Filling Missing Paths: Modeling Co-occurrences of Word Pairs and Dependency Paths for Recognizing Lexical Semantic Relations

Koki Washio and **Tsuneaki Kato**
Department of Language and Information Sciences
Graduate School of Arts and Sciences
The University of Tokyo
3-8-1, Komaba, Meguroku, Tokyo 153-8902 Japan
{kokiwashio@g.ecc, kato@boz.c}.u-tokyo.ac.jp

Abstract

Recognizing lexical semantic relations between word pairs is an important task for many applications of natural language processing. One of the mainstream approaches to this task is to exploit the lexico-syntactic paths connecting two target words, which reflect the semantic relations of word pairs. However, this method requires that the considered words co-occur in a sentence. This requirement is hardly satisfied because of Zipf's law, which states that most content words occur very rarely. In this paper, we propose novel methods with a neural model of $P(path|w_1, w_2)$ to solve this problem. Our proposed model of $P(path|w_1, w_2)$ can be learned in an unsupervised manner and can generalize the co-occurrences of word pairs and dependency paths. This model can be used to augment the path data of word pairs that do not co-occur in the corpus, and extract features capturing relational information from word pairs. Our experimental results demonstrate that our methods improve on previous neural approaches based on dependency paths and successfully solve the focused problem.

1 Introduction

The semantic relations between words are important for many natural language processing tasks, such as recognizing textual entailment (Dagan et al., 2010) and question answering (Yang et al., 2017). Moreover, these relations have been also used as features for neural methods in machine translation (Sennrich and Haddow, 2016) and relation extraction (Xu et al., 2015). This type of information is provided by manually-created semantic taxonomies, such as WordNet (Fellbaum, 1998). However, these resources are expensive to expand manually and have limited domain coverage. Thus, the automatic detection of lexico-semantic relations has been studied for several decades.

One of the most popular approaches is based on patterns that encode a specific kind of relationship (synonym, hypernym, etc.) between adjacent words. This type of approach is called a path-based method. Lexico-syntactic patterns between two words provide information on semantic relations. For example, if we see the pattern, "animals such as a dog" in a corpus, we can infer that *animal* is a hypernym of *dog*. On the basis of this assumption, Hearst (1992) detected the hypernymy relation of two words from a corpus based on several handcrafted lexico-syntactic patterns, e.g., *X such as Y*. Snow et al. (2004) used as features indicative dependency paths, in which target word pairs co-occurred, and trained a classifier with data to detect hypernymy relations.

In recent studies, Shwartz et al. (2016) proposed a neural path-based model that encoded dependency paths between two words into low-dimensional dense vectors with recurrent neural networks (RNN) for hypernymy detection. This method can prevent sparse feature space and generalize indicative dependency paths for detecting lexico-semantic relations. Their model outperformed the previous state-of-the-art path-based method. Moreover, they demonstrated that these dense path representations capture complementary information with word embeddings that contain individual word features. This was indicated by the experimental result that showed the combination of path representations and word embeddings improved classification performance. In addition, Shwartz and Dagan (2016) showed that the neural path-based approach, combined with word embeddings, is effective in recognizing multiple semantic relations.

Although path-based methods can capture the relational information between two words, these methods can obtain clues only for word pairs that

Proceedings of NAACL-HLT 2018, pages 1123–1133
New Orleans, Louisiana, June 1 - 6, 2018.

co-occur in a corpus. Even with a very large corpus, it is almost impossible to observe a co-occurrence of arbitrary word pairs. Thus, path-based methods are still limited in terms of the number of word pairs that are correctly classified.

To address this problem, we propose a novel method with modeling $P(path|w_1, w_2)$ in a neural unsupervised manner, where w_1 and w_2 are the two target words, and $path$ is a dependency path that can connect the joint co-occurrence of w_1 and w_2. A neural model of $P(path|w_1, w_2)$ can generalize co-occurrences of word pairs and dependency paths, and infer plausible dependency paths which connect two words that do not co-occur in a corpus. After unsupervised learning, this model can be used in two ways:

- Path data augmentation through predicting dependency paths that are most likely to co-occur with a given word pair.
- Feature extraction of word pairs, capturing the information of dependency paths as contexts where two words co-occur.

While previous supervised path-based methods used only a small portion of a corpus, combining our models makes it possible to use an entire corpus for learning process.

Experimental results for four common datasets of multiple lexico-semantic relations show that our methods improve the classification performance of supervised neural path-based models.

2 Background

2.1 Supervised Lexical Semantic Relation Detection

Supervised lexical semantic relation detection represents word pairs (w_1, w_2) as feature vectors $\boldsymbol{v}$ and trains a classifier with these vectors based on training data. For word pair representations $\boldsymbol{v}$, we can use the distributional information of each word and path information in which two words co-occur.

Several methods exploit word embeddings (Mikolov et al., 2013; Levy and Goldberg, 2014; Pennington et al., 2014) as distributional information. These methods use a combination of each word's embeddings, such as vector concatenation (Baroni et al., 2012; Roller and Erk, 2016) or vector difference (Roller et al., 2014; Weeds et al., 2014; Vylomova et al., 2016), as word pair representations. While these distributional supervised methods do not require co-occurrences of two words in a sentence, Levy et al. (2015) notes that these methods do not learn the relationships between two words but rather the separate property of each word, i.e., whether or not each word tends to have a target relation.

