A Neural Transition-based Model for Nested Mention Recognition

Bailin Wang
University of Massachusetts
Amherst
bailinwang@cs.umass.edu

Wei Lu
Singapore University of Technology
and Design
luwei@sutd.edu.sg

Yu Wang and **Hongxia Jin**
Samsung Research America
{yu.wang1, hongxia.jin}@samsung.com

Abstract

It is common that entity mentions can contain other mentions recursively. This paper introduces a scalable transition-based method to model the nested structure of mentions. We first map a sentence with nested mentions to a designated forest where each mention corresponds to a constituent of the forest. Our shift-reduce based system then learns to construct the forest structure in a bottom-up manner through an action sequence whose maximal length is guaranteed to be three times of the sentence length. Based on Stack-LSTM which is employed to efficiently and effectively represent the states of the system in a continuous space, our system is further incorporated with a character-based component to capture letter-level patterns. Our model achieves the state-of-the-art results on ACE datasets, showing its effectiveness in detecting nested mentions.[1]

1 Introduction

There has been an increasing interest in named entity recognition or more generally recognizing entity mentions[2] (Alex et al., 2007; Finkel and Manning, 2009; Lu and Roth, 2015; Muis and Lu, 2017) that the nested hierarchical structure of entity mentions should be taken into account to better facilitate downstream tasks like question answering (Abney et al., 2000), relation extraction (Mintz et al., 2009; Liu et al., 2017), event extraction (Riedel and McCallum, 2011; Li et al., 2013), and coreference resolution (Soon et al., 2001; Ng and Cardie, 2002; Chang et al., 2013). Practically, the mentions with nested structures frequently exist in news (Doddington et al., 2004) and biomedical documents (Kim et al., 2003). For example in

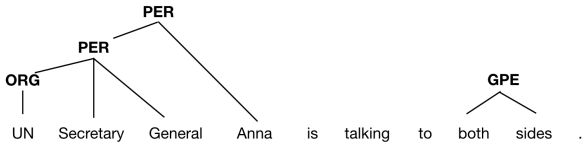

Figure 1: An example sentence of nested mentions represented in the structure of forest. PER:Person, ORG:Organization, GPE:Geo-Political Entity.

Figure 1, "UN Secretary General" of type Person also contains "UN" of type Organization.

Traditional sequence labeling models such as conditional random fields (CRF) (Lafferty et al., 2001) do not allow hierarchical structures between segments, making them incapable to handle such problems. Finkel and Manning (2009) presented a chart-based parsing approach where each sentence with nested mentions is mapped to a rooted constituent tree. The issue of using a chart-based parser is its cubic time complexity in the number of words in the sentence.

To achieve a scalable and effective solution for recognizing nested mentions, we design a transition-based system which is inspired by the recent success of employing transition-based methods for constituent parsing (Zhang and Clark, 2009) and named entity recognition (Lou et al., 2017), especially when they are paired with neural networks (Watanabe and Sumita, 2015). Generally, each sentence with nested mentions is mapped to a forest where each outermost mention forms a tree consisting of its inner mentions. Then our transition-based system learns to construct this forest through a sequence of shift-reduce actions. Figure 1 shows an example of such a forest. In contrast, the tree structure by Finkel and Manning (2009) further uses a root node to connect all tree elements. Our forest representation eliminates the root node so that the number of actions required to

[1] We make our implementation available at https://github.com/berlino/nest-trans-em18.

[2] Mentions are defined as references to entities that could be named, nominal or pronominal (Florian et al., 2004).

construct it can be reduced significantly.

Following (Dyer et al., 2015), we employ Stack-LSTM to represent the system's state, which consists of the states of input, stack and action history, in a continuous space incrementally. The (partially) processed nested mentions in the stack are encoded with recursive neural networks (Socher et al., 2013) where composition functions are used to capture dependencies between nested mentions. Based on the observation that letter-level patterns such as capitalization and prefix can be beneficial in detecting mentions, we incorporate a character-level LSTM to capture such morphological information. Meanwhile, this character-level component can also help deal with the out-of-vocabulary problem of neural models. We conduct experiments in three standard datasets. Our system achieves the state-of-the-art performance on ACE datasets and comparable performance in GENIA dataset.

