End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures

Makoto Miwa
Toyota Technological Institute
Nagoya, 468-8511, Japan
makoto-miwa@toyota-ti.ac.jp

Mohit Bansal
Toyota Technological Institute at Chicago
Chicago, IL, 60637, USA
mbansal@ttic.edu

Abstract

We present a novel end-to-end neural model to extract entities and relations between them. Our recurrent neural network based model captures both word sequence and dependency tree substructure information by stacking bidirectional tree-structured LSTM-RNNs on bidirectional sequential LSTM-RNNs. This allows our model to jointly represent both entities and relations with shared parameters in a single model. We further encourage detection of entities during training and use of entity information in relation extraction via entity pretraining and scheduled sampling. Our model improves over the state-of-the-art feature-based model on end-to-end relation extraction, achieving 12.1% and 5.7% relative error reductions in F1-score on ACE2005 and ACE2004, respectively. We also show that our LSTM-RNN based model compares favorably to the state-of-the-art CNN based model (in F1-score) on nominal relation classification (SemEval-2010 Task 8). Finally, we present an extensive ablation analysis of several model components.

1 Introduction

Extracting semantic relations between entities in text is an important and well-studied task in information extraction and natural language processing (NLP). Traditional systems treat this task as a pipeline of two separated tasks, i.e., named entity recognition (NER) (Nadeau and Sekine, 2007; Ratinov and Roth, 2009) and relation extraction (Zelenko et al., 2003; Zhou et al., 2005), but recent studies show that end-to-end (joint) modeling of entity and relation is important for high performance (Li and Ji, 2014; Miwa and Sasaki, 2014) since relations interact closely with entity information. For instance, to learn that *Toefting* and *Bolton* have an *Organization-Affiliation (ORG-AFF)* relation in the sentence *Toefting transferred to Bolton*, the entity information that *Toefting* and *Bolton* are *Person* and *Organization* entities is important. Extraction of these entities is in turn encouraged by the presence of the context words *transferred to*, which indicate an employment relation. Previous joint models have employed feature-based structured learning. An alternative approach to this end-to-end relation extraction task is to employ automatic feature learning via neural network (NN) based models.

There are two ways to represent relations between entities using neural networks: recurrent/recursive neural networks (RNNs) and convolutional neural networks (CNNs). Among these, RNNs can *directly* represent essential linguistic structures, i.e., word sequences (Hammerton, 2001) and constituent/dependency trees (Tai et al., 2015). Despite this representation ability, for relation classification tasks, the previously reported performance using long short-term memory (LSTM) based RNNs (Xu et al., 2015b; Li et al., 2015) is worse than one using CNNs (dos Santos et al., 2015). These previous LSTM-based systems mostly include limited linguistic structures and neural architectures, and do not model entities and relations jointly. We are able to achieve improvements over state-of-the-art models via end-to-end modeling of entities and relations based on richer LSTM-RNN architectures that incorporate complementary linguistic structures.

Word sequence and tree structure are known to be complementary information for extracting relations. For instance, dependencies between words

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1105–1116,
Berlin, Germany, August 7-12, 2016. ©2016 Association for Computational Linguistics

are not enough to predict that *source* and *U.S.* have an *ORG-AFF* relation in the sentence *"This is ...", one U.S. source said*, and the context word *said* is required for this prediction. Many traditional, feature-based relation classification models extract features from both sequences and parse trees (Zhou et al., 2005). However, previous RNN-based models focus on only one of these linguistic structures (Socher et al., 2012).

We present a novel end-to-end model to extract relations between entities on both word sequence and dependency tree structures. Our model allows joint modeling of entities and relations in a single model by using both bidirectional sequential (left-to-right and right-to-left) and bidirectional tree-structured (bottom-up and top-down) LSTM-RNNs. Our model first detects entities and then extracts relations between the detected entities using a single incrementally-decoded NN structure, and the NN parameters are jointly updated using both entity and relation labels. Unlike traditional incremental end-to-end relation extraction models, our model further incorporates two enhancements into training: entity pretraining, which pretrains the entity model, and scheduled sampling (Bengio et al., 2015), which replaces (unreliable) predicted labels with gold labels in a certain probability. These enhancements alleviate the problem of low-performance entity detection in early stages of training, as well as allow entity information to further help downstream relation classification.

On end-to-end relation extraction, we improve over the state-of-the-art feature-based model, with 12.1% (ACE2005) and 5.7% (ACE2004) relative error reductions in F1-score. On nominal relation classification (SemEval-2010 Task 8), our model compares favorably to the state-of-the-art CNN-based model in F1-score. Finally, we also ablate and compare our various model components, which leads to some key findings (both positive and negative) about the contribution and effectiveness of different RNN structures, input dependency relation structures, different parsing models, external resources, and joint learning settings.

2 Related Work

LSTM-RNNs have been widely used for sequential labeling, such as clause identification (Hammerton, 2001), phonetic labeling (Graves and Schmidhuber, 2005), and NER (Hammerton, 2003). Recently, Huang et al. (2015) showed that building a conditional random field (CRF) layer on top of bidirectional LSTM-RNNs performs comparably to the state-of-the-art methods in the part-of-speech (POS) tagging, chunking, and NER.

