Learning Term Embeddings for Hypernymy Identification

Zheng Yu[1], **Haixun Wang**[2], **Xuemin Lin**[1,3], **Min Wang**[2]
[1] East China Normal University, China
zyu.0910@gmail.com
[2] Google Research, USA, {haixun, minwang}@google.com
[3] University of New South Wales, Australia
lxue@cse.unsw.edu.au

Abstract

Hypernymy identification aims at detecting if isA relationship holds between two words or phrases. Most previous methods are based on lexical patterns or the Distributional Inclusion Hypothesis, and the accuracy of such methods is not ideal. In this paper, we propose a simple yet effective supervision framework to identify hypernymy relations using distributed term representations (a.k.a term embeddings). First, we design a distance-margin neural network to learn term embeddings based on some pre-extracted hypernymy data. Then, we apply such embeddings as term features to identify positive hypernymy pairs through a supervision method. Experimental results demonstrate that our approach outperforms other supervised methods on two popular datasets and the learned term embeddings has better quality than existing term distributed representations with respect to hypernymy identification.

1 Introduction

In this paper, we are concerned with the problem of hypernymy identification. Specifically, given a pair of words or phrases (X, Y), our goal is to check if the hypernymy relationship (a.k.a. the isA relationship) holds between X and Y, that is, if X names a broad category that includes Y [Snow *et al.*, 2004].

1.1 The Hypernymy Relationship

Given a hypernymy pair (X, Y), we call X a hypernym of Y and Y a hyponym of X. Examples of hypernymy pairs include (`scientist`, `Einstein`), (`car brand`, `jaguar`), (`mammal`, `jaguar`), etc.

Hypernymy relationship plays a critical role in language understanding because it enables generalization, which lies at the core of human cognition. As a result, the hypernymy relationship is the backbone of almost every taxonomy, ontology, and semantic network. To understand its significance, consider two terms a and b, and $R(a, b)$, which denotes a certain relationship between a and b. It can be shown that the hypernymy relationship may be used to generalize R, or in other words, it may enable us to know more about R. For example, if we know b' is a hypernym or hyponym of b, chances are $R(a, b')$ also holds. More specifically, let R denote the head-modifier relationship, that is, $R(a, b)$ holds if a is more likely to be the head and b the modifier when a and b appear together. For instance, we may have R(`charger`, `iphone`) because when people search for `charger iphone`, their intent is usually to find a `charger` for an `iphone`. Now, given that we know (`smart phone`, `iphone`) is a hypernymy pair, then it is very likely R(`charger`, `smart phone`) also holds. This kind of inferencing can be extended and we may derive R(`case`, `iphone`), R(`charger`, `galaxy s4`), simply because `case` and `charger` are co-hyponyms for the concept of `accessory`, and `iphone` and `galaxy s4` are co-hyponyms for the concept of `smart phone`.

Clearly, the hypernymy relationship is powerful for inferencing, and as a result, hypernymy identification has many applications, including ontology construction [Suchanek *et al.*, 2008], machine reading [Etzioni *et al.*, 2006], question answering [McNamee *et al.*, 2008], etc.

1.2 State-of-the-Art Approaches

Hypernymy identification is a long-standing and challenging research topic. We briefly review two major approaches, namely pattern-based methods and distributional models.

Pattern-based methods. Much work [Miller, 1995; Liu and Singh, 2004; Suchanek *et al.*, 2007; Wu *et al.*, 2012] has focused on creating a large data set of hypernymy pairs. To tell if the hypernymy relationship holds for a pair (X, Y), it simply checks if (X, Y) is in the data set. To create the dataset, it relies on information extraction over large corpra using some syntactic patterns that indicate the hypernymy relationship. A well-known syntactic pattern for this purpose is the Hearst pattern [Hearst, 1992], for instance, the pattern "... NP_0 such as NP_1, NP_2, ..." may indicate NP_0 is a hypernym of NP_1, NP_2, etc.

This method is simple and efficient, but generally it has low precision and low recall because information extraction is error prone and the text corpus is always sparse (people do not express every possible hypernymy relationship in a Hearst pattern and in an information extraction friendly way). Consider two sentences : `...cities in Asian`

countries such as Tokyo... and ...cities in Asian countries such as Japan... Naive extraction may lead to (Asian country, Tokyo) or (city, Japan). The ambiguity of a natural language compounded by data sparsity makes syntactic patterns based methods less robust.

Distributional models. Another approach for hypernymy identification is based on the Distributional Inclusion Hypothesis (abbreviated DIH) [Zhitomirsky-Geffet and Dagan, 2005; 2009], which assumes that hypernyms have broader contexts than hyponyms. Distributional models typically represent a term by its textual contexts in the form of a high-dimensional vector [Turney and Pantel, 2010]. For example, the context features of iphone may include imac, handset, psp, smart phone, etc., and the values for those features are certain statistics (e.g., PPMI) between the term and the feature. The DIH states that if X is a hypernym of Y, then most context features of Y are also features of X. For example, if X is animal and Y is cat, most features of cat are also features of animal, but at least some features of animal do not apply to cat. For instance, the term rights co-occurrs with animal frequently, but not so much for cat.

Several measures [Weeds *et al.*, 2004; Clarke, 2009; Kotlerman *et al.*, 2010; Lenci and Benotto, 2012] have been proposed to identify the hypernymy relationship based on the DIH. For example, Weeds et al [2004] proposed a very simple measure to compute the weighted inclusion of features of Y within the features of X:

$$WeedsPrec(X,Y) = \frac{\sum_{f \in F_X \cap F_Y} w_Y(f)}{\sum_{f \in F_Y} w_Y(f)}$$

Here, F_X is the set of features of X and $w_X(f)$ is the weight of feature f for X.

There are two major issues for this approach. First, the inclusion hypothesis is not always correct, that is, hypernyms do not always have broader context features. For example, American is a hypernym of Obama, but Obama definitely have some features that are not strongly associated with American. Second, the proposed measures in the distributional space are not able to perfectly discriminate the hypernymy relationship from other semantic relationships such as co-hyponym and meronymy.

Recently, Roller et al. [2014] propose a simple supervision distributional model to weight the importance of different context features. The hypernymy identification task benefits greatly from this feature selection approach, and its accuracy is significantly improved. But problems are that the supervised models are still based on the DIH, and the selected features are domain dependent, which means they are heavily related to the training set and are not global indicators of hypernymy.

1.3 Our Approach

In this paper, we propose a new approach for hypernymy identification. We use term embeddings to encode hypernymy properties. Term embeddings typically represent a term by a dense, low-dimensional, real-valued vector. Much previous work [Bengio *et al.*, 2003; Collobert *et al.*, 2011; Mikolov *et al.*, 2013a] has focused on learning such representations from term co-occurrence data so that similar terms have similar embeddings. However, co-occurrence based similarity is not enough for our purpose, for we need to tell if two terms form a specific relationship instead of just frequently co-occur. To go beyond co-occurrence based similarity, we design a neural network model to learn term embeddings that encode hypernymy properties. We use as training data a set of hypernymy pairs extracted from a web corpus [Wu *et al.*, 2012]. Our model is shown to be able to generalize from the training dataset to encode hypernymy properties for unseen pairs. We then use term embeddings as input features to further train a supervised classifier for identifying hypernymy. Compared with the state-of-the-art supervised distributional method [Roller *et al.*, 2014], our approach gets better accuracy, and the learned term embeddings are not domain dependent; moreover, we show that our term embeddings are superior to other traditional embeddings in the task of hypernymy identification since using our term embeddings achieves much higher accuracy than using other embeddings.

1.4 Paper organization

The rest of this paper is organized as follows. In Section 2, we describe hypernymy relationships in data driven knowledge bases and background of term embeddings. In Section 3, we introduce the learning method for term embeddings and the architecture of our neural network model. In Section 4, we present our embedding-based hypernymy classifier. We report experimental results in Section 5, and give conclusion in Section 6.

2 Background

Training Corpus. We use data in Probase [Wu *et al.*, 2012] for training term embeddings. Probase extracts hypernymy data from billions of Web pages using pattern-based method. We denote each hypernymy relation as a triple (v, u, q), where v is the hypernym term, u is the hyponym term, and $q = n(v, u)$ is the number of times u and v occur together in hypernymy patterns. We filter out some rare examples to clean the data. Specially, we filter out i) triplets that have frequency q less than 5, and terms that have frequency less than 10. At last we obtain a training corpus of $5,796,987$ triplets, including about 1 million unique terms, $906,241$ hyponyms, and $221,612$ hypernyms. We use $S_u = \{u_1, u_2, ..., u_m\}$ and $S_v = \{v_1, v_2, ..., v_n\}$ to represent the set of hyponym and hypernym respectively. Note that $S_u \cap S_v \neq \phi$. From another point of view, the training set can be regarded as a matrix, whose rows represent hyponym terms and columns represent hypernym terms, and the cell value is the frequency q.

