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), ontology 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 `Entity-Destination`(e_1, e_2).

Traditional relation classification approaches rely largely on feature representation (Kambhatla, 2004), or kernel design (Zelenko et al., 2003; Bunescu 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; dos 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.

*Corresponding authors.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1785–1794,
Lisbon, Portugal, 17-21 September 2015.

- Direction matters. Dependency trees are a kind of directed graph. The dependency relation between *into* and *region* is `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, *water* and *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.

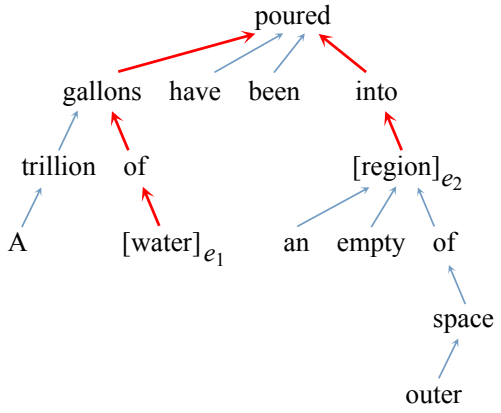


Figure 1: The dependency parse tree corresponding to the sentence "A trillion gallons of water have been poured into an empty region of outer space." Red lines indicate the shortest dependency path between entities *water* and *region*. An edge $a \rightarrow b$ refers to a being governed by b. Dependency types are labeled by the parser, but not presented in the figure for clarity.

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 *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 (GuoDong 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. dos 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, which is built upon the dependency path. Subsection 3.5 customizes a dropout strategy for our network to alleviate overfitting. We finally present our training objective in Subsection 3.6.

3.1 Overview

Figure 2 depicts the overall architecture of our SDP-LSTM network.

First, a sentence is parsed to a dependency tree by the Stanford parser;[1] the shortest dependency path (SDP) is extracted as the input of our network. Along the SDP, four different types of information—referred to as *channels*—are used, including the words, POS tags, grammatical relations, and WordNet hypernyms. (See Figure 2a.) In each channel, discrete inputs, e.g., words, are mapped to real-valued vectors, called *embeddings*, which capture the underlying meanings of the inputs.

Two recurrent neural networks (Figure 2b) pick up information along the left and right sub-paths of the SDP, respecitvely. (The path is separated by the common ancestor node of two entities.) Long short term memory (LSTM) units are used in the recurrent networks 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 $\mathrm{softmax}$ output layer for classification. (See again Figure 2a.)

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 $\texttt{Entity-Destination}(e_1, e_2)$, rather than $\texttt{Entity-Destination}(e_2, e_1)$.

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

[1] http://nlp.stanford.edu/software/lex-parser.shtml

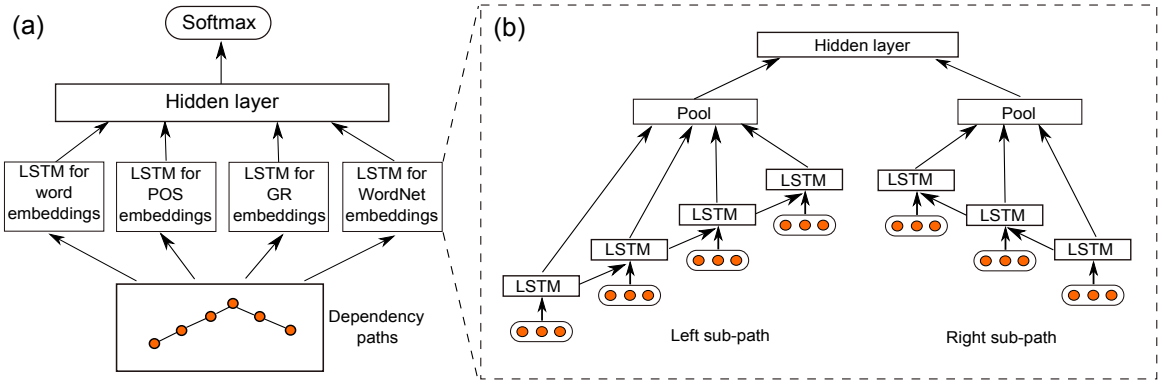


Figure 2: (a) The overall architecture of SDP-LSTM. (b) One channel of the recurrent neural networks built upon the shortest dependency path. The channels are words, part-of-speech (POS) tags, grammatical relations (abbreviated as *GR* in the figure), and WordNet hypernyms.