For relation classification, in addition to traditional feature/kernel-based approaches (Zelenko et al., 2003; Bunescu and Mooney, 2005), several neural models have been proposed in the SemEval-2010 Task 8 (Hendrickx et al., 2010), including embedding-based models (Hashimoto et al., 2015), CNN-based models (dos Santos et al., 2015), and RNN-based models (Socher et al., 2012). Recently, Xu et al. (2015a) and Xu et al. (2015b) showed that the shortest dependency paths between relation arguments, which were used in feature/kernel-based systems (Bunescu and Mooney, 2005), are also useful in NN-based models. Xu et al. (2015b) also showed that LSTM-RNNs are useful for relation classification, but the performance was worse than CNN-based models. Li et al. (2015) compared separate sequence-based and tree-structured LSTM-RNNs on relation classification, using basic RNN model structures.

Research on tree-structured LSTM-RNNs (Tai et al., 2015) fixes the direction of information propagation from bottom to top, and also cannot handle an arbitrary number of typed children as in a typed dependency tree. Furthermore, no RNN-based relation classification model simultaneously uses word sequence and dependency tree information. We propose several such novel model structures and training settings, investigating the simultaneous use of bidirectional sequential and bidirectional tree-structured LSTM-RNNs to jointly capture linear and dependency context for end-to-end extraction of relations between entities.

As for end-to-end (joint) extraction of relations between entities, all existing models are feature-based systems (and no NN-based model has been proposed). Such models include structured prediction (Li and Ji, 2014; Miwa and Sasaki, 2014), integer linear programming (Roth and Yih, 2007; Yang and Cardie, 2013), card-pyramid parsing (Kate and Mooney, 2010), and global probabilistic graphical models (Yu and Lam, 2010; Singh et al., 2013). Among these, structured prediction methods are state-of-the-art on several corpora. We present an improved, NN-based alternative for the end-to-end relation extraction.

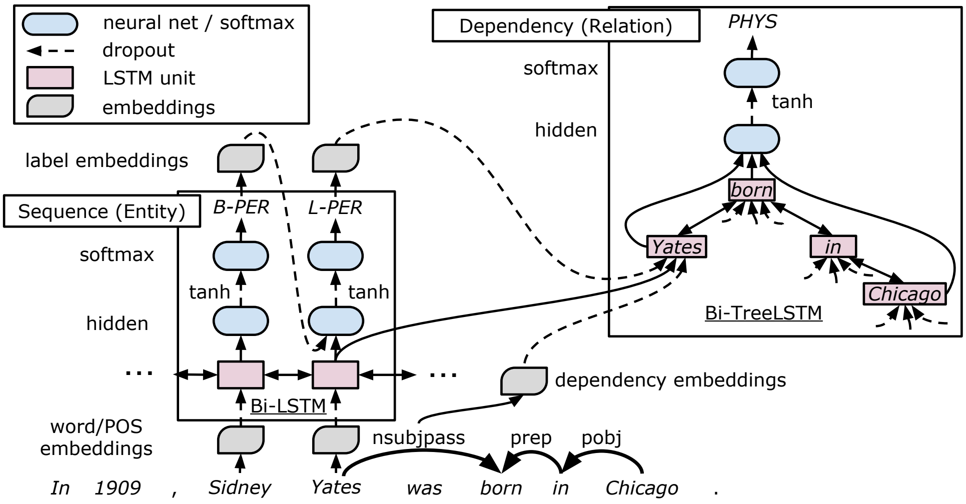

Fig. 1: Our incrementally-decoded end-to-end relation extraction model, with bidirectional sequential and bidirectional tree-structured LSTM-RNNs.

3 Model

We design our model with LSTM-RNNs that represent both word sequences and dependency tree structures, and perform end-to-end extraction of relations between entities on top of these RNNs. Fig. 1 illustrates the overview of the model. The model mainly consists of three representation layers: a word embeddings layer (embedding layer), a word sequence based LSTM-RNN layer (sequence layer), and finally a dependency subtree based LSTM-RNN layer (dependency layer). During decoding, we build greedy, left-to-right entity detection on the sequence layer and realize relation classification on the dependency layers, where each subtree based LSTM-RNN corresponds to a relation candidate between two detected entities. After decoding the entire model structure, we update the parameters simultaneously via backpropagation through time (BPTT) (Werbos, 1990). The dependency layers are stacked on the sequence layer, so the embedding and sequence layers are **shared** by both entity detection and relation classification, and the shared parameters are affected by both entity and relation labels.

3.1 Embedding Layer

The embedding layer handles embedding representations. n_w, n_p, n_d and n_e-dimensional vectors $v^{(w)}$, $v^{(p)}$, $v^{(d)}$ and $v^{(e)}$ are embedded to words, part-of-speech (POS) tags, dependency types, and entity labels, respectively.

3.2 Sequence Layer

The sequence layer represents words in a linear sequence using the representations from the embedding layer. This layer represents sentential context information and maintains entities, as shown in bottom-left part of Fig. 1.