Term Embeddings. Term embedding is a dense, low-dimensional, and continuous valued representation of word and phrase. Compared with distributional representation of terms, term embeddings are more compact, and they capture useful syntactic and semantic properties of the terms. For example, in the embedding space, the nearest neighbors of cat

include cats, dog, puppy, and so on. Term embeddings are currently widely used in NLP tasks [Glorot *et al.*, 2011; Socher *et al.*, 2011; Turney, 2013], and many training approaches have been proposed [Bengio *et al.*, 2003; Turian *et al.*, 2010; Collobert *et al.*, 2011]. Recently, Mikolov et al. [2013a] introduced a log-linear model, namely the Skip-gram, to efficiently induce term embeddings on a very large-scale corpus. The Skip-gram model adopts log-linear classifiers to predict context terms within a predefined range given the current term embedding as input. The embeddings (denoted as word2vec) are trained to maximize the log-likelihood over the entire corpus through stochastic gradient descent (SGD) method. Many applications have shown the good performance of word2vec [Mikolov *et al.*, 2013a; Kim, 2014; Hill *et al.*, 2014], and in this study, we compare our term embeddings with word2vec embeddings in the task of hypernymy identification.

3 Dynamic Distance-Margin Model

In this section, we introduce a dynamic distance-margin model based on neural networks to encode hypernymy properties into term embeddings.

3.1 Embeddings for the Hypernymy Relationship

We give each term x two embeddings, namely, the hyp**O**nym embedding $O(x)$ and the hyp**E**rnym embedding $E(x)$. The reason we give each term two embeddings is because hypernymy is not a symmetric relation: we need to discriminate the roles of the two terms in a hypernymy relation. We use $O(x)$ to represent x when x functions as a hyponym, as for example, bird in (animal, bird), and we use $E(x)$ to represent x when x functions as a hypernym, as for example, bird in (bird, robin).

We want to learn embeddings that encode the hypernymy relationship. More specifically, we hope that the learned embeddings have the following three properties:

1. hyponym-hypernym similarity: If u is a hyponym of v, then $O(u)$ is similar to $E(v)$, for example, $O(\texttt{dog}) \sim E(\texttt{animal})$;
2. co-hyponym similarity: If u and v are co-hyponyms, i.e. they have some same hypernyms, then $O(u)$ is similar to $O(v)$, for example, $O(\texttt{dog}) \sim O(\texttt{cat})$;
3. co-hypernym similarity: If u and v share a lot of hyponyms, then $E(u)$ and $E(v)$ are similar, for example, $E(\texttt{car}) \sim E(\texttt{auto})$.

3.2 Learning Embeddings

Let S_u be the set of all hyponyms, and S_v be the set of all hypernyms. We aim to train two functions to map terms into a d-dimensional embedding space R^d.

$$O : S_u \to \mathbb{R}^d$$
$$E : S_v \to \mathbb{R}^d$$

For a hypernymy relationship $x = (v, u, q)$, our goal is to ensure $O(u)$ is close to $E(v)$. We choose 1-norm distance as the distance measure since it is simple yet effective to compute the degree of similarity between two embeddings, and it enables fast SGD learning. The distance measure function of x is:

$$f(x) = ||O(u) - E(v)||_1 \quad (1)$$

We adopt a pairwise training strategy. Specifically, let $x = (v, u, q)$ be a positive hypernymy relationship, and let x' be a negative hypernymy relationship, which in the form of $x' = (v', u, 0)$ or $x' = (v, u', 0)$. We want to ensure $f(x)$ is smaller than $f(x')$ by a certain margin. More specifically, our objective function for one pairwise training is:

$$f(x) \leq f(x') - m(x, x') \quad (2)$$

where $m(x, x')$ is the margin.

Not every hypernymy relationship is equal. For a hypernymy example (v, u, q), the frequency q denotes the strength or the popularity of the relationship. We adopt two mechanisms to reflect this in our training.

- For each hypernymy relationship $x = (v, u, q)$ in the training set, we randomly create q corrupted relationships $x'_1, \cdots, x'_q$, and we perform pairwise training for every pair (x, x'_i). This means our model places more weight on popular hypernymy relationships than those rare ones. We create corrupted relationships as follows: we replace u or v with a random $u' \in S_u$ or $v' \in S_v$. The result is a corrupted triple in the form of (v, u', q') or (v', u, q'), where q' is the frequency. In practice, we often have $q' = 0$, because it is unlikely a randomly corrupted pair of terms exist in the training set.
- The margin in Eq 2 is not a constant, but a function of x and its corrupted version x'. We devise the margin function $m(x, x')$ to put more emphasis on popular hypernymy relationships. Specifically, we define the margin as follows:

  $$m(x, x') = log(q + 1) - log(q' + 1) = log\frac{q + 1}{q' + 1}$$

  In other words, we require a popular sample to have a bigger margin than a rare sample to their corresponding negative samples. This is why we call our model a **dynamic distance-margin model**.

