Classifying Relations via Long Short Term Memory Networks along Shortest Dependency Paths

Yan Xu,† Lili Mou,† Ge Li,† Yunchuan Chen, Hao Peng,† Zhi Jin†∗
†Software Institute, Peking University, 100871, P. R. China
{xuyan14,lige,zhijin}@sei.pku.edu.cn, {doublepower.mou,penghao.pku}@gmail.com
‡University of Chinese Academy of Sciences, chenyunchuan11@mails.ucas.ac.cn

Abstract

Relation classification is an important research arena in the field of natural language processing (NLP). In this paper, we present SDP-LSTM, a novel neural network to classify the relation of two entities in a sentence. Our neural architecture leverages the shortest dependency path (SDP) between two entities; multichannel recurrent neural networks, with long short term memory (LSTM) units, pick up heterogeneous information along the SDP. Our proposed model has several distinct features: (1) The shortest dependency paths retain most relevant information (to relation classification), while eliminating irrelevant words in the sentence. (2) The multichannel LSTM networks allow effective information integration from heterogeneous sources over the dependency paths. (3) A customized dropout strategy regularizes the neural network to alleviate overfitting. We test our model on the SemEval 2010 relation classification task, and achieve an $F_1$-score of 83.7%, higher than competing methods in the literature.

1 Introduction

Relation classification is an important NLP task. It plays a key role in various scenarios, e.g., information extraction (Wu and Weld, 2010), question answering (Yao and Van Durme, 2014), medical informatics (Wang and Fan, 2014), ontolgy learning (Xu et al., 2014), etc. The aim of relation classification is to categorize into predefined classes the relations between pairs of marked entities in given texts. For instance, in the sentence “A trillion gallons of [water]$_{e_1}$ have been poured into an empty [region]$_{e_2}$ of outer space,” the entities water and region are of relation $\text{Entity-Destination} (e_1, e_2)$.

Traditional relation classification approaches rely largely on feature representation (Kambhatla, 2004), or kernel design (Zelenko et al., 2003; Buneșcu and Mooney, 2005). The former method usually incorporates a large set of features; it is difficult to improve the model performance if the feature set is not very well chosen. The latter approach, on the other hand, depends largely on the designed kernel, which summarizes all data information. Deep neural networks, emerging recently, provide a way of highly automatic feature learning (Bengio et al., 2013), and have exhibited considerable potential (Zeng et al., 2014; Santos et al., 2015). However, human engineering—that is, incorporating human knowledge to the network’s architecture—is still important and beneficial.

This paper proposes a new neural network, SDP-LSTM, for relation classification. Our model utilizes the shortest dependency path (SDP) between two entities in a sentence; we also design a long short term memory (LSTM)-based recurrent neural network for information processing. The neural architecture is mainly inspired by the following observations.

- Shortest dependency paths are informative (Fundel et al., 2007; Chen et al., 2014). To determine the two entities’ relation, we find it mostly sufficient to use only the words along the SDP: they concentrate on most relevant information while diminishing less relevant noise. Figure 1 depicts the dependency parse tree of the aforementioned sentence. Words along the SDP form a trimmed phrase (gallons of water poured into region) of the original sentence, which conveys much information about the target relation. Other words, such as a, trillion, outer space, are less informative and may bring noise if not dealt with properly.
Direction matters. Dependency trees are a kind of directed graph. The dependency relation between \textit{into} and \textit{region} is \texttt{PREP}; such relation hardly makes any sense if the directed edge is reversed. Moreover, the entities’ relation distinguishes its directionality, that is, $r(a, b)$ differs from $r(b, a)$, for a same given relation $r$ and two entities $a, b$. Therefore, we think it necessary to let the neural model process information in a direction-sensitive manner. Out of this consideration, we separate an SDP into two sub-paths, each from an entity to the common ancestor node. The extracted features along the two sub-paths are concatenated to make final classification.

Linguistic information helps. For example, with prior knowledge of hyponymy, we know “water is a kind of substance.” This is a hint that the entities, \textit{water} and \textit{region}, are more of Entity-Destination relation than, say, Communication-Topic. To gather heterogeneous information along SDP, we design a multichannel recurrent neural network. It makes use of information from various sources, including words themselves, POS tags, WordNet hypernyms, and the grammatical relations between governing words and their children.

