Task-Oriented Learning of Word Embeddings
for Semantic Relation Classification

Kazuma Hashimoto[†], Pontus Stenetorp[‡], Makoto Miwa[§], and Yoshimasa Tsuruoka[†]
†The University of Tokyo, 3-7-1 Hongo, Bunkyo-ku, Tokyo, Japan
`{hassy,tsuruoka}@logos.t.u-tokyo.ac.jp`
‡University College London, London, United Kingdom
`pontus@stenetorp.se`
§Toyota Technological Institute, 2-12-1 Hisakata, Tempaku-ku, Nagoya, Japan
`makoto-miwa@toyota-ti.ac.jp`

Abstract

We present a novel learning method for word embeddings designed for relation classification. Our word embeddings are trained by predicting words between noun pairs using lexical relation-specific features on a large unlabeled corpus. This allows us to explicitly incorporate relation-specific information into the word embeddings. The learned word embeddings are then used to construct feature vectors for a relation classification model. On a well-established semantic relation classification task, our method significantly outperforms a baseline based on a previously introduced word embedding method, and compares favorably to previous state-of-the-art models that use syntactic information or manually constructed external resources.

1 Introduction

Automatic classification of semantic relations has a variety of applications, such as information extraction and the construction of semantic networks (Girju et al., 2007; Hendrickx et al., 2010). A traditional approach to relation classification is to train classifiers using various kinds of features with class labels annotated by humans. Carefully crafted features derived from lexical, syntactic, and semantic resources play a significant role in achieving high accuracy for semantic relation classification (Rink and Harabagiu, 2010).

In recent years there has been an increasing interest in using *word embeddings* as an alternative to traditional hand-crafted features. Word embeddings are represented as real-valued vectors and capture syntactic and semantic similarity between words. For example, *word2vec*[1] (Mikolov et al., 2013b) is a well-established tool for learning word embeddings. Although word2vec has successfully been used to learn word embeddings, these kinds of word embeddings capture only co-occurrence relationships between words (Levy and Goldberg, 2014). While simply adding word embeddings trained using window-based contexts as additional features to existing systems has proven valuable (Turian et al., 2010), more recent studies have focused on how to tune and enhance word embeddings for specific tasks (Bansal et al., 2014; Boros et al., 2014; Chen et al., 2014; Guo et al., 2014; Nguyen and Grishman, 2014) and we continue this line of research for the task of relation classification.

In this work we present a learning method for word embeddings specifically designed to be useful for relation classification. The overview of our system and the embedding learning process are shown in Figure 1. First we train word embeddings by predicting each of the words between noun pairs using lexical relation-specific features on a large unlabeled corpus. We then use the word embeddings to construct lexical feature vectors for relation classification. Lastly, the feature vectors are used to train a relation classification model.

We evaluate our method on a well-established semantic relation classification task and compare it to a baseline based on word2vec embeddings and previous state-of-the-art models that rely on either manually crafted features, syntactic parses or external semantic resources. Our method significantly outperforms the word2vec-based baseline, and compares favorably with previous state-of-the-art models, despite relying only on lexi-

---

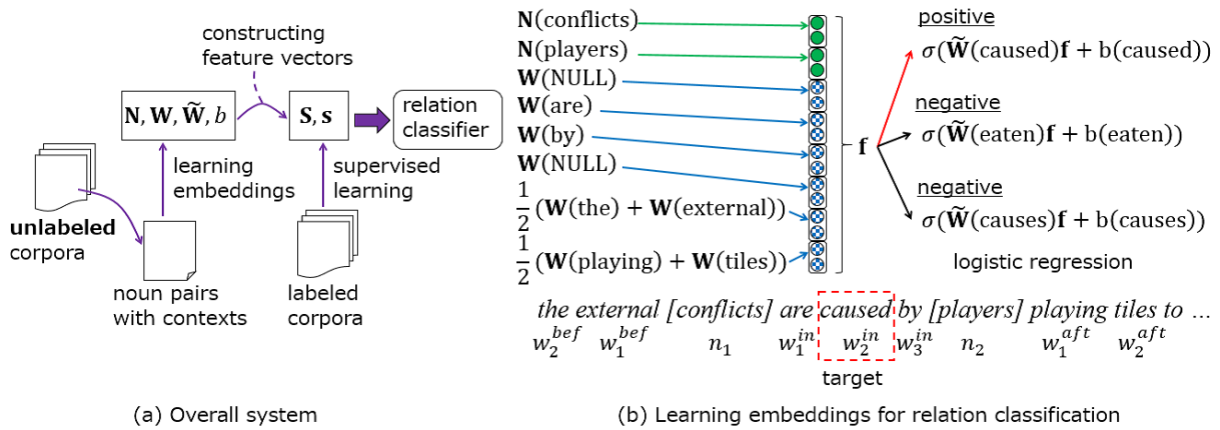[1] `https://code.google.com/p/word2vec/`.

Figure 1: The overview of our system (a) and the embedding learning method (b). In the example sentence, each of *are*, *caused*, and *by* is treated as a target word to be predicted during training.

cal level features and no external annotated resources. Furthermore, our qualitative analysis of the learned embeddings shows that n-grams of our embeddings capture salient syntactic patterns similar to semantic relation types.