In contrast, supervised path-based methods can capture relational information between two words. These methods represent a word pair as the set of lexico-syntactic paths, which connect two target words in a corpus (Snow et al., 2004). However, these methods suffer from sparse feature space, as they cannot capture the similarity between indicative lexico-syntactic paths, e.g., *X is a species of Y* and *X is a kind of Y*.

2.2 Neural Path-based Method

A neural path-based method can avoid the sparse feature space of the previous path-based methods (Shwartz et al., 2016; Shwartz and Dagan, 2016). Instead of treating an entire dependency path as a single feature, this model encodes a sequence of edges of a dependency path into a dense vector using a long short-term memory network (LSTM) (Hochreiter and Schmidhuber, 1997).

A dependency path connecting two words can be extracted from the dependency tree of a sentence. For example, given the sentence "A dog is a mammal," with *X* = *dog* and *Y* = *mammal*, the dependency path connecting the two words is `X/NOUN/nsubj/> be/VERB/ROOT/- Y/NOUN/attr/<`. Each edge of a dependency path is composed of a lemma, part of speech (POS), dependency label, and dependency direction.

Shwartz et al. (2016) represents each edge as the concatenation of its component embeddings:

$$\boldsymbol{e} = [\boldsymbol{v}_l; \boldsymbol{v}_{pos}; \boldsymbol{v}_{dep}; \boldsymbol{v}_{dir}] \quad (1)$$

where $\boldsymbol{v}_l, \boldsymbol{v}_{pos}, \boldsymbol{v}_{dep}$,and $\boldsymbol{v}_{dir}$ represent the embedding vectors of the lemma, POS, dependency label, and dependency direction respectively. This edge vector $\boldsymbol{e}$ is an input of the LSTM at each time step. Here, $\boldsymbol{h_t}$, the hidden state at time step t, is abstractly computed as:

$$\boldsymbol{h}_t = LSTM(\boldsymbol{h}_{t-1}, \boldsymbol{e}_t) \quad (2)$$

where $LSTM$ computes the current hidden state given the previous hidden state $\boldsymbol{h}_{t-1}$ and the current input edge vector $\boldsymbol{e}_t$ along with the LSTM architecture. The final hidden state vector $\boldsymbol{o}_p$ is

treated as the representation of the dependency path p.

When classifying a word pair (w_1, w_2), the word pair is represented as the average of the dependency path vectors that connect two words in a corpus:

$$\begin{aligned} \boldsymbol{v}_{(w_1,w_2)} &= \boldsymbol{v}_{paths(w_1,w_2)} \\ &= \frac{\sum_{p \in paths(w_1,w_2)} f_{p,(w_1,w_2)} \cdot \boldsymbol{o}_p}{\sum_{p \in paths(w_1,w_2)} f_{p,(w_1,w_2)}} \end{aligned} \quad (3)$$

where $paths(w_1, w_2)$ is the set of dependency paths that connects w_1 and w_2 in the corpus, and $f_{p,(w_1,w_2)}$ is the frequency of p in $paths(w_1, w_2)$. The final output of the network is calculated as follows:

$$\boldsymbol{y} = softmax(\boldsymbol{W}\boldsymbol{v}_{(w_1,w_2)} + \boldsymbol{b}) \quad (4)$$

where $\boldsymbol{W} \in \mathbb{R}^{|c| \times d}$ is a linear transformation matrix, $\boldsymbol{b} \in \mathbb{R}^{|c|}$ is a bias parameter, $|c|$ is the number of the output class, and d is the size of $\boldsymbol{v}_{(w_1,w_2)}$.

This neural path-based model can be combined with distributional methods. Shwartz et al. (2016) concatenated $\boldsymbol{v}_{paths(w_1,w_2)}$ to the word embeddings of w_1 and w_2, redefining $\boldsymbol{v}_{(w_1,w_2)}$ as:

$$\boldsymbol{v}_{(w_1,w_2)} = [\boldsymbol{v}_{w_1}; \boldsymbol{v}_{paths(w_1,w_2)}; \boldsymbol{v}_{w_2}] \quad (5)$$

where $\boldsymbol{v}_{w_1}$ and $\boldsymbol{v}_{w_2}$ are word embeddings of w_1 and w_2, respectively. This integrated model, named LexNET, exploits both path information and distributional information, and has high generalization performance for lexical semantic relation detection.

2.3 Missing Path Problem

All path-based methods, including the neural ones, suffer from data sparseness as they depend on word pair co-occurrences in a corpus. However, we cannot observe all co-occurrences of semantically related words even with a very large corpus because of Zipf's law, which states that the frequency distribution of words has a long tail; in other words, most words occur very infrequently (Hanks, 2009). In this paper, we refer to this phenomenon as the missing path problem.

This missing path problem leads to the fact that path-based models cannot find any clues for two words that do not co-occur. Thus, in the neural path-based method, $paths(w_1, w_2)$ for these word pairs is padded with an empty path, like `UNK-lemma/UNK-POS/UNK-dep/UNK-dir`. However, this process makes path-based classifiers unable to distinguish between semantically-related pairs with no co-occurrences and those that have no semantic relation.

In an attempt to solve this problem, Necsulescu et al. (2015) proposed a method that used a graph representation of a corpus. In this graph, words and dependency relations were denoted as nodes and labeled directed edges, respectively. From this graph representation, paths linking two target words can be extracted through bridging words, even if the two words do not co-occur in the corpus. They represent word pairs as the sets of paths linking word pairs on the graph and train a support vector machine classifier with training data, thereby improving recall. However, the authors reported that this method still suffered from data sparseness.

In this paper, we address this missing path problem, which generally restricts path-based methods, by neural modeling $P(path|w_1, w_2)$.

3 Method

We present a novel method for modeling $P(path|w_1, w_2)$. The purpose of this method is to address the missing path problem by generalizing the co-occurrences of word pairs and dependency paths. To model $P(path|w_1, w_2)$, we used the context-prediction approach (Collobert and Weston, 2008; Mikolov et al., 2013; Levy and Goldberg, 2014; Pennington et al., 2014), which is a widely used method for learning word embeddings. In our proposed method, word pairs and dependency paths are represented as embeddings that are updated with unsupervised learning through predicting $path$ from w_1 and w_2 (Section 3.1).

After the learning process, our model can be used to (1) augment path data by predicting the plausibility of the co-occurrence of two words and a dependency path (Section 3.2); and to (2) extract useful features from word pairs, which reflect the information of co-occurring dependency paths (Section 3.3).

3.1 Unsupervised Learning

There are many possible ways to model $P(path|w_1, w_2)$. In this paper, we present a straightforward and efficient architecture, similar to the skip-gram with negative sampling (Mikolov et al., 2013).

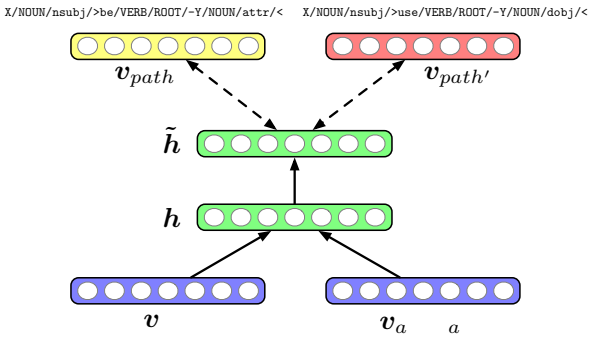

Figure 1: An illustration of our network for modeling $P(path|w_1, w_2)$. Given a word pair $(dog, animal)$, our model makes $\tilde{\boldsymbol{h}}$ of $(dog, animal)$ similar to $\boldsymbol{v}_{path}$ of the observed co-occurring dependency path `X/NOUN/nsubj/> be/VERB/ROOT/- Y/NOUN/attr/<` and dissimilar to $\boldsymbol{v}_{path'}$ of the unobserved paths, such as `X/NOUN/nsubj/> use/VERB/ROOT/- Y/NOUN/dobj/<`, through unsupervised learning.

Figure 1 depicts our network structure, which is described below.

Data and Network Architecture

We are able to extract many triples $(w_1, w_2, path)$ from a corpus after dependency parsing. We denote a set of these triples as D. These triples are the instances used for the unsupervised learning of $P(path|w_1, w_2)$. Given $(w_1, w_2, path)$, our model learns through predicting $path$ from w_1 and w_2.

We encode word pairs into dense vectors as follows:

$$\boldsymbol{h}_{(w1,w2)} = tanh(\boldsymbol{W}_1[\boldsymbol{v}_{w_1}; \boldsymbol{v}_{w_2}] + \boldsymbol{b}_1) \quad (6)$$

$$\tilde{\boldsymbol{h}}_{(w_1,w_2)} = tanh(\boldsymbol{W}_2\boldsymbol{h}_{(w1,w2)} + \boldsymbol{b}_2) \quad (7)$$

where $[\boldsymbol{v}_{w_1}; \boldsymbol{v}_{w_2}]$ is the concatenation of the word embeddings of w_1 and w_2; $\boldsymbol{W}_1$, $\boldsymbol{b}_1$, $\boldsymbol{W}_2$, and $\boldsymbol{b}_2$ are the parameter matrices and bias parameters of the two linear transformations; and $\tilde{\boldsymbol{h}}_{(w_1,w_2)}$ is the representation of the word pair.

We associate each $path$ with the embedding $\boldsymbol{v}_{path}$, initialized randomly. While we use a simple way to represent dependency paths in this paper, LSTM can be used to encode each path in the way described in Section 2.2. If LSTM is used, learning time increases but similarities among paths will be captured.

Objective

We used the negative sampling objective for training (Mikolov et al., 2013). Given the word pair representations $\tilde{\boldsymbol{h}}_{(w_1,w_2)}$ and the dependency path representations $\boldsymbol{v}_{path}$, our model was trained to distinguish real $(w_1, w_2, path)$ triples from incorrect ones. The log-likelihood objective is as follows:

$$L = \sum_{(w_1,w_2,path)\in D} \log \sigma(\boldsymbol{v}_{path} \cdot \tilde{\boldsymbol{h}}_{(w_1,w_2)}) + \sum_{(w_1,w_2,path')\in D'} \log \sigma(-\boldsymbol{v}_{path'} \cdot \tilde{\boldsymbol{h}}_{(w_1,w_2)}) \quad (8)$$

where, D' is the set of randomly generated negative samples. We constructed n triples $(w_1, w_2, path')$ for each $(w_1, w_2, path) \in D$, where n is a hyperparameter and each $path'$ is drawn according to its unigram distribution raised to the $3/4$ power. The objective L was maximized using the stochastic gradient descent algorithm.