For effective information propagation and integration, our model leverages LSTM units during recurrent propagation. We also customize a new dropout strategy for our SDP-LSTM network to alleviate the problem of overfitting. To the best of our knowledge, we are the first to use LSTM-based recurrent neural networks for the relation classification task.

We evaluate our proposed method on the SemEval 2010 relation classification task, and achieve an $F_1$-score of 83.7%, higher than competing methods in the literature.

In the rest of this paper, we review related work in Section 2. In Section 3, we describe our SDP-LSTM model in detail. Section 4 presents quantitative experimental results. Finally, we have our conclusion in Section 5.

2 Related Work

Relation classification is a widely studied task in the NLP community. Various existing methods mainly fall into three classes: feature-based, kernel-based, and neural network-based.

In feature-based approaches, different sets of features are extracted and fed to a chosen classifier (e.g., logistic regression). Generally, three types of features are often used. Lexical features concentrate on the entities of interest, e.g., entities \textit{per se}, entity POS, entity neighboring information. Syntactic features include chunking, parse trees, etc. Semantic features are exemplified by the concept hierarchy, entity class, entity mention. Kambhatla (2004) uses a maximum entropy model to combine these features for relation classification. However, different sets of handcrafted features are largely complementary to each other (e.g., hypernyms versus named-entity tags), and thus it is hard to improve performance in this way (Zhou et al., 2005).

Kernel-based approaches specify some measure of similarity between two data samples, without explicit feature representation. Zelenko et al. (2003) compute the similarity of two trees by utilizing their common subtrees. Bunescu and Mooney (2005) propose a shortest path dependency kernel for relation classification. Its main idea is that the relation strongly relies on the dependency path between two given entities. Wang (2008) provides a systematic analysis of several kernels and show that relation extraction can bene-
fit from combining convolution kernel and syntactic features. Plank and Moschitti (2013) introduce semantic information into kernel methods in addition to considering structural information only. One potential difficulty of kernel methods is that all data information is completely summarized by the kernel function (similarity measure), and thus designing an effective kernel becomes crucial.

Deep neural networks, emerging recently, can learn underlying features automatically, and have attracted growing interest in the literature. Socher et al. (2011) propose a recursive neural network (RNN) along sentences’ parse trees for sentiment analysis; such model can also be used to classify relations (Socher et al., 2012). Hashimoto et al. (2013) explicitly weight phrases’ importance in RNNs to improve performance. Ebrahimi and Dou (2015) rebuild an RNN on the dependency path between two marked entities. Zeng et al. (2014) explore convolutional neural networks, by which they utilize sequential information of sentences. Santos et al. (2015) also use the convolutional network; besides, they propose a ranking loss function with data cleaning, and achieve the state-of-the-art result in SemEval-2010 Task 8.

In addition to the above studies, which mainly focus on relation classification approaches and models, other related research trends include information extraction from Web documents in a semi-supervised manner (Bunescu and Mooney, 2007; Banko et al., 2007), dealing with small datasets without enough labels by distant supervision techniques (Mintz et al., 2009), etc.

3 The Proposed SDP-LSTM Model

In this section, we describe our SDP-LSTM model in detail. Subsection 3.1 delineates the overall architecture of our model. Subsection 3.2 presents the rationale of using SDPs. Four different information channels along the SDP are explained in Subsection 3.3. Subsection 3.4 introduces the recurrent neural network with long short term memory (LSTM) units for effective information propagation. A max pooling layer thereafter gathers information from LSTM nodes in each path. The pooling layers from different channels are concatenated, and then connected to a hidden layer. Finally, we have a softmax output layer for classification. (See again Figure 2b.)

3.2 The Shortest Dependency Path

The dependency parse tree is naturally suitable for relation classification because it focuses on the action and agents in a sentence (Socher et al., 2014). Moreover, the shortest path between entities, as discussed in Section 1, condenses most illuminating information for entities’ relation.

We also observe that the sub-paths, separated by the common ancestor node of two entities, provide strong hints for the relation’s directionality. Take Figure 1 as an example. Two entities water and region have their common ancestor node, poured, which separates the SDP into two parts:

\[ \text{[water]}_{e_1} \rightarrow \text{of} \rightarrow \text{gallons} \rightarrow \text{poured} \]

and

\[ \text{poured} \leftarrow \text{into} \leftarrow \text{[region]}_{e_2} \]

The first sub-path captures information of \( e_1 \), whereas the second sub-path is mainly about \( e_2 \). By examining the two sub-paths separately, we know \( e_1 \) and \( e_2 \) are of relation Entity-Destination(\( e_1, e_2 \)), rather than Entity-Destination(\( e_2, e_1 \)).