2 Related Work

A traditional approach to relation classification is to train classifiers in a supervised fashion using a variety of features. These features include lexical bag-of-words features and features based on syntactic parse trees. For syntactic parse trees, the paths between the target entities on constituency and dependency trees have been demonstrated to be useful (Bunescu and Mooney, 2005; Zhang et al., 2006). On the shared task introduced by Hendrickx et al. (2010), Rink and Harabagiu (2010) achieved the best score using a variety of hand-crafted features which were then used to train a Support Vector Machine (SVM).

Recently, word embeddings have become popular as an alternative to hand-crafted features (Collobert et al., 2011). However, one of the limitations is that word embeddings are usually learned by predicting a target word in its context, leading to only local co-occurrence information being captured (Levy and Goldberg, 2014). Thus, several recent studies have focused on overcoming this limitation. Le and Mikolov (2014) integrated paragraph information into a word2vec-based model, which allowed them to capture paragraph-level information. For dependency parsing, Bansal et al. (2014) and Chen et al. (2014) found ways to improve performance by integrating dependency-based context information into their embeddings.

Bansal et al. (2014) trained embeddings by defining parent and child nodes in dependency trees as contexts. Chen et al. (2014) introduced the concept of feature embeddings induced by parsing a large unannotated corpus and then learning embeddings for the manually crafted features. For information extraction, Boros et al. (2014) trained word embeddings relevant for event role extraction, and Nguyen and Grishman (2014) employed word embeddings for domain adaptation of relation extraction. Another kind of task-specific word embeddings was proposed by Tang et al. (2014), which used sentiment labels on tweets to adapt word embeddings for a sentiment analysis tasks. However, such an approach is only feasible when a large amount of labeled data is available.

3 Relation Classification Using Word Embedding-based Features

We propose a novel method for learning word embeddings designed for relation classification. The word embeddings are trained by *predicting each word between noun pairs*, given the corresponding low-level features for relation classification. In general, to classify relations between pairs of nouns the most important features come from the pairs themselves and the words between and around the pairs (Hendrickx et al., 2010). For example, in the sentence in Figure 1 (b) there is a *cause-effect* relationship between the two nouns *conflicts* and *players*. To classify the relation, the most common features are the noun pair (*conflicts, players*), the words between the noun pair (*are, caused, by*), the words before the pair (*the, external*), and the words after the pair (*playing, tiles,*

269

to, ...). As shown by Rink and Harabagiu (2010), the words between the noun pairs are the most effective among these features. Our main idea is to treat the most important features (the words between the noun pairs) as the targets to be predicted and other lexical features (noun pairs, words outside them) as their contexts. Due to this, we expect our embeddings to capture relevant features for relation classification better than previous models which only use window-based contexts.

In this section we first describe the learning process for the word embeddings, focusing on lexical features for relation classification (Figure 1 (b)). We then propose a simple and powerful technique to construct features which serve as input for a softmax classifier. The overview of our proposed system is shown in Figure 1 (a).

3.1 Learning Word Embeddings

Assume that there is a noun pair $\mathbf{n} = (n_1, n_2)$ in a sentence with M_{in} words between the pair and M_{out} words before and after the pair:

- $\mathbf{w}_{in} = (w_1^{in}, \ldots, w_{M_{in}}^{in})$,

- $\mathbf{w}_{bef} = (w_1^{bef}, \ldots, w_{M_{out}}^{bef})$, and

- $\mathbf{w}_{aft} = (w_1^{aft}, \ldots, w_{M_{out}}^{aft})$.

Our method predicts each target word $w_i^{in} \in \mathbf{w}_{in}$ using three kinds of information: $\mathbf{n}$, words around w_i^{in} in $\mathbf{w}_{in}$, and words in $\mathbf{w}_{bef}$ and $\mathbf{w}_{aft}$. Words are embedded in a d-dimensional vector space and we refer to these vectors as word embeddings. To discriminate between words in $\mathbf{n}$ from those in $\mathbf{w}_{in}$, $\mathbf{w}_{bef}$, and $\mathbf{w}_{aft}$, we have two sets of word embeddings: $\mathbf{N} \in \mathbb{R}^{d \times |\mathcal{N}|}$ and $\mathbf{W} \in \mathbb{R}^{d \times |\mathcal{W}|}$. $\mathcal{W}$ is a set of words and $\mathcal{N}$ is also a set of words but contains only nouns. Hence, the word *cause* has two embeddings: one in $\mathbf{N}$ and another in $\mathbf{W}$. In general *cause* is used as a noun and a verb, and thus we expect the noun embeddings to capture the meanings focusing on their noun usage. This is inspired by some recent work on word representations that explicitly assigns an independent representation for each word usage according to its part-of-speech tag (Baroni and Zamparelli, 2010; Grefenstette and Sadrzadeh, 2011; Hashimoto et al., 2013; Hashimoto et al., 2014; Kartsaklis and Sadrzadeh, 2013).

A feature vector $\mathbf{f} \in \mathbb{R}^{2d(2+c) \times 1}$ is constructed to predict w_i^{in} by concatenating word embeddings:

$$\mathbf{f} = [\mathbf{N}(n_1); \mathbf{N}(n_2); \mathbf{W}(w_{i-1}^{in}); \ldots; \mathbf{W}(w_{i-c}^{in});$$
$$\mathbf{W}(w_{i+1}^{in}); \ldots; \mathbf{W}(w_{i+c}^{in});$$
$$\frac{1}{M_{out}} \sum_{j=1}^{M_{out}} \mathbf{W}(w_j^{bef}); \frac{1}{M_{out}} \sum_{j=1}^{M_{out}} \mathbf{W}(w_j^{aft})] .$$

$$(1)$$

$\mathbf{N}(\cdot)$ and $\mathbf{W}(\cdot) \in \mathbb{R}^{d \times 1}$ corresponds to each word and c is the context size. A special *NULL* token is used if $i - j$ is smaller than 1 or $i + j$ is larger than M_{in} for each $j \in \{1, 2, \ldots, c\}$.