We represent the word sequence in a sentence with bidirectional LSTM-RNNs (Graves et al., 2013). The LSTM unit at t-th word consists of a collection of n_{l_s}-dimensional vectors: an input gate i_t, a forget gate f_t, an output gate o_t, a memory cell c_t, and a hidden state h_t. The unit receives an n-dimensional input vector x_t, the previous hidden state h_{t-1}, and the memory cell c_{t-1}, and calculates the new vectors using the following equations:

$$\begin{aligned} i_t &= \sigma\left(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}\right), \quad (1)\\ f_t &= \sigma\left(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}\right),\\ o_t &= \sigma\left(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}\right),\\ u_t &= \tanh\left(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)}\right),\\ c_t &= i_t\odot u_t + f_t\odot c_{t-1},\\ h_t &= o_t\odot \tanh(c_t), \end{aligned}$$

where σ denotes the logistic function, $\odot$ denotes element-wise multiplication, W and U are weight matrices, and b are bias vectors. The LSTM unit at t-th word receives the concatenation of word and POS embeddings as its input vector: $x_t = \left[v_t^{(w)}; v_t^{(p)}\right]$. We also concatenate the hidden state vectors of the two directions' LSTM units corresponding to each word (denoted as $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$) as its output vector, $s_t = \left[\overrightarrow{h_t}; \overleftarrow{h_t}\right]$, and pass it to the subsequent layers.

3.3 Entity Detection

We treat entity detection as a sequence labeling task. We assign an entity tag to each word using a commonly used encoding scheme BILOU (Begin, Inside, Last, Outside, Unit) (Ratinov and Roth, 2009), where each entity tag represents the entity type and the position of a word in the entity. For example, in Fig. 1, we assign *B-PER* and *L-PER* (which denote the beginning and last words of a person entity type, respectively) to each word in *Sidney Yates* to represent this phrase as a *PER* (person) entity type.

We perform entity detection on top of the sequence layer. We employ a two-layered NN with an n_{h_e}-dimensional hidden layer $h^{(e)}$ and a softmax output layer for entity detection.

$$\begin{aligned} h_t^{(e)} &= \tanh\left(W^{(e_h)}[s_t; v_{t-1}^{(e)}] + b^{(e_h)}\right) \quad (2) \\ y_t &= \text{softmax}\left(W^{(e_y)} h_t^{(e)} + b^{(e_y)}\right) \quad (3) \end{aligned}$$

Here, W are weight matrices and b are bias vectors.

We assign entity labels to words in a greedy, left-to-right manner.[1] During this decoding, we use the predicted label of a word to predict the label of the next word so as to take label dependencies into account. The NN above receives the concatenation of its corresponding outputs in the sequence layer and the label embedding for its previous word (Fig. 1).

3.4 Dependency Layer

The dependency layer represents a relation between a pair of two target words (corresponding to a relation candidate in relation classification) in the dependency tree, and is in charge of relation-specific representations, as is shown in top-right part of Fig. 1. This layer mainly focuses on the *shortest path* between a pair of target words in the dependency tree (i.e., the path between the least common node and the two target words) since these paths are shown to be effective in relation classification (Xu et al., 2015a). For example, we show the shortest path between *Yates* and *Chicago* in the bottom of Fig. 1, and this path well captures the key phrase of their relation, i.e., *born in*.

We employ bidirectional tree-structured LSTM-RNNs (i.e., bottom-up and top-down) to represent a relation candidate by capturing the dependency structure around the target word pair. This bidirectional structure propagates to each node not only the information from the leaves but also information from the root. This is especially important for relation classification, which makes use of argument nodes near the bottom of the tree, and our top-down LSTM-RNN sends information from the top of the tree to such near-leaf nodes (unlike in standard bottom-up LSTM-RNNs).[2] Note that the two variants of tree-structured LSTM-RNNs by Tai et al. (2015) are not able to represent our target structures which have a variable number of typed children: the Child-Sum Tree-LSTM does not deal with types and the N-ary Tree assumes a fixed number of children. We thus propose a new variant of tree-structured LSTM-RNN that shares weight matrices Us for same-type children and also allows variable number of children. For this variant, we calculate n_{l_t}-dimensional vectors in the LSTM unit at t-th node with $C(t)$ children using following equations:

$$\begin{aligned} i_t &= \sigma\left(W^{(i)} x_t + \sum_{l \in C(t)} U_{m(l)}^{(i)} h_{tl} + b^{(i)}\right), \quad (4) \\ f_{tk} &= \sigma\left(W^{(f)} x_t + \sum_{l \in C(t)} U_{m(k)m(l)}^{(f)} h_{tl} + b^{(f)}\right), \\ o_t &= \sigma\left(W^{(o)} x_t + \sum_{l \in C(t)} U_{m(l)}^{(o)} h_{tl} + b^{(o)}\right), \\ u_t &= \tanh\left(W^{(u)} x_t + \sum_{l \in C(t)} U_{m(l)}^{(u)} h_{tl} + b^{(u)}\right), \\ c_t &= i_t \odot u_t + \sum_{l \in C(t)} f_{tl} \odot c_{tl}, \\ h_t &= o_t \odot \tanh(c_t), \end{aligned}$$

where $m(\cdot)$ is a type mapping function.