Following the above intuition, we design two recurrent neural networks, which propagate

---

1. http://nlp.stanford.edu/software/lex-parser.shtml
bottom-up from the entities to their common ancestor. In this way, our model is direction-sensitive.

3.3 Channels

We make use of four types of information along the SDP for relation classification. We call them channels as these information sources do not interact during recurrent propagation. Detailed channel descriptions are as follows.

- **Word representations.** Each word in a given sentence is mapped to a real-valued vector by looking up in a word embedding table. Unsupervisedly trained on a large corpus, word embeddings are thought to be able to well capture words' syntactic and semantic information (Mikolov et al., 2013b).

- **Part-of-speech tags.** Since word embeddings are obtained on a generic corpus of a large scale, the information they contain may not agree with a specific sentence. We deal with this problem by allying each input word with its POS tag, e.g., noun, verb, etc. In our experiment, we only take into use a coarse-grained POS category, containing 15 different tags.

- **Grammatical relations.** The dependency relations between a governing word and its children makes a difference in meaning. A same word pair may have different dependency relation types. For example, "beats _nsubj_ it" is distinct from "beats _dobj_ it." Thus, it is necessary to capture such grammatical relations in SDPs. In our experiment, grammatical relations are grouped into 19 classes, mainly based on a coarse-grained classification (De Marneffe et al., 2006).

- **WordNet hypernyms.** As illustrated in Section 1, hyponymy information is also useful for relation classification. (Details are not repeated here.) To leverage WordNet hypernyms, we use a tool developed by Ciaramita and Altun (2006). The tool assigns a hypernym to each word, from 41 predefined concepts in WordNet, e.g., noun.food, verb.motion, etc. Given its hypernym, each word gains a more abstract concept, which helps to build a linkage between different but conceptual similar words.

As we can see, POS tags, grammatical relations, and WordNet hypernyms are also discrete (like words per se). However, no prevailing embedding learning method exists for POS tags, say. Hence, we randomly initialize their embeddings, and tune them in a supervised fashion during training. We notice that these information sources contain much fewer symbols, 15, 19, and 41, than the vocabulary size (greater than 25,000). Hence, we believe our strategy of random initialization is feasible, because they can be adequately tuned during supervised training.

3.4 Recurrent Neural Network with Long Short Term Memory Units

The recurrent neural network is suitable for modeling sequential data by nature, as it keeps a hid-
input and recurrent connections, respectively. The hidden state \( h_t \), which changes with input data at each step accordingly. We use the recurrent network to gather information along each sub-path in the SDP (Figure 2).

The hidden state \( h_t \), for the \( t \)-th word in the sub-path, is a function of its previous state \( h_{t-1} \) and the current word \( x_t \). Traditional recurrent networks have a basic interaction, that is, the input is linearly transformed by a weight matrix and non-linearly squashed by an activation function. Formally, we have

\[
h_t = f(W_{in}x_t + W_{rec}h_{t-1} + b_h)
\]

where \( W_{in} \) and \( W_{rec} \) are weight matrices for the input and recurrent connections, respectively. \( b_h \) is a bias term for the hidden state vector, and \( f_h \) a non-linear activation function (e.g., tanh).

One problem of the above model is known as gradient vanishing or exploding. The training of neural networks requires gradient back-propagation. If the propagation sequence (path) is too long, the gradient may probably either grow, or decay, exponentially, depending on the magnitude of \( W_{rec} \). This leads to the difficulty of training.

Long short term memory (LSTM) units are proposed in Hochreiter (1998) to overcome this problem. The main idea is to introduce an adaptive gating mechanism, which decides the degree to which LSTM units keep the previous state and memorize the extracted features of the current data input. Many LSTM variants have been proposed in the literature. We adopt in our method a variant introduced by Zaremba and Sutskever (2014), also used in Zhu et al. (2014).

