# Simple Customization of Recursive Neural Networks for Semantic Relation Classification

**Kazuma Hashimoto†, Makoto Miwa††, Yoshimasa Tsuruoka†, and Takashi Chikayama†**
†The University of Tokyo, 3-7-1 Hongo, Bunkyo-ku, Tokyo, Japan
{hassy, tsuruoka, chikayama}@logos.t.u-tokyo.ac.jp
††The University of Manchester, 131 Princess Street, Manchester, M1 7DN, UK
makoto.miwa@manchester.ac.uk

## Abstract

In this paper, we present a recursive neural network (RNN) model that works on a syntactic tree. Our model differs from previous RNN models in that the model allows for an explicit weighting of important phrases for the target task. We also propose to average parameters in training. Our experimental results on semantic relation classification show that both phrase categories and task-specific weighting significantly improve the prediction accuracy of the model. We also show that averaging the model parameters is effective in stabilizing the learning and improves generalization capacity. The proposed model marks scores competitive with state-of-the-art RNN-based models.

## 1 Introduction

Recursive Neural Network (RNN) models are promising deep learning models which have been applied to a variety of natural language processing (NLP) tasks, such as sentiment classification, compound similarity, relation classification and syntactic parsing (Hermann and Blunsom, 2013; Socher et al., 2012; Socher et al., 2013). RNN models can represent phrases of arbitrary length in a vector space of a fixed dimension. Most of them use minimal syntactic information (Socher et al., 2012).

Recently, Hermann and Blunsom (2013) proposed a method for leveraging syntactic information, namely CCG combinatory operators, to guide composition of phrases in RNN models. While their models were successfully applied to binary sentiment classification and compound similarity tasks, there are questions yet to be answered, e.g., whether such enhancement is beneficial in other NLP tasks as well, and whether a similar improvement can be achieved by using syntactic information of more commonly available types such as phrase categories and syntactic heads.

In this paper, we present a supervised RNN model for a semantic relation classification task. Our model is different from existing RNN models in that important phrases can be explicitly weighted for the task. Syntactic information used in our model includes part-of-speech (POS) tags, phrase categories and syntactic heads. POS tags are used to assign vector representations to word-POS pairs. Phrase categories are used to determine which weight matrices are chosen to combine phrases. Syntactic heads are used to determine which phrase is weighted during combining phrases. To incorporate task-specific information, phrases on the path between entity pairs are further weighted.

The second contribution of our work is the introduction of parameter averaging into RNN models. In our preliminary experiments, we observed that the prediction performance of the model often fluctuates significantly between training iterations. This fluctuation not only leads to unstable performance of the resulting models, but also makes it difficult to fine-tune the hyperparameters of the model. Inspired by Swersky et al. (2010), we propose to average the model parameters in the course of training. A recent technique for deep learning models of similar vein is *dropout* (Hinton et al., 2012), but averaging is simpler to implement.

Our experimental results show that our model per-



*Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1372–1376,
Seattle, Washington, USA, 18-21 October 2013. 

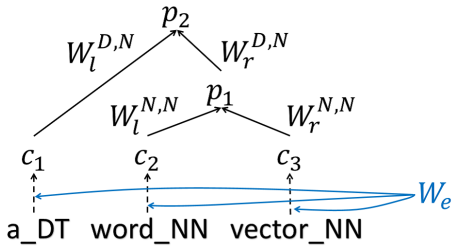


Figure 1: A recursive representations of a phrase "a word vector" with POS tags of the words (DT, NN and NN respectively). For example, the two word-POS pairs "word_NN" and "vector_NN" with a syntactic category $N$ are combined to represent the phrase "word vector".

forms better than standard RNN models. By averaging the model parameters, our model achieves performance competitive with the MV-RNN model in Socher et al. (2012), without using computationally expensive word-dependent matrices.

## 2 An Averaged RNN Model with Syntax

Our model is a supervised RNN that works on a binary syntactic tree. As our first step to leverage information available in the tree, we distinguish words with the same spelling but POS tags in the vector space. Our model also uses different weight matrices dependent on the phrase categories of child nodes (phrases or words) in combining phrases. Our model further weights those nodes that appear to be important.

Compositional functions of our model follow those of the SU-RNN model in Socher et al. (2013).

### 2.1 Word-POS Vector Representations

Our model simply assigns vector representations to word-POS pairs. For example, a word "caused" can be represented in two ways: "caused_VBD" and "caused_VBN". The vectors are represented as column vectors in a matrix $\boldsymbol{W}_e \in \mathbb{R}^{d\times|\mathbb{V}|}$, where $d$ is the dimension of a vector and $\mathbb{V}$ is a set of all word-POS pairs we consider.

### 2.2 Compositional Functions with Syntax

In construction of parse trees, we associate each of the tree node with its $d$-dimensional vector representation computed from vector representations of its subtrees. For leaf nodes, we look up word-POS vector representations in $\mathbb{V}$. Figure 1 shows an example of such recursive representations. A parent vector $\boldsymbol{p} \in \mathbb{R}^{d\times 1}$ is computed from its direct child vectors $\boldsymbol{c}_l$ and $\boldsymbol{c}_r \in \mathbb{R}^{d\times 1}$:

$$\boldsymbol{p} = \tanh(\alpha_l \boldsymbol{W}_l^{T_{c_l},T_{c_r}} \boldsymbol{c}_l + \alpha_r \boldsymbol{W}_r^{T_{c_l},T_{c_r}} \boldsymbol{c}_r + \boldsymbol{b}^{T_{c_l},T_{c_r}}),$$

where $\boldsymbol{W}_l^{T_{c_l},T_{c_r}}$ and $\boldsymbol{W}_r^{T_{c_l},T_{c_r}} \in \mathbb{R}^{d\times d}$ are weight matrices that depend on the phrase categories of $\boldsymbol{c}_l$ and $\boldsymbol{c}_r$. Here, $\boldsymbol{c}_l$ and $\boldsymbol{c}_r$ have phrase categories $T_{c_l}$ and $T_{c_r}$ respectively (such as *N*, *V*, etc.). $\boldsymbol{b}^{T_{c_l},T_{c_r}} \in \mathbb{R}^{d\times 1}$ is a bias vector. To incorporate the importance of phrases into the model, two subtrees of a node may have different weights $\alpha_l \in [0,1]$ and $\alpha_r (= 1-\alpha_l)$, taking phrase importance into account. The value of $\alpha_l$ is manually specified and automatically applied to all nodes based on prior knowledge about the task. In this way, we can compute vector representations for phrases of arbitrary length. We denote a set of such matrices as $\boldsymbol{W}_{lr}$ and bias vectors as $\boldsymbol{b}$.

### 2.3 Objective Function and Learning

As with other RNN models, we add on the top of a node $\boldsymbol{x}$ a $\mathrm{softmax}$ classifier. The classifier is used to predict a $K$-class distribution $\boldsymbol{d}(\boldsymbol{x}) \in \mathbb{R}^{K\times 1}$ over a specific task to train our model:

$$\boldsymbol{d}(\boldsymbol{x}) = \mathrm{softmax}(\boldsymbol{W}^{label}\boldsymbol{x} + \boldsymbol{b}^{label}), \qquad (1)$$

where $\boldsymbol{W}^{label} \in \mathbb{R}^{K\times d}$ is a weight matrix and $\boldsymbol{b}^{label} \in \mathbb{R}^{K\times 1}$ is a bias vector. We denote $\boldsymbol{t}(\boldsymbol{x}) \in \mathbb{R}^{K\times 1}$ as the target distribution vector at node $\boldsymbol{x}$. $\boldsymbol{t}(\boldsymbol{x})$ has a 0-1 encoding: the entry at the correct label of $\boldsymbol{t}(\boldsymbol{x})$ is 1, and the remaining entries are 0. We then compute the cross entropy error between $\boldsymbol{d}(\boldsymbol{x})$ and $\boldsymbol{t}(\boldsymbol{x})$:

$$E(\boldsymbol{x}) = -\sum_{k=1}^{K} \boldsymbol{t}_k(\boldsymbol{x}) \log \boldsymbol{d}_k(\boldsymbol{x}),$$

and define an objective function as the sum of $E(\boldsymbol{x})$ over all training data:

$$J(\theta) = \sum_{\boldsymbol{x}} E(\boldsymbol{x}) + \frac{\lambda}{2}\|\theta\|^2,$$

where $\theta = (\boldsymbol{W}_e, \boldsymbol{W}_{lr}, \boldsymbol{b}, \boldsymbol{W}^{label}, \boldsymbol{b}^{label})$ is the set of our model parameters that should be learned. $\lambda$ is a vector of regularization parameters.

To compute $\boldsymbol{d}(\boldsymbol{x})$, we can directly leverage any other nodes' feature vectors in the same tree. We denote such additional feature vectors as $\boldsymbol{x}'_i \in \mathbb{R}^{d \times 1}$, and extend Eq. (1):

$$\boldsymbol{d}(\boldsymbol{x}) = \mathrm{softmax}(\boldsymbol{W}^{label}\boldsymbol{x} + \sum_i \boldsymbol{W}_i^{add}\boldsymbol{x}'_i + \boldsymbol{b}^{label}),$$

where $\boldsymbol{W}_i^{add} \in \mathbb{R}^{K \times d}$ are weight matrices for additional features. We denote these matrices $\boldsymbol{W}_i^{add}$ as $\boldsymbol{W}^{add}$. We also add $\boldsymbol{W}^{add}$ to $\theta$:

$$\theta = (\boldsymbol{W}_e, \boldsymbol{W}_{lr}, \boldsymbol{b}, \boldsymbol{W}^{label}, \boldsymbol{W}^{add}, \boldsymbol{b}^{label}).$$

The gradient of $J(\theta)$

$$\frac{\partial J(\theta)}{\partial \theta} = \sum_{\boldsymbol{x}} \frac{\partial E(\boldsymbol{x})}{\partial \theta} + \lambda\theta$$

is efficiently computed via backpropagation through structure (Goller and Küchler, 1996). To minimize $J(\theta)$, we use batch L-BFGS[1] (Hermann and Blunsom, 2013; Socher et al., 2012).

### 2.4 Averaging

We use averaged model parameters

$$\overline{\theta} = \frac{1}{T+1}\sum_{t=0}^{T}\theta_t$$

at test time, where $\theta_t$ is the vector of model parameters after $t$ iterations of the L-BFGS optimization. Our preliminary experimental results suggest that averaging $\theta$ except $\boldsymbol{W}_e$ works well.

## 3 Experimental Settings

We used the Enju parser (Miyao and Tsujii, 2008) for syntactic parsing. We used 13 phrase categories given by Enju.

### 3.1 Task: Semantic Relation Classification

We evaluated our model on a semantic relation classification task: SemEval 2010 Task 8 (Hendrickx et al., 2010). Following Socher et al. (2012), we regarded the task as a 19-class classification problem. There are 8,000 samples for training, and 2,717 for test. For the validation set, we randomly sampled 2,182 samples from the training data.

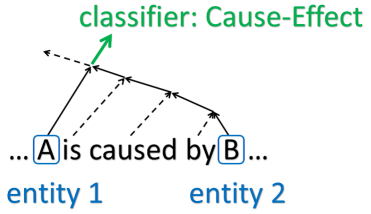


Figure 2: Classifying the relation between two entities.

To predict a class label, we first find the minimal phrase that covers the target entities and then use the vector representation of the phrase (Figure 2).

As explained in Section 2.3, we can directly connect features on any other nodes to the softmax classifier. In this work, we used three such internal features: two vector representations of target entities and one averaged vector representation of words between the entities[2].

### 3.2 Weights on Phrases

We tuned the weight $\alpha_l$ (or $\alpha_r$) introduced in Section 2.2 for this particular task. There are two factors: syntactic heads and syntactic path between target entities. Our model puts a weight $\beta \in [0.5, 1]$ on head phrases, and $1-\beta$ on the others. For relation classification tasks, syntactic paths between target entities are important (Zhang et al., 2006), so our model also puts another weight $\gamma \in [0.5, 1]$ on phrases on the path, and $1-\gamma$ on the others. When both child nodes are on the path or neither of them on the path, we set $\gamma = 0.5$. The two weight factors are summed up and divided by 2 to be the final weights $\alpha_l$ and $\alpha_r$ to combine the phrases. For example, we set $\alpha_l = \frac{(1-\beta)+\gamma}{2}$ and $\alpha_r = \frac{\beta+(1-\gamma)}{2}$ when the right child node is the head and the left child node is on the path.

### 3.3 Initialization of Model Parameters and Tuning of Hyperparameters

We initialized $\boldsymbol{W}_e$ with 50-dimensional word vectors[3] trained with the model of Collobert et

[1] We used libLBFGS provided at `http://www.chokkan.org/software/liblbfgs/`.

[2] Socher et al. (2012) used richer features including words around entity pairs in their implementation.

[3] The word vectors are provided at `http://ronan.collobert.com/senna/`. We used the vectors without any modifications such as normalization.

| Method | F1 (%) |
|---|---|
| Our model | 79.4 |
| RNN | 74.8 |
| MV-RNN | 79.1 |
| RNN w/ POS, WordNet, NER | 77.6 |
| MV-RNN w/ POS, WordNet, NER | 82.4 |
| SVM w/ bag-of-words | 73.1 |
| SVM w/ lexical and semantic features | 82.2 |

Table 1: Comparison of our model with other methods on SemEval 2010 task 8.

| Method | F1 (%) |
|---|---|
| Our model | 79.4 |
| Our model w/o phrase categories (PC) | 77.7 |
| Our model w/o head weights (HW) | 78.8 |
| Our model w/o path weights (PW) | 78.7 |
| Our model w/o averaging (AVE) | 76.9 |
| Our model w/o PC, HW, PW, AVE | 74.1 |

Table 2: Contributions of syntactic and task-specific information and averaging.

al. (2011), and $\boldsymbol{W}_{lr}$ with $\frac{I}{2} + \varepsilon$, where $I \in \mathbb{R}^{d \times d}$ is an identity matrix. Here, $\varepsilon$ is zero-mean gaussian random variable with a variance of $0.01$. The initialization of $\boldsymbol{W}_{lr}$ is the same as that of Socher et al. (2013). The remaining model parameters were initialized with $0$.

We tuned hyperparameters in our model using the validation set for each experimental setting. The hyperparameters include the regularization parameters for $\boldsymbol{W}_e$, $\boldsymbol{W}_{lr}$, $\boldsymbol{W}^{label}$ and $\boldsymbol{W}^{add}$, and the weights $\beta$ and $\gamma$. For example, the best performance for our model with all the proposed methods was obtained with the values: $10^{-6}$, $10^{-4}$, $10^{-3}$, $10^{-3}$, $0.7$ and $0.9$ respectively.

## 4 Results and Discussion

Table 1 shows the performance of our model and that of previously reported systems on the test set. The performance of an SVM system with bag-of-words features was reported in Rink and Harabagiu (2010), and the performance of the RNN and MV-RNN models was reported in Socher et al. (2012). Our model achieves an F1 score of 79.4% and outperforms the RNN model (74.8% F1) as well as the simple SVM-based system (73.1% F1). More notably, the score of our model is competitive with that of the MV-RNN model (79.1% F1), which is computationally much more expensive. Readers are referred to Hermann and Blunsom (2013) for the discussion about the computational complexity of the MV-RNN model. We improved the performance of RNN models on the task without much increasing the complexity. This is a significant practical advantage of our model, although its expressive power is not the same as that of the MV-RNN model.

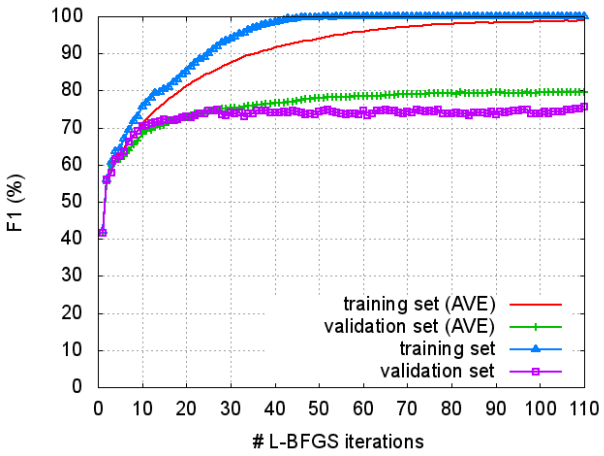


Figure 3: F1 vs Training iterations.

Our model outperforms the RNN model with one lexical and two semantic external features: POS tags, WordNet hypernyms and named entity tags (NER) of target word pairs (external features). The MV-RNN model with external features shows better performance than our model. An SVM with rich lexical and semantic features (Rink and Harabagiu, 2010) also outperforms ours. Note, however, that this is not a fair comparison because those models use rich external resources such as WordNet and named entity tags.

### 4.1 Contributions of Proposed Methods

We conducted additional experiments to quantify the contributions of phrase categories, heads, paths and averaging to our classification score. As shown in Table 2, our model without phrase categories, heads or paths still outperforms the RNN model with external features. On the other hand, our model without averaging yields a lower score than the RNN model with external features, though it is still bet-

ter than the RNN model alone. Without utilizing these four properties, our model obtained only 74.1% F1. These results indicate that syntactic and task-specific information and averaging contribute to the performance improvement. The improvement is achieved by a simple modification of compositional functions in RNN models.

### 4.2 Effects of Averaging in Training

Figure 3 shows the training curves in terms of F1 scores. These curves clearly demonstrate that parameter averaging helps to stabilize the learning and improve generalization capacity.

## 5 Conclusion

We have presented an averaged RNN model for semantic relation classification. Our experimental results show that syntactic information such as phrase categories and heads improves the performance, and the task-specific weighting is also beneficial. The results also demonstrate that averaging the model parameters not only stabilizes the learning but also improves the generalization capacity of the model. As future work, we plan to combine deep learning models with richer information such as predicate-argument structures.

## Acknowledgments

We thank the anonymous reviewers for their insightful comments.

## References

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. *Natural Language Processing (almost) from Scratch*. In *JMLR*, 12:2493–2537.

Christoph Goller and Andreas Küchler. 1996. *Learning Task-Dependent Distributed Representations by Backpropagation Through Structure*. In *ICNN*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano and Stan Szpakowicz. 2010. *SemEval-2010 Task 8: Multi-Way Classication of Semantic Relations Between Pairs of Nominals*. In *SemEval 2010*.

Karl Moritz Hermann and Phil Blunsom. 2013. *The Role of Syntax in Vector Space Models of Compositional Semantics*. In *ACL*.

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

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

Bryan Rink and Sanda Harabagiu. 2010. *UTD: Classifying Semantic Relations by Combining Lexical and Semantic Resources*. In *SemEval 2010*.

Richard Socher, Brody Huval, Christopher D. Manning and Andrew Y. Ng. 2012. *Semantic Compositionality Through Recursive Matrix-Vector Spaces*. In *EMNLP*.

Richard Socher, John Bauer, Christopher D. Manning and Andrew Y. Ng. 2013. *Parsing with Compositional Vector Grammars*. In *ACL*.

Kevin Swersky, Bo Chen, Ben Marlin and Nando de Freitas. 2010. *A tutorial on stochastic approximation algorithms for training Restricted Boltzmann Machines and Deep Belief Nets*. In *ITA workshop*.

Min Zhang, Jie Zhang, Jian Su and Guodong Zhou. 2006. *A Composite Kernel to Extract Relations between Entities with Both Flat and Structured Features*. In *COLING/ACL*.