-score of 73.74% on a manually labeled test dataset. Further experiments show that our method is complementary to the previous manually-built hierarchy extension methods.

For future work, we aim to improve word embedding learning under the guidance of hypernym–hyponym relations. By including the hypernym–hyponym relation constraints while training word embeddings, we expect to improve the embeddings such that they become more suitable for this task.

Acknowledgment

Special thanks to Shiqi Zhao, Zhenghua Li, Wei Song and the anonymous reviewers for insightful comments and suggestions. We also thank Xinwei Geng and Hongbo Cai for their help in the experiments.

References

[1] G. A. Miller, “WordNet: A lexical database for English,” *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[2] F. M. Suchanek, G. Kasneci, and G. Weikum, “Yago: A large ontology from Wikipedia and WordNet,” *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 6, no. 3, pp. 203–217, 2008.

[3] R. Fu, B. Qin, and T. Liu, “Exploiting multiple sources for open-domain hypernym discovery,” in *Proc. EMNLP*, 2013, pp. 1224–1234.

[4] M. A. Hearst, “Automatic acquisition of hyponyms from large text corpora,” in *Proc. 14th Conf. Comput. Linguist.-Vol. 2*, 1992, pp. 539–545, Assoc. for Comput. Linguist..

[5] R. Snow, D. Jurafsky, and A. Y. Ng, “Learning syntactic patterns for automatic hypernym discovery,” in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. Cambridge, MA: MIT Press, 2005, pp. 1297–1304.

[6] L. Kotlerman, I. Dagan, I. Szpektor, and M. Zhitomirsky-Geffet, “Directional distributional similarity for lexical inference,” *Nat. Lang. Eng.*, vol. 16, no. 4, pp. 359–389, 2010.

[7] A. Lenci and G. Benotto, “Identifying hypernyms in distributional semantic spaces,” in *Proc. 6th Int. Workshop Semantic Eval.*, 2012, pp. 75–79, Assoc. for Comput. Linguist..

[8] T. Mikolov, W.-t. Yih, and G. Zweig, “Linguistic regularities in continuous space word representations,” in *Proc. NAACL-HLT*, 2013, pp. 746–751.

[9] R. Fu, J. Guo, B. Qin, W. Che, H. Wang, and T. Liu, “Learning semantic hierarchies via word embeddings,” in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguist. (Vol. 1: Long Papers)*, Baltimore, MD, USA, Jun. 2014, pp. 1199–1209 [Online]. Available: http://www.aclweb.org/anthology/P14-1113, Assoc. for Comput. Linguist.

[10] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, “A neural probabilistic language model,” *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, 2003.

[11] A. Mnih and G. E. Hinton, “A scalable hierarchical distributed language model,” *Adv. Neural Inf. Process. Syst.*, pp. 1081–1088, 2008.

[12] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.

[13] P. Dhillon, D. P. Foster, and L. H. Ungar, “Multi-view learning of word embeddings via cca,” *Adv. Neural Inf. Process. Syst.*, pp. 199–207, 2011.

[14] P. D. Turney, “Similarity of semantic relations,” *Comput. Linguist.*, vol. 32, no. 3, pp. 379–416, 2006.

[15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Adv. Neural Inf. Process. Syst. 26*, 2013, pp. 3111–3119 [Online]. Available: http://media.nips.cc/nipsbooks/nipspapers/paper_files/nips26/1421.pdf

[16] O. Levy and Y. Goldberg, “Linguistic regularities in sparse and explicit word representations,” in *Proc. 18th Conf. Comput. Nat. Lang. Learn.*, Baltimore, MD, USA, Jun. 2014, Assoc. Comput. Linguist..

[17] Y. Goldberg and O. Levy, “word2vec explained: Deriving mikolov et al. ’s negative-sampling word-embedding method,” in *Proc. CoRR*, 2014, vol. abs/1402.3722.

[18] B. Recht, C. Re, S. Wright, and F. Niu, “Hogwild: A lock-free approach to parallelizing stochastic gradient descent,” *Adv. Neural Inf. Process. Syst.*, pp. 693–701, 2011.

[19] H. Ritzema, *Drainage principles and applications*. Highlands Ranch, CO, USA: Water Resources, 1994.

