Leveraging Linguistic Structures for Named Entity Recognition with Bidirectional Recursive Neural Networks

Peng-Hsuan Li
National Taiwan University
No. 1, Sec. 4, Roosevelt Rd.
Taipei 10617, Taiwan
jacobvsdanniel@gmail.com

Ruo-Ping Dong
National Tsing Hua University
No. 101, Sec. 2, Kuang-Fu Rd.
Hsinchu 30013, Taiwan
dongruoping@gmail.com

Yu-Siang Wang
National Taiwan University
No. 1, Sec. 4, Roosevelt Rd.
Taipei 10617, Taiwan
b03202047@ntu.edu.tw

Ju-Chieh Chou
National Taiwan University
No. 1, Sec. 4, Roosevelt Rd.
Taipei 10617, Taiwan
jjery2243542@gmail.com

Wei-Yun Ma
Academia Sinica
No. 128, Sec. 2, Academia Rd.
Taipei 11529, Taiwan
ma@iis.sinica.edu.tw

Abstract

In this paper, we utilize the linguistic structures of texts to improve named entity recognition by BRNN-CNN, a special bidirectional recursive network attached with a convolutional network. Motivated by the observation that named entities are highly related to linguistic constituents, we propose a constituent-based BRNN-CNN for named entity recognition. In contrast to classical sequential labeling methods, the system first identifies which text chunks are possible named entities by whether they are linguistic constituents. Then it classifies these chunks with a constituency tree structure by recursively propagating syntactic and semantic information to each constituent node. This method surpasses current state-of-the-art on OntoNotes 5.0 with automatically generated parses.

1 Introduction

Named Entity Recognition (NER) can be seen as a combined task of locating named entity chunks of texts and classifying which named entity category a chunk falls into. Traditional approaches label each token in texts as a part of a named entity chunk, e.g. "person_begin", and achieve high performances in several benchmark datasets (Ratinov and Roth, 2009; Passos et al., 2014; Chiu and Nichols, 2016).

Being formulated as a sequential labeling problem, NER systems could be naturally implemented by recurrent neural networks. These networks process a token at a time, taking, for each token, the hidden features of its previous token as well as its raw features to compute its own hidden features. Then they classify each token by these hidden features. With both forward and backward directions, networks learn how to propagate the information of a token sequence to each token. Chiu and Nichols (2016) utilize a variation of recurrent networks, bidirectional LSTM, attached with a CNN, which learns character-level features instead of handcrafting. They accomplish state-of-the-art results on both CoNLL-2003 (Tjong Kim Sang and De Meulder, 2003) and OntoNotes 5.0 (Hovy et al., 2006; Pradhan et al., 2013) datasets.

Classical sequential labeling approaches take little information about phrase structures of sentences. However, according to our analysis, most named entity chunks are actually linguistic constituents, e.g. noun phrases. This motivates us to focus on a constituent-based approach for NER where the NER problem is transformed into a named entity classification task on every node of a

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2647–2652
Copenhagen, Denmark, September 7–11, 2017.

Algorithm 1 Binarization

```
1:  function BINARIZE(node)
2:      n ← node.children.length
3:      if n > 2 then
4:          if HEAD-FINDER(node) ≠ node.children[n] then
5:              newChild ← GROUP(node.children[1..n-1])
6:              node.children ← [newChild, node.children[n]]
7:          else
8:              newChild ← GROUP(node.children[2..n])
9:              node.children ← [node.children[1], newChild]
10:         newChild.pos ← node.pos
11:     for child in node.children do
12:         BINARIZE(child)
```

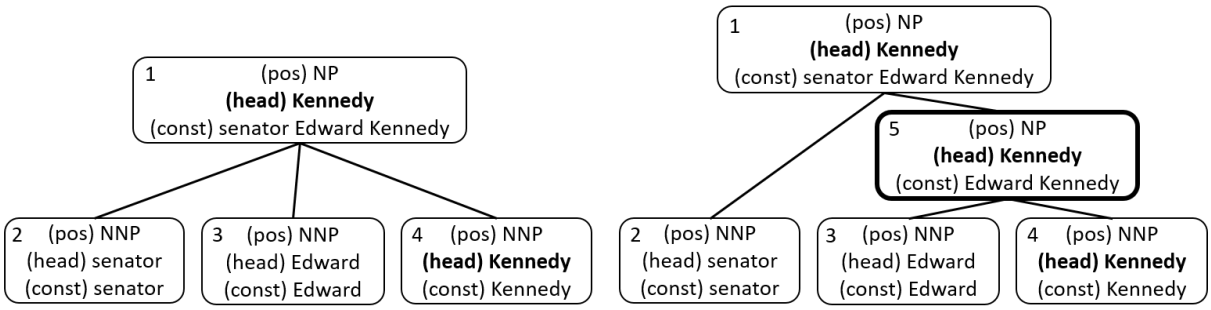

Figure 1: Applying Algorithm 1 to the parse of *senator Edward Kennedy*.

constituency structure.