2 Related Work

Entity mention recognition with nested structures has been explored first with rule-based approaches (Zhang et al., 2004; Zhou et al., 2004; Zhou, 2006) where the authors first detected the innermost mentions and then relied on rule-based post-processing methods to identify outer mentions. McDonald et al. (2005) proposed a structured multi-label model to represent overlapping segments in a sentence. but it came with a cubic time complexity in the number of words. Alex et al. (2007) proposed several ways to combine multiple conditional random fields (CRF) (Lafferty et al., 2001) for such tasks. Their best results were obtained by cascading several CRF models in a specific order while each model is responsible for detecting mentions of a particular type. However, such an approach cannot model nested mentions of the same type, which frequently appear.

Lu and Roth (2015) and Muis and Lu (2017) proposed new representations of mention hypergraph and mention separator to model *overlapping mentions*. However, the nested structure is not guaranteed in such approaches since overlapping structures additionally include the *crossing structures*[3], which rarely exist in practice (Lu and Roth, 2015). Also, their representations did not model the dependencies between nested mentions explicitly, which may limit their performance. In contrast, the chart-based parsing method (Finkel and Manning, 2009) can capture the dependencies between nested mentions with composition rules which allow an outer entity to be influenced by its contained entities. However, their cubic time complexity makes them not scalable to large datasets.

$$\begin{array}{ll} \text{Initial State} & [\phi, 0, \phi] \\ \text{Final State} & [S, n, A] \end{array}$$

$$\textsc{Shift}\quad \frac{[S,\ i,\ A]}{[S|w,\ i{+}1,\ A|\textsc{Shift}]}$$

$$\textsc{Reduce-X}\quad \frac{[S|t_1 t_0,\ i,\ A]}{[S|X,\ i,\ A|\textsc{Reduce-X}]}$$

$$\textsc{Unary-X}\quad \frac{[S|t_0,\ i,\ A]}{[S|X,\ i,\ A|\textsc{Unary-X}]}$$

Figure 2: Deduction rules. $[S, i, A]$ denotes stack, buffer front index and action history respectively.

As neural network based approaches are proven effective in entity or mention recognition (Collobert et al., 2011; Lample et al., 2016; Huang et al., 2015; Chiu and Nichols, 2016; Ma and Hovy, 2016), recent efforts focus on incorporating neural components for recognizing nested mentions. Ju et al. (2018) dynamically stacked multiple LSTM-CRF layers (Lample et al., 2016), detecting mentions in an inside-out manner until no outer entities are extracted. Katiyar and Cardie (2018) used recurrent neural networks to extract features for a hypergraph which encodes all nested mentions based on the BILOU tagging scheme.

3 Model

Specifically, given a sequence of words $\{x_0, x_1, \ldots, x_n\}$, the goal of our system is to output a set of mentions where nested structures are allowed. We use the forest structure to model the nested mentions scattered in a sentence, as shown in Figure 1. The mapping is straightforward: each outermost mention forms a tree where the mention is the root and its contained mentions correspond to constituents of the tree.[4]

3.1 Shift-Reduce System

Our transition-based model is based on the shift-reduce parser for constituency parsing (Watan-

[3]For example, in a four-word sentence ABCD, the phrase ABC and BCD together form a crossing structure.

[4]Note that words that are not contained in any mention each forms a single-node tree.

abe and Sumita, 2015), which adopts (Zhang and Clark, 2009; Sagae and Lavie, 2005). Generally, our system employs a stack to store (partially) processed nested elements. The system's state is defined as $[S, i, A]$ which denotes stack, buffer front index and action history respectively. In each step. an action is applied to change the system's state.

Our system consists of three types of transition actions, which are also summarized in Figure 2:

- SHIFT pushes the next word from buffer to the stack.
- REDUCE-X pops the top two items t_0 and t_1 from the tack and combines them as a new tree element $\{\text{X} \rightarrow t_0 t_1\}$ which is then pushed onto the stack.
- UNARY-X pops the top item t_0 from the stack and constructs a new tree element $\{\text{X} \rightarrow t_0\}$ which is pushed back to the stack.