We give the overall loss function J across all training data as follows:

$$J = \sum_{x=(u,v,q)} \sum_{j=1}^{q} max(0, f(x) - f(x'_j) + m(x, x'_j)) \quad (3)$$

We argue that, by minimizing the loss function J, the resulting embeddings satisfy the three properties we mentioned above. It is easy to understand how hyponym-hypernym similarity is achieved, because our training objective is to make $O(u)$ and $E(v)$ similar so that the 1-norm distance between them will be small enough to minimize J. For example, as shown in Table 1, for hyponym-hypernym similarity, the 1-norm distance between $O(\texttt{iphone})$ and $E(\texttt{device})$ is 15.4, while that between $O(\texttt{iphone})$ and $E(\texttt{animal})$ is 30.6. To see how co-hyponym similarity is achieved, consider term $u = \texttt{dog}$ and term $v = \texttt{cat}$. Because they share a hypernym $w = \texttt{animal}$, both $O(u)$ and $O(v)$ are optimized

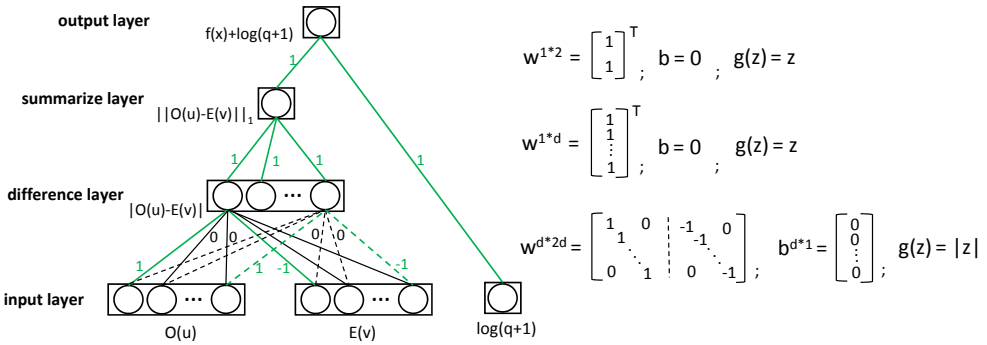

Figure 1: The Architecture of our dynamic margin model. The input is a concatenation vector of embeddings $O(u)$ and $E(v)$, and the output is $f(x) + log(q + 1)$. For the corresponding corrupted x', we can get $f(x') + log(q' + 1)$ as the output. Then we compute the loss $J = f(x) - f(x') + log(q + 1) - log(q' + 1)$, and optimize the embeddings through SGD.

to have small distances to $E(w)$, that is, $O(u) \sim E(w)$ and $O(v) \sim E(w)$. In other words, w draws $O(u)$ and $O(v)$ closer to each other. Furthermore, since w can take other values such as `mammal`, `pet`, and so on, each of which makes $O(u)$ and $O(v)$ closer, we will finally get $O(u) \sim O(v)$. As shown in Table 1, `google` is much closer to `yahoo` than to `banana` in the embedding space. For the same reason, for two terms, such as `company` and `firm`, that share a number of hyponyms, their embeddings are likely to be close in the embedding space.

property type	example		distance
hyponym-hypernym similarity: $O(u) \sim E(v)$	positive	iphone-device	15.4
		iphone-phone	16.0
	negative	iphone-animal	30.6
		iphone-fruit	37.9
co-hyponym similarity: $O(u) \sim O(v)$	positive	google-yahoo	6.1
		google-amazon	10.3
	negative	google-banana	27.1
		google-cat	27.5
co-hypernym similarity: $E(u) \sim E(v)$	positive	company-firm	12.1
		company-brand	14.5
	negative	company-fruit	24.9
		company-animal	30.5