Our method then estimates a conditional probability $p(w|\mathbf{f})$ that the target word is a word w given the feature vector $\mathbf{f}$, using a logistic regression model:

$$p(w|\mathbf{f}) = \sigma(\tilde{\mathbf{W}}(w) \cdot \mathbf{f} + b(w)) , \qquad (2)$$

where $\tilde{\mathbf{W}}(w) \in \mathbb{R}^{2d(2+c) \times 1}$ is a weight vector for w, $b(w) \in \mathbb{R}$ is a bias for w, and $\sigma(x) = \frac{1}{1+e^{-x}}$ is the logistic function. Each column vector in $\tilde{\mathbf{W}} \in \mathbb{R}^{2d(c+1) \times |\mathcal{W}|}$ corresponds to a word. That is, we assign a logistic regression model for each word, and we can train the embeddings using the one-versus-rest approach to make $p(w_i^{in}|\mathbf{f})$ larger than $p(w'|\mathbf{f})$ for $w' \neq w_i^{in}$. However, naively optimizing the parameters of those logistic regression models would lead to prohibitive computational cost since it grows linearly with the size of the vocabulary.

When training we employ several procedures introduced by Mikolov et al. (2013b), namely, *negative sampling*, a modified unigram noise distribution and *subsampling*. For negative sampling the model parameters $\mathbf{N}$, $\mathbf{W}$, $\tilde{\mathbf{W}}$, and b are learned by maximizing the objective function $J_{unlabeled}$:

$$\sum_{\mathbf{n}} \sum_{i=1}^{M_{in}} \left(\log(p(w_i^{in}|\mathbf{f})) + \sum_{j=1}^{k} \log(1 - p(w_j'|\mathbf{f})) \right) ,$$

$$(3)$$

where w_j' is a word randomly drawn from the unigram noise distribution weighted by an exponent of 0.75. Maximizing $J_{unlabeled}$ means that our method can discriminate between each target word and k noise words given the target word's context. This approach is much less computationally expensive than the one-versus-rest approach and has proven effective in learning word embeddings.

To reduce redundancy during training we use subsampling. A training sample, whose target word is w, is discarded with the probability $P_d(w) = 1 - \sqrt{\frac{t}{p(w)}}$, where t is a threshold which is set to 10^{-5} and $p(w)$ is a probability corresponding to the frequency of w in the training corpus. The more frequent a target word is, the more likely it is to be discarded. To further emphasize infrequent words, we apply the subsampling approach not only to target words, but also to noun pairs; concretely, by drawing two random numbers r_1 and r_2, a training sample whose noun pair is (n_1, n_2) is discarded if $P_d(n_1)$ is larger than r_1 or $P_d(n_2)$ is larger than r_2.

Since the feature vector $\mathbf{f}$ is constructed as defined in Eq. (1), at each training step, $\tilde{\mathbf{W}}(w)$ is updated based on information about what pair of nouns surrounds w, what word n-grams appear in a small window around w, and what words appear outside the noun pair. Hence, the weight vector $\tilde{\mathbf{W}}(w)$ captures rich information regarding the target word w.

3.2 Constructing Feature Vectors

Once the word embeddings are trained, we can use them for relation classification. Given a noun pair $\mathbf{n} = (n_1, n_2)$ with its context words $\mathbf{w}_{in}$, $\mathbf{w}_{bef}$, and $\mathbf{w}_{aft}$, we construct a feature vector to classify the relation between n_1 and n_2 by concatenating three kinds of feature vectors:

$\mathbf{g}_{\mathrm{n}}$ the word embeddings of the noun pair,

$\mathbf{g}_{\mathrm{in}}$ the averaged n-gram embeddings between the pair, and

$\mathbf{g}_{\mathrm{out}}$ the concatenation of the averaged word embeddings in $\mathbf{w}_{bef}$ and $\mathbf{w}_{aft}$.

The feature vector $\mathbf{g}_{\mathrm{n}} \in \mathbb{R}^{2d \times 1}$ is the concatenation of $\mathbf{N}(n_1)$ and $\mathbf{N}(n_2)$:

$$\mathbf{g}_{\mathrm{n}} = [\mathbf{N}(n_1); \mathbf{N}(n_2)] . \qquad (4)$$

Words between the noun pair contribute to classifying the relation, and one of the most common ways to incorporate an arbitrary number of words is treating them as a bag of words. However, word order information is lost for bag-of-words features such as averaged word embeddings. To incorporate the word order information, we first define n-gram embeddings $\mathbf{h}_i \in \mathbb{R}^{4d(1+c) \times 1}$ between the

noun pair:

$$\begin{aligned} \mathbf{h}_i = [&\mathbf{W}(w_{i-1}^{in}); \dots ; \mathbf{W}(w_{i-c}^{in}); \\ &\mathbf{W}(w_{i+1}^{in}); \dots ; \mathbf{W}(w_{i+c}^{in}); \tilde{\mathbf{W}}(w_i^{in})] . \end{aligned} \qquad (5)$$