[20] W. Che, Z. Li, and T. Liu, “Ltp: A chinese language technology platform,” in *Proc. Coling'10: Demonstrations*, Beijing, China, Aug. 2010, pp. 13–16 [Online]. Available: http://www.aclweb.org/anthology/C10-3004

[21] S. Siegel and N. J. Castellan Jr, *Nonparametric statistics for the behavioral sciences*. New York, NY, USA: McGraw-Hill, 1988.

[22] Y. Zhang, S. Vogel, and A. Waibel, “Interpreting bleu/nist scores: How much improvement do we need to have a better system?,” in *Proc. 4rth Int. Conf. Lang. Resources Eval.*, 2004, pp. 2051–2054.

[23] R. Evans, “A framework for named entity recognition in the open domain,” *Recent Adv. Nat. Lang. Process. III: Sel. Papers from RANLP*, vol. 260, pp. 267–274, 2004.

[24] R. M. Ortega-Mendoza, L. Villaseñor-Pineda, and M. Montes-y Gómez, “Using lexical patterns for extracting hyponyms from the web,” in *MICAI 2007: Advances in Artificial Intelligence*. New York, NY, USA: Springer, 2007, pp. 904–911.

[25] E. T. K. Sang, “Extracting hypernym pairs from the web,” in *Proc. 45th Annu. Meeting ACL Interact. Poster Demonstrat. Sess.*, 2007, pp. 165–168, Assoc. Comput. Linguist..

[26] P. McNamee, R. Snow, P. Schone, and J. Mayfield, “Learning named entity hyponyms for question answering,” in *Proc. 3ird Int. Joint Conf. Nat. Lang. Process.*, 2008, pp. 799–804.

[27] A. Ritter, S. Soderland, and O. Etzioni, “What is this, anyway: Automatic hypernym discovery,” in *Proc. 2009AAAI Spring Symp. Learn. Reading Learn. Read*, 2009, pp. 88–93.

[28] J. Weeds, D. Weir, and D. McCarthy, “Characterising measures of lexical distributional similarity,” in *Proc. 20th Int. Conf. Comput. Linguist.*, 2004, p. 1015, Assoc. Comput. Linguist..

[29] M. Geffet and I. Dagan, “The distributional inclusion hypotheses and lexical entailment,” in *Proc. 43rd Annu. Meeting Assoc. Comput. Linguist.*, 2005, pp. 107–114, Assoc. Comput. Linguist..

[30] R. Bhagat, P. Pantel, E. H. Hovy, and M. Rey, “Ledir: An unsupervised algorithm for learning directionality of inference rules,” in *Proc. EMNLP-CoNLL*, 2007, pp. 161–170.

[31] I. Szpektor, E. Shnarch, and I. Dagan, “Instance-based evaluation of entailment rule acquisition,” in *Proc. 45th Annu. Meeting Assoc. Comput. Linguist.*, Prague, Czech Republic, Jun. 2007, pp. 456–463 [Online]. Available: http://www.aclweb.org/anthology/P07-1058, Association for Computational Linguistics

[32] D. Clarke, “Context-theoretic semantics for natural language: An overview,” in *Proc. Workshop Geometrical Models Nat. Lang. Semant.*, 2009, pp. 112–119, Assoc. Comput. Linguist..

[33] R. Snow, D. Jurafsky, and A. Y. Ng, “Semantic taxonomy induction from heterogenous evidence,” in *Proc. 21st Int. Conf. Comput. Linguist. and 44th Annu. Meeting Assoc. Comput. Linguist.*, Sydney, Australia, Jul. 2006, pp. 801–808 [Online]. Available: http://www.aclweb.org/anthology/P06-1101, Assoc. Comput. Linguist.

[34] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations,” in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 609–616, Assoc. Comput. Linguist..

[35] D. Ciresan, U. Meier, and J. Schmidhuber, “Multi-column deep neural networks for image classification,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recogn. (CVPR)*, 2012, pp. 3642–3649.

[36] G. E. Dahl, D. Yu, L. Deng, and A. Acero, “Context-dependent pretrained deep neural networks for large-vocabulary speech recognition,” *IEEE/ACM Trans. Audio, Speech, Lang. Process. (TASLP)*, vol. 20, no. 1, pp. 30–42, Jan. 2012.