3.2 Path Data Augmentation

After the unsupervised learning described above, our model of $P(path|w_1, w_2)$ can assign the plausibility score $\sigma(\boldsymbol{v}_{path} \cdot \tilde{\boldsymbol{h}}_{(w_1,w_2)})$ to the co-occurrences of a word pair and a dependency path. We can then append the plausible dependency paths to $paths(w_1, w_2)$, the set of dependency paths that connects w_1 and w_2 in the corpus, based on these scores.

We calculate the score of each dependency path given $(X = w_1, Y = w_2)$ and append the k dependency paths with the highest scores to $paths(w_1, w_2)$, where k is a hyperparameter. We perform the same process given $(X = w_2, Y = w_1)$ with the exception of swapping the `X` and `Y` in the dependency paths to be appended. As a result, we add $2k$ dependency paths to the set of dependency paths for each word pair. Through this data augmentation, we can obtain plausible dependency paths even when word pairs do not co-occur in the corpus. Note that we retain the empty path indicators of $paths(w_1, w_2)$, as we believe that this information contributes to classifying two unrelated words.

3.3 Feature Extractor of Word Pairs

Our model can be used as a feature extractor of word pairs. We can exploit $\tilde{\boldsymbol{h}}_{(w_1,w_2)}$ to represent the word pair (w_1, w_2). This representation captures the information of co-occurrence dependency paths of (w_1, w_2) in a generalized fashion. Thus, $\tilde{\boldsymbol{h}}_{(w_1,w_2)}$ is used to construct the pseudo-path representation $\boldsymbol{v}_{p-paths(w_1,w_2)}$. With our model, we represent the word pair (w_1, w_2) as

datasets	relations
K&H+N	hypernym, meronym, co-hyponym, random
BLESS	hypernym, meronym, co-hyponym, random
ROOT09	hypernym, co-hyponym, random
EVALution	hypernym, meronym, attribute, synonym, antonym, holonym, substance meronym

Table 1: The relation types in each dataset.

follows:

$$\boldsymbol{v}_{p-paths(w_1,w_2)} = [\tilde{\boldsymbol{h}}_{(w_1,w_2)}; \tilde{\boldsymbol{h}}_{(w_2,w_1)}] \quad (9)$$

This representation can be used for word pair classification tasks, such as lexical semantic relation detection.

4 Experiment

In this section, we examine how our method improves path-based models on several datasets for recognizing lexical semantic relations. In this paper, we focus on major noun relations, such as hypernymy, co-hypernymy, and meronymy.

4.1 Dataset

We relied on the datasets used in Shwartz and Dagan (2016); K&H+N (Necsulescu et al., 2015). BLESS (Baroni and Lenci, 2011), EVALution (Santus et al., 2015), and ROOT09 (Santus et al., 2016). These datasets were constructed with knowledge resources (e.g., WordNet, Wikipedia), crowd-sourcing, or both. We used noun pair instances of these datasets.[1] Table 1 displays the relations in each dataset used in our experiments. Note that we removed the two relations *Entails* and *MemberOf* with few instances from EVALution following Shwartz and Dagan (2016). For data splitting, we used the presplitted train/val/test sets from Shwartz and Dagan (2016) after removing all but the noun pairs from each set.

4.2 Corpus and Dependency Parsing

For path-based methods, we used the June 2017 Wikipedia dump as a corpus and extracted $(w_1, w_2, path)$ triples of noun pairs using the dependency parser of spaCy[2] to construct D. In this process, w_1 and w_2 were lemmatized with spaCy. We only used the dependency paths which occurred at least five times following the implementation of Shwartz and Dagan (2016).[3]

datasets	instances	instances with paths	proportion
K&H+N	57509	8866	15.4%
BLESS	14558	8775	60.3%
ROOT09	8602	6582	76.5%
EVALution	3240	3199	98.7%

Table 2: The number and proportion of instances whose dependency path is obtained from each dataset

Table 2 displays the number of instances and the proportion of the instances for which at least one dependency path was obtained.

4.3 Baseline

We conducted experiments with three neural path-based methods. The implementation details below follow those in Shwartz and Dagan (2016). We implemented all models using Chainer.[4]

Neural Path-Based Model (NPB). We implemented and trained the neural path-based model described in Section 2.2. We used the two-layer LSTM with 60-dimensional hidden units. An input vector was composed of embedding vectors of the lemma (50 dims), POS (4 dims), dependency label (5 dims), and dependency direction (1 dim). Regularization was applied by a dropout on each of the components embeddings (Iyyer et al., 2015; Kiperwasser and Goldberg, 2016).

LexNET. We implemented and trained the integrated model LexNET as described in Section 2.2. The LSTM details are the same as in the NPB model.