Concretely, the LSTM-based recurrent neural network comprises four components: an input gate \( i_t \), a forget gate \( f_t \), an output gate \( o_t \), and a memory cell \( c_t \) (depicted in Figure 3 and formalized through Equations 1-6 as below).

The three adaptive gates \( i_t \), \( f_t \), and \( o_t \) depend on the previous state \( h_{t-1} \) and the current input \( x_t \) (Equations 1-3). An extracted feature vector \( g_t \) is also computed, by Equation 4, serving as the candidate memory cell.

\[
i_t = \sigma(W_i \cdot x_t + U_i \cdot h_{t-1} + b_i) \quad (1)
\]
\[
f_t = \sigma(W_f \cdot x_t + U_f \cdot h_{t-1} + b_f) \quad (2)
\]
\[
o_t = \sigma(W_o \cdot x_t + U_o \cdot h_{t-1} + b_o) \quad (3)
\]
\[
g_t = \tanh(W_g \cdot x_t + U_g \cdot h_{t-1} + b_g) \quad (4)
\]

The current memory cell \( c_t \) is a combination of the previous cell content \( c_{t-1} \) and the candidate content \( g_t \), weighted by the input gate \( i_t \) and forget gate \( f_t \), respectively. (See Equation 5 below.)

\[
c_t = i_t \odot g_t + f_t \odot c_{t-1} \quad (5)
\]

The output of LSTM units is the recurrent network’s hidden state, which is computed by Equation 6 as follows.

\[
h_t = o_t \odot \tanh(c_t) \quad (6)
\]

In the above equations, \( \sigma \) denotes a sigmoid function; \( \odot \) denotes element-wise multiplication.

### 3.5 Dropout Strategies

A good regularization approach is needed to alleviate overfitting. Dropout, proposed recently by Hinton et al. (2012), has been very successful on feed-forward networks. By randomly omitting feature detectors from the network during training, it can obtain less interdependent network units and achieve better performance. However, the conventional dropout does not work well with recurrent neural networks with LSTM units, since dropout may hurt the valuable memorization ability of memory units.

As there is no consensus on how to drop out LSTM units in the literature, we try several dropout strategies for our SDP-LSTM network:

- Dropout embeddings;
- Dropout inner cells in memory units, including \( i_t, g_t, o_t, c_t \), and \( h_t \); and
• Dropout the penultimate layer.

As we shall see in Section 4.2, dropping out LSTM units turns out to be inimical to our model, whereas the other two strategies boost in performance.

The following equations formalize the dropout operations on the embedding layers, where $D$ denotes the dropout operator. Each dimension in the embedding vector, $x_t$, is set to zero with a predefined dropout rate.

$$i_t = \sigma(W_i \cdot D(x_t) + U_i \cdot h_{t-1} + b_i) \quad (7)$$

$$f_t = \sigma(W_f \cdot D(x_t) + U_f \cdot h_{t-1} + b_f) \quad (8)$$

$$o_t = \sigma(W_o \cdot D(x_t) + U_o \cdot h_{t-1} + b_o) \quad (9)$$

$$g_t = \tanh \left( W_g \cdot D(x_t) + U_g \cdot h_{t-1} + b_g \right) \quad (10)$$

### 3.6 Training Objective

The SDP-LSTM described above propagates information along a sub-path from an entity to the common ancestor node (of the two entities). A max pooling layer packs, for each sub-path, the recurrent network’s states, $h$’s, to a fixed vector by taking the maximum value in each dimension.

Such architecture applies to all channels, namely, words, POS tags, grammatical relations, and WordNet hypernyms. The pooling vectors in these channels are concatenated, and fed to a fully connected hidden layer. Finally, we add a softmax output layer for classification. The training objective is the penalized cross-entropy error, given by

$$J = - \sum_{t=1}^{n_c} t_i \log y_i + \lambda \left( \sum_{i=1}^{n_w} \| W_i \|_F^2 + \sum_{i=1}^{n_U} \| U_i \|_F^2 \right)$$

where $t \in \mathbb{R}^{n_c}$ is the one-hot represented ground truth and $y \in \mathbb{R}^{n_c}$ is the estimated probability for each class by softmax. ($n_c$ is the number of target classes.) $\| \cdot \|_F$ denotes the Frobenius norm of a matrix; $\omega$ and $\nu$ are the numbers of weight matrices (for $W$’s and $U$’s, respectively). $\lambda$ is a hyper-parameter that specifies the magnitude of penalty on weights. Note that we do not add $\ell_2$ penalty to bias parameters.