Note that $\tilde{\mathbf{W}}$ can also be used and that the value used for n is $(2c+1)$. As described in Section 3.1, $\tilde{\mathbf{W}}$ captures meaningful information about each word and after the first embedding learning step we can treat the embeddings in $\tilde{\mathbf{W}}$ as features for the words. Mnih and Kavukcuoglu (2013) have demonstrated that using embeddings like those in $\tilde{\mathbf{W}}$ is useful in representing the words. We then compute the feature vector $\mathbf{g}_{\mathrm{in}}$ by averaging $\mathbf{h}_i$:

$$\mathbf{g}_{\mathrm{in}} = \frac{1}{M_{in}} \sum_{i=1}^{M_{in}} \mathbf{h}_i . \qquad (6)$$

We use the averaging approach since M_{in} depends on each instance. The feature vector $\mathbf{g}_{\mathrm{in}}$ allows us to represent word sequences of arbitrary lengths as fixed-length feature vectors using the simple operations: concatenation and averaging.

The words before and after the noun pair are sometimes important in classifying the relation. For example, in the phrase "pour n_1 into n_2", the word *pour* should be helpful in classifying the relation. As with Eq. (1), we use the concatenation of the averaged word embeddings of words before and after the noun pair to compute the feature vector $\mathbf{g}_{\mathrm{out}} \in \mathbb{R}^{2d \times 1}$:

$$\mathbf{g}_{\mathrm{out}} = \frac{1}{M_{out}} [\sum_{j=1}^{M_{out}} \mathbf{W}(w_j^{bef}); \sum_{j=1}^{M_{out}} \mathbf{W}(w_j^{aft})] . \qquad (7)$$

As described above, the overall feature vector $\mathbf{e} \in \mathbb{R}^{4d(2+c) \times 1}$ is constructed by concatenating $\mathbf{g}_{\mathrm{n}}$, $\mathbf{g}_{\mathrm{in}}$, and $\mathbf{g}_{\mathrm{out}}$. We would like to emphasize that we only use simple operations: averaging and concatenating the learned word embeddings. The feature vector $\mathbf{e}$ is then used as input for a softmax classifier, without any complex transformation such as matrix multiplication with non-linear functions.

3.3 Supervised Learning

Given a relation classification task we train a softmax classifier using the feature vector $\mathbf{e}$ described in Section 3.2. For each k-th training sample with a corresponding label l_k among L predefined labels, we compute a conditional probability given

its feature vector $\mathbf{e}_k$:

$$p(l_k|\mathbf{e}_k) = \frac{\exp(\mathbf{o}(l_k))}{\sum_{i=1}^{L} \exp(\mathbf{o}(i))}, \qquad (8)$$

where $\mathbf{o} \in \mathbb{R}^{L \times 1}$ is defined as $\mathbf{o} = \mathbf{S}\mathbf{e}_k + \mathbf{s}$, and $\mathbf{S} \in \mathbb{R}^{L \times 4d(2+c)}$ and $\mathbf{s} \in \mathbb{R}^{L \times 1}$ are the softmax parameters. $\mathbf{o}(i)$ is the i-th element of $\mathbf{o}$. We then define the objective function as:

$$J_{labeled} = \sum_{k=1}^{K} \log(p(l_k|\mathbf{e}_k)) - \frac{\lambda}{2}\|\theta\|^2 . \qquad (9)$$

K is the number of training samples and λ controls the L-2 regularization. $\theta = (\mathbf{N}, \mathbf{W}, \tilde{\mathbf{W}}, \mathbf{S}, \mathbf{s})$ is the set of parameters and $J_{labeled}$ is maximized using AdaGrad (Duchi et al., 2011). We have found that *dropout* (Hinton et al., 2012) is helpful in preventing our model from overfitting. Concretely, elements in $\mathbf{e}$ are randomly omitted with a probability of 0.5 at each training step. Recently dropout has been applied to deep neural network models for natural language processing tasks and proven effective (Irsoy and Cardie, 2014; Paulus et al., 2014).

In what follows, we refer to the above method as **RelEmb**. While RelEmb uses only low-level features, a variety of useful features have been proposed for relation classification. Among them, we use dependency path features (Bunescu and Mooney, 2005) based on the untyped binary dependencies of the Stanford parser to find the shortest path between target nouns. The dependency path features are computed by averaging word embeddings from $\mathbf{W}$ on the shortest path, and are then concatenated to the feature vector $\mathbf{e}$. Furthermore, we directly incorporate semantic information using word-level semantic features from Named Entity (NE) tags and WordNet hypernyms, as used in previous work (Rink and Harabagiu, 2010; Socher et al., 2012; Yu et al., 2014). We refer to this extended method as **RelEmb**$_{\text{FULL}}$. Concretely, RelEmb$_{\text{FULL}}$ uses the same binary features as in Socher et al. (2012). The features come from NE tags and WordNet hypernym tags of target nouns provided by a sense tagger (Ciaramita and Altun, 2006).

4 Experimental Settings

4.1 Training Data

For pre-training we used a snapshot of the English Wikipedia[2] from November 2013. First,

we extracted 80 million sentences from the original Wikipedia file, and then used *Enju*[3] (Miyao and Tsujii, 2008) to automatically assign part-of-speech (POS) tags. From the POS tags we used *NN*, *NNS*, *NNP*, or *NNPS* to locate noun pairs in the corpus. We then collected training data by listing pairs of nouns and the words between, before, and after the noun pairs. A noun pair was omitted if the number of words between the pair was larger than 10 and we consequently collected 1.4 billion pairs of nouns and their contexts[4]. We used the 300,000 most frequent words and the 300,000 most frequent nouns and treated out-of-vocabulary words as a special *UNK* token.

4.2 Initialization and Optimization

We initialized the embedding matrices $\mathbf{N}$ and $\mathbf{W}$ with zero-mean gaussian noise with a variance of $\frac{1}{d}$. $\tilde{\mathbf{W}}$ and b were zero-initialized. The model parameters were optimized by maximizing the objective function in Eq. (3) using stochastic gradient ascent. The learning rate was set to α and linearly decreased to 0 during training, as described in Mikolov et al. (2013a). The hyperparameters are the embedding dimensionality d, the context size c, the number of negative samples k, the initial learning rate α, and M_{out}, the number of words outside the noun pairs. For hyperparameter tuning, we first fixed α to 0.025 and M_{out} to 5, and then set d to {50, 100, 300}, c to {1, 2, 3}, and k to {5, 15, 25}.

At the supervised learning step, we initialized $\mathbf{S}$ and $\mathbf{s}$ with zeros. The hyperparameters, the learning rate for AdaGrad, λ, M_{out}, and the number of iterations, were determined via 10-fold cross validation on the training set for each setting. Note that M_{out} can be tuned at the supervised learning step, adapting to a specific dataset.

5 Evaluation

5.1 Evaluation Dataset

We evaluated our method on the SemEval 2010 Task 8 data set[5] (Hendrickx et al., 2010), which involves predicting the semantic relations between

---

[3]Despite Enju being a syntactic parser we only use the POS tagger component. The accuracy of the POS tagger is about 97.2% on the WSJ corpus.

[4]The training data, the training code, and the learned model parameters used in this paper are publicly available at http://www.logos.t.u-tokyo.ac.jp/~hassy/publications/conll2015/

[5]http://docs.google.com/View?docid=dfvxd49s_36c28v9pmw.

noun pairs in their contexts. The dataset, containing 8,000 training and 2,717 test samples, defines nine classes (*Cause-Effect, Entity-Origin, etc.*) for ordered relations and one class (*Other*) for other relations. Thus, the task can be treated as a 19-class classification task. Two examples from the training set are shown below.

(a) Financial [stress]$_{E_1}$ is one of the main causes of [divorce]$_{E_2}$

(b) The [burst]$_{E_1}$ has been caused by water hammer [pressure]$_{E_2}$

Training example (a) is classified as *Cause-Effect(E_1, E_2)* which denotes that E_2 is an effect caused by E_1, while training example (b) is classified as *Cause-Effect(E_2, E_1)* which is the inverse of *Cause-Effect(E_1, E_2)*. We report the official macro-averaged F1 scores and accuracy.

5.2 Models

To empirically investigate the performance of our proposed method we compared it to several baselines and previously proposed models.

5.2.1 Random and word2vec Initialization

Rand-Init. The first baseline is RelEmb itself, but without applying the learning method on the unlabeled corpus. In other words, we train the softmax classifier from Section 3.3 on the labeled training data with randomly initialized model parameters.

W2V-Init. The second baseline is RelEmb using word embeddings learned by word2vec. More specifically, we initialize the embedding matrices $\mathbf{N}$ and $\mathbf{W}$ with the word2vec embeddings. Related to our method, word2vec has a set of weight vectors similar to $\tilde{\mathbf{W}}$ when trained with negative sampling and we use these weight vectors as a replacement for $\tilde{\mathbf{W}}$. We trained the word2vec embeddings using the *CBOW* model with subsampling on the full Wikipedia corpus. As with our experimental settings, we fix the learning rate to 0.025, and investigate several hyperparameter settings. For hyperparameter tuning we set the embedding dimensionality d to $\{50, 100, 300\}$, the context size c to $\{1, 3, 9\}$, and the number of negative samples k to $\{5, 15, 25\}$.

5.2.2 SVM-Based Systems

A simple approach to the relation classification task is to use SVMs with standard binary bag-of-words features. The bag-of-words features included the noun pairs and words between, before, and after the pairs, and we used LIBLINEAR[6] as our classifier.

5.2.3 Neural Network Models

Socher et al. (2012) used Recursive Neural Network (RNN) models to classify the relations. Subsequently, Ebrahimi and Dou (2015) and Hashimoto et al. (2013) proposed RNN models to better handle the relations. These methods rely on syntactic parse trees.

Yu et al. (2014) introduced their novel Factor-based Compositional Model (FCM) and presented results from several model variants, the best performing being FCM$_{\mathrm{EMB}}$ and FCM$_{\mathrm{FULL}}$. The former only uses word embedding information and the latter relies on dependency paths and NE features, in addition to word embeddings.

Zeng et al. (2014) used a Convolutional Neural Network (CNN) with WordNet hypernyms. Noteworthy in relation to the RNN-based methods, the CNN model does not rely on parse trees. More recently, dos Santos et al. (2015) have introduced CR-CNN by extending the CNN model and achieved the best result to date. The key point of CR-CNN is that it improves the classification score by omitting the noisy class "Other" in the dataset described in Section 5.1. We call CR-CNN using the "Other" class CR-CNN$_{\mathrm{Other}}$ and CR-CNN omitting the class CR-CNN$_{\mathrm{Best}}$.

