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 stacks bidirectional sequential LSTM-RNNs and bidirectional tree-structured LSTM-RNNs to capture both word sequence and dependency tree substructure information. This allows our model to jointly represent both entities and relations with shared parameters. We further encourage detection of entities during training and use of entity information in relation extraction via curriculum learning and scheduled sampling. Our model improves over the state-of-the-art feature-based model on end-to-end relation extraction, achieving 3.5% and 4.8% relative error reductions in F-score on ACE2004 and ACE2005, respectively. We also show improvements over the state-of-the-art convolutional neural network based model on nominal relation classification (SemEval-2010 Task 8), with 2.5% relative error reduction in F-score.

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 manual 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 fit well to NLP models, since they 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, combined 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 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 entity and relation extraction model based on both bidirectional sequential (left-to-right and right-to-left) and bidirectional tree-structured (bottom-up and top-down) LSTM-RNNs, to represent both word sequence and dependency tree structures, and to allow joint modeling of entities and relations in a single model. Our model also incorporates curriculum learning (Bengio et al., 2009) and scheduled sampling (Bengio et al., 2015) to alleviate the problem of low-performance entity detection in early stages of training, as well as to allow entity information to further help downstream relation extraction. On end-to-end entity and relation extraction, we improve over the state-of-the-art feature-based model, with 3.5% (ACE2004) and 4.8% (ACE2005) relative error reductions in F-score. On nominal relation classification (SemEval-2010 Task 8), our model gets a 2.5% relative error reduction in F-score over the state-of-the-art CNN-based model. Finally, we also ablate and compare our various model components, which leads to some key findings about the contribution and effectiveness of different RNN structures, input dependency relation structures, 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 the 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 neu-

ral 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.

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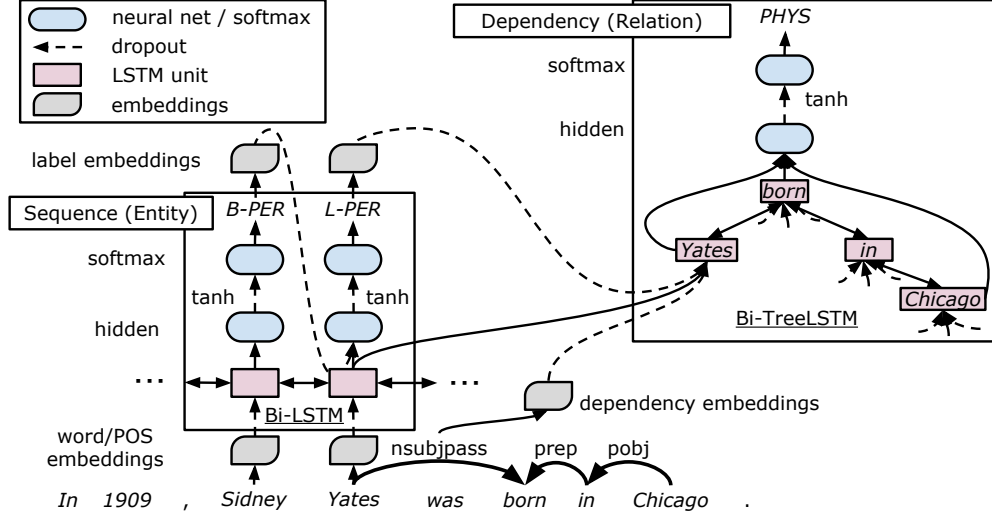
Fig. 1: Our end-to-end relation extraction model, with bidirectional sequential and bidirectional tree-structured LSTM-RNNs.

lations 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, a word sequence based LSTM-RNN layer, and finally a dependency subtree based LSTM-RNN layer.