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* $\xrightarrow{\text{nsubj}}$ *it*" is distinct from "*beats* $\xrightarrow{\text{dobj}}$ *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).[2] 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-

[2] http://sourceforge.net/projects/supersensetag

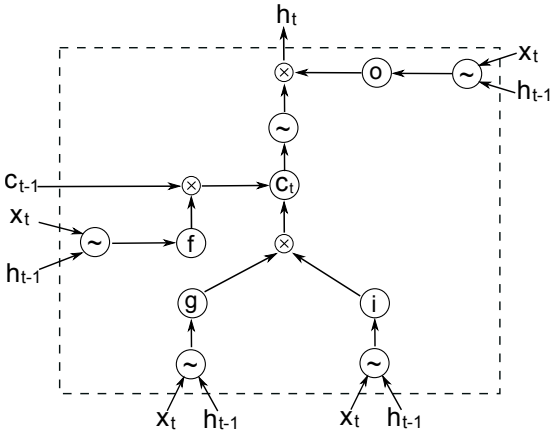


Figure 3: A long short term memory unit. h: hidden unit. c: memory cell. i: input gate. f: forget gate. o: output gate. g: candidate cell. $\otimes$: element-wise multiplication. $\sim$: activation function.

den state vector $\boldsymbol{h}$, 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 2b).

The hidden state $\boldsymbol{h}_t$, for the t-th word in the sub-path, is a function of its previous state $\boldsymbol{h}_{t-1}$ and the current word $\boldsymbol{x}_t$. Traditional recurrent networks have a basic interaction, that is, the input is linearly transformed by a weight matrix and nonlinearly squashed by an activation function. Formally, we have

$$\boldsymbol{h}_t = f(W_{in}\boldsymbol{x}_t + W_{rec}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h)$$

where W_{in} and W_{rec} are weight matrices for the input and recurrent connections, respectively. $\boldsymbol{b}_h$ 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. (2015).

Concretely, the LSTM-based recurrent neural network comprises four components: an input gate $\boldsymbol{i}_t$, a forget gate $\boldsymbol{f}_t$, an output gate $\boldsymbol{o}_t$, and a memory cell $\boldsymbol{c}_t$ (depicted in Figure 3 and formalized through Equations 1–6 as bellow).

The three adaptive gates $\boldsymbol{i}_t$, $\boldsymbol{f}_t$, and $\boldsymbol{o}_t$ depend on the previous state $\boldsymbol{h}_{t-1}$ and the current input $\boldsymbol{x}_t$ (Equations 1–3). An extracted feature vector $\boldsymbol{g}_t$ is also computed, by Equation 4, serving as the candidate memory cell.

$$\boldsymbol{i}_t = \sigma(W_i{\cdot}\boldsymbol{x}_t + U_i{\cdot}\boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \quad (1)$$
$$\boldsymbol{f}_t = \sigma(W_f{\cdot}\boldsymbol{x}_t + U_f{\cdot}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \quad (2)$$
$$\boldsymbol{o}_t = \sigma(W_o{\cdot}\boldsymbol{x}_t + U_o{\cdot}\boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \quad (3)$$
$$\boldsymbol{g}_t = \tanh(W_g{\cdot}\boldsymbol{x}_t + U_g{\cdot}\boldsymbol{h}_{t-1} + \boldsymbol{b}_g) \quad (4)$$

The current memory cell $\boldsymbol{c}_t$ is a combination of the previous cell content $\boldsymbol{c}_{t-1}$ and the candidate content $\boldsymbol{g}_t$, weighted by the input gate $\boldsymbol{i}_t$ and forget gate $\boldsymbol{f}_t$, respectively. (See Equation 5 below.)

$$\boldsymbol{c}_t = \boldsymbol{i}_t \otimes \boldsymbol{g}_t + \boldsymbol{f}_t \otimes \boldsymbol{c}_{t-1} \quad (5)$$

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

$$\boldsymbol{h}_t = \boldsymbol{o}_t \otimes \tanh(\boldsymbol{c}_t) \quad (6)$$

In the above equations, σ denotes a $\mathrm{sigmoid}$ function; $\otimes$ 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, $\boldsymbol{x}_t$, is set to zero with a predefined dropout rate.

$$\boldsymbol{i}_t = \sigma(W_i \cdot D(\boldsymbol{x}_t) + U_i \cdot \boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \quad (7)$$

$$\boldsymbol{f}_t = \sigma(W_f \cdot D(\boldsymbol{x}_t) + U_f \cdot \boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \quad (8)$$

$$\boldsymbol{o}_t = \sigma(W_o \cdot D(\boldsymbol{x}_t) + U_o \cdot \boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \quad (9)$$

$$\boldsymbol{g}_t = \tanh\Big(W_g \cdot D(\boldsymbol{x}_t) + U_g \cdot \boldsymbol{h}_{t-1} + \boldsymbol{b}_g\Big) \quad (10)$$

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, $\boldsymbol{h}$'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 $\mathrm{softmax}$ output layer for classification. The training objective is the penalized cross-entropy error, given by

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

where $\boldsymbol{t} \in \mathbb{R}^{n_c}$ is the one-hot represented ground truth and $\boldsymbol{y} \in \mathbb{R}^{n_c}$ is the estimated probability for each class by $\mathrm{softmax}$. (n_c is the number of target classes.) $\|\cdot\|_F$ denotes the Frobenius norm of a matrix; ω and υ are the numbers of weight matrices (for W's and U's, respectively). λ is a hyperparameter that specifies the magnitude of penalty on weights. Note that we do not add ℓ_2 penalty to biase 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., 2009). 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]$_{e_2}$.

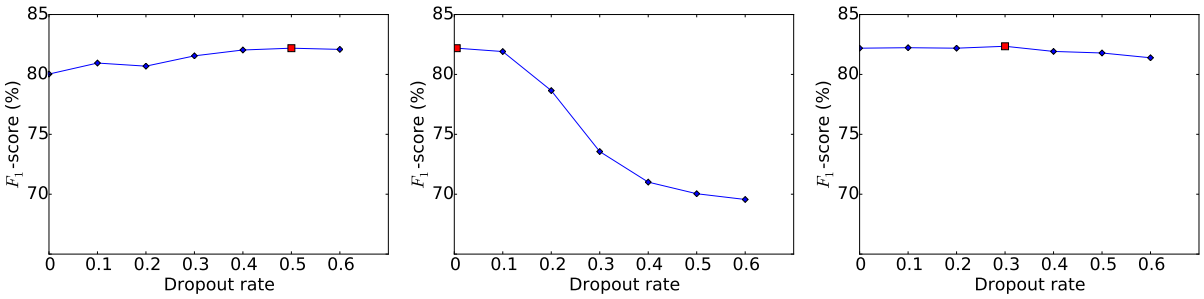
> Financial [stress]$_{e_1}$ is one of the main causes of [divorce]$_{e_2}$.

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.



(a) Dropout word embeddings (b) Dropout inner cells of memory units (c) Dropout the penultimate layer

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.

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 ℓ_2 penalty for weights with coefficient 10^{-5}, which was chosen by validation from the set $\{10^{-2}, 10^{-3}, \cdots, 10^{-7}\}$.

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 4b). 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. (2009) 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. dos 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 an F_1-score of 82.2%. These results demonstrate

Classifier	Feature set	F_1
SVM	POS, WordNet, prefixes and other morphological features, depdency 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, depedency 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 1: Comparison of relation classification systems. The "†" remark refers to special treatment for the `Other` class.

the effectiveness of LSTM and directionality in relation classification.

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%.

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

Table 2: Effect of different channels.

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

relations.

- 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

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction for the web. In *Proceedings of twentieth International Joint Conference on Artificial Intelligence*, volume 7, pages 2670–2676.

Yoshua Bengio, Aaron Courville, and Pierre Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Razvan C. Bunescu and Raymond 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.

Razvan Bunescu and Raymond Mooney. 2007. Learning to extract relations from the web using minimal supervision. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, volume 45, pages 576–583.

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.

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.

Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the International Conference on Language Resources and Evaluation*, volume 6, pages 449–454.

Cıcero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of 53rd Annual Meeting of the Association for Computational Linguistics*, pages 626–634.

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.

Katrin Fundel, Robert Küffner, and Ralf Zimmer. 2007. Relexłrelation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371.

Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 427–434.

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.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. 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)*, pages 94–99.

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

Sepp 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.

Nanda 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.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

Mike Mintz, Steven Bills, Rion Snow, and Dan 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.

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. *arXiv preprint arXiv:1504.01106*.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1498–1507.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher 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.

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.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218.

Chang Wang and James 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)*, volume 1, pages 828–838.

Mengqiu Wang. 2008. A re-examination of dependency path kernels for relation extraction. In *Proceedings of the Third International Joint Conference on Natural Language Processing*, pages 841–846.

Fei Wu and Daniel S. Weld. 2010. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127.

Yan Xu, Ge Li, Lili Mou, and Yangyang Lu. 2014. Learning non-taxonomic relations on demand for ontology extension. *International Journal of Software Engineering and Knowledge Engineering*, 24(08):1159–1175.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 956–966.

Mo Yu, Matthew Gormley, and Mark Dredze. 2014. Factor-based compositional embedding models. In *NIPS Workshop on Learning Semantics*.

Wojciech Zaremba and Ilya Sutskever. 2014. Learning to execute. *arXiv preprint arXiv:1410.4615*.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *The Journal of Machine Learning Research*, 3:1083–1106.

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.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over tree structures. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1604–1612.