5.3 Results and Discussion

The scores on the test set for SemEval 2010 Task 8 are shown in Table 1. RelEmb achieves 82.8% of F1 which is better than those of almost all models compared and comparable to that of the previous state of the art, except for CR-CNN$_{\mathrm{Best}}$. Note that RelEmb does not rely on external semantic features and syntactic parse features[7]. Furthermore, RelEmb$_{\mathrm{FULL}}$ achieves 83.5% of F1. We calculated a confidence interval $(82.0, 84.9)$ $(p < 0.05)$ using bootstrap resampling (Noreen, 1989).

5.3.1 Comparison with the Baselines

RelEmb significantly outperforms not only the Rand-Init baseline, but also the W2V-Init baseline.

---

[6]http://www.csie.ntu.edu.tw/~cjlin/liblinear/.

[7]While we use a POS tagger to locate noun pairs, RelEmb does not explicitly use POS features at the supervised learning step.

	Features for classifiers	F1 / ACC (%)
RelEmb$_{\text{FULL}}$	embeddings, dependency paths, WordNet, NE	83.5 / 79.9
RelEmb	embeddings	82.8 / 78.9
RelEmb (W2V-Init)	embeddings	81.8 / 77.7
RelEmb (Rand-Init)	embeddings	78.2 / 73.5
SVM	bag of words	76.5 / 72.0
SVM (Rink and Harabagiu, 2010)	bag of words, POS, dependency paths, WordNet, paraphrases, TextRunner, Google n-grams, etc.	82.2 / 77.9
CR-CNN$_{\text{Best}}$ (dos Santos et al., 2015)	embeddings, word position embeddings	84.1 / n/a
FCM$_{\text{FULL}}$ (Yu et al., 2014)	embeddings, dependency paths, NE	83.0 / n/a
CR-CNN$_{\text{Other}}$ (dos Santos et al., 2015)	embeddings, word position embeddings	82.7 / n/a
CRNN (Ebrahimi and Dou, 2015)	embeddings, parse trees, WordNet, NE, POS	82.7 / n/a
CNN (Zeng et al., 2014)	embeddings, WordNet	82.7 / n/a
MVRNN (Socher et al., 2012)	embeddings, parse trees, WordNet, NE, POS	82.4 / n/a
FCM$_{\text{EMB}}$ (Yu et al., 2014)	embeddings	80.6 / n/a
RNN (Hashimoto et al., 2013)	embeddings, parse trees, phrase categories, etc.	79.4 / n/a

Table 1: Scores on the test set for SemEval 2010 Task 8.

These results show that our task-specific word embeddings are more useful than those trained using window-based contexts. A point that we would like to emphasize is that the baselines are unexpectedly strong. As was noted by Wang and Manning (2012), we should carefully implement strong baselines and see whether complex models can outperform these baselines.

5.3.2 Comparison with SVM-Based Systems

RelEmb performs much better than the bag-of-words-based SVM. This is not surprising given that we use a large unannotated corpus and embeddings with a large number of parameters. RelEmb also outperforms the SVM system of Rink and Harabagiu (2010), which demonstrates the effectiveness of our task-specific word embeddings, despite our only requirement being a large unannotated corpus and a POS tagger.

5.3.3 Comparison with Neural Network Models

RelEmb outperforms the RNN models. In our preliminary experiments, we have found some undesirable parse trees when computing vector representations using RNN-based models and such parsing errors might hamper the performance of the RNN models.

FCM$_{\text{FULL}}$, which relies on dependency paths and NE features, achieves a better score than that of RElEmb. Without such features, RelEmb outperforms FCM$_{\text{EMB}}$ by a large margin. By incorporating external resources, RelEmb$_{\text{FULL}}$ outperforms FCM$_{\text{FULL}}$.

RelEmb compares favorably to CR-CNN$_{\text{Other}}$, despite our method being less computationally expensive than CR-CNN$_{\text{Other}}$. When classifying an instance, the number of the floating number multiplications is $4d(2 + c)L$ in our method since our method requires only one matrix-vector product for the softmax classifier as described in Section 3.3. c is the window size, d is the word embedding dimensionality, and L is the number of the classes. In CR-CNN$_{\text{Other}}$, the number is $(Dc(d + 2d')N + DL)$, where D is the dimensionality of the convolution layer, d' is the position embedding dimensionality, and N is the average length of the input sentences. Here, we omit the cost of the hyperbolic tangent function in CR-CNN$_{\text{Other}}$ for simplicity. Using the best hyperparameter settings, the number is roughly 3.8×10^4 in our method, and 1.6×10^7 in CR-CNN$_{\text{Other}}$ assuming N is 10. dos Santos et al. (2015) also boosted the score of CR-CNN$_{\text{Other}}$ by omitting the noisy class "Other" by a ranking-based classifier, and achieved the best score (CR-CNN$_{\text{Best}}$). Our results may also be improved by using the same technique, but the technique is dataset-dependent, so we did not incorporate the technique.

5.4 Analysis on Training Settings

We perform analysis of the training procedure focusing on RelEmb.