To investigate appropriate structures to represent relations between two target word pairs, we experiment with three structure options. We primarily employ the shortest path structure (**SP-Tree**), which captures the core dependency path between a target word pair and is widely used in relation classification models, e.g., (Bunescu and

[1] We also tried beam search but this did not show improvements in initial experiments.

[2] We also tried to use one LSTM-RNN by connecting the root (Paulus et al., 2014), but preparing two LSTM-RNNs showed slightly better performance in our initial experiments.

Mooney, 2005; Xu et al., 2015a). We also try two other dependency structures: **SubTree** and **FullTree**. SubTree is the subtree under the lowest common ancestor of the target word pair. This provides additional modifier information to the path and the word pair in SPTree. FullTree is the full dependency tree. This captures context from the entire sentence. While we use one node type for SPTree, we define two node types for SubTree and FullTree, i.e., one for nodes on shortest paths and one for all other nodes. We use the type mapping function $m(\cdot)$ to distinguish these two nodes types.

3.5 Stacking Sequence and Dependency Layers

We stack the dependency layers (corresponding to relation candidates) on top of the sequence layer to incorporate both word sequence and dependency tree structure information into the output. The dependency-layer LSTM unit at the t-th word receives as input $x_t = \left[s_t; v_t^{(d)}; v_t^{(e)}\right]$, i.e., the concatenation of its corresponding hidden state vectors s_t in the sequence layer, dependency type embedding $v_t^{(d)}$ (denotes the type of dependency to the parent[3]), and label embedding $v_t^{(e)}$ (corresponds to the predicted entity label).

3.6 Relation Classification

We incrementally build relation candidates using all possible combinations of the last words of detected entities, i.e., words with L or U labels in the BILOU scheme, during decoding. For instance, in Fig. 1, we build a relation candidate using *Yates* with an *L-PER* label and *Chicago* with an *U-LOC* label. For each relation candidate, we realize the dependency layer d_p (described above) corresponding to the path between the word pair p in the relation candidate, and the NN receives a relation candidate vector constructed from the output of the dependency tree layer, and predicts its relation label. We treat a pair as a negative relation when the detected entities are wrong or when the pair has no relation. We represent relation labels by type and direction, except for negative relations that have no direction.

The relation candidate vector is constructed as the concatenation $d_p = [\uparrow h_{p_A}; \downarrow h_{p_1}; \downarrow h_{p_2}]$, where $\uparrow h_{p_A}$ is the hidden state vector of the top LSTM unit in the bottom-up LSTM-RNN (representing the lowest common ancestor of the target word pair p), and $\downarrow h_{p_1}$, $\downarrow h_{p_2}$ are the hidden state vectors of the two LSTM units representing the first and second target words in the top-down LSTM-RNN.[4] All the corresponding arrows are shown in Fig. 1.

Similarly to the entity detection, we employ a two-layered NN with an n_{h_r}-dimensional hidden layer $h^{(r)}$ and a softmax output layer (with weight matrices W, bias vectors b).

$$h_p^{(r)} = \tanh\left(W^{(r_h)} d_p + b^{(r_h)}\right) \quad (5)$$

$$y_p = \text{softmax}\left(W^{(r_y)} h_t^{(r)} + b^{(r_y)}\right) \quad (6)$$

We construct the input d_p for relation classification from tree-structured LSTM-RNNs stacked on sequential LSTM-RNNs, so the contribution of sequence layer to the input is indirect. Furthermore, our model uses words for representing entities, so it cannot fully use the entity information. To alleviate these problems, we directly concatenate the average of hidden state vectors for each entity from the sequence layer to the input d_p to relation classification, i.e., $d'_p = \left[d_p; \frac{1}{|I_{p_1}|}\sum_{i \in I_{p_1}} s_i; \frac{1}{|I_{p_2}|}\sum_{i \in I_{p_2}} s_i\right]$ (**Pair**), where I_{p_1} and I_{p_2} represent sets of word indices in the first and second entities.[5]

Also, we assign two labels to each word pair in prediction since we consider both left-to-right and right-to-left directions. When the predicted labels are inconsistent, we select the positive and more confident label, similar to Xu et al. (2015a).

3.7 Training

We update the model parameters including weights, biases, and embeddings by BPTT and Adam (Kingma and Ba, 2015) with gradient clipping, parameter averaging, and L2-regularization (we regularize weights W and U, not the bias terms b). We also apply dropout (Srivastava et al., 2014) to the embedding layer and to the final hidden layers for entity detection and relation classification.

We employ two enhancements, **scheduled sampling** (Bengio et al., 2015) and **entity pretraining**, to alleviate the problem of unreliable prediction of entities in the early stage of training,

[3] We use the dependency to the parent since the number of children varies. Dependency types can also be incorporated into $m(\cdot)$, but this did not help in initial experiments.

[4] Note that the order of the target words corresponds to the direction of the relation, not the positions in the sentence.