Table 1: Some examples of the three similarity properties

3.3 Neural Network Architecture

We minimize the loss function J using the stochastic gradient descent (SGD) method [Robbins and Monro, 1951] over the neural network architecture shown in Figure 1. The dimensionality of the *input layer* is $2d$, as we concatenate of the hyponym embedding $O(u)$ and the hypernym embedding $E(v)$, each of which is d-dimensional. At the beginning of the training, we initialize all embeddings to be uniformly random in $[-0.1, +0.1]^d$. The *difference layer* captures the error between $O(u)$ and $E(v)$ on each dimension. The activation function of this layer is $g(z) = |z|$, where z is a d-dimensional vector and $z^i = O(u)^i - E(v)^i$. That is, the i_{th} neuron only computes the error between $O(u)^i$ and $E(v)^i$. Therefore, the weight of edge $z^i \leftrightarrow O(u)^i$ is 1 and the weight of edge $z^i \leftrightarrow E(v)^i$ is -1, and all other edges' weights are 0. As shown in Figure 1, the activation function on each layer is linear and simple, and all parameters (w, b) in the neural network are constant numbers. Thus, in the training process, we only need to update the term embeddings without optimizing parameters, which markedly simplify the training process.

4 Supervised Hypernymy Identification

Our learned term embeddings have a property that for a pair of positive and negative hypernymy relationships x and x', we have $f(x) < f(x')$. The question is, can we directly use this property for hypernymy identification? That is, does there exist a threshold δ such that for any x that satisfies $f(x) < \delta$ then x is a positive hypernymy relationship, and if not then a negative one?

Unfortunately, it is hard to find a uniform threshold to ideally separate all positive and negative hypernymy pairs, because the distance-margin is only able to determine the margin between $f(x)$ and $f(x')$. It does not provide an absolute threshold for hypernymy identification for any x. In fact, certain positive and negative hypernymy pairs have similar distances. For example, the 1-norm distance between $O($`goldfish`$)$ and $E($`pond fish`$)$ is 21.4, and `pond fish` is a hypernym of `goldfish`. However, the distance between $O($`salmon`$)$ and $E($`crop`$)$ is also 21.4, but `crop` is not a hypernym of `salmon` (see Table 2 for more examples).

hyponym	hypernym	distance	isHypernymy
goldfish	pond fish	21.4	true
salmon	crop	21.4	false
tokyo	large urban area	23.9	true
robin	snake	23.9	false

Table 2: The 1-norm distances between some positive and negative hypernymy examples. The distance alone cannot distinguish positive hypernymy relations from negative ones.

Instead of using the distance between $O(u)$ and $E(v)$ to decide whether (v, u) is a positive hypernymy pair, we build a classifier that uses $O(u)$, $E(v)$, and other signals as features

for hypernymy identification. Specifically, we use a Support Vector Machine (SVM) for this purpose. Given a pair of terms (v, u), the input feature is the concatenation of u's embedding and v's embedding as well as the 1-norm distance between them, or simply, $O(u) + E(v) + ||O(u) - E(v)||_1$, where "+" denotes concatenation. The 1-norm distance feature is further normalized using $d' = \frac{d-d_{min}}{d_{max}-d_{min}}$. Thus, the dimension of the feature vector is $2d + 1$. As we will show in the experiment result, the 1-norm distance feature plays an important role in the task of hypernymy identification.

5 Experiments

We compare our approach with state-of-the-art supervised distributional methods [Roller *et al.*, 2014]. To evaluate the quality of our term embeddings, we compare with word2vec [Mikolov *et al.*, 2013b] as input to SVM for hypernymy identification.

5.1 Evaluation Data Sets

We use two datasets, BLESS [Baroni and Lenci, 2011] and ENTAILMENT [Baroni *et al.*, 2012] for evaluation. BLESS contains 200 distinct, unambiguous concepts, each of which is involved with other words in some relations. We extract from BLESS 14,547 tuples to form 3 datasets: hypernymy-meronymy, hypernymy-coordinate, and hypernymy-random. Specifically, i) hypernymy-meronymy consists of tuples of either hypernymy relation or meronymy relation (a.k.a. part-of relation, e.g. $\langle dog, paw \rangle$); ii) hypernymy-coordinate consists tuples of either hypernymy relation or coordinate relation (two terms that have the same concepts, e.g. $\langle dog, cat \rangle$); and iii) hypernymy-random consists of tuples of hypernymy relation or random relation (e.g. $\langle dog, laptop \rangle$). The ENTAILMENT dataset consists of 2,770 pairs, with equal number of positive and negative examples of hypernymy relations. Altogether, there are 1,376 unique hyponyms and 1,016 unique hypernyms. The negative examples are generated by randomly permuting the hypernyms of the positive examples.

5.2 Supervised Models

We evaluate the following three supervised models.