3.1 Embedding Layer

The embedding layer handles word 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 employ bidirectional LSTM-RNNs (Zaremba and Sutskever, 2014) to represent the word sequence in a sentence. The LSTM unit at t-th word consists of a collection of d-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:

$$i_t = \sigma\left(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}\right), \quad (1)$$

$$f_t = \sigma\left(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}\right),$$

$$o_t = \sigma\left(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}\right),$$

$$u_t = \tanh\left(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)}\right),$$

$$c_t = i_t \odot u_t + f_t \odot c_{t-1},$$

$$h_t = o_t \odot \tanh(c_t),$$

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

output layer for entity detection.

$$h_t^{(e)} = \tanh\left(W^{(e_h)}[s_t; v_{t-1}^{(e)}] + b^{(e_h)}\right) \quad (2)$$

$$y_t = \mathrm{softmax}\left(W^{(e_y)}h_t^{(e)} + b^{(e_y)}\right) \quad (3)$$

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 target words 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 extraction, 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

---

[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.

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 vectors in the LSTM unit at t-th node with $C(t)$ children using following equations:

$$i_t = \sigma\left(W^{(i)}x_t + \sum_{l \in C(t)} U_{m(l)}^{(i)}h_{tl} + b^{(i)}\right), \quad (4)$$

$$f_{tk} = \sigma\left(W^{(f)}x_t + \sum_{l \in C(t)} U_{m(k)m(l)}^{(f)}h_{tl} + b^{(f)}\right),$$

$$o_t = \sigma\left(W^{(o)}x_t + \sum_{l \in C(t)} U_{m(l)}^{(o)}h_{tl} + b^{(o)}\right),$$

$$u_t = \tanh\left(W^{(u)}x_t + \sum_{l \in C(t)} U_{m(l)}^{(u)}h_{tl} + b^{(u)}\right),$$

$$c_t = i_t \odot u_t + \sum_{l \in C(t)} f_{tl} \odot c_{tl},$$

$$h_t = o_t \odot \tanh(c_t),$$

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

To investigate appropriate structures to represent relations between target word pairs, we experiment with three structure options. We primarily employ the shortest path structure (**SPTree**), which captures the core dependency path between a target word pair and is widely used in relation extraction models, e.g., (Bunescu and 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 layer 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). Next, the output relation candidate vector, which is passed to the subsequent relation classification softmax layer, 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.

3.6 Relation Classification

We build relation candidates using the last words of entities, i.e., words with L or U labels in the BILOU scheme. 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, the NN receives the output of the dependency tree layer d_p (described above) corresponding to the path between the word pair p in the candidate, and predicts its relation label.[5] Similarly to the entity detection, we employ a two-layered NN with an 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 both sequence and tree-structured LSTM-RNNs, but 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.[6]

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 back-propagation through time (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 **scheduled sampling** (Bengio et al., 2015) in entity detection. 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.

We also incorporate **curriculum learning** (Bengio et al., 2009), where we pretrain the entity detection model using the training data to encourage building positive relation instances from the detected entities in training.

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.

---

[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]We represent relation labels by type and direction, except for negative relations that have no direction.

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

ACE05 defines 7 coarse-grained entity types[7] and 6 coarse-grained relation types between entities.[8] We use the same data splits and preprocessing as Li and Ji (2014).[9] We report the micro precision, recall, and F-scores 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, and we treat a relation as correct when its type and argument entities are correct.

ACE04 defines the same 7 coarse-grained entity types as ACE05 (Doddington et al., 2004), but defines 7 coarse-grained relation types.[10] We follow the cross-validation setting of Chan and Roth (2011) and Li and Ji (2014)[11], and the preprocessing and evaluation metrics of ACE05.

SemEval-2010 Task 8 defines 9 relation types between nominals[12] and a tenth type *Other* when two nouns have none of these relations (Hendrickx et al., 2010). 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.

4.2 Experimental Settings

We implemented our model using the *cnn* library.[13] We parsed the texts using the Stanford neural dependency parser (Chen and Manning, 2014) with the original Stanford Dependencies. Based on prelim-

inary tuning, we fixed embedding dimensions n_w to 200, n_p, n_d, n_e to 25, and dimensions of intermediate layers (d of LSTM-RNNs and h_e, h_r of hidden layers) to 100. We initialized word vectors via word2vec (Mikolov et al., 2013) trained on Wikipedia[14] and randomly initialized all other parameters. We tuned hyper-parameters using development sets for SemEval-2010 Task 8 and ACE05. For ACE04, we directly employed the best parameters for ACE05. Such 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), and the number of epochs for training and pretraining in curriculum learning ($\leq$ 100).[15] 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) 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 curriculum learning or scheduled sampling, and the performance significantly degraded when we removed both of them (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 the entity detection performance, but this degraded the recall in relation extraction. This indicates that entity label information is helpful in detecting relations.