[5] Note that we do not show this **Pair** in Fig.1 for simplicity.

and to encourage building positive relation instances from the detected entities. In scheduled sampling, we use gold labels as prediction in the probability of ϵ_i that depends on the number of epochs i during training if the gold labels are legal. As for ϵ_i, we choose the inverse sigmoid decay $\epsilon_i = k/(k + \exp(i/k))$, where $k(\geq 1)$ is a hyper-parameter that adjusts how often we use the gold labels as prediction. Entity pretraining is inspired by (Pentina et al., 2015), and we pretrain the entity detection model using the training data before training the entire model parameters.

4 Results and Discussion

4.1 Data and Task Settings

We evaluate on three datasets: ACE05 and ACE04 for end-to-end relation extraction, and SemEval-2010 Task 8 for relation classification. We use the first two datasets as our primary target, and use the last one to thoroughly analyze and ablate the relation classification part of our model.

ACE05 defines 7 coarse-grained entity types and 6 coarse-grained relation types between entities. We use the same data splits, preprocessing, and task settings as Li and Ji (2014). We report the primary micro F1-scores as well as micro precision and recall on both entity and relation extraction to better explain model performance. We treat an entity as correct when its type and the region of its head are correct. We treat a relation as correct when its type and argument entities are correct; we thus treat all non-negative relations on wrong entities as false positives.

ACE04 defines the same 7 coarse-grained entity types as ACE05 (Doddington et al., 2004), but defines 7 coarse-grained relation types. We follow the cross-validation setting of Chan and Roth (2011) and Li and Ji (2014), and the preprocessing and evaluation metrics of ACE05.

SemEval-2010 Task 8 defines 9 relation types between nominals and a tenth type *Other* when two nouns have none of these relations (Hendrickx et al., 2010). We treat this *Other* type as a negative relation type, and no direction is considered. The dataset consists of 8,000 training and 2,717 test sentences, and each sentence is annotated with a relation between two given nominals. We randomly selected 800 sentences from the training set as our development set. We followed the official task setting, and report the official macro-averaged F1-score (Macro-F1) on the 9 relation types.

For more details of the data and task settings, please refer to the supplementary material.

4.2 Experimental Settings

We implemented our model using the *cnn* library.[6] We parsed the texts using the Stanford neural dependency parser[7] (Chen and Manning, 2014) with the original Stanford Dependencies. Based on preliminary tuning, we fixed embedding dimensions n_w to 200, n_p, n_d, n_e to 25, and dimensions of intermediate layers (n_{l_s}, n_{l_t} of LSTM-RNNs and n_{h_e}, n_{h_r} of hidden layers) to 100. We initialized word vectors via word2vec (Mikolov et al., 2013) trained on Wikipedia[8] and randomly initialized all other parameters. We tuned hyper-parameters using development sets for ACE05 and SemEval-2010 Task 8 to achieve high primary (Micro- and Macro-) F1-scores.[9] For ACE04, we directly employed the best parameters for ACE05. The hyper-parameter settings are shown in the supplementary material. For SemEval-2010 Task 8, we also omitted the entity detection and label embeddings since only target nominals are annotated and the task defines no entity types. Our statistical significance results are based on the Approximate Randomization (AR) test (Noreen, 1989).

4.3 End-to-end Relation Extraction Results

Table 1 compares our model with the state-of-the-art feature-based model of Li and Ji (2014)[10] on final test sets, and shows that our model performs better than the state-of-the-art model.

To analyze the contributions and effects of the various components of our end-to-end relation extraction model, we perform ablation tests on the ACE05 development set (Table 2). The performance slightly degraded without scheduled sampling, and the performance significantly degraded when we removed entity pretraining or removed both ($p<0.05$). This is reasonable because the model can only create relation instances when both of the entities are found and, without these enhancements, it may get too late to find some relations. Removing label embeddings did not affect

[6] https://github.com/clab/cnn

[7] http://nlp.stanford.edu/software/stanford-corenlp-full-2015-04-20.zip

[8] https://dumps.wikimedia.org/enwiki/20150901/

[9] We did not tune the precision-recall trade-offs, but doing so can specifically improve precision further.

[10] Other work on ACE is not comparable or performs worse than the model by Li and Ji (2014).

Corpus	Settings	Entity			Relation		
		P	R	F1	P	R	F1
ACE05	Our Model (SPTree)	0.829	**0.839**	**0.834**	0.572	**0.540**	**0.556**
	Li and Ji (2014)	**0.852**	0.769	0.808	**0.654**	0.398	0.495
ACE04	Our Model (SPTree)	0.808	**0.829**	**0.818**	0.487	**0.481**	**0.484**
	Li and Ji (2014)	**0.835**	0.762	0.797	**0.608**	0.361	0.453

Table 1: Comparison with the state-of-the-art on the ACE05 test set and ACE04 dataset.

Settings	Entity			Relation		
	P	R	F1	P	R	F1
Our Model (SPTree)	0.815	0.821	0.818	0.506	0.529	0.518
−Entity pretraining (EP)	0.793	0.798	0.796	0.494	0.491	0.492*
−Scheduled sampling (SS)	0.812	0.818	0.815	0.522	0.490	0.505
−Label embeddings (LE)	0.811	0.821	0.816	0.512	0.499	0.505
−Shared parameters (Shared)	0.796	0.820	0.808	0.541	0.482	0.510
−EP, SS	0.781	0.804	0.792	0.509	0.479	0.494*
−EP, SS, LE, Shared	0.800	0.815	0.807	0.520	0.452	0.484**

Table 2: Ablation tests on the ACE05 development dataset. * denotes significance at p<0.05, ** denotes p<0.01.

Settings	Entity			Relation		
	P	R	F1	P	R	F1
SPTree	0.815	0.821	0.818	0.506	0.529	0.518
SubTree	0.812	0.818	0.815	0.525	0.506	0.515
FullTree	0.806	0.816	0.811	0.536	0.507	0.521
SubTree (-SP)	0.803	0.816	0.810	0.533	0.495	0.514
FullTree (-SP)	0.804	0.817	0.811	0.517	0.470	0.492*
Child-Sum	0.806	0.819	0.8122	0.514	0.499	0.506
SPSeq	0.801	0.813	0.807	0.500	0.523	0.511
SPXu	0.809	0.818	0.813	0.494	0.522	0.508

Table 3: Comparison of LSTM-RNN structures on the ACE05 development dataset.

the entity detection performance, but this degraded the recall in relation classification. This indicates that entity label information is helpful in detecting relations.

We also show the performance without sharing parameters, i.e., embedding and sequence layers, for detecting entities and relations (−**Shared parameters**); we first train the entity detection model, detect entities with the model, and build a *separate* relation extraction model using the detected entities, i.e., without entity detection. This setting can be regarded as a pipeline model since two separate models are trained sequentially. Without the shared parameters, both the performance in entity detection and relation classification drops slightly, although the differences are not significant. When we removed all the enhancements, i.e., scheduled sampling, entity pretraining, label embedding, and shared parameters, the performance is significantly worse than SPTree (p<0.01), showing that these enhancements provide complementary benefits to end-to-end relation extraction.

Next, we show the performance with different LSTM-RNN structures in Table 3. We first compare the three input dependency structures (SPTree, SubTree, FullTree) for tree-structured LSTM-RNNs. Performances on these three structures are almost same when we distinguish the nodes in the shortest paths from other nodes, but when we do not distinguish them (-SP), the information outside of the shortest path, i.e., FullTree (-SP), significantly hurts performance ($p<0.05$). We then compare our tree-structured LSTM-RNN (SPTree) with the Child-Sum tree-structured LSTM-RNN on the shortest path of Tai et al. (2015). Child-Sum performs worse than our SPTree model, but not with as big of a decrease as above. This may be because the difference in the models appears only on nodes that have multiple children and all the nodes except for the least common node have one child.

We finally show results with two counterparts of sequence-based LSTM-RNNs using the shortest path (last two rows in Table 3). **SPSeq** is a bidirectional LSTM-RNN on the shortest path. The LSTM unit receives input from the sequence layer concatenated with embeddings for the surrounding dependency types and directions. We concatenate the outputs of the two RNNs for the relation candidate. **SPXu** is our adaptation of the shortest path LSTM-RNN proposed by Xu et al. (2015b) to match our sequence-layer based model.[11] This has two LSTM-RNNs for the left and right subpaths of the shortest path. We first calculate the max pooling of the LSTM units for each of these two RNNs, and then concatenate the outputs of the pooling for the relation candidate. The comparison with these sequence-based LSTM-RNNs indicates that a tree-structured LSTM-RNN is comparable to sequence-based ones in representing shortest paths.

Overall, the performance comparison of the LSTM-RNN structures in Table 3 show that for end-to-end relation extraction, *selecting the appropriate tree structure representation of the input (i.e., the shortest path) is more important than the choice of the LSTM-RNN structure on that input (i.e., sequential versus tree-based).*

4.4 Relation Classification Analysis Results

To thoroughly analyze the relation classification part alone, e.g., comparing different LSTM structures, architecture components such as hidden layers and input information, and classification task settings, we use the SemEval-2010 Task 8. This dataset, often used to evaluate NN models for relation classification, annotates only relation-related nominals (unlike ACE datasets), so we can focus cleanly on the relation classification part.

Settings	Macro-F1
No External Knowledge Resources	
Our Model (SPTree)	**0.844**
dos Santos et al. (2015)	0.841
Xu et al. (2015a)	0.840
+WordNet	
Our Model (SPTree + WordNet)	0.855
Xu et al. (2015a)	**0.856**
Xu et al. (2015b)	0.837

Table 4: Comparison with state-of-the-art models on SemEval-2010 Task 8 test-set.

Settings	Macro-F1
SPTree	0.851
SubTree	0.839
FullTree	0.829∗
SubTree (-SP)	0.840
FullTree (-SP)	0.828∗
Child-Sum	0.838
SPSeq	0.844
SPXu	0.847

Table 5: Comparison of LSTM-RNN structures on SemEval-2010 Task 8 development set.