We pretrained word embeddings by *word2vec* (Mikolov et al., 2013a) on the English Wikipedia corpus; other parameters are initialized randomly. We apply stochastic gradient descent (with mini-batch 10) for optimization; gradients are computed by standard back-propagation. Training details are further introduced in Section 4.2.

### 4 Experiments

In this section, we present our experiments in detail. Our implementation is built upon [Mou et al. (2015)]. Section 4.1 introduces the dataset; Section 4.2 describes hyperparameter settings. In Section 4.3, we compare SDP-LSTM’s performance with other methods in the literature. We also analyze the effect of different channels in Section 4.4.

#### 4.1 Dataset

The SemEval-2010 Task 8 dataset is a widely used benchmark for relation classification (Hendrickx et al., 2010). The dataset contains 8,000 sentences for training, and 2,717 for testing. We split 1/10 samples out of the training set for validation.

The target contains 19 labels: 9 directed relations, and an undirected Other class. The directed relations are list as below.

- Cause-Effect
- Component-Whole
- Content-Container
- Entity-Destination
- Entity-Origin
- Message-Topic
- Member-Collection
- Instrument-Agency
- Product-Producer

In the following are illustrated two sample sentences with directed relations.

[People]$_{e_1}$ have been moving back into [downtown]$_{e_2}$.

Financial [stress]$_{e_1}$ is one of the main causes of [divorce]$_{e_2}$.

The target labels are Entity-Destination ($e_1, e_2$), and Cause-Effect($e_1, e_2$), respectively.

The dataset also contains an undirected Other class. Hence, there are 19 target labels in total. The undirected Other class takes in entities that do not fit into the above categories, illustrated by the following example.

A misty [ridge]$_{e_1}$ uprises from the [surge]$_{e_2}$.

We use the official macro-averaged $F_1$-score to evaluate model performance. This official measurement excludes the Other relation. Nonetheless, we have no special treatment of Other class in our experiments, which is typical in other studies.
4.2 Hyperparameters and Training Details

This subsection presents hyperparameter tuning for our model. We set word-embeddings to be 200-dimensional; POS, WordNet hyponymy, and grammatical relation embeddings are 50-dimensional. Each channel of the LSTM network contains the same number of units as its source embeddings (either 200 or 50). The penultimate hidden layer is 100-dimensional. As it is not feasible to perform full grid search for all hyperparameters, the above values are chosen empirically.

We add $\ell_2$ penalty for weights with coefficient $10^{-5}$, which was chosen by validation from the set \{10$^{-2}$, 10$^{-3}$, \ldots, 10$^{-7}$\}.

We thereafter validate the proposed dropout strategies in Section 3.5. Since network units in different channels do not interact with each other during information propagation, we herein take one channel of LSTM networks to assess the efficacy. Taking the word channel as an example, we first drop out word embeddings. Then with a fixed dropout rate of word embeddings, we test the effect of dropping out LSTM inner cells and the penultimate units, respectively.

We find that, dropout of LSTM units hurts the model, even if the dropout rate is small, 0.1, say (Figure 3b). Dropout of embeddings improves model performance by 2.16% (Figure 4a); dropout of the penultimate layer further improves by 0.16% (Figure 4c). This analysis also provides, for other studies, some clues for dropout in LSTM networks.

4.3 Results

Table 4 compares our SDT-LSTM with other state-of-the-art methods. The first entry in the table presents the highest performance achieved by traditional feature engineering. [Hendrickx et al. (2010)] leverage a variety of handcrafted features, and use SVM for classification; they achieve an $F_1$-score of 82.2%.

Neural networks are first used in this task in [Socher et al. (2012)]. They build a recursive neural network (RNN) along a constituency tree for relation classification. They extend the basic RNN with matrix-vector interaction and achieve an $F_1$-score of 82.4%.

[Zeng et al. (2014)] treat a sentence as sequential data and exploit the convolutional neural network (CNN); they also integrate word position information into their model. [Santos et al. (2015)] design a model called CR-CNN; they propose a ranking-based cost function and elaborately diminish the impact of the Other class, which is not counted in the official $F_1$-measure. In this way, they achieve the state-of-the-art result with the $F_1$-score of 84.1%. Without such special treatment, their $F_1$-score is 82.7%.

[Yu et al. (2014)] propose a Feature-rich Compositional Embedding Model (FCM) for relation classification, which combines unlexicalized linguistic contexts and word embeddings. They achieve an $F_1$-score of 83.0%.

Our proposed SDT-LSTM model yields an $F_1$-score of 83.7%. It outperforms existing competing approaches, in a fair condition of softmax with cross-entropy error.

It is worth to note that we have also conducted two controlled experiments: (1) Traditional RNN without LSTM units, achieving an $F_1$-score of 82.8%; (2) LSTM network over the entire dependency path (instead of two sub-paths), achieving

![Figure 4: $F_1$-scores versus dropout rates. We first evaluate the effect of dropout embeddings (a). Then the dropout of the inner cells (b) and the penultimate layer (c) is tested with word embeddings being dropped out by 0.5.](image-url)
Table 1: Comparison of relation classification systems. The “†” remark refers to special treatment for the Other class.

| Classifier | Feature set | $F_1$ |
|------------|-------------|-------|
| SVM        | POS, WordNet, prefixes and other morphological features, dependency parse, Levin classes, PropBank, FanmeNet, NomLex-Plus, Google n-gram, paraphrases, TextRunner | 82.2 |
| RNN        | Word embeddings | 74.8 |
|            | Word embeddings, POS, NER, WordNet | 77.6 |
| MVRNN      | Word embeddings | 79.1 |
|            | Word embeddings, POS, NER, WordNet | 82.4 |
| CNN        | Word embeddings | 69.7 |
|            | Word embeddings, word position embeddings, WordNet | 82.7 |
| Chain CNN  | Word embeddings, POS, NER, WordNet | 82.7 |
| FCM        | Word embeddings | 80.6 |
|            | Word embeddings, dependency parsing, NER | 83.0 |
| CR-CNN     | Word embeddings | 82.8† |
|            | Word embeddings, position embeddings | 82.7 |
|            | Word embeddings, position embeddings | 84.1† |
| SDP-LSTM   | Word embeddings | 82.4 |
|            | Word embeddings, POS embeddings, WordNet embeddings, grammar relation embeddings | 83.7 |

Table 2: Effect of different channels.

| Channels | $F_1$ |
|----------|-------|
| Word embeddings | 82.35 |
| + POS embeddings (only) | 82.98 |
| + GR embeddings (only) | 83.21 |
| + WordNet embeddings (only) | 83.03 |
| + POS + GR + WordNet embeddings | 83.70 |

4.4 Effect of Different Channels

This subsection analyzes how different channels affect our model. We first used word embeddings only as a baseline; then we added POS tags, grammatical relations, and WordNet hypernyms, respectively; we also combined all these channels into our models. Note that we did not try the latter three channels alone, because each single of them (e.g., POS) does not carry much information.

We see from Table 2 that word embeddings alone in SDP-LSTM yield a remarkable performance of 82.35%, compared with CNNs 69.7%, RNNs 74.9–79.1%, and FCM 80.6%.

Adding either grammatical relations or WordNet hypernyms outperforms other existing methods (data cleaning not considered here). POS tagging is comparatively less informative, but still boosts the $F_1$-score by 0.63%.

We notice that, the boosts are not simply added when channels are combined. This suggests that these information sources are complementary to each other in some linguistic aspects. Nonetheless, incorporating all four channels further pushes the $F_1$-score to 83.70%.

5 Conclusion

In this paper, we propose a novel neural network model, named SDP-LSTM, for relation classification. It learns features for relation classification iteratively along the shortest dependency path. Several types of information (word themselves, POS tags, grammatical relations and WordNet hypernyms) along the path are used. Meanwhile, we leverage LSTM units for long-range information propagation and integration. We demonstrate the effectiveness of SDP-LSTM by evaluating the model on SemEval-2010 relation classification task, outperforming existing state-of-art methods (in a fair condition without data cleaning). Our result sheds some light in the relation classification task as follows.

- The shortest dependency path can be a valuable resource for relation classification, covering mostly sufficient information of target
Classifying relation is a challenging task due to the inherent ambiguity of natural languages and the diversity of sentence expression. Thus, integrating heterogeneous linguistic knowledge is beneficial to the task.