(1) SVM+Emb This is our method that uses SVM and hypernymy embeddings. The input is a 2d+1-dimensional vector: $\langle O(u); E(v); \|O(u) - E(v)\|_1 \rangle$, where d is the dimension of our hypernymy embeddings. SVM is trained using a RBF kernel with $\gamma = 0.03125$ and penalty term $C = 8.0$.

(2) SVM+Word2Vec The model is configured the same way as SVM+Emb, but the input is word2vec embeddings, that is $\langle w2v(u); w2v(v); \|w2v(u) - w2v(v)\|_1 \rangle$. The word2vec embeddings are trained using the Skip-gram Model [Mikolov *et al.*, 2013b] on One Billion Word Language Modeling Benchmark [Chelba *et al.*, 2013].

(3) Diff+TypeDM$_{300}$ This is a supervised logistic regression model proposed by Roller et al. [2014]. We select this model for its good performance. We first represent terms in a distributional vector space that is extracted from TypeDM tensors [Baroni and Lenci, 2011]. The TypeDM contains a large number of weighted tuples in the form of $\langle \langle w_1, l, w_2 \rangle, \sigma \rangle$, where w_1 and w_2 are content terms, l is a syntagmatic relationship, and σ is a weight estimating the saliency of the relationship. For example, $\langle \langle marine, use, bomb \rangle, 82.1 \rangle$ is a TypeDM tuple.

We construct vectors for every unique noun w_1 using the set of $\langle l, w_2 \rangle$ pairs as dimensions and the corresponding σ as dimension values. We further reduce this distributional space to 300 dimensions using Singular Value Decomposition, so every term is represented as a 300-dimensional vector. Given a pair of terms (v, u) (here we also use v, u to denote their reduced distributional vectors), the input feature vector includes two parts $\langle f; g \rangle$, where $f_i = \frac{u_i}{\|u\|} - \frac{v_i}{\|v\|}, g_i = f_i^2$.

5.3 Experiment 1

For each BLESS dataset (hypernymy-coordinate, hypernymy-meronymy, hypernymy-random), we hold out one target concept and train on the remaining 199 ones. The hold-out concept and its relatum constitute the test data, and we exclude from the training set any pair containing a relatum that appears in the test set. We report the average accuracy across all concepts. For ENTAILMENT data, we do the evaluation using the same method – we hold out one hyponym for test and train on all remaining hyponyms, and we also compute the average accuracy across all hyponyms. In this experiment, both our term embeddings and word2vec embeddings are 100 dimensional (denoted as the subscript $_{100}$). Furthermore, to see the effect of 1-norm distance, we remove the distance value from the input features. That is, for SVM+Emb model, the input feature vector is changed to $\langle O(u); E(v) \rangle$, and for SVM+Word2Vec model, the input feature vector is changed to $\langle w2v(u); w2v(v) \rangle$. We use the superscript ^{2} to denote this setting.

Results Table 3 shows the performance of the three supervised methods on both BLESS and ENTAILMENT data sets. First, we compare the results of SVM+Emb$_{100}$, SVM+Word2Vec$_{100}$ and Diff+TypeDM$_{300}$. It is obvious that our approach, which uses $\langle O(u); E(v); \|O(u) - E(v)\|_1 \rangle$ as input features, has the highest average accuracy for all data sets. The Diff+TypeDM$_{300}$ model performs slightly worse, while the SVM+Word2Vec$_{100}$ model gets the worst result, especially for the hypernymy-coordinate data set and hypernymy-meronymy data set. This is because word2vec embeddings capture only co-occurrence based similarity, which is not enough for the classifier to discriminate hypernymy from other semantic relationships, especially the coordinate and meronymy. For example, as far as word2vec is concerned, `cat` is highly similar to `dog`, `paws`, and `animal`. But in fact, `(cat, dog)` is of the coordinate relationship, `(cat, animal)` the hypernymy relationship, and `(cat, paws)` the meronymy relationship. Their word2vec input features are too similar for the classifier to make correct predication.

Our term embeddings, however, enable the classifier to perform hypernymy identification. The reason is that, in our in-

Data set	BLESS			ENTAILMENT
	hyper-coord	hyper-mero	hyper-random	
SVM+Emb$_{100}$	92.3	88.7	91.8	87.5
SVM+Emb$^2_{100}$	88.2	86.9	90.1	85.9
SVM+Word2Vec$_{100}$	75.8	72.9	84.3	81.6
SVM+Word2Vec$^2_{100}$	75.7	72.6	84.4	81.2
Diff+TypeDM$_{300}$	86.5	85.7	88.2	84.6

Table 3: Average accuracy (%) of the supervised models obtained on BLESS and ENTAILMENT.

Dimension	Training Time (s/epoch)	BLESS			ENTAILMENT
		hyper-coord	hyper-mero	hyper-random	
50	52	89.7	86.2	89.9	86.1
100	89	92.3	88.7	91.8	87.5
200	156	92.7	89.0	92.3	88.0
300	263	92.6	89.0	92.4	88.1

Table 4: The training time for embeddings and the average accuracy (%) for SVM+Emb model that uses different dimensional embeddings. We ran a single-threaded program on one machine powered by an Intel Core(TM) i5-2400 3.1-GHz with 8GB memory, running Linux.

put features $\langle O(u); E(v); \|O(u) - E(v)\|_1 \rangle$, the two functions O and E are able to characterize the hypernymy relationship held by u and v. Let us consider the same example as above. For `cat`, we use $O(cat)$ as its embeddings, and the candidate hypernyms `dog`, `paws` and `animal` are represented by $O(dog)$, $E(paws)$ and $E(animal)$. These hypernym embeddings are not similar to each other, and it is easier for the classifier to identify `animal` is a hypernym of `cat` because only $E(animal)$ is similar to $O(cat)$.

From the results of SVM+Emb$_{100}$ and SVM+Emb$^2_{100}$, we conclude that the 1-norm distance plays an important role in our model. It improves the average accuracy on all test data sets. However, the distance feature is not so important for the SVM+Word2Vec model. Naturally, for our term embeddings, the 1-norm distance between a hyponym and a hypernym is able to reflect the hypernymy relationship more or less, but the distance between two word2vec embeddings does not have this signal.

5.4 Experiment 2

To further demonstrate the generalization ability of our term embeddings, we carry out the following two experiments. i) We train a classifier on BLESS data and use the same classifier for testing on ENTAILMENT data. Concretely, for each hyponym in ENTAILMENT, we take the pairs containing that hyponym as test data, and we use the BLESS hypernymy-random data set for training. Again, we exclude from the training set any pair containing a term that appears in the test set. At last, we compute the average accuracy across all hyponyms in ENTAILMENT. ii) We use ENTAILMENT data for training and we perform test on BLESS hypernymy-random data. Since the SVM+Word2Vec model is not as good as other supervised models in identifying hypernymy relationships, here we only compare our approach with the Diff+TypeDM$_{300}$ model.

Results We see from Table 5 that our model performs much better than Diff+TypeDM$_{300}$. Although using different domain data for training and test could bring down the performance of both models, the accuracy achieved by Diff+TypeDM$_{300}$ has a much bigger decline than the accuracy achieved by our model. This demonstrates that the hypernymy properties encoded in our term embeddings have a great generalization ability, and they can be used as global features to indicate hypernymy relationship. However the performance of Diff+TypeDM$_{300}$ model is heavily dependent on the training data, because the selected features (i.e. the terms on which the model puts more weights) are directly related to the training data. For example, after training on BLESS, most of the selected features belong to animal, plants and artifacts, since the concept terms in BLESS are mainly from the three broader categories. This is one limit of Diff+TypeDM$_{300}$ model, and it is also an issue that the authors [Roller *et al.*, 2014] want to solve in future work.

Model	Training Set	Test Set	Average Accuracy
SVM+Emb$_{100}$	BLESS	ENTAIL	83.7
	ENTAIL	BLESS	87.1
Diff+TypeDM$_{300}$	BLESS	ENTAIL	72.8
	ENTAIL	BLESS	74.6

Table 5: Average Accuracy (%) of our SVM+EMB$_{100}$ model and Diff+Word2Vec$_{300}$ model obtained on BLESS and ENTAILMENT. Every time we use one data set for training and use the other one for test (ENTAIL represents ENTAILMENT and BLESS refers to the hypernymy-random data set).

5.5 Experiment 3

We learn embeddings of different dimensions (50, 200, and 300) using our dynamic distance-margin model. We then use them for hypernymy identification to see whether the performance is related to the size of embeddings.

Table 4 shows the training time of embeddings as well as the performance of our SVM+Emb model on all data sets. With the increase of dimension, the average accuracy achieved by our model improved slowly. When the dimension

increases from 50 to 200, the accuracy is improved slightly, after that, increasing the dimension nearly has no effect on the performance. We argue that for the task of hypernymy identification, dimension between 100 and 200 is enough to capture all useful information.