To classify constituents and take into account their structures, we propose BRNN-CNN, a special bidirectional recursive neural network attached with a convolutional network. For each sentence, a constituency parse where every node represents a meaningful chunk of the sentence, i.e. a constituent, is first generated. Then BRNN-CNN recursively computes hidden state features of every node and classifies each node by these hidden features. To capture structural linguistic information, bidirectional passes are applied so that each constituent sees what it is composed of as well as what is containing it, both in a near-to-far fashion.

Our main contribution is the introduction of a novel constituent-based BRNN-CNN for named entity recognition, which successfully utilizes the linguistic structures of texts by recursive neural networks. We show that it achieves better scores than current state-of-the-art on OntoNotes 5.0, where good parses can be automatically generated. Additionally, we analyze the effects of only considering constituents and the effects of constituency parses.

2 Related Work

Collobert et al. (2011) achieved near state-of-the-art performance on CoNLL-2003 NER with an end-to-end neural network which had minimal feature engineering and external data. Chiu and Nichols (2016) achieved the current state-of-the-art on both CoNLL-2003 and OntoNotes 5.0 NER with a sequential bidirectional LSTM-CNN. They also did extensive studies of additional features such as character type, capitalization, and Senna and DBpedia lexicons.

Finkel and Manning (2009) explored training a parser for an NER-suffixed grammar, jointly tackling parsing and NER. They achieved competitive results on OntoNotes with a CRF-CFG parser.

Recursive neural networks have been successfully applied for parsing and sentiment analysis on Stanford sentiment treebank (Socher et al., 2010, 2013a,b; Tai et al., 2015). Their recursive networks, such as RNTN and Tree-LSTM, do sentiment combinations on phrase structures in a bottom-up fashion, showing the potential of such models in computing semantic compositions.

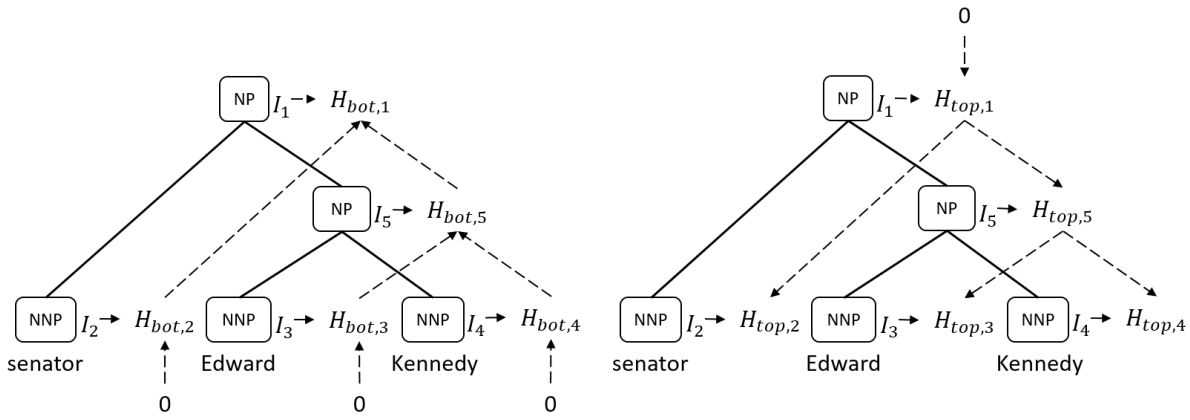

Figure 2: The bottom-up and top-down hidden layers applied to the binarized tree in Figure 1.

3 Method

For each input sentence, our constituent-based BRNN-CNN first extracts features from its constituency parse, then recursively classifies each constituent, and finally resolves conflicting predictions.

3.1 Preparing Constituency Structures

For a sentence and its associated constituency parse, our system first sets three features for each node: a POS, a word, and a head. While constituency tags and words should come readily, semantic head words are determined by a rule-based head finder (Collins, 1999). Additionally, a fourth feature vector is added to each node to utilize lexicon knowledge. The 3-bit vector records if the constituent of a node matches some phrases in each of the three SENNA (Collobert et al., 2011) lexicons of persons, organizations, and locations.

The system then tries to generate more plausible constituents while preserving linguistic structures by applying a binarization process which groups excessive child nodes around the head children. The heuristic is that a head constituent is usually modified by its siblings in a near to far fashion. Algorithm 1 shows the recursive procedure called for the root node of a parse. Figure 1 shows the application of the algorithm to the parse of *senator Edward Kennedy*. With the heuristic that *Edward* modifies the head node *Kennedy* before *senator*. The binarization process successfully adds a new node *Edward Kennedy* that corresponds to a person name.

3.2 Computing Word Embeddings

For each word, our network retrieves one embedding from a trainable lookup table initialized by GloVe (Pennington et al., 2014). However, to capture the morphology information of a word and help dealing with unseen words, the network computes another character-level embedding. Inspired by Kim et al. (2016), the network passes one-hot character vectors through a series of convolutional and highway layers to generate the embedding. These two embeddings are concatenated as the final embedding of a word.

3.3 Computing Hidden Features

Given a constituency parse tree, where every node represents a constituent, our network recursively computes two hidden state features for every node.

First, for each node i with left sibling l and right sibling r, the raw feature vector I_i is formed by concatenating the one-hot POS vectors of i, l, r, the head embeddings of i, l, r, the word embedding of i, the lexicon vector of i, and the mean of word embeddings in the sentence. Then, with the the set of child nodes C and the parent node p, the hidden feature vectors $H_{bot,i}$ and $H_{top,i}$ are computed by 2 hidden layers:

$$H_{bot,i} = ReLU((I_i \| \sum_{c \in C} H_{bot,c}) W_{bot} + b_{bot}) \quad (1)$$

$$H_{top,i} = ReLU((I_i \| H_{top,p}) W_{top} + b_{top}) \quad (2)$$

where Ws are weight matrices, bs are bias vectors, and $ReLU(x) = max(0, x)$. In cases when some needed neighboring nodes do not exist, or when i is a nonterminal and does not have a word, zero

		Validation			Test		
Model	Parser	Precision	Recall	F1	Precision	Recall	F1
BRNN-CNN	gold	86.6	87.0	86.77	88.9	88.9	88.92
BRNN	gold	87.5	86.7	87.11	89.5	88.3	88.91
BRNN-CNN	auto	85.5	84.7	85.08	88.0	86.5	**87.21**
BRNN	auto	86.0	84.7	85.34	88.0	86.2	87.10
Bidirectional Tree-LSTM	auto	85.2	84.5	84.84	87.3	86.2	86.74
Sequential Recurrent NN	-	83.1	83.7	83.38	84.5	84.4	84.40
Finkel and Manning (2009)	gold	-	-	-	84.04	80.86	82.42
Durrett and Klein (2014)	-	-	-	-	85.22	82.89	84.04
Chiu and Nichols (2016)	-	-	-	-	-	-	86.41

Table 1: Experiment results on whole dataset. BRNN is BRNN-CNN deprived of character-level embeddings. Human-labeled parses and automatically generated parses are indicated by gold and auto respectively. Finkel and Manning used gold parses in training a joint model for parsing and NER.

Model	BC	BN	MZ	NW	TC	WB
Test set size (# tokens)	32488	23209	17875	49235	10976	18945
Test set size (# entities)	1697	2184	1163	4696	380	1137
Finkel and Manning (2009)	78.66	87.29	82.45	85.50	67.27	72.56
Durrett and Klein (2014)	78.88	87.39	82.46	87.60	72.68	76.17
Chiu and Nichols (2016)	85.23	89.93	84.45	88.39	72.39	78.38
BRNN-CNN-auto	**85.98**	**90.96**	**84.93**	**89.18**	**73.18**	**80.39**

Table 2: F1 scores on different data sources. From left to right: broadcast conversation, broadcast news, magazine, newswire, telephone conversation, and blogs & newsgroups.

vectors are used as the missing parts of raw or hidden features.

Figure 2 shows the applications of the equations to the binarized tree in Figure 1. The computations are done recursively in two directions. The bottom-up direction computes the semantic composition of the subtree of each node, and the top-down counterpart propagates to that node the linguistic structures which contain the subtree. Together, hidden features of a constituent capture its structural linguistic information.

In addition, each hidden layer can be extended to a deep hidden network. For example, a 2-layer top-down hidden network is given by

$$H_{t\alpha,i} = ReLU((I_i \| H_{t\alpha,p})W_{t\alpha} + b_{t\alpha})$$

$$H_{t\beta,i} = ReLU((H_{t\alpha,i} \| H_{t\beta,p})W_{t\beta} + b_{t\beta})$$

where $t\alpha$ represents the first top-down hidden layer and $t\beta$ represents the second. Our best model is tuned to have 3 layers for both directions.

3.4 Forming Consistent Predictions

Given hidden features for every node, our network computes a probability distribution of named entity classes plus a special non-entity class by an output layer. For each node i with left sibling l and right sibling r, the probability distribution O_i is computed by an output layer:

$$O_i = \sigma((H_i \| H_l \| H_r)W_{out} + b_{out}) \quad (3)$$

where $H_x = H_{bot,x} + H_{top,x}$, $x \in \{i, l, r\}$, and $\sigma(x) = (1 + e^{-x})^{-1}$. If a sibling does not exist, zero vectors are used as its hidden states. Should deep hidden layers be deployed, the last hidden layer is used.

Finally, the system makes predictions for a sentence by collecting the constituents whose most probable classes are named entity classes. However, nodes whose ancestors are already predicted as named entities are ignored to prevent predicting overlapping named entities.

4 Evaluation

We evaluate our system on OntoNotes 5.0 NER (Hovy et al., 2006; Pradhan et al., 2013) and analyze it with several ablation studies. The project sources are publicly available on `https://github.com/jacobvsdanniel/tf_rnn`.

Split	Token	NE	Constituent
Train	1,088,503	81,828	93.3 → 97.3
Validate	147,724	11,066	92.8 → 97.0
Test	152,728	11,257	92.9 → 97.2

Table 3: Dataset statistics for OntoNotes 5.0.

4.1 Training and Tuning

To train the model, we minimize the cross entropy loss of the softmax class probabilities in Equation 3 by the Adam optimizer (Kingma and Ba, 2014). Other details such as hyperparameters are documented in the supplemental materials as well as the public repository.

4.2 OntoNotes 5.0 NER

OntoNotes 5.0 annotates 18 types of named entities for diverse sources of texts. Like other previous work (Durrett and Klein, 2014; Chiu and Nichols, 2016), we use the format and the train-validate-test split provided by CoNLL-2012. In addition, both gold and auto parses are available.

Table 3 shows the dataset statistics. The last column shows the percentages of named entities that correspond to constituents of auto parses before and after binarization.

Table 1 and Table 2 compare our results with others on the whole dataset and different sources respectively. The sample mean, standard deviation, and sample count of BRNN-auto and Chiu and Nichols' model are 87.10, 0.14, 3 and 86.41, 0.22, 10 respectively. By one-tailed Welch's T-test, the former significantly surpasses the latter with 99% confidence level (0.000489 p-value).

4.3 Analysis of the Approach

The training and validation sets contain 1,236,227 tokens and 92,894 named entities, of which 90,371 correspond to some constituents of binarized auto parses. This backs our motivation that more than 97% named entities are linguistic constituents, and 52,729 of them are noun phrases.

Essentially, the constituent-based approach filters out the other 3% named entities that cross constituent boundaries (Figure 3), i.e. 3% loss of recall. We dig into this problem by analyzing a sequential labeling recurrent network (the sixth model in Table 1). The simple model performs reasonably well, but its non-constituent predictions are mostly false positive. In fact, it slightly improves if all non-constituent predictions are removed in post-processing, i.e., the precision gain of focusing on constituents is more significant than the recall loss. This is one advantage of our system over other sequential models, which try to learn and predict non-constituent named entities but do not perform well.

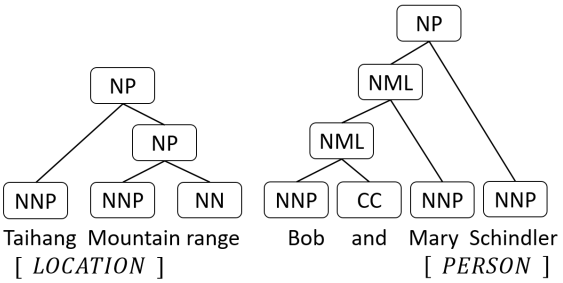

Figure 3: Two sample named entities that cross different branches of syntax parses.

In addition, to analyze the effects of constituency structures, we test our models with different qualities of parses (gold vs. auto in Table 1). The significant F1 differences suggest that structural linguistic information is crucial and can be learned by our model.

5 Conclusion

We have demonstrated a novel constituent-based BRNN-CNN for named entity recognition which successfully utilizes constituency structures and surpasses the current state-of-the-art on OntoNotes 5.0 NER. Instead of propagating information by word orders as normal recurrent networks, the model is able to recursively propagate structural linguistic information to every constituent. Experiments show that when a good parser is available, the approach will be a good alternative to traditional sequential labeling token-based NER.

Named entities that cross constituent boundaries are analyzed and we find out that a naïve sequential labeling model has difficulty predicting them without too many false positives. While avoiding them is one of the strengths of our model, generating more consistent parses to reduce this kind of named entities would be one possible direction for future research.

Acknowledgments

This work is supported by the 2016 Summer Internship Program of IIS, Academia Sinica.

References

Jason P.C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics*, 2:477–490.

Jenny Rose Finkel and Christopher D. Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 326–334. Association for Computational Linguistics.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60. Association for Computational Linguistics.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980 [cs.LG]*.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *Proceedings of the Eighteenth Conference on Computational Language Learning*, pages 78–86.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152.

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 '09, pages 147–155, Stroudsburg, PA, USA. Association for Computational Linguistics.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013a. Parsing with compositional vector grammars. In *Proceedings of the ACL conference*.

Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

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 Processingg*, pages 1556–1566.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.