LexNET_h. This model, a variant of LexNET, has an additional hidden layer between the output layer and $\boldsymbol{v}_{(w_1,w_2)}$ of Equation (5). Because of this additional hidden layer, this model can take into account the interaction of the path information

[1] We focused only noun pairs to shorten the unsupervised learning time, though this restriction is not necessary for our methods and the unsupervised learning is still tractable.

[2] https://spacy.io

[3] https://github.com/vered1986/LexNET

[4] https://chainer.org

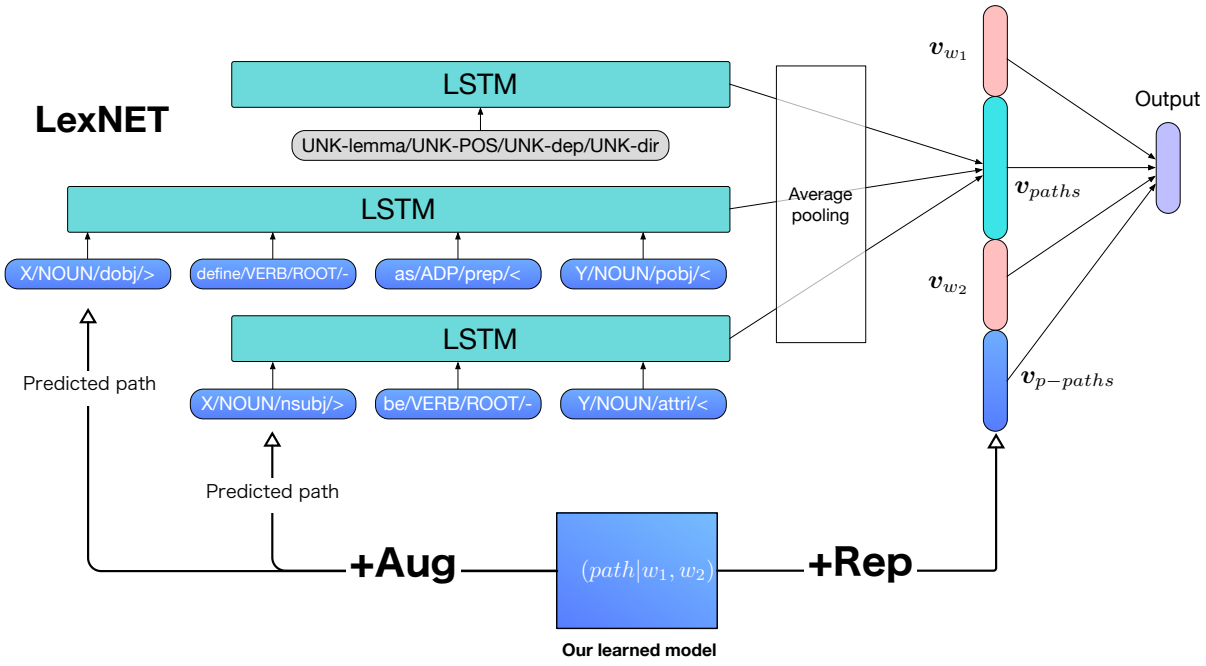

Figure 2: Illustration of +Aug and +Rep applied to LexNET. +Aug predicts plausible paths from two word embeddings, and these paths are fed into the LSTM path encoder. +Rep concatenates the pseudo-path representation $\boldsymbol{v}_{p-paths(w_1,w_2)}$ with the penultimate layer of LexNET

and distributional information of two word embeddings. The size of the additional hidden layer was set to 60.

Following Shwartz and Dagan (2016), we optimized each model using Adam (whose learning rate is 0.001) while tuning the dropout rate dr among $\{0.0, 0.2, 0.4\}$ on the validation set. The minibatch size was set to 100.

We initialized the lemma embeddings of LSTM and concatenated the word embeddings of LexNET with the pretrained 50-dimensional GloVe vector.[5] Training was stopped if performance on the validation set did not improve for seven epochs, and the best model for test evaluation was selected based on the score of the validation set.

4.4 Our Method

We implemented and trained our model of $P(path|w_1, w_2)$, described in Section 3.1, as follows. We used the most frequent 30,000 paths connecting nouns as the context paths for unsupervised learning. We initialized word embeddings with the same pretrained GloVe vector as the baseline models. For unsupervised learning data, we extracted $(w_1, w_2, path)$, whose w_1 and w_2 are included in the vocabulary of the GloVe vector, and whose $path$ is included in the context paths, from D. The number of these triples was 217,737,765.

We set the size of $\boldsymbol{h}_{(w_1,w_2)}$, $\tilde{\boldsymbol{h}}_{(w_1,w_2)}$, and $\boldsymbol{v}_{path}$ for context paths to 100. The negative sampling size n was set to 5. We trained our model for five epochs using Adam (whose learning rate is 0.001). The minibatch size was 100. To preserve the distributional regularity of the pretrained word embeddings, we did not update the input word embeddings during the unsupervised learning.

With our trained model, we applied the two methods described in Section 3.2 and 3.3 to the NPB and LexNET models as follows:

+Aug. We added the most plausible $2k$ paths to each $paths(w_1, w_2)$ as in Section 3.2. We tuned $k \in \{1, 3, 5\}$ on the validation set.