5.4.1 Effects of Tuning Hyperparameters

In Tables 2 and 3, we show how tuning the hyperparameters of our method and word2vec affects

c	d	$k=5$	$k=15$	$k=25$
1	50	80.5	81.0	80.9
	100	80.9	81.3	81.2
2	50	80.9	81.3	81.3
	100	81.3	81.6	81.7
3	50	81.0	81.0	81.5
	100	81.3	81.9	**82.2**
	300	-	-	82.0

Table 2: Cross-validation results for RelEmb.

c	d	$k=5$	$k=15$	$k=25$
1	50	80.5	80.7	80.9
	100	81.1	81.2	81.0
	300	81.2	**81.3**	81.2
3	50	80.4	80.7	80.8
	100	81.0	81.0	80.9
9	50	80.0	79.8	80.2
	100	80.3	80.4	80.1

Table 3: Cross-validation results for the W2V-Init.

the classification results using 10-fold cross validation on the training set. The same split is used for each setting, so all results are comparable to each other. The best settings for the cross validation are used to produce the results reported in Table 1.

Table 2 shows F1 scores obtained by RelEmb. The results for $d = 50, 100$ show that RelEmb benefits from relatively large context sizes. The n-gram embeddings in RelEmb capture richer information by setting c to 3 compared to setting c to 1. Relatively large numbers of negative samples also slightly boost the scores. As opposed to these trends, the score does not improve using $d = 300$. We use the best setting ($c = 3$, $d = 100$, $k = 25$) for the remaining analysis. We note that RelEmb$_{\text{FULL}}$ achieves an F1-score of 82.5.

We also performed similar experiments for the W2V-Init baseline, and the results are shown in Table 3. In this case, the number of negative samples does not affect the scores, and the best score is achieved by $c = 1$. As discussed in Bansal et al. (2014), the small context size captures the syntactic similarity between words rather than the topical similarity. This result indicates that syntactic similarity is more important than topical similarity for this task. Compared to the word2vec embeddings, our embeddings capture not only local context information using word order, but also long-

$\mathbf{g}_{\text{n}}$	$\mathbf{g}_{\text{in}}$	$\mathbf{g}'_{\text{in}}$	$\mathbf{g}_{\text{n}}, \mathbf{g}_{\text{in}}$	$\mathbf{g}_{\text{n}}, \mathbf{g}_{\text{in}}, \mathbf{g}_{\text{out}}$
61.8	70.2	68.2	81.1	82.2

Table 4: Cross-validation results for ablation tests.

Method	Score
RelEmb $\mathbf{N}$	0.690
RelEmb $\mathbf{W}$	0.599
W2V-Init	0.687

Table 5: Evaluation on the WordSim-353 dataset.

range co-occurrence information by being tailored for the specific task.

5.4.2 Ablation Tests

As described in Section 3.2, we concatenate three kinds of feature vectors, $\mathbf{g}_{\text{n}}$, $\mathbf{g}_{\text{in}}$, and $\mathbf{g}_{\text{out}}$, for supervised learning. Table 4 shows classification scores for ablation tests using 10-fold cross validation. We also provide a score using a simplified version of $\mathbf{g}_{\text{in}}$, where the feature vector $\mathbf{g}'_{\text{in}}$ is computed by averaging the word embeddings $[\mathbf{W}(w_i^{in}); \tilde{\mathbf{W}}(w_i^{in})]$ of the words between the noun pairs. This feature vector $\mathbf{g}'_{\text{in}}$ then serves as a bag-of-words feature.

Table 4 clearly shows that the averaged n-gram embeddings contribute the most to the semantic relation classification performance. The difference between the scores of $\mathbf{g}_{\text{in}}$ and $\mathbf{g}'_{\text{in}}$ shows the effectiveness of our averaged n-gram embeddings.

5.4.3 Effects of Dropout

At the supervised learning step we use dropout to regularize our model. Without dropout, our performance drops from 82.2% to 81.3% of F1 on the training set using 10-fold cross validation.

5.4.4 Performance on a Word Similarity Task

As described in Section 3.1, we have the noun-specific embeddings $\mathbf{N}$ as well as the standard word embeddings $\mathbf{W}$. We evaluated the learned embeddings using a word-level semantic evaluation task called *WordSim-353* (Finkelstein et al., 2001). This dataset consists of 353 pairs of nouns and each pair has an averaged human rating which corresponds to a semantic similarity score. Evaluation is performed by measuring Spearman's rank correlation between the human ratings and the cosine similarity scores of the embeddings. Table 5 shows the evaluation results. We used the best settings reported in Table 2 and 3 since our method

Cause-Effect(E$_1$,E$_2$)				Content-Container(E$_1$,E$_2$)				Message-Topic(E$_1$,E$_2$)			
resulted	poverty	caused	the	inside	was	inside	a	discuss	magazines	relating	to
caused	stability	caused	the	in	was	in	a	explaining	to	discuss	aspects
generated	coast	resulted	in	hidden	hidden	in	a	discussing	concerned	about	NULL
cause	fire	caused	due	was	was	inside	the	relating	interview	relates	to
causes	that	resulted	in	stored	was	hidden	in	describing	to	discuss	the

Cause-Effect(E$_2$,E$_1$)				Content-Container(E$_2$,E$_1$)				Message-Topic(E$_2$,E$_1$)			
after	caused	by	radiation	full	NULL	full	of	subject	were	related	in
from	caused	by	infection	included	was	full	of	related	was	related	in
caused	stomach	caused	by	contains	a	full	NULL	discussed	been	discussed	in
triggered	caused	by	genetic	contained	a	full	and	documented	is	related	through
due	anger	caused	by	stored	a	full	forty	received	the	subject	of

Table 6: Top five unigrams and trigrams with the highest scores for six classes.

is designed for relation classification and it is not clear how to tune the hyperparameters for the word similarity task. As shown in the result table, the noun-specific embeddings perform better than the standard embeddings in our method, which indicates the noun-specific embeddings capture more useful information in measuring the semantic similarity between nouns. The performance of the noun-specific embeddings is roughly the same as that of the word2vec embeddings.