6 Conclusion

We propose a supervised distributed approach for hypernymy identification. We introduce a dynamic distance-margin model to learn term embeddings that capture hypernymy properties, and we train an SVM classifier for hypernymy identification using the embeddings as features. Our approach has better accuracy than state-of-the-art supervised distributional methods, and our learned hypernymy embeddings have good generalization ability. Moreover, compared with other co-occurrence based term embeddings, our embeddings have a great advantage for the task of hypernymy identification.

Acknowledgments

This work is supported by NSFC61232006, ARC DP120104168, ARC DP140103578, and ARC DP150102728. We thank the anonymous reviewers for their detailed comments.

References

[Baroni and Lenci, 2011] Marco Baroni and Alessandro Lenci. How we blessed distributional semantic evaluation. In *GEMS*, pages 1–10, 2011.

[Baroni *et al.*, 2012] Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. Entailment above the word level in distributional semantics. In *EACL*, pages 23–32, 2012.

[Bengio *et al.*, 2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, pages 1137–1155, 2003.

[Chelba *et al.*, 2013] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. Technical report, Google, 2013.

[Clarke, 2009] Daoud Clarke. Context-theoretic semantics for natural language: An overview. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, 2009.

[Collobert *et al.*, 2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.

[Etzioni *et al.*, 2006] Oren Etzioni, Michele Banko, and Michael J. Cafarella. Machine reading. In *AAAI*, pages 1517–1519, 2006.

[Glorot *et al.*, 2011] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*, pages 513–520, 2011.

[Hearst, 1992] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING*, pages 539–545, 1992.

[Hill *et al.*, 2014] Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *CoRR*, abs/1408.3456, 2014.

[Kim, 2014] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

[Kotlerman *et al.*, 2010] Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16(4):359–389, 2010.

[Lenci and Benotto, 2012] Alessandro Lenci and Giulia Benotto. Identifying hypernyms in distributional semantic spaces. In *SemEval*, pages 75–79, 2012.

[Liu and Singh, 2004] H. Liu and P. Singh. Conceptnet — a practical commonsense reasoning tool-kit. *BT Technology Journal*, 22(4):211–226, 2004.

[McNamee *et al.*, 2008] Paul McNamee, Rion Snow, Patrick Schone, and James Mayfield. Learning named entity hyponyms for question answering. In *IJCNLP*, pages 799–804, 2008.

[Mikolov *et al.*, 2013a] Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168, 2013.

[Mikolov *et al.*, 2013b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[Miller, 1995] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995.

[Robbins and Monro, 1951] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[Roller *et al.*, 2014] Stephen Roller, Katrin Erk, and Gemma Boleda. Inclusive yet selective: Supervised distributional hypernymy detection. In *COLING: Technical Papers*, pages 1025–1036, 2014.

[Snow *et al.*, 2004] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In *NIPS*, pages 1297–1304, 2004.

[Socher *et al.*, 2011] Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. Parsing natural scenes and natural language with recursive neural networks. In *ICML*, pages 129–136, 2011.

[Suchanek *et al.*, 2007] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge. In *WWW*, pages 697–706. ACM, 2007.

[Suchanek *et al.*, 2008] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: A large ontology from wikipedia and wordnet. *J. Web Sem.*, 6(3):203–217, 2008.

[Turian *et al.*, 2010] J. Turian, L. Ratinov, and Y. Bengio. Word representations: A simple and general method for semi-supervised learning. In *ACL*, 2010.

[Turney and Pantel, 2010] Peter D. Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *J. Artif. Int. Res.*, 37(1):141–188, 2010.

[Turney, 2013] Peter D. Turney. Distributional semantics beyond words: Supervised learning of analogy and paraphrase. *TACL*, 1:353–366, 2013.

[Weeds *et al.*, 2004] Julie Weeds, David J. Weir, and Diana McCarthy. Characterising measures of lexical distributional similarity. In *COLING*, pages 1015–1021, 2004.

[Wu *et al.*, 2012] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Qili Zhu. Probase: a probabilistic taxonomy for text understanding. In *SIGMOD*, pages 481–492, 2012.

[Zhitomirsky-Geffet and Dagan, 2005] Maayan Zhitomirsky-Geffet and Ido Dagan. The distributional inclusion hypotheses and lexical entailment. In *ACL*, pages 107–114, 2005.

[Zhitomirsky-Geffet and Dagan, 2009] Maayan Zhitomirsky-Geffet and Ido Dagan. Bootstrapping distributional feature vector quality. *Comput. Linguist.*, 35(3):435–461, 2009.