We also show the performance without sharing parameters for detecting entities and relations (−**Shared parameters**); we first train the entity de-

---

[7]Facility (*FAC*), Geo-Political Entities (*GPE*), Location (*LOC*), Organization (*ORG*), Person (*PER*), Vehicle (*VEH*) and Weapon (*WEA*).

[8]Artifact (*ART*), Gen-Affiliation (*GEN-AFF*), Org-Affiliation (*ORG-AFF*), Part-Whole (*PART-WHOLE*), Person-Social (*PER-SOC*) and Physical (*PHYS*).

[9]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 use *entities* and *relations* to refer to *entity mentions* and *relation mentions* in ACE for brevity.

[10]*PYS*, *PER-SOC*, Employment / Membership / Subsidiary (*EMP-ORG*), *ART*, *PER/ORG* affiliation (*Other-AFF*), *GPE* affiliation (*GPE-AFF*), and Discourse (*DISC*).

[11]We removed *DISC* and did 5-fold CV on *bnews* and *nwire* subsets (348 documents).

[12] *Cause-Effect*, *Instrument-Agency*, *Product-Producer*, *Content-Container*, *Entity-Origin*, *Entity-Destination*, *Component-Whole*, *Member-Collection* and *Message-Topic*

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

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

[15]Here, numbers in parentheses show the range tried for the hyper-parameters. Also, for SemEval-2010 Task 8, we omitted the entity detection layer and label embeddings since only target nominals are annotated and the task defines no entity types.

Corpus	Settings	Entity			Relation		
		P	R	F1	P	R	F1
ACE05	Our Model (SpTree)	**0.855**	**0.812**	**0.833**	**0.658**	**0.429**	**0.519**
	Li and Ji (2014)	0.852	0.769	0.808	0.654	0.398	0.495
ACE04	Our Model (SpTree)	0.833	**0.792**	**0.812**	0.561	**0.408**	**0.472**
	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)	0.841	0.792	0.816	0.605	0.425	0.499
−Curriculum learning (CL)	0.816	0.778	0.797	0.598	0.402	0.481
−Scheduled sampling (SS)	0.833	0.782	0.807	0.606	0.392	0.476
−Label embeddings (LE)	0.844	0.792	0.817	0.648	0.374	0.475
−Shared parameters (Shared)	0.821	0.794	0.807	0.598	0.401	0.480
−CL, SS	0.807	0.766	0.786	0.579	0.390	0.466
−CL, SS, LE, Shared	0.815	0.778	0.796	0.602	0.342	0.436

Table 2: Ablation tests on the ACE05 development dataset.

Settings	Macro-F1
Our Model (SpTree)	**0.845**
dos Santos et al. (2015)	0.841
Xu et al. (2015a)	0.840

Table 3: Comparison with state-of-the-art CNN models (no external knowledge resources) on SemEval-2010 Task 8 test-set.

Settings	Macro-F1
SPTree	0.848
SubTree	0.826
FullTree	0.823
Child-Sum	0.834
SPSeq	0.843
SPXu	0.840

Table 4: Comparison of LSTM-RNN structures on SemEval-2010 Task 8 development set.

tection model, detect entities with the model, and build a separate relation extraction model on the top of the detected entities (note that we cannot use the curriculum learning in this case). Without the shared parameters, both the performance in entity detection and relation extraction drops slightly, although the differences are not significant. When we removed all the enhancements, i.e. scheduled sampling, curriculum learning, label embedding, and shared parameters, the performance is significantly worse than Sp-Tree (p<0.001), showing that these enhancements provide complementary benefits to end-to-end relation extraction.

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 and solely on the relation classification part.