[37] Z.-H. Ling, L. Deng, and D. Yu, “Modeling spectral envelopes using restricted boltzmann machines and deep belief networks for statistical parametric speech synthesis,” *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 21, no. 10, pp. 2129–2139, Oct. 2013.

[38] T. Mikolov, “Statistical language models based on neural networks,” Ph.D. dissertation, Brno Univ. of Technol., Brno, Czech Republic, 2012.

[39] E. Arisoy, S. F. Chen, B. Ramabhadran, and A. Sethy, “Converting neural network language models into back-off language models for efficient decoding in automatic speech recognition,” *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 22, no. 1, pp. 184–192, Jan. 2014.

[40] H.-S. Le, I. Oparin, A. Allauzen, J. Gauvain, and F. Yvon, “Structured output layer neural network language models for speech recognition,” *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 21, no. 1, pp. 197–206, Jan. 2013.

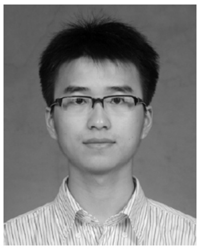
[41] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, “Semi-supervised recursive autoencoders for predicting sentiment distributions,” in *Proc. Conf. Empir. Meth. Nat. Lang. Process.*, 2011, pp. 151–161, Assoc. Comput. Linguist..

[42] R. Socher, E. H. Huang, J. Pennin, C. D. Manning, and A. Ng, “Dynamic pooling and unfolding recursive autoencoders for paraphrase detection,” *Adv. Neural Inf. Process. Syst.*, pp. 801–809, 2011.

[43] J. Turian, L. Ratinov, and Y. Bengio, “Word representations: A simple and general method for semi-supervised learning,” in *Proc. 48th Annu. Meeting Assoc. Comput. Linguist.*, 2010, pp. 384–394, Assoc. Comput. Linguist..

[44] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural language processing (almost) from scratch,” *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, 2011.

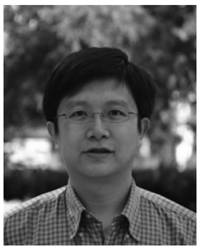
Ruiji Fu received his Ph.D., M.S., and B.S. in computer science from the Harbin Institute of Technology in 2014, 2009, and 2007, respectively. He is a Senior Researcher with iFlytek Co., Ltd now. His current research interests include natural language processing, text mining, and open information extraction.



Jiang Guo is a Ph.D. candidate at the Harbin Institute of Technology (HIT), China. He received his M.S. and B.S. both in computer science from HIT, in 2012 and 2010, respectively. His current research interest is representation learning and its applications on natural language processing.

Bing Qin received her Ph.D. in computer science from the Harbin Institute of Technology (HIT), China, in 2005. She is a Full Professor in the School of Computer Science and Technology, HIT. Her research interests include natural language processing, text mining, and opinion mining.

Wanxiang Che received his Ph.D. in computer science from the Harbin Institute of Technology (HIT), China, in 2008. He is a full time Associate Professor in the School of Computer Science and Technology, HIT. His current research interests include natural language processing and information retrieval.



Haifeng Wang is a Vice President of Baidu, the chair of Department of Language Information Engineering of Peking University, and a Visiting Professor at the Harbin Institute of Technology. In 1999, He received his Ph.D. in computer science from the Harbin Institute of Technology. Soon after, he was an Associate Researcher at Microsoft Research China from 1999 to 2000, a Research Scientist at iSilk.com (Hong Kong) from 2000 to 2002, and the Chief Research Scientist and Deputy Director at Toshiba (China) R&D Center until Jan. 2010. He is also the immediate past president of the Association for Computational Linguistics (ACL). He has served as program chair, workshop chair, tutorial chair, area chair, industry chair, and sponsorship chair for several top conferences including SIGIR, ACL, IJCAI, KDD, COLING, IJCNLP, etc., as well as associate editor, guest editor and reviewers for some academic journals.

Ting Liu received his Ph.D. in computer science from the Harbin Institute of Technology (HIT), China, in 1998. He is a Full Professor in the School of Computer Science and Technology, HIT. His current research interests include natural language processing, information retrieval, and social computing.