We first report official test set results in Table 4. Our novel LSTM-RNN model is comparable to both the state-of-the-art CNN-based models on this task with or without external sources, i.e., WordNet, unlike the previous best LSTM-RNN model (Xu et al., 2015b).[12]

Next, we compare different LSTM-RNN structures in Table 5. As for the three input dependency structures (SPTree, SubTree, FullTree), FullTree performs significantly worse than other structures regardless of whether or not we distinguish the nodes in the shortest paths from the other nodes, which hints that the information outside of the shortest path significantly hurts the performance ($p<0.05$). We also compare our tree-structured LSTM-RNN (SPTree) with sequence-based LSTM-RNNs (SPSeq and SPXu) and tree-structured LSTM-RNNs (Child-Sum). All these LSTM-RNNs perform slightly worse than our SP-

[11] This is different from the original one in that we use the sequence layer and we concatenate the embeddings for the input, while the original one prepared individual LSTM-RNNs for different inputs and concatenated their outputs.

[12] When incorporating WordNet information into our model, we prepared embeddings for WordNet hypernyms extracted by SuperSenseTagger (Ciaramita and Altun, 2006) and concatenated the embeddings to the input vector (the concatenation of word and POS embeddings) of the sequence LSTM. We tuned the dimension of the WordNet embeddings and set it to 15 using the development dataset.

Settings	Macro-F1
SPTree	0.851
−Hidden layer	0.839
−Sequence layer	0.840
−Pair	0.844
−Pair, Sequence layer	0.827∗
Stanford PCFG	0.844
+WordNet	0.854
Left-to-right candidates	0.843
Neg. sampling (Xu et al., 2015a)	0.848

Table 6: Model setting ablations on SemEval-2010 development set.

Tree model, but the differences are small.

Overall, for relation classification, although the performance comparison of the LSTM-RNN structures in Table 5 produces different results on FullTree as compared to the results on ACE05 in Table 3, the trend still holds that selecting the appropriate tree structure representation of the input is more important than the choice of the LSTM-RNN structure on that input.

Finally, Table 6 summarizes the contribution of several model components and training settings on SemEval relation classification. We first remove the hidden layer by directly connecting the LSTM-RNN layers to the softmax layers, and found that this slightly degraded performance, but the difference was small. We then skip the sequence layer and directly use the word and POS embeddings for the dependency layer. Removing the sequence layer[13] or entity-related information from the sequence layer (−Pair) slightly degraded performance, and, on removing both, the performance dropped significantly ($p<0.05$). This indicates that the sequence layer is necessary but the last words of nominals are almost enough for expressing the relations in this task.

When we replace the Stanford neural dependency parser with the Stanford lexicalized PCFG parser (Stanford PCFG), the performance slightly dropped, but the difference was small. This indicates that the selection of parsing models is not critical. We also included WordNet, and this slightly improved the performance (+WordNet), but the difference was small. Lastly, for the generation of relation candidates, generating only left-to-right candidates slightly degraded the performance, but the difference was small and hence the creation of right-to-left candidates was not critical. Treating the inverse relation candidate as a negative instance (Negative sampling) also performed comparably to other generation methods in our model (unlike Xu et al. (2015a), which showed a significance improvement over generating only left-to-right candidates).

5 Conclusion

We presented a novel end-to-end relation extraction model that represents both word sequence and dependency tree structures by using bidirectional sequential and bidirectional tree-structured LSTM-RNNs. This allowed us to represent both entities and relations in a single model, achieving gains over the state-of-the-art, feature-based system on end-to-end relation extraction (ACE04 and ACE05), and showing favorably comparable performance to recent state-of-the-art CNN-based models on nominal relation classification (SemEval-2010 Task 8).

Our evaluation and ablation led to three key findings. First, the use of both word sequence and dependency tree structures is effective. Second, training with the shared parameters improves relation extraction accuracy, especially when employed with entity pretraining, scheduled sampling, and label embeddings. Finally, the shortest path, which has been widely used in relation classification, is also appropriate for representing tree structures in neural LSTM models.

Acknowledgments

We thank Qi Li, Kevin Gimpel, and the anonymous reviewers for dataset details and helpful discussions.

References

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. *arXiv preprint arXiv:1506.03099*.

Razvan C Bunescu and Raymond 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. ACL.

[13] Note that this setting still uses some sequence layer information since it uses the entity-related information (Pair).

Yee Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 551–560, Portland, Oregon, USA, June. ACL.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar, October. ACL.

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, Sydney, Australia, July. ACL.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ace) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC-2004)*, Lisbon, Portugal, May. European Language Resources Association (ELRA).

Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 626–634, Beijing, China, July. ACL.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

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

James Hammerton. 2001. Clause identification with long short-term memory. In *Proceedings of the 2001 workshop on Computational Natural Language Learning-Volume 7*, page 22. ACL.

James Hammerton. 2003. Named entity recognition with long short-term memory. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 172–175. ACL.

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. 2015. Task-oriented learning of word embeddings for semantic relation classification. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 268–278, Beijing, China, July. ACL.