5.5 Qualitative Analysis on the Embeddings

Using the n-gram embeddings $\mathbf{h}_i$ in Eq. (5), we inspect which n-grams are relevant to each relation class after the supervised learning step of RelEmb. When the context size c is 3, we can use at most 7-grams. The learned weight matrix $\mathbf{S}$ in Section 3.3 is used to detect the most relevant n-grams for each class. More specifically, for each n-gram embedding $(n = 1, 3)$ in the training set, we compute the dot product between the n-gram embedding and the corresponding components in $\mathbf{S}$. We then select the pairs of n-grams and class labels with the highest scores. In Table 6 we show the top five n-grams for six classes. These results clearly show that the n-gram embeddings capture salient syntactic patterns which are useful for the relation classification task.

6 Conclusions and Future Work

We have presented a method for learning word embeddings specifically designed for relation classification. The word embeddings are trained using large unlabeled corpora to capture lexical features for relation classification. On a well-established semantic relation classification task our method significantly outperforms the baseline based on word2vec. Our method also compares favorably to previous state-of-the-art models that rely on syntactic parsers and external semantic resources, despite our method requiring only access to an unannotated corpus and a POS tagger. For future work, we will investigate how well our method performs on other domains and datasets and how relation labels can help when learning embeddings in a semi-supervised learning setting.

References

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring Continuous Word Representations for Dependency Parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 809–815.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are Vectors, Adjectives are Matrices: Representing Adjective-Noun Constructions in Semantic Space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193.

Emanuela Boros, Romaric Besançon, Olivier Ferret, and Brigitte Grau. 2014. Event Role Extraction using Domain-Relevant Word Representations. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1852–1857.

Razvan Bunescu and Raymond Mooney. 2005. A Shortest Path Dependency Kernel for Relation Extraction. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 724–731.

Wenliang Chen, Yue Zhang, and Min Zhang. 2014. Feature Embedding for Dependency Parsing. In

Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pages 816–826.

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-Coverage Sense Disambiguation and Information Extraction with a Supersense Sequence Tagger. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 594–602.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Cicero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying Relations by Ranking with Convolutional Neural Networks. In *Proceedings of the Joint Conference of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*. to appear.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Javid Ebrahimi and Dejing Dou. 2015. Chain Based RNN for Relation Classification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1244–1249.

Lev Finkelstein, Gabrilovich Evgenly, Matias Yossi, Rivlin Ehud, Solan Zach, Wolfman Gadi, and Ruppin Eytan. 2001. Placing Search in Context: The Concept Revisited. In *Proceedings of the Tenth International World Wide Web Conference*.

Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret. 2007. SemEval-2007 Task 04: Classification of Semantic Relations between Nominals. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 13–18.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental Support for a Categorical Compositional Distributional Model of Meaning. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1394–1404.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Revisiting Embedding Features for Simple Semi-supervised Learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 110–120.

Kazuma Hashimoto, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Simple Customization of Recursive Neural Networks for Semantic Relation Classification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1372–1376.

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. 2014. Jointly Learning Word Representations and Composition Functions Using Predicate-Argument Structures. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1544–1555.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations between Pairs of Nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 33–38.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

Ozan Irsoy and Claire Cardie. 2014. Deep Recursive Neural Networks for Compositionality in Language. In *Advances in Neural Information Processing Systems 27*, pages 2096–2104.

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. 2013. Prior Disambiguation of Word Tensors for Constructing Sentence Vectors. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1590–1601.

Quoc Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, ICML '14, pages 1188–1196.

Omer Levy and Yoav Goldberg. 2014. Neural Word Embedding as Implicit Matrix Factorization. In *Advances in Neural Information Processing Systems 27*, pages 2177–2185.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of Workshop at the International Conference on Learning Representations*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Yusuke Miyao and Jun'ichi Tsujii. 2008. Feature Forest Models for Probabilistic HPSG Parsing. *Computational Linguistics*, 34(1):35–80, March.

277

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems 26*, pages 2265–2273.

Thien Huu Nguyen and Ralph Grishman. 2014. Employing Word Representations and Regularization for Domain Adaptation of Relation Extraction. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 68–74.

Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses: An Introduction*. Wiley-Interscience.

Romain Paulus, Richard Socher, and Christopher D Manning. 2014. Global Belief Recursive Neural Networks. In *Advances in Neural Information Processing Systems 27*, pages 2888–2896.

Bryan Rink and Sanda Harabagiu. 2010. UTD: Classifying Semantic Relations by Combining Lexical and Semantic Resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 256–259.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic Compositionality through Recursive Matrix-Vector Spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1555–1565.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word Representations: A Simple and General Method for Semi-Supervised Learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394.

Sida Wang and Christopher Manning. 2012. Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 90–94.

Mo Yu, Matthew R. Gormley, and Mark Dredze. 2014. Factor-based Compositional Embedding Models. In *Proceedings of Workshop on Learning Semantics at the 2014 Conference on Neural Information Processing Systems*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation Classification via Convolutional Deep Neural Network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344.

Min Zhang, Jie Zhang, Jian Su, and GuoDong Zhou. 2006. A Composite Kernel to Extract Relations between Entities with Both Flat and Structured Features. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 825–832.