+Rep. We concatenated $\boldsymbol{v}_{p-paths(w_1,w_2)}$ in Equation (9) with the penultimate layer. To focus on the pure contribution of unsupervised learning, we did not update this component during supervised learning.

Figure 2 illustrates +Aug and +Rep applied to LexNET in the case where the two target words, w_1 and w_2, do not co-occur in the corpus.

[5] https://nlp.stanford.edu/projects/glove/

Models	K&H+N	BLESS	ROOT09	EVALution
NPB	0.495	0.773	0.731	0.463
NPB+Aug	**0.897**	**0.842**	**0.778**	**0.489**

Table 3: Classification performance of the neural path-based model (NPB) and that with the path data augmentation (NPB+Aug).

Models	K&H+N	BLESS	ROOT09	EVALution
LexNET	0.969	0.922	0.776	0.539
LexNET_h	0.968	0.927	0.810	0.540
LexNET+Aug	**0.970**	0.927	0.806	0.545
LexNET+Rep	**0.970**	**0.944**	**0.832**	0.565
LexNET+Aug+Rep	0.969	0.942	0.820	**0.567**

Table 4: Classification performance of the integrated model, LexNET and LexNET_h, and those with our methods, +Aug and +Rep.

5 Result

In this section we examine how our methods improved the baseline models. Following the previous research (Shwartz and Dagan, 2016), the performance metrics were the "averaged" $F1$ of scikit-learn (Pedregosa et al., 2011), which computes the $F1$ for each relation, and reports their average weighted by the number of true instances for each relation.

5.1 Path-based Model and Path Data Augmentation

We examined whether or not our path data augmentation method +Aug contributes to the neural path-based method. The results are displayed in Table 3.

Applying our path data augmentation method improved the classification performance on each dataset. Especially for K&H+N, the large dataset where the three-fourths of word pairs had no paths, our method significantly improved the performance. This result shows that our path data augmentation effectively solves the missing path problem. Moreover, the model with our method outperforms the baseline on EVALution, in which nearly all word pairs co-occurred in the corpus. This indicates that the predicted paths provide useful information and enhance the path-based classification. We examine the paths that were predicted by our model of $P(path|w_1, w_2)$ in Section 6.1.

5.2 Integrated Model and Our Methods

We investigated how our methods using modeling $P(path|w_1, w_2)$ improved the baseline integrated model, LexNET. Table 4 displays the results.

Our proposed methods, +Aug and +Rep, improved the performance of LexNET on each dataset.[6] Moreover, the best score on each dataset was achieved by the model to which our methods were applied. These results show that our methods are also effective with the integrated models based on path information and distributional information.

The table also shows that LexNET+Rep outperforms LexNET_h, though the former has fewer parameters to be tuned during the supervised learning than the latter. This indicates that the word pair representations of our model capture information beyond the interaction of two word embeddings. We investigate the properties of our word pair representation in Section 6.2.

Finally, We found that applying both methods did not necessarily yield the best performance. A possible explanation for this is that applying both methods is redundant, as both +Aug and +Rep depend on the same model of $P(path|w_1, w_2)$.

6 Analysis

In this section, we investigate the properties of the predicted dependency paths and word pair representations of our model.

6.1 Predicted Dependency Paths

We extracted the word pairs of BLESS without co-occurring dependency paths and predicted the

[6]The improvement for K&H+N is smaller than those for the others. We think this owes to most instances of this dataset being correctly classified only by distributional information. This view is supported by Shwartz and Dagan (2016), in which LexNET hardly outperformed a distributional method for this dataset.

Word pair	Relation	Predicted paths
X = "jacket", Y = "commodity"	hypernym	**X/NOUN/nsubj/> be/VERB/ROOT/- shooter/NOUN/attr/< Y/NOUN/compound/<**
		X/NOUN/nsubj/> be/VERB/ROOT/- Y/NOUN/attr/< manufacture/VERB/acl/<
		red/ADJ/amod/< X/NOUN/nsubj/> be/VERB/ROOT/- Y/NOUN/attr/<
X = "goose", Y = "creature"	hypernym	**X/NOUN/nsubj/> be/VERB/ROOT/- species/NOUN/attr/< of/ADP/prep/< Y/NOUN/pobj/< of/ADP/prep/>**
		X/NOUN/nsubj/> be/VERB/ROOT/- specie/NOUN/attr/< of/ADP/prep/< Y/NOUN/pobj/< in/ADP/prep/>
		X/NOUN/pobj/> of/ADP/ROOT/- bird/NOUN/pobj/< Y/NOUN/conj/<
X = "owl", Y = "rump"	meronym	**X/NOUN/ROOT/- represent/VERB/relcl/< Y/NOUN/nsubj/<**
		X/NOUN/nsubj/> have/VERB/ROOT/- Y/NOUN/dobj/< be/VERB/relcl/>
		all/DET/det/< X/NOUN/nsubj/> have/VERB/ROOT/- Y/NOUN/dobj/<
X = "mug", Y = "plastic"	meronym	X/NOUN/pobj/> of/ADP/ROOT/- arm/NOUN/pobj/< Y/NOUN/conj/<
		the/DET/det/< X/NOUN/nsubjpass/> make/VERB/ROOT/- from/ADP/prep/< Y/NOUN/pobj/<
		X/NOUN/compound/> gun/NOUN/ROOT/- Y/NOUN/appos/<
X = "carrot", Y = "beans"	co-hyponym	**X/NOUN/compound/> leaf/NOUN/ROOT/- Y/NOUN/conj/<**
		X/NOUN/compound/> specie/NOUN/ROOT/- Y/NOUN/conj/<
		X/NOUN/dobj/> use/VERB/ROOT/- in/ADP/prep/< Y/NOUN/pobj/< of/ADP/prep/>
X = "cello", Y = "kazoo"	co-hyponym	**X/NOUN/dobj/> play/VERB/ROOT/- guitar/NOUN/dobj/< Y/NOUN/conj/<**
		X/NOUN/pobj/> for/ADP/ROOT/- piano/NOUN/pobj/< Y/NOUN/conj<
		X/NOUN/pobj/> on/ADP/ROOT/- drum/NOUN/pobj/< Y/NOUN/conj/<

Table 5: Predicted paths with our model for a word pair of each relation in BLESS.

plausible dependency paths of those pairs with our model of $P(path|w_1, w_2)$. The examples are displayed in Table 5 at the top three paths. We used the bold style for the paths that we believe to be indicative or representative for a given relationship.

Our model predicted plausible and indicative dependency paths for each relation, although the predicted paths also contain some implausible or unindicative ones. For hypernymy, our model predicted variants of the is-a path according to domains, such as *X is Y manufactured* in the clothing domain and *X is a species of Y* in the animal domain. For $(owl, rump)$, which is a meronymy pair, the top predicted path was *X that Y represent*. This is not plausible for $(owl, rump)$ but is indicative for meronymy, particularly member-of relations. Moreover, domain-independent paths which indicate meronymy, such as *all X have Y*, were predicted. For $(mug, plastic)$, one of the predicted paths, *X is made from Y*, is also a domain-independent indicative path for meronymy. For co-hypernymy, our model predicted domain-specific paths, which indicate that two nouns are of the same kind. For examples, given *X leaf and Y* and *X specie and Y* of $(carrot, beans)$, we can infer that both X and Y are plants or vegetables. Likewise, given *play X, guitar, and Y* of $(cello, kazoo)$, we can infer that both X and Y are musical instruments. These examples show that our path data augmentation is effective for the missing path problem and enhances path-based models.

6.2 Visualizing Word Pair Representations

We visualized the word pair representations $\boldsymbol{v}_{p-paths(w_1,w_2)}$ to examine their specific properties. In BLESS, every pair was annotated with 17 domain class labels. For each domain, we reduced the dimensionality of the representations using t-SNE (Maaten and Hinton, 2008) and plotted the data points of the hypernyms, co-hyponyms, and meronyms. We compared our representations with the concatenation of two word embeddings (pre-trained 50-dimensional GloVe). The examples are displayed in Figure 3.

We found that our representations (the top row in Figure 3) grouped the word pairs according to their semantic relation in some specific domains based only on unsupervised learning. This property is desirable for the lexical semantic relation detection task. In contrast to our representations,

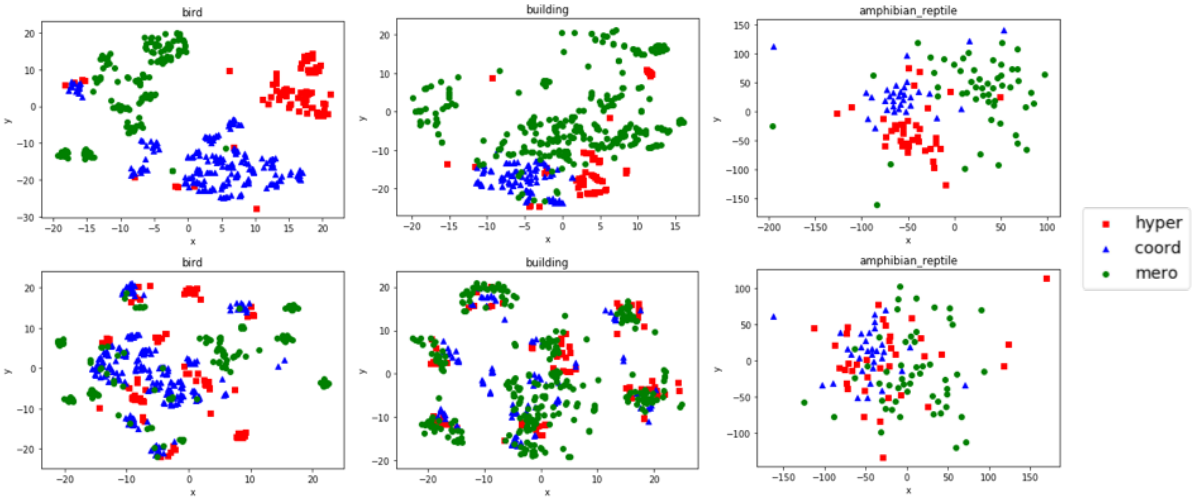

Figure 3: Visualization of the our word pair representations $\boldsymbol{v}_{p-paths(w_1,w_2)}$ (top row) and the concatenation of two word embeddings (bottom row) using t-SNE in some domains. The two axes of each plot, x and y, are the reduced dimensions using t-SNE.

the concatenation of word embeddings (the bottom row in Figure 3) has little or no such tendency in all domains. The data points of the concatenation of word embeddings are scattered or jumbled. This is because the concatenation of word embeddings cannot capture the relational information of word pairs but only the distributional information of each word (Levy et al., 2015).