Treating the shortest dependency path as two sub-paths, mapping two different neural networks, helps to capture the directionality of relations.

LSTM units are effective in feature detection and propagation along the shortest dependency path.

Acknowledgments
This research is supported by the National Basic Research Program of China (the 973 Program) under Grant No. 2015CB352201 and the National Natural Science Foundation of China under Grant Nos. 61232015 and 91318301.

References

[Banko et al.2007] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction for the web. In IJCAI, volume 7, pages 2670–2676.

[Bengio et al.2013] Y. Bengio, A. Courville, and P. Vincent. 2013. Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8):1798–1828.

[Bunescu and Mooney2005] R. C. Bunescu and R. J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing, pages 724–731. Association for Computational Linguistics.

[Bunescu and Mooney2007] R. Bunescu and R. Mooney. 2007. Learning to extract relations from the web using minimal supervision. In Annual meeting-association for Computational Linguistics, volume 45, page 576.

[Chen et al.2014] Yun-Nung Chen, Dilek Hakkani-Tur, and Gokan Tur. 2014. Deriving local relational surface forms from dependency-based entity embeddings for unsupervised spoken language understanding. In Spoken Language Technology Workshop (SLT), 2014 IEEE, pages 242–247. IEEE.

[Ciaramita and Altun2006] M. Ciaramita and Y. 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. Association for Computational Linguistics.

[De Marneffe et al.2006] M. C. De Marneffe, B. MacCartney, and C. D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In Proceedings of LREC, volume 6, pages 449–454.

[Ebrahimi and Dou2015] J. Ebrahimi and D. Dou. 2015. Chain based rnn for relation classification. In HLT-NAACL.

[Fundel et al.2007] K. Fundel, R. Küffner, and R. Zimmer. 2007. RelEx—Relation extraction using dependency parse trees. Bioinformatics, 23(3):365–371.

[Hashimoto et al.2013] K. Hashimoto, M. Miwa, Y. Tsuruoka, and T. Chikayama. 2013. Simple customization of recursive neural networks for semantic relation classification. In EMNLP, pages 1372–1376.

[Hendrickx et al.2010] I. Hendrickx, S. N. Kim, and Z. et al. Kozareva. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions. Association for Computational Linguistics.

[Hinton et al.2012] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580.

[Hochreiter1998] S. Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6(02):107–116.

[Kambhatla2004] N. Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In Proceedings of the ACL 2004 on Interactive poster and demonstration sessions, page 22. Association for Computational Linguistics.

[Mikolov et al.2013a] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. 2013a. Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems, pages 3111–3119.

[Mikolov et al.2013b] T. Mikolov, W. T. Yih, and G. Zweig. 2013b. Linguistic regularities in continuous space word representations. In HLT-NAACL.

[Mintz et al.2009] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In Proceedings of the Joint Conference of the 47th Annual Meeting of
the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2, pages 1003–1011. Association for Computational Linguistics.

[Mou et al.2015] L. Mou, H. Peng, G. Li, Y. Xu, L. Zhang, and Z. Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. arXiv preprint arXiv:1504.01106.

[Plank and Moschitti2013] B. Plank and A. Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In ACL (1), pages 1498–1507.

[Santos et al.2015] C. N. d. Santos, B. Xiang, and B. Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics.

[Socher et al.2011] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 151–161. Association for Computational Linguistics.

[Socher et al.2012] R. Socher, B. Huval, C. D. Manning, and A. 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. Association for Computational Linguistics.

[Socher et al.2014] R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. Transactions of the Association for Computational Linguistics, 2:207–218.

[Wang and Fan2014] C. Wang and J. Fan. 2014. Medical relation extraction with manifold models. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 828–838, Baltimore, Maryland, June. Association for Computational Linguistics.

[Wang2008] M. Wang. 2008. A re-examination of dependency path kernels for relation extraction. In IJCNLP, pages 841–846.

[Wu and Weld2010] F. Wu and D. S. Weld. 2010. Open information extraction using wikipedia. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 118–127. Association for Computational Linguistics.

[Xu et al.2014] Y. Xu, G. Li, L. Mou, and Y. Lu. 2014. Learning non-taxonomic relations on demand for ontology extension. International Journal of Software Engineering and Knowledge Engineering, 24(08):1159–1175.