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 the 5th International Workshop on Semantic Evaluation*, pages 33–38, Uppsala, Sweden, July. ACL.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Rohit J. Kate and Raymond Mooney. 2010. Joint entity and relation extraction using card-pyramid parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 203–212, Uppsala, Sweden, July. ACL.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR 2015*, San Diego, CA, May.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 402–412, Baltimore, Maryland, June. ACL.

Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard Hovy. 2015. When are tree structures necessary for deep learning of representations? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2304–2314, Lisbon, Portugal, September. ACL.

Wei Lu and Dan Roth. 2015. Joint mention extraction and classification with mention hypergraphs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 857–867, Lisbon, Portugal, September. ACL.

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.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1858–1869, Doha, Qatar, October. ACL.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses : An Introduction*. Wiley-Interscience, April.

Romain Paulus, Richard Socher, and Christopher D Manning. 2014. Global belief recursive neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2888–2896. Curran Associates, Inc.

Anastasia Pentina, Viktoriia Sharmanska, and Christoph H. Lampert. 2015. Curriculum learning of multiple tasks. In *IEEE Conference on Computer Vision and Pattern Recognition CVPR*, pages 5492–5500, Boston, MA, USA, June.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado, June. ACL.

Dan Roth and Wen-Tau Yih, 2007. *Global Inference for Entity and Relation Identification via a Linear Programming Formulation*. MIT Press.

Sameer Singh, Sebastian Riedel, Brian Martin, Jiaping Zheng, and Andrew McCallum. 2013. Joint inference of entities, relations, and coreference. In *Proceedings of the 2013 workshop on Automated knowledge base construction*, pages 1–6. ACM.

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, Jeju Island, Korea, July. ACL.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China, July. ACL.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

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

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 the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1785–1794, Lisbon, Portugal, September. ACL.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1640–1649, Sofia, Bulgaria, August. ACL.

Xiaofeng Yu and Wai Lam. 2010. Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach. In *Coling 2010: Posters*, pages 1399–1407, Beijing, China, August. Coling 2010 Organizing Committee.

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

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 427–434, Ann Arbor, Michigan, June. ACL.

A Supplemental Material

A.1 Data and Task Settings

ACE05 defines 7 coarse-grained entity types: Facility (*FAC*), Geo-Political Entities (*GPE*), Location (*LOC*), Organization (*ORG*), Person (*PER*), Vehicle (*VEH*) and Weapon (*WEA*), and 6 coarse-grained relation types between entities: Artifact (*ART*), Gen-Affiliation (*GEN-AFF*), Org-Affiliation (*ORG-AFF*), Part-Whole (*PART-WHOLE*), Person-Social (*PER-SOC*) and Physical (*PHYS*). We removed the *cts*, *un* subsets, and used a 351/80/80 train/dev/test split. We removed duplicated entities and relations, and resolved nested entities. We used head spans for entities. We follow the settings by (Li and Ji, 2014), and we did not use the full mention boundary unlike Lu and Roth (2015). We use *entities* and *relations* to refer to *entity mentions* and *relation mentions* in ACE for brevity.

ACE04 defines the same 7 coarse-grained entity types as ACE05 (Doddington et al., 2004), but defines 7 coarse-grained relation types: *PYS*, *PER-SOC*, Employment / Membership / Subsidiary (*EMP-ORG*), *ART*, *PER/ORG* affiliation (*Other-AFF*), *GPE* affiliation (*GPE-AFF*), and Discourse (*DISC*). We follow the cross-validation setting of Chan and Roth (2011) and Li and Ji (2014). We removed *DISC* and did 5-fold CV on *bnews* and *nwire* subsets (348 documents). We use the same preprocessing and evaluation metrics of ACE05.

SemEval-2010 Task 8 defines 9 relation types between nominals (*Cause-Effect*, *Instrument-Agency*, *Product-Producer*, *Content-Container*, *Entity-Origin*, *Entity-Destination*, *Component-Whole*, *Member-Collection* and *Message-Topic*), and a tenth type *Other* when two nouns have none of these relations (Hendrickx et al., 2010). We treat this *Other* type as a negative relation type, and no direction is considered. The dataset consists of 8,000 training and 2,717 test sentences, and each sentence is annotated with a relation between two given nominals. We randomly selected 800 sentences from the training set as our development set. We followed the official task setting, and report the official macro-averaged F1-score (Macro-F1) on the 9 relation types.

A.2 Hyper-parameter Settings

Here we show the hyper-parameters and the range tried for the hyper-parameters in parentheses. Hyper-parameters include the initial learning rate (5e-3, 2e-3, 1e-3, 5e-4, 2e-4, 1e-4), the regularization parameter (1e-4, 1e-5, 1e-6, 1e-7), dropout probabilities (0.0, 0.1, 0.2, 0.3, 0.4, 0.5), the size of gradient clipping (1, 5, 10, 50, 100), scheduled sampling parameter k (1, 5, 10, 50, 100), the number of epochs for training and entity pretraining ($\leq$ 100), and the embedding dimension of WordNet hypernym (5, 10, 15, 20, 25, 30).