This visualization further shows that our word pair representations can be used as pseudo-path representations to alleviate the missing path problem.

7 Conclusion

In this paper, we proposed the novel methods with modeling $P(path|w_1, w_2)$ to solve the missing path problem. Our neural model of $P(path|w_1, w_2)$ can be learned from a corpus in an unsupervised manner, and can generalize co-occurrences of word pairs and dependency paths. We demonstrated that this model can be applied in the two ways: (1) to augment path data by predicting plausible paths for a given word pair, and (2) to extract from word pairs useful features capturing co-occurring path information. Finally, our experiments demonstrated that our methods can improve upon the previous models and successfully solve the missing path problem.

In future work, we will explore unsupervised learning with a neural path encoder. Our model bears not only word pair representations but also dependency path representations as context vectors. Thus, we intend to apply these representations to various tasks, which path representations contribute to.

Acknowledgments

This work was supported by JSPS KAKENHI Grant numbers JP17H01831, JP15K12873.

References

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 23–32. http://www.aclweb.org/anthology/E12-1004.

Marco Baroni and Alessandro Lenci. 2011. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on Geometrical Models of Natural Language Semantics*. Association for Computational Linguistics, pages 1–10. http://www.aclweb.org/anthology/W11-2501.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*. ACM, pages 160–167.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2010. Recognizing textual entailment: Rational, evaluation and approaches erratum. *Natural Language Engineering* 16(1):105–105. https://doi.org/10.1017/S1351324909990234.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, Mass.

Patrick Hanks. 2009. The impact of corpora on dictionaries. In Paul Baker, editor, *Contemporary Corpus Linguistics*, Continuum, London, Great Britain, pages 214–236.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING 1992 Volume 2: The 15th International Conference on Computational Linguistics*. http://www.aclweb.org/anthology/C92-2082.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1681–1691. https://doi.org/10.3115/v1/P15-1162.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association of Computational Linguistics* 4:313–327. http://www.aclweb.org/anthology/Q16-1023.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 302–308. https://doi.org/10.3115/v1/P14-2050.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 970–976. https://doi.org/10.3115/v1/N15-1098.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research* 9(Nov):2579–2605.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. Curran Associates Inc., USA, NIPS'13, pages 3111–3119. http://dl.acm.org/citation.cfm?id=2999792.2999959.

Silvia Necsulescu, Sara Mendes, David Jurgens, Núria Bel, and Roberto Navigli. 2015. Reading between the lines: Overcoming data sparsity for accurate classification of lexical relationships. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics, pages 182–192. https://doi.org/10.18653/v1/S15-1021.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research* 12(Oct):2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1532–1543. https://doi.org/10.3115/v1/D14-1162.

Stephen Roller and Katrin Erk. 2016. Relations such as hypernymy: Identifying and exploiting hearst patterns in distributional vectors for lexical entailment. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2163–2172. https://doi.org/10.18653/v1/D16-1234.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, pages 1025–1036. http://www.aclweb.org/anthology/C14-1097.

Enrico Santus, Alessandro Lenci, Tin-Shing Chiu, Qin Lu, and Chu-Ren Huang. 2016. Nine features in a random forest to learn taxonomical semantic relations. In *LREC*. Portorož, Slovenia.

Enrico Santus, Frances Yung, Alessandro Lenci, and Chu-Ren Huang. 2015. Evalution 1.0: an evolving semantic dataset for training and evaluation of distributional semantic models. In *Proceedings of The 4th Workshop on Linked Data in Linguistics (LDL-2015)*. Association for Computational Linguistics, pages 64–69. https://doi.org/10.18653/v1/W15-4208.

Rico Sennrich and Barry Haddow. 2016. Linguistic Input Features Improve Neural Machine Translation. In *Proceedings of the First Conference on Machine Translation*. Association for Computational Linguistics, Berlin, Germany, pages 83–91. http://www.aclweb.org/anthology/W16-2209.pdf.

Vered Shwartz and Ido Dagan. 2016. Path-based vs. distributional information in recognizing lexical semantic relations. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex-V), in COLING*. Osaka, Japan. http://www.aclweb.org/anthology/W16-5304.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 2389–2398. https://doi.org/10.18653/v1/P16-1226.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems 17*, MIT Press, Cambridge, MA, pages 1297–1304. http://books.nips.cc/papers/files/nips17/NIPS2004_0887.pdf.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. 2016. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1671–1682. https://doi.org/10.18653/v1/P16-1158.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, pages 2249–2259. http://www.aclweb.org/anthology/C14-1212.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1785–1794. http://aclweb.org/anthology/D15-1206.

Shuo Yang, Lei Zou, Zhongyuan Wang, Jun Yan, and Ji-Rong Wen. 2017. Efficiently answering technical questions-a knowledge graph approach. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. pages 3111–3118.