Since the shift-reduce system assumes unary and binary branching, we binarize the trees in each forest in a left-branching manner. For example, if three consecutive words A, B, C are annotated as Person, we convert it into a binary tree $\{Person \rightarrow \{Person* \rightarrow A, B\}, C\}$ where $Person*$ is a temporary label for $Person$. Hence, the X in reduce- actions will also include such temporary labels.

Note that since most words are not contained in any mention, they are only shifted to the stack and not involved in any reduce- or unary- actions. An example sequence of transitions can be found in Figure 3. Our shift-reduce system is different from previous parsers in terms of the terminal state. 1) It does not require the terminal stack to be a rooted tree. Instead, the final stack should be a forest consisting of multiple nested elements with tree structures. 2) To conveniently determine the ending of our transition process, we add an auxiliary symbol \$ to each sentence. Once it is pushed to the stack, it implies that all deductions of actual words are finished. Since we do not allow unary rules between labels like X1 $\rightarrow$ X2, the length of maximal action sequence is $3n$.[5]

3.2 Action Constraints

To make sure that each action sequence is valid, we need to make some hard constraints on the action to take. For example, reduce- action can only be conducted when there are at least two elements in the stack. Please see the Appendix for the full list of restrictions. Formally, we use $\mathcal{V}(S, i, A)$ to denote the valid actions given the parser state. Let us denote the feature vector for the parser state at time step k as $\mathbf{p}_k$. The distribution of actions is computed as follows:

$$p(z_k \mid \mathbf{p}_k) = \frac{\exp\left(\mathbf{w}_{z_k}^\top \mathbf{p}_k + b_{z_k}\right)}{\sum_{z' \in \mathcal{V}(S,i,A)} \exp\left(\mathbf{w}_{z'}^\top \mathbf{p}_k + b_{z'}\right)} \quad (1)$$

where $\mathbf{w}_z$ is a column weight vector for action z, and b_z is a bias term.

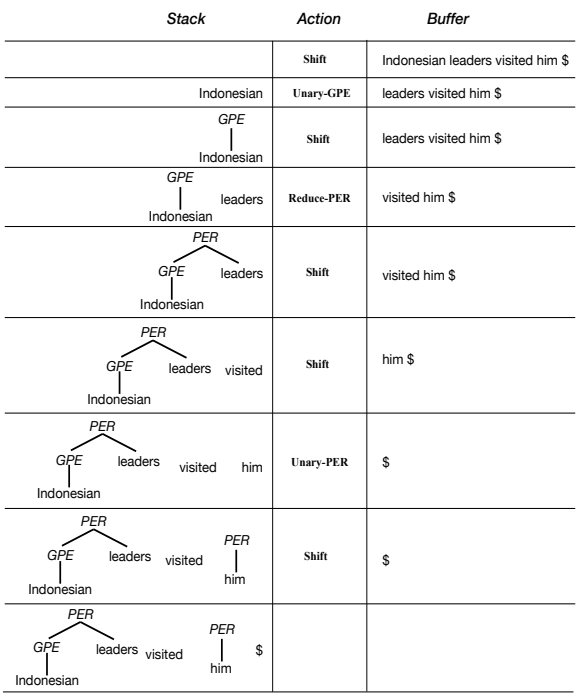

Figure 3: An example sequence of transition actions for the sentence "Indonesian leaders visited him". \$ is the special symbol indicating the termination of transitions. PER:Person, GPE:Geo-Political Entity.

3.3 Neural Transition-based Model

We use neural networks to learn the representation of the parser state, which is $\mathbf{p}_k$ in (1).

Representation of Words

Words are represented by concatenating three vectors:

$$\mathbf{e}_{x_i} = [\mathbf{e}_{w_i}, \mathbf{e}_{p_i}, \mathbf{c}_{w_i}] \quad (2)$$

where $\mathbf{e}_{w_i}$ and $\mathbf{e}_{p_i}$ denote the embeddings for i-th word and its POS tag respectively. $\mathbf{c}_{w_i}$ denotes the representation learned by a character-level model

[5] In this case, each word is shifted (n) and involved in a unary action (n). Then all elements are reduced to a single node ($n-1$). The last action is to shift the symbol \$.

using a bidirectional LSTM. Specifