Chain based RNN for Relation Classification

Javid Ebrahimi and **Dejing Dou**
Department of Computer and Information Science, University of Oregon
Eugene, Oregon 97403, USA
{javid,dou}@cs.uoregon.edu

Abstract

We present a novel approach for relation classification, using a recursive neural network (RNN), based on the shortest path between two entities in a dependency graph. Previous works on RNN are based on constituency-based parsing because phrasal nodes in a parse tree can capture compositionality in a sentence. Compared with constituency-based parse trees, dependency graphs can represent relations more compactly. This is particularly important in sentences with distant entities, where the parse tree spans words that are not relevant to the relation. In such cases RNN cannot be trained effectively in a timely manner. However, due to the lack of phrasal nodes in dependency graphs, application of RNN is not straightforward. In order to tackle this problem, we utilize dependency constituent units called *chains*. Our experiments on two relation classification datasets show that Chain based RNN provides a shallower network, which performs considerably faster and achieves better classification results.

1 Introduction

Relation extraction is the task of finding relations between entities in text, which is useful for several tasks such as information extraction, summarization, and question answering (Wu and Weld, 2010). For instance, in the sentence: those "cancers" were caused by radiation "exposures," the two entities have a *cause-effect* relation. As reported in detail (SaraWagi, 2008), one approach to the problem involves supervised methods where the models rely on lexical, syntactic, and semantic features to classify relations between pairs of entities. The downside of this approach is that one has to retrain the model for other domains with different target relations. Thus it is not scalable to the web, where thousands of (previously-unseen) relations exist (Banko et al., 2007). To address this problem, Open Information Extraction is proposed, which does not require supervision. In these systems (Banko et al., 2007; Mausam et al., 2012), patterns based on lexical, syntactic, POS, and dependency features are extracted. While these patterns give good precision, they suffer from low recall (Banko and Etzioni, 2008). This is because they fail to extract patterns which have not been pre-specified, and thereby are unable to generalize.

Recursive Neural Network (RNN) has proven to be highly successful in capturing semantic compositionality in text and has improved the results of several Natural Language Processing tasks (Socher et al., 2012; Socher et al., 2013). Previous applications of Recursive Neural Networks (RNN) to supervised relation extraction (Socher et al., 2012; Hashimoto et al., 2013; Khashabi, 2013) are based on constituency-based parsers. These RNNs may span words that do not contribute to the relation. We investigate the incorporation of dependency parsing into RNN that can give a more compact representation of relations.

Our contribution is introducing a compositional account of dependency graphs that can match RNN's recursive nature, and can be applied to relation classification. We study different data structures that incorporate dependency trees into RNNs.

Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL, pages 1244–1249,
Denver, Colorado, May 31 – June 5, 2015.

One of these structures produces a compact full binary tree that compared with the constituency-based RNN, has higher classification accuracy and saves up to 70% in the training time.

2 Related Work

At the core of deep learning techniques for NLP, lies the vector based word representation, which maps words to an n-dimensional space. Having word vectors as parameters makes neural models flexible in finding different word embeddings for separate tasks (Collobert and Weston, 2008). Recursive Neural Network (RNN) is a recursive deep architecture that can learn feature representation of words, phrases and sentences.

As an example, in (Socher et al., 2010), each node in the parse tree is associated with a vector and at each internal node p, there exists a composition function that takes its input from its children $c_1 \in \mathbb{R}^n$ and $c_2 \in \mathbb{R}^n$.

$$p = f(c_1, c_2) = tanh(W \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + b) \quad (1)$$

The matrix $W \in \mathbb{R}^{n \times 2n}$ is the global composition parameter, b is the bias term, and the output of the function $p \in \mathbb{R}^n$ is another vector in the space of inputs. Socher et al. (2012) propose Matrix-Vector Recursive Neural Network (MV-RNN), where instead of using only vectors for words, an additional matrix for each word is used to capture operator semantics in language. To apply RNN to relation classification, they find the path in the parse tree between the two entities and apply compositions bottom up. Hashimoto et al. (2013) follow the same design but introduce a different composition function. They make use of word-POS pairs and use untied weights based on phrase categories of the pair.

Socher et al. (2014) introduce a dependency-based RNN that extracts features from a dependency graph whose composition function has major differences from ours. Their function consists of a linear sum of unary compositions, while our function is a binary composition of children. Our work is also related to (Bunescu and Mooney, 2005), where the similarity between the words on the path connecting two entities in the dependency graph is used to devise a Kernel function.

3 Chain based RNN

While constituency-based parsing seems to be a reasonable choice for compositionality in general, it may not be the best choice for all NLP tasks. In particular, for relation classification, one may prefer to use a structure that encodes more information about the relations between the words in a sentence. To this end, we use dependency-based parsing that provides a one-to-one correspondence between nodes in a dependency graph (DG).

DGs are significantly different from constituency parse trees since they lack phrasal nodes. More precisely, the internal nodes where the nonlinear combinations take place, do not exist in DGs. Therefore, we modify the original RNN and present a dependency-based RNN for relation classification. In our experiments, we restrict ourselves to trees where each dependent has only one head. We also use the example in Figure 1 for better illustration; in this example the arguments of the relation are $child$ and $cradle$.

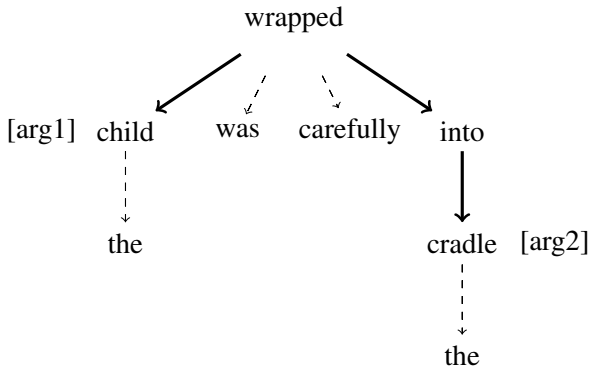

Figure 1: DG: the child was carefully wrapped into the cradle.

We apply compositions on the words on the shortest path between entities. From a linguistic point of view, this type of composition is related to the concept of $chain$ or dependency constituent unit in DGs (Osborne, 2005).

> **Chain**: The words A ... B ... C ... (order irrelevant) form a chain iff A immediately dominates (is the parent of) B and C, or if A immediately dominates B and B immediately dominates C.

Based on this definition, *child wrapped, into cradle, wrapped into cradle, child wrapped into cradle* all qualify as a chain while *child was* does not. To illustrate the motivation to use dependency parsing, consider the sentence:

> The hidden "camera," found by a security guard, was hidden in a business card-sized "box" placed at an unmanned ATM.

The shortest path between entities is:

> camera → found ← hidden ← in ← box

Using dependency parsing, we only need four compositions for this chain, which results in 86% decrease against constituency-based parsing.

Now with all words represented as vectors, we need to find a reduced dimensional representation of the chain in fixed size. To this end, we transform this chain to a data structure, the root of which represents the extracted features.

3.1 Fixed Structure

We cannot use an off-the-shelf syntax parser to create a tree for the chain because the chain may not necessarily be a coherent English statement. Thus, we build two Directed Acyclic Graph (DAG) structures by heuristics. The idea is to start from argument(s) and recursively combine dependent-head pairs to the (common) ancestor i.e., each head is combined with the subtree below itself. In the simplest case: $a \rightarrow b$ results in $p = f(a, b)$.

The subtlety of this approach lies in the treatment of the word with two dependents. We use two methods to handle such a node: 1) including it in only one composition as in Figure 2 or 2) including it in two compositions and sum their results as in Figure 3.

Both structures produce a DAG where each internal node has two children and there is only one node with two non-leaf children. We now prove that this greedy algorithm results in a full binary tree for the first case. We skip the proof of the algorithm for the second case which produces a full binary DAG.

Lemma: *There is at most one node with exactly two none-leaf children in the tree.*

Proof. If one of the arguments is an ancestor of the other argument e.g., arg1 → ... → arg2, then

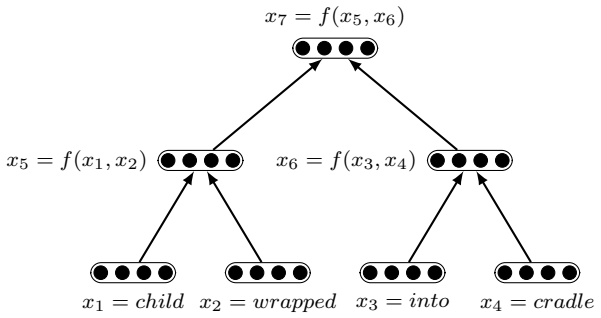

Figure 2: a fixed tree example

obviously every head on the chain has exactly one dependent. Combination of each head and its subtree's output vector results in a full binary node in the tree. If the arguments have a common ancestor p e.g., arg1 → ... p ... ← arg2, then that particular node has two dependents. In this case, the parent is combined with either its left or right subtrees, and its result is combined with the output of the other child. No other head has this property; otherwise, p is not the common ancestor.

Theorem: *The algorithm converts a chain to a full binary tree.*

Proof. The leaves of the tree are words of the chain. By applying the lemma, there exists one root and all internal nodes have exactly two children.

Note that we only consider dependency trees as the input; so each pair of arguments has a unique common ancestor. Concretely, having a connected graph leads to at least one such ancestor and having only one head for each node (being a tree) leads to exactly one such ancestor.

3.2 Predicted Tree Structure

Instead of using a deterministic approach to create the tree, we can use Recursive Autoencoders (RAE) to find the best representation of the chain. This model is similar to (Socher et al., 2011) with some modification in implementation. Socher et al. (2011) use a semi supervised method where the objective function is a weighted sum of the supervised and unsupervised error. We achieved better results with a pipeline where first, during pre-training, the unsupervised autoencoder predicts the structure of RNN and then during training, the supervised cross entropy error is minimized.

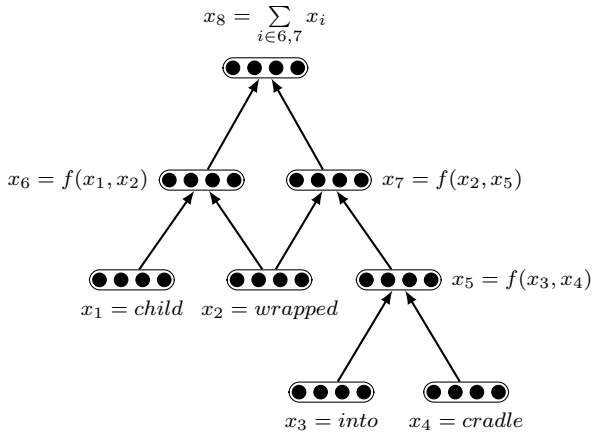

Figure 3: a fixed DAG example

4 Learning

To predict the label of the relation, a softmax classifier is added on top of the tree. i.e., $y_i = softmax(P_n^T W^{label})$ where $L \in \mathbb{R}^k$, k is the number of classes, and P_n is the final vector on top of the tree for sentence n. The objective function is the sum of cross entropy error at all the nodes, for all the sentences in the training set.

$$E(\theta) = -\sum_n \sum_k t_n^k \log y_n^k + \frac{\lambda}{2} \parallel \theta \parallel^2 \quad (2)$$

The vectors for target, predicted labels, and regularization parameters are denoted by t_n, y_n and λ respectively. We initialize the word vectors with pretrained 50-dimensional words from (Collobert and Weston, 2008) and initialize other parameters by a normal distribution with mean of 0 and standard deviation of 0.01. Derivatives are computed by backpropagation through structure (Goller and Kuchler, 1996) and L-BFGS is used for optimization.

5 Experiments

In this section we discuss our experimental results on two datasets for relation classification. To derive the dependency tree for each sentence, we use arc-eager MaltParser (Goldberg and Nivre, 2012). We set the hyper-parameters through a validation set for the first dataset and use them for the second dataset too. Similar to the previous works, a few internal features were also added e.g., depth of the tree, distance between entities, context words, and the type of dependencies in our model. We found that using dependency types inside the composition function as in typed functions worsens the results.

5.1 SemEval-2010 Task 8

This data set consists of 10017 sentences and nine types of relations between nominals (Hendrickx et al., 2010). Table 1 compares the results of our tree based chain RNN (C-RNN), DAG based chain RNN (DC-RNN) and the autoencoder based one (C-RNN-RAE) with other RNN models and the best system participating (Rink and Harabagiu, 2010) in the task. Evaluation of the systems is done by comparing the F-measure of their best runs. The best system (Rink and Harabagiu, 2010) uses SVM with many sets of features. We add some external features using supersense sequence tagger (Ciaramita and Altun, 2006). Adding POS tags, WordNet hypernyms, and named entity tags (NER) of the two arguments helps C-RNN improve the results.

We implement SDT-RNN (Socher et al., 2014) which has similar complexity as our model but has significantly lower F-measure. SDT-RNN also performs much better when considering only the words on the path between entities; confirming our hypothesis about the effectiveness of chains. This can be attributed to the intuitive advantage of dependency trees where the shortest path between entities captures most of the information about the relation (Bunescu and Mooney, 2005).

As it can bee seen in Table 1, C-RNN achieves the best results. The baseline RNN, uses a global composition function and $\mathbb{R}^{50}$ vectors for each word. We also use the same number of model parameters.

The advantage of our approach is that our models are computationally less expensive compared with other RNN models. MV-RNN (Socher et al., 2012) uses an additional matrix $\mathbb{R}^{50\times50}$ for each word, resulting in a 50 fold increase in the number of model parameters. POS-RNN (Hashimoto et al., 2013) uses untied weight matrices and POS based word vectors that results in about 100% increase in the number of model parameters compared with C-RNN.

Relations with long distances between entities are harder to classify. This is illustrated in Figure 4 where MV-RNN and C-RNN are compared. Considering three bins for the distance between two en-

Method	F-measure	Feature sets
RNN	74.8	-
SDT-RNN	75.12	-
MV-RNN	79.1	-
POS-RNN	79.4	-
DC-RNN	77.36	-
C-RNN-RAE	78.78	-
C-RNN	**79.68**	-
SVM	82.2	POS, WordNet, Levine classes, PropBank, FrameNet, TextRunner, paraphrases, Google *n*-grams, NormLex-Plus, morphological features, dependency parse features
MV-RNN	82.4	POS, NER, WordNet
C-RNN	**82.66**	POS, NER, WordNet

Table 1: Results on SemEval 2010 relation classification task with the feature sets used. C-RNN outperforms all RNN based models. By including three extra features, it achieves the state-of-the-art performance.

tities, the figure shows what fraction of test instances are misclassified in each bin. Both classifiers make more errors when the distance between entities is longer than 10. The performance of the two classifiers for distances less than five is quite similar while C-RNN has the advantage in classifying more relations correctly when the distance increases.

5.2 SemEval-2013 Task 9.b

To further illustrate the advantage of C-RNN over MV-RNN, we evaluate our work on another data set. See Table 2. In this task, the goal is to extract interactions between drug mentions in text. The corpus (Segura-Bedmar et al., 2013) consists of 1,017 texts that were manually annotated with a total of 5021 drug-drug interactions of four types: *mechanism, effect, advise* and *int*.

Method	Precision	Recall	F=measure
MV-RNN	74.07	65.53	67.84
C-RNN	**75.31**	**66.19**	**68.64**

Table 2: Results on SemEval 2013 Drug-Drug Interaction task

5.3 Training Time

Dependency graphs can represent relations more compactly by utilizing only the words on the shortest path between entities. C-RNN uses a sixth of neural computations of MV-RNN. More precisely, there is an 83% decrease in the number of *tanh* evaluations. Consequently, as demonstrated by Figure 5, C-RNN runs 3.21 and 1.95 times faster for SemEval 2010 and SemEval 2013 respectively.

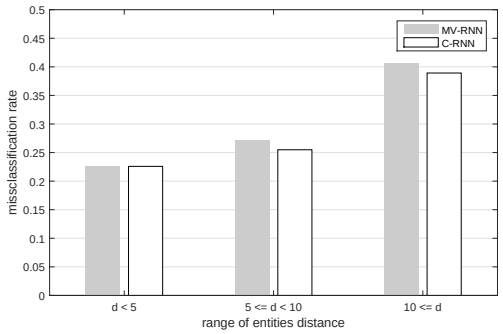

Figure 4: Misclassification based on entities distance in three bins. More errors occur with entities separated by more than ten words. C-RNN performs better in bottleneck long distances.

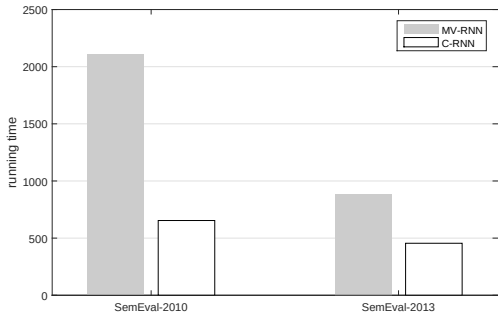

Figure 5: Training time measured by seconds. Experiments were run on a cluster node with 6 core 2.66GHz cpu.

6 Conclusions

Recently, Recursive Neural Network (RNN) has found a wide appeal in the Machine Learning community. This deep architecture has been applied in several NLP tasks including relation classification. We present an RNN architecture based on a compositional account of dependency graphs. The proposed RNN model is based on the shortest path between entities in a dependency graph. The resulting shallow network is superior for supervised learning in terms of speed and accuracy. We improve the classification results and save up to 70% in training time compared with a constituency-based RNN . The limitation of our Chain based RNN is that it assumes the named entities to be known in advance. This requires a separate named entity recognizer and cannot extract the entities jointly with the relation classifier.

Acknowledgment

This work is partially supported by the NIH grant R01GM103309.

References

Michele Banko and Oren Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *Proceedings of ACL*, pages 28–36.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of IJCAI*, pages 2670–2676.

Razvan Bunescu and Raymond J. Mooney. 2005. A Shortest Path Dependency Kernel for Relation Extraction . In *Proceedings of HLT/EMNLP*, pages 724–731.

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of EMNLP*, pages 594–602.

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of COLING*, pages 959–976.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of ICNN*, pages 347–352.

Kazuma Hashimoto, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Simple customization of recursive neural networks for semantic relation classification. In *Proceedings of EMNLP*, pages 1372–1376.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of SemEval*, pages 33–38.

Daniel Khashabi. 2013. On the recursive neural networks for relation extraction and entity recognition. Technical report, UIUC.

Mausam, Michael D Schmitz, Robert E. Bart, Stephen Soderland, and Oren Etzioni. 2012. Open Language Learning for Information Extraction . In *Proceedings of EMNLP*, pages 523–534.

Timothy Osborne. 2005. Beyond the constituent: a dependency grammar analysis of chains. *Folia Linguistica*, 39(3-4):251–297.

Bryan Rink and Sanda Harabagiu. 2010. Utd: Classifying semantic relations by combining lexical and semantic resources. In *Proceedings of SemaEval*, pages 256–259.

Sunita SaraWagi. 2008. Information Extraction. In *Foundations and Trends in Databases,Volume 1 Issue 3*, pages 261–377.

Isabel Segura-Bedmar, Paloma Martínez, and María Herrero Zazo. 2013. Semeval-2013 task 9 : Extraction of drug-drug interactions from biomedical texts (DDIExtraction 2013). In *Proceedings of SemEval*, pages 341–350.

Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2010. Learning Continuous Phrase Representations and Syntactic Parsing with Recursive Neural Networks. In *Deep Learning and Unsupervised Feature Learning Workshop, NIPS*, pages 1–9.

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 EMNLP*, 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 EMNLP*, pages 1201–1211.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of EMNLP*, pages 1631–1642.

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. *TACL*, 2:207–218.

Fei Wu and Daniel S. Weld. 2010. Open information extraction using wikipedia. In *Proceeding of ACL*, pages 118–127.