We first report official test set results in Table 3, where our novel LSTM-RNN model is comparable to and better than both the state-of-the-art CNN-based models on this task, whereas the previous best LSTM-RNN model (even when using external WordNet) achieved only **0.837** (Xu et al., 2015b).[16]

Next, we compare different LSTM-RNN structures in Table 4. We first compare the three input dependency structures (SPTree, SubTree, FullTree) for tree-structured LSTM-RNNs, which hints that the information outside of the shortest path signif-

---

[16] We only show previous work's results without external knowledge sources (e.g., WordNet, NER) for fair comparison. With inclusion of such sources, previous work's best reported performance is 0.856 using WordNet (Xu et al., 2015a). The reported best performance with LSTM-RNNs is 0.837, but with using WordNet (Xu et al., 2015b). The authors did not provide results without using external resources.

icantly hurts the performance (p<0.05), even if we distinguish the nodes in the shortest paths from other nodes. We also compare our tree-structured LSTM-RNN (SPTree) with the Child-Sum tree-LSTM 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 further show results with two counterparts of sequence-based LSTM-RNNs using the shortest path. **SPSeq** is bidirectional LSTM-RNN on the shortest path. The LSTM unit receives input from the sequence layer concatenated with embeddings for 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.[17] This has two LSTM-RNNs for the left and right sub-paths 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 4 shows that for relation classification, *selecting the appropriate tree structure representation of the input (i.e., SPTree versus SubTree and FullTree) is more important than the choice of the LSTM-RNN structure on that input (i.e., sequential versus tree-based).*

Table 5 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. We also skip the sequence layer and directly use the word and POS embeddings for the dependency layer.[18] Removing one of hidden

---

[17]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 input and concatenated the outputs of the LSTM-RNNs.

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

Settings	Macro-F1
SPTree	0.848
−Hidden layer	0.835
−Sequence layer	0.841
−Hidden & Sequence layers	0.807
−Pair	0.844
−Pair, Sequence layer	0.827
Left-to-right candidates	0.836
Negative sampling (Xu et al., 2015a)	0.842

Table 5: Model setting ablations on SemEval-2010 dev-set.

and sequence layers slightly degraded performance, and when removing of both layers, the performance significantly dropped (p<0.001). Removing entity-related information from the sequence layer (−Pair) did not affect the performance much, but on removing it together with the sequence layer, 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.

Finally, 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 an end-to-end relation extraction model using bidirectional sequential and bidirectional tree-structured LSTM-RNNs to represent both word sequence and dependency tree structures. 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 over 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

curriculum learning, 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 and Kevin Gimpel for dataset details and helpful discussions.

References

[Bengio et al.2009] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48, New York, NY, USA. ACM.

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

[Bunescu and Mooney2005] 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.

[Chan and Roth2011] 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.

[Chen and Manning2014] 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.

[Doddington et al.2004] 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).

[dos Santos et al.2015] 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.

[Graves and Schmidhuber2005] Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

[Hammerton2001] 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.

[Hammerton2003] 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.

[Hashimoto et al.2015] 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.

[Hendrickx et al.2010] 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.

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

[Kate and Mooney2010] 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.

[Kingma and Ba2015] Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR 2015*, San Diego, CA, May.

[Li and Ji2014] 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.

[Li et al.2015] 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.

[Mikolov et al.2013] 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.

[Miwa and Sasaki2014] 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.

[Nadeau and Sekine2007] David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

[Noreen1989] Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses : An Introduction*. Wiley-Interscience, April.

[Paulus et al.2014] 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.

[Ratinov and Roth2009] 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.

[Roth and Yih2007] Dan Roth and Wen-Tau Yih, 2007. *Global Inference for Entity and Relation Identification via a Linear Programming Formulation*. MIT Press.

[Singh et al.2013] 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.

[Socher et al.2012] 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.

[Srivastava et al.2014] 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.

[Tai et al.2015] 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.

[Xu et al.2015a] 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.

[Xu et al.2015b] 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.

[Yang and Cardie2013] 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.

[Yu and Lam2010] 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.

[Zaremba and Sutskever2014] Wojciech Zaremba and Ilya Sutskever. 2014. Learning to execute. *arXiv preprint arXiv:1410.4615*.

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

[Zhou et al.2005] 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.