Improved Relation Classification by Deep Recurrent Neural Networks with Data Augmentation

Yan Xu,[1,*,‡] **Ran Jia,**[1,*] **Lili Mou,**[1] **Ge Li,**[1,†] **Yunchuan Chen,**[2] **Yangyang Lu,**[1] **Zhi Jin**[1,†]
[1]Key Laboratory of High Confidence Software Technologies (Peking University),
Ministry of Education, China; Institute of Software, Peking University
{xuyan14,lige,luyy11,zhijin}@sei.pku.edu.cn
{jiaran1994,doublepower.mou}gmail.com
[2]University of Chinese Academy of Sciences chenyunchuan11@mails.ucas.ac.cn

Abstract

Nowadays, neural networks play an important role in the task of relation classification. By designing different neural architectures, researchers have improved the performance to a large extent in comparison with traditional methods. However, existing neural networks for relation classification are usually of shallow architectures (e.g., one-layer convolutional neural networks or recurrent networks). They may fail to explore the potential representation space in different abstraction levels. In this paper, we propose deep recurrent neural networks (DRNNs) for relation classification to tackle this challenge. Further, we propose a data augmentation method by leveraging the directionality of relations. We evaluated our DRNNs on the SemEval-2010 Task 8, and achieve an F_1-score of 86.1%, outperforming previous state-of-the-art recorded results.[1]

1 Introduction

Classifying relations between two entities in a given context is an important task in natural language processing (NLP). Take the following sentence as an example: "Jewelry and other smaller [valuables]$_{e_1}$ were locked in a [safe]$_{e_2}$ or a closet with a deadbolt." The marked entities *valuables* and *safe* are of relation Content-Container(e_1, e_2). Relation classification plays a key role in various NLP applications, and has become a hot research topic in recent years.

Nowadays, neural network-based approaches have made significant improvement in relation classification, compared with traditional methods based on either human-designed features (Kambhatla, 2004; Hendrickx et al., 2009) or kernels (Bunescu and Mooney, 2005; Plank and Moschitti, 2013). For example, Zeng et al. (2014) and Xu et al. (2015a) utilize convolutional neural networks (CNNs) for relation classification. Xu et al. (2015b) apply long short term memory (LSTM)-based recurrent neural networks (RNNs) along the shortest dependency path. Nguyen and Grishman (2015) build ensembles of gated recurrent unit (GRU)-based RNNs and CNNs.

We have noticed that these neural models are typically designed in shallow architectures, e.g., one layer of CNN or RNN, whereas evidence in the deep learning community suggests that deep architectures are more capable of information integration and abstraction (Graves et al., 2013; Hermans and Schrauwen, 2013; Irsoy and Cardie, 2014). A natural question is then whether such deep architectures are beneficial to the relation classification task.

In this paper, we propose the deep recurrent neural networks (DRNNs) to classify relations. The deep RNNs can explore the representation space in different levels of abstraction and granularity. By visualizing how RNN units are related to the ultimate classification, we demonstrate that different layers indeed learn different representations: low-level layers enable sufficient information mix, while high-level layers are more capable of precisely locating the information relevant to the target relation between

[*]Equal contribution. [†]Corresponding authors. [‡]Yan Xu is currently a research scientist at Inveno Co., Ltd.
[1]Code released on https://sites.google.com/site/drnnre/

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1461–1470, Osaka, Japan, December 11-17 2016.

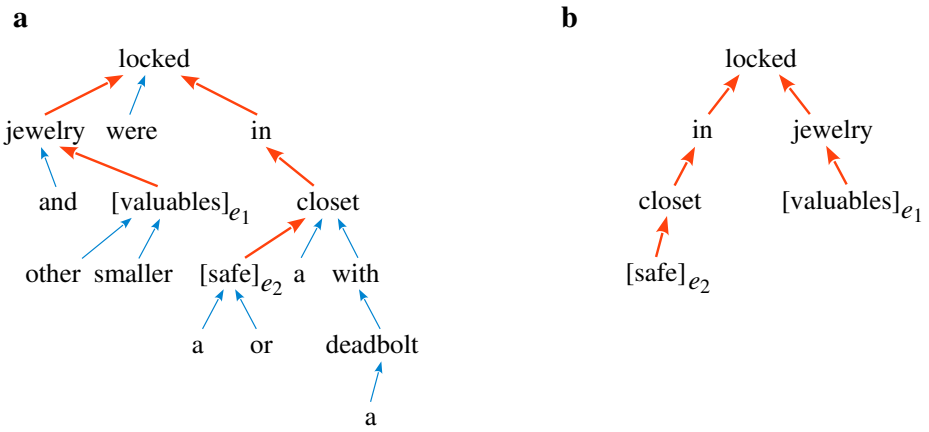


Figure 1: (a) The dependency parse tree corresponding to the sentence "Jewelry and other smaller [valuables]$_{e_1}$ were locked in a [safe]$_{e_2}$ or a closet with a deadbolt." Red arrows indicate the shortest dependency path between e_1 and e_2. (b) The augmented data sample.

two entities. Following our previous work (Xu et al., 2015b), we leverage the shortest dependency path (SDP, Figure 1) as the backbone of our RNNs.

We further observe that the relationship between two entities are directed. Two sub-paths, separated by entities' common ancestor, can be mapped to `subject-predicate` and `object-predicate` components of a relation. By changing the order of these two sub-paths, we obtain a new data sample with the inversed relationship (Figure 1b). Such data augmentation technique can provide additional data samples without using external data resources.

We evaluated our proposed method on the SemEval-2010 relation classification task. Even if we do not apply data augmentation, the DRNNs model has achieved a high performance of 84.2% F_1-score with a depth of 3, but the performance decreases when the depth is too large. This is because the deep RNN is a large model, which necessitates more data samples for training. Applying data augmentation can alleviate the problem of data sparseness and sustain a deeper RNN to improve the performance to 86.1%. The results show that both our deep networks and the data augmentation strategy have contributed to the relation classification task, and that they are coupled well together for further performance improvement.

The rest of this paper is organized as follows. Section 2 reviews related work; Section 3 describes our DRNNs model in detail. Section 4 presents in-depth experimental results. Finally, we have conclusion in Section 5.

2 Related Work

Traditional methods for relation classification mainly fall into two groups: feature-based or kernel-based. The former approaches extract different types of features and feed them into a classifier, e.g., a maximum entropy model (Kambhatla, 2004). Various features, including lexical, syntactic, as well as semantic ones, are shown to be useful to relation classification (Hendrickx et al., 2009). By contrast, kernel-based methods do not have explicit feature representations, but require predefined similarity measure of two data samples. Bunescu and Mooney (2005) design a kernel along the shortest dependency path (SDP) between two entities by observing that the relation strongly relies on SDPs. Plank and Moschitti (2013) combine structural information and semantic information in a tree kernel.

Neural networks have now become a prevailing technique in this task. Socher et al. (2011) design a recursive neural network along the constituency parse tree. Hashimoto et al. (2013), also on the basis of recursive networks, emphasize more on important phrases; Ebrahimi and Dou (2015) restrict recursive networks to SDP. In our previous study (Xu et al., 2015b), we introduce SDP-based recurrent neural network to classify relations.

Zeng et al. (2014), on the other hand, apply CNNs to relation classification. Along this line, dos Santos et al. (2015) replace the common softmax loss function with a ranking loss in their CNN model. Xu et

al. (2015a) design a negative sampling method for SDP-based CNNs.

Besides, representative hybrid models of CNNs and recursive/recurrent networks include Liu et al. (2015) and Nguyen and Grishman (2015).

3 The Proposed Methodology

In this section, we describe our methodology in detail. Subsection 3.1 provides an overall picture of our DRNNs model. Subsections 3.2 and 3.3 describe deep recurrent neural networks. The proposed data augmentation technique is introduced in Subsection 3.4. Finally, we present our training objective in Subsection 3.5.

3.1 Overview

Figure 2 depicts the overall architecture of the DRNNs model. Given a sentence and its dependency parse tree,[1] we follow our previous work (Xu et al., 2015b) and build DRNNs on the shortest dependency path (SDP), which serves as a backbone. In particular, an RNN picks up information along each sub-path, separated by the common ancestor of marked entities. Also, we take advantage of four information channels, namely, word embeddings, POS embeddings, grammatical relation embeddings, and WordNet embeddings.

Different from Xu et al. (2015b), we design deep RNNs with up to four hidden layers so as to capture information in different levels of abstraction. For each RNN layer, max pooling gathers information from different recurrent nodes. Notice that the four channels (with eight sub-paths) are processed in a similar way. Then all pooling layers are concatenated and fed into a hidden layer for information integration. Finally, we have a softmax output layer for classification.

3.2 Recurrent Neural Networks on Shortest Dependency Path

In this subsection, we introduce a single layer of RNN based on SDP, serving as a building block of our deep architecture.

Compared with a raw word sequence or a whole parse tree, the shortest dependency path (SDP) between two entities has two main advantages. First, it reduces irrelevant information; second, grammatical relations between words focus on the action and agents in a sentence and are naturally suitable for relation classification. Existing studies have demonstrated the effectiveness of SDP (Ebrahimi and Dou, 2015; Liu et al., 2015; Xu et al., 2015b; Xu et al., 2015a); details are not repeated here.

Focused on the SDP, an RNN keeps a hidden state vector $\boldsymbol{h}$, changing with the input word at each step accordingly. Concretely, the hidden state $\boldsymbol{h}_t$, for the t-th word in the sub-path, depends on its previous state $\boldsymbol{h}_{t-1}$ and the current word's embedding $\boldsymbol{x}_t$. For the simplicity and without loss of generality, we use vanilla recurrent networks with perceptron-like interaction, that is, the input is linearly transformed by a weight matrix and non-linearly squashed by an activation function, i.e.,

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

where W_{in} and W_{rec} are weight matrices for the input and recurrent connections, respectively. $\boldsymbol{b}_h$ is a bias term, and f is a non-linear activation function (ReLU in our experiment).

3.3 Deep Recurrent Neural Networks

Although an RNN, as described above, is suitable for picking information along a sequence (a subpath in our task) by its iterative nature, the machine learning community suggests that deep architectures may be more capable of information integration, and can capture different levels of abstraction.

A single-layer RNN can be viewed that it is deep along *time steps*. When unfolded, however, the RNN has only one hidden layer to capture the current input, as well as to retain the information in its previous step. In this sense, single-layer RNNs are actually shallow in information processing (Hermans and Schrauwen, 2013; Irsoy and Cardie, 2014).

[1] Parsed by the Stanford parser (de Marneffe et al., 2006).

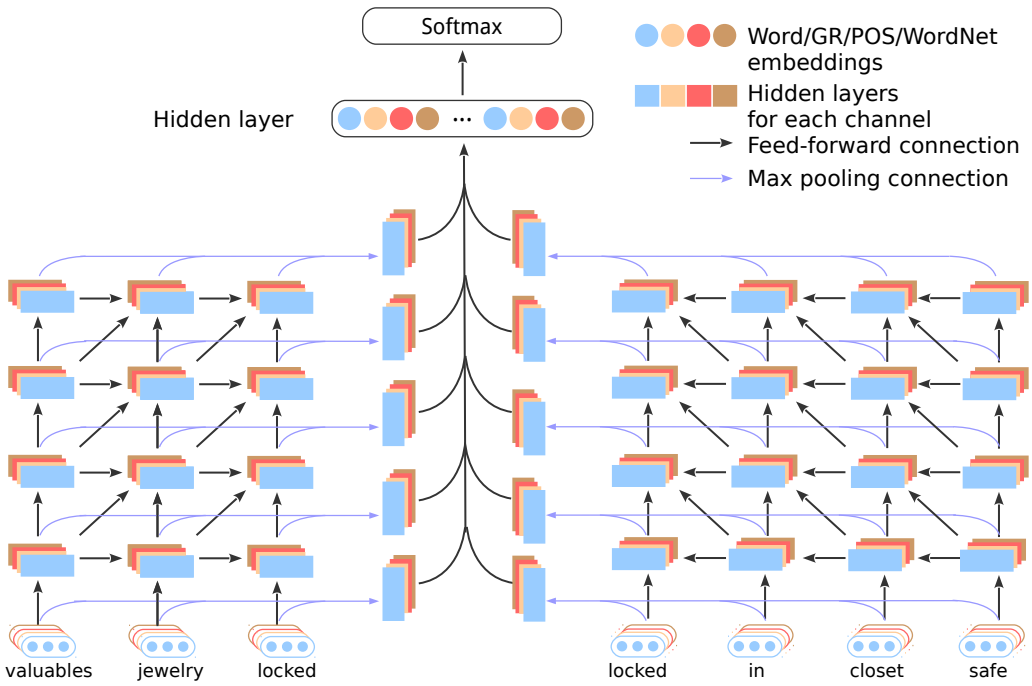


Figure 2: The overall architecture of DRNNs. Two recurrent neural networks pick up information along the shortest dependency path, separated by its common ancestor. We use four information channels, namely words, part-of-speech tags, grammatical relations (GR), and WordNet hypernyms.

In the relation classification task, words along SDPs provide information from different perspectives. On the one hand, the marked entities themselves are informative. On the other hand, the entities' common ancestor (typically verbs) tells how the two entities are related to each other. Such heterogeneous information might necessitate more complex machinery than a single RNN layer.

Following such intuition, we investigate deep RNNs by stacking multiple hidden layers on the top of one another, that is, every layer treats its previous layer as input, and computes its activation similar to Equation 1. Formally, we have

$$\boldsymbol{h}_t^{(i)} = f(W_{\text{in}}^{(i-1)}\boldsymbol{h}_t^{(i-1)} + W_{\text{rec}}^{(i)}\boldsymbol{h}_{t-1}^{(i)} + W_{\text{cross}}^{(i-1)}\boldsymbol{h}_{t-1}^{(i-1)} + \boldsymbol{b}^{(i)}) \tag{2}$$

where the subscripts refer to time steps, and superscripts indicate the layer number. To enhance information propagation, we add a "cross" connection for hidden layers ($i \geq 2$) from the lower layer in the previous time step, given by $W_{\text{cross}}^{(i-1)}\boldsymbol{h}_{t-1}^{(i-1)}$ in Equation 2. (See also $\nearrow$ and $\nwarrow$ arrows in Figure 2).

3.4 Data Augmentation

Neural networks, especially deep ones, are likely to be prone to overfitting. The SemEval-2010 relation classification dataset, we use, comprises only several thousand samples, which may not fully sustain the training of deep RNNs.

To mitigate this problem, we propose a data augmentation technique for relation classification by making use of the directionality of relationships.

The two sub-paths

$[\text{valuables}]_{e_1} \rightarrow \text{jewelry} \rightarrow \text{locked}$

$\text{locked} \leftarrow \text{in} \leftarrow \text{closet} \leftarrow [\text{safe}]_{e_2}$

in Figure 1, for example, can be mapped to the `subject-predicate` and `object- predicate` components in the relation `Content-Container`(e_1, e_2). If we change the order of these two sub-paths, we obtain

$[\text{safe}]_{e_1} \rightarrow \text{closet} \rightarrow \text{in} \rightarrow \text{locked}$
$\text{locked} \leftarrow \text{jewelry} \leftarrow [\text{valuables}]_{e_2}$

Then the relationship becomes `Container-Content`(e_1, e_2), which is exactly the inverse of `Content-Container`(e_1, e_2). In this way, we can augment the dataset without using additional resources.

3.5 Training Objective

For each recurrent layer and embedding layer (over each sub-path for each channel), we apply a max pooling layer to gather information. In total, we have 40 pools, which are concatenated and fed to a hidden layer for information integration.

Finally, a softmax layer outputs the estimated probability that two sub-paths (s^{left} and s^{right}) are of relation r. For a single data sample i, we apply the standard cross-entropy loss, denoted as $J(s_i^{\text{left}}, s_i^{\text{right}}, r_i)$. With the data augmentation technique, our overall training objective is

$$J = \sum_{i=1}^{m} J(s_i^{\text{left}}, s_i^{\text{right}}, r_i) + J(s_i^{\text{right}}, s_i^{\text{left}}, r_i^{-1}) + \lambda \sum_{i=1}^{\omega} \|W_i\|_F$$

where r^{-1} refers to the inverse of relation r. m is the number of data samples in the original training set. ω is the number of weight matrices in DRNNs. λ is a regularization coefficient, and $\|\cdot\|_F$ denotes Frobenius norm of a matrix.

For decoding (predicting the relation of an unseen sample), the data augmentation technique provides new opportunities, because we can use the probability of $r(e_1, e_2)$, $r^{-1}(e_2, e_1)$, or both. Section 4.3 provides detailed discussion.

4 Experiments

In this section, we present our experiments in detail. Subsection 4.1 introduces the dataset; Subsection 4.2 describes hyperparameter settings. We discuss the details of data augmentation in Subsection 4.3 and the rationale for using RNNs in Subsection 4.4. Subsection 4.5 compares our DRNNs model with other methods in the literature. In Subsection 4.6, we have quantitative and qualitative analysis of how the depth affects our model.

4.1 Dataset

We evaluated our DRNNs model on the SemEval-2010 Task 8 dataset, which is an established benchmark for relation classification (Hendrickx et al., 2009). The dataset contains 8000 sentences for training, and 2717 for testing. We split 800 samples out of the training set for validation.

There are 9 directed relations and an undirected default relation `Other`; thus, we have 19 different labels in total. However, the `Other` class is not taken into consideration when we compute the official measures.

4.2 Hyperparameter Settings

This subsection presents hyperparameters of our proposed model. We basically followed the settings in our previous work (Xu et al., 2015b). Word embeddings were 200-dimensional, pretrained ourselves using `word2vec` (Mikolov et al., 2013) on the Wikipedia corpus; embeddings in other channels were 50-dimensional initialized randomly. The hidden layers in each channel had the same number of units as their embeddings (either 200 or 50); the penultimate hidden layer was 100-dimensional. An ℓ_2 penalty of 10^{-5} was also applied as in Xu et al. (2015b), but we chose the dropout rate by validation with a granularity of 5% for our model variants (with different depths).

We also chose the depth of DRNNs by validation from the set $\{1, 2, \cdots, 6\}$. The 3-layer and 4-layer DRNNs yield the highest performance with and without data augmentation, respectively. Section 4.6 provides both quantitative and qualitative analysis regarding the effect of depth.

We applied mini-batched stochastic gradient descent for optimization, where gradients were computed by standard back-propagation.

Variant of Data augmentation	F_1
No Augmentation	84.16
Augment all relations	83.43
Augment `Other` only	83.01
Augment directed relations only	86.10

Table 1: Comparing variants of data augmentation.

	Depth	
	1	2
CNN	84.01	83.78
RNN	84.43	85.04

Table 2: Comparing CNNs and RNNs (also using F_1-score as the measurement).

4.3 Data Augmentation Details

As mentioned in Section 4.1, the SemEval-2010 Task 8 dataset contains an undirected class `Other` in addition to 9 directed relations (18 classes). For data augmentation, it is natural that the inversed `Other` relation is also in the `Other` class itself. However, if we augment all the relations, we observe a performance degradation of 0.7% (Table 1). We deem the `Other` class contains mainly noise, and is inimical to our model. Then we conducted another experiment where we only augmented the `Other` class. The result verifies our conjecture as we obtained an even larger degradation of 1.1% in this setting.

The pilot experiments suggest that we should take into consideration unfavorable noise when performing data augmentation. In this experiment, if we reverse the directed relations only and leave the `Other` class intact, the performance is improved by a large margin of 1.9%. This shows that our proposed data augmentation technique does help to mitigate the problem of data sparseness, if we carefully rule out the impact of noise.

During validation and testing, we shall decode the target label of an unseen data sample (with two entities e_1 and e_2). Through data augmentation, we are equipped with the probability of $r^{-1}(e_2, e_1)$ in addition to $r(e_1, e_2)$. In our experiment, we tried several settings and chose to use $r^{-1}(e_2, e_1)$ only, because it yields the highest the validation result. We think this is probably because the `Other` class brings more noise to r than r^{-1}, as the `Other` class is not augmented (and hence asymmetric).

We would like to point out that our data augmentation method is a general technique for relation classification, which is not *ad hoc* to a specific dataset; that the methodology for dealing with noise is also potentially applicable to other datasets.

4.4 RNNs vs. CNNs

As both RNNs and CNNs are prevailing neural models for NLP, we are curious whether deep architectures are also beneficial to CNNs. We tried a CNN with a sliding window of size 3 based on SDPs, similar to Xu et al. (2015a); other settings were as our DRNNs.

The results are shown in Table 2. We observe that a single layer of CNN is also effective, yielding an F_1-score slightly worse than our RNN. But the deep architecture hurts the performance of CNNs in this task. One plausible explanation is that, when convolution is performed, the beginning and end of a sentence are typically padded with a special symbol or simply zero. However, the shortest dependency path between two entities is usually not very long ($\sim$4 on average). Hence, sentence boundaries may play a large role in convolution, which makes CNNs vulnerable.

On the contrary, RNNs can deal with sentence boundaries smoothly, and the performance continues to increase with up to 4 hidden layers. (Details are deferred to Subsection 4.6.)

4.5 Overall Performance

Table 3 compares our DRNNs model with previous state-of-the-art methods.[2] The first entry in the table presents the highest performance achieved by traditional feature-based methods. Hendrickx et al. (2009) feed a variety of handcrafted features to the SVM classifier and achieve an F_1-score of 82.2%.

Recent performance improvements on this dataset are mostly achieved with the help of neural networks. In an early study, Socher et al. (2012) build a recursive network on constituency trees, but

[2]This paper was preprinted on arXiv on 14 Jan 2016.

Model	Features	F_1
SVM (Hendrickx et al., 2009)	POS, WordNet, prefixes and other morphological features, depdency parse, Levin classes, PropBank, FanmeNet, NomLex-Plus, Google n-gram, paraphrases, TextRunner	82.2
RNN (Socher et al., 2011)	Word embeddings	74.8
	+ POS, NER, WordNet	77.6
MVRNN (Socher et al., 2012)	Word embeddings	79.1
	+ POS, NER, WordNet	82.4
CNN (Zeng et al., 2014)	Word embeddings	69.7
	+ position embeddings, WordNet	82.7
Chain CNN (Ebrahimi and Dou, 2015)	Word embeddings, POS, NER, WordNet	82.7
CR-CNN (dos Santos et al., 2015)	Word embeddings	82.8
	+ position embeddings	84.1
FCM (Yu et al., 2014)	Word embeddings	80.6
	+ dependency parsing, NER	83.0
SDP-LSTM (Xu et al., 2015b)	Word embeddings	82.4
	Word + POS + GR + WordNet embeddings	83.7
DepNN (Liu et al., 2015)	Word embeddings + WordNet	83.0
	Word embeddings + NER	83.6
depLCNN (Xu et al., 2015a)	Word + WordNet + words around nominals	83.7
	+ negative sampling from NYT dataset	85.6
Ensemble Methods (Nguyen and Grishman, 2015)	Word+POS+NER+WordNet embeddings, CNNs, RNNs + Stacking	83.4
	Word+POS+NER+WordNet embeddings, CNNs, RNNs + Voting	84.1
DRNNs	Word+POS+GR+WordNet embeddings w/o data augmentation	84.2
	+ data augmentation	**86.1**

Table 3: Comparison of previous relation classification systems.

achieve a performance worse than Hendrickx et al. (2009). They extend their recursive network with matrix-vector interaction and elevate the F_1-score to 82.4%. Ebrahimi and Dou (2015) restrict the recursive network to SDP, which is slightly better than a sentence-wide network. In our previous study (Xu et al., 2015b), we introduce recurrent neural networks based on SDP and improve the F_1-score to 83.7%.

In the school of convolution, Zeng et al. (2014) construct a CNN on the word sequence; they also integrate word position embeddings, which benefit the CNN architecture. dos Santos et al. (2015) propose a similar CNN model, named CR-CNN, by replacing the common softmax cost function with a ranking-based cost function. By diminishing the impact of the `Other` class, they achieve an F_1-score of 84.1%. Xu et al. (2015a) design an SDP-based CNN with negative sampling, improving the performance to 85.6%.

Hybrid models of CNNs and RNNs do not appear to be very useful, achieving up to an F_1-score of 84.1% (Liu et al., 2015; Nguyen and Grishman, 2015).

Yu et al. (2014) propose a Feature-rich Compositional Embedding Model (FCM), which combines unlexicalized linguistic contexts and word embeddings. They do not use neural networks (at least in the usual sense) and achieve an F_1-score of 83.0%.

Our DRNNs model, along with data augmentation, achieves an F_1-score of 86.1%. Even if we do not apply data augmentation, the DRNNs model yields 84.2% F_1-score, which is also the highest score achieved without special treatment to the noisy `Other` class. The above results show the effectiveness of DRNNs, especially trained with a large (augmented) dataset.

4.6 Analysis of DRNNs' Depth

In this subsection, we analyze the effect of depth in our DRNNs model. We have tested the depth from the set $\{1, 2, \cdots, 6\}$, and plot the results in Figure 3. Initially, the performance increases if the depth is larger in both settings with and without augmentation. However, if we do not augment data, the performance peaks when the depth is 3. Provided with augmented training samples, the F_1-score continues to increase with up to 4 layers, and ends up with an F_1-score of 86.1%.

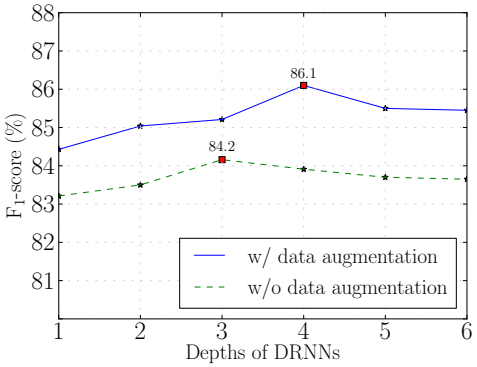


Figure 3: Analysis of the depth.[3]

We next investigate how RNN units in different layers are related to the ultimate task of interest. This is accomplished by tracing back information from pooling layers. Noticing that the pooling layer takes maximum value in each dimension, we can compute how much a hidden layer's units are gathered by pooling for further processing. In this way, we are able to demonstrate the information flow in RNN hidden units. We plot three examples in Figure 4. Here, rectangles refer to RNN hidden layers, unfolded along time. (Rounded rectangles are word embeddings.) The intensity of color reflects the ratio of the pooling proportion.

- Sample 1: "Until 1864 $[\text{vessels}]_{e_1}$ in the service of certain UK public offices defaced the Red Ensign with the $[\text{badge}]_{e_2}$ of their office" with label `Instrument-Agency`(e_2, e_1). Its two sub-paths of SDP are

  $[\text{vessels}]_{e_1} \rightarrow \text{until} \rightarrow \text{defaced}$
  $\text{defaced} \leftarrow \text{with} \leftarrow [\text{badge}]_{e_2}$

  From Figure 4a, we see that entities like *vessels* and *badge* are darker than the verb phrase *defaced with* on the embedding layer. When information is propagating horizontally and vertically, these entities are getting lighter, while the verb phrase becomes darker gradually. Intuitively, we think that, considering the relation `Instrument-Agency`(e_2, e_1), it is less informative with only two entities *vessels* and *badge*. When adding the semantic of verb phrase *defaced with*, we are more aware of the target relation.

- Sample 2: "Most of the $[\text{verses}]_{e_1}$ of the plantation songs had some reference to $[\text{freedom}]_{e_2}$" with label `Message-Topic`(e_1, e_2). Its two sub-paths of SDP are

  $[\text{verses}]_{e_1} \rightarrow \text{of} \rightarrow \text{most} \rightarrow \text{had}$
  $\text{had} \leftarrow \text{reference} \leftarrow \text{to} \leftarrow [\text{freedom}]_{e_2}$

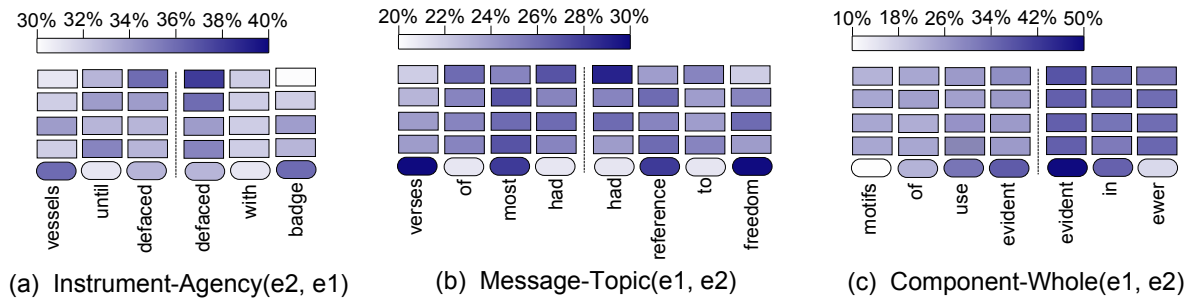
  Similar to Sample 1, we see from Figure 4b that the color of the "pivot" verb *had* is getting darker vertically, and becomes the darkest in the fourth RNN layer, indicating the highest pooling portion. This is probably because *had* links two ends of the relation, `Message` and `Topic`.

- Sample 3: "A more spare, less robust use of classical $[\text{motifs}]_{e_1}$ is evident in a $[\text{ewer}]_{e_2}$ of 1784-85" with label `Component-Whole`(e_1, e_2). Its two sub-paths of SDP are

  $[\text{motifs}]_{e_1} \rightarrow \text{of} \rightarrow \text{use} \rightarrow \text{evident}$
  $\text{evident} \leftarrow \text{in} \leftarrow [\text{ewer}]_{e_2}$

  Different from Figures 4a and 4b, higher layers pay more attention to entities rather than their ancestor. In this example, *motifs* and *ewer* appear to be more relevant to the relation `Component-Whole` than their common ancestor *evident*. The pooling proportion of entities (*motifs*, *ewer*) is increasing, while other words' proportion is decreasing.

[3] Using vanilla RNN with a depth of 1, we obtained a slightly better accuracy in this paper than Xu et al. (2015b).



(a) Instrument-Agency(e2, e1) (b) Message-Topic(e1, e2) (c) Component-Whole(e1, e2)

Figure 4: Visualization of information propagation along multiple RNN layers.

We summarize our findings as follows. (1) Pooled information usually peaks at one or a few words in the embedding layer. This makes sense because there is no information flow in this layer. (2) Information scatters over a wider range in hidden layers, showing that the recurrent propagation does mix information. (3) For a higher-level layer, the network pays more attention to those words that are more relevant to the relation, but whether entities or their common ancestor is more relevant is not consistent among different data samples.

5 Conclusion

In this paper, we proposed deep recurrent neural networks, named DRNNs, to improve the performance of relation classification. The DRNNs model, consisting of several RNN layers, explores the representation space of different abstraction levels. By visualizing DRNNs' units, we demonstrated that high-level layers are more capable of integrating information relevant to target relations. In addition, we have designed a data augmentation strategy by leveraging the directionality of relations. When evaluated on the SemEval dataset, our DRNNs model results in substantial performance boost. The performance generally improves when the depth increases; with a depth of 4, our model reaches the highest F_1-measure of 86.1%.

Acknowledgments

We thank all reviewers for their constructive comments. This research is supported by the National Basic Research Program of China (the 973 Program) under Grant No. 2015CB352201, the National Natural Science Foundation of China under Grant Nos. 61232015, 91318301, 61421091, and 61502014.

References

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.

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.

Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649.

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.

Michiel Hermans and Benjamin Schrauwen. 2013. Training and analysing deep recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 190–198.

Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 720–728.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 178–181.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng WANG. 2015. A dependency-based neural network for relation classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 285–290.

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

Thien Huu Nguyen and Ralph Grishman. 2015. Combining neural networks and log-linear models to improve relation extraction. *arXiv preprint arXiv:1511.05926*.

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.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pages 536–540.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pages 1785–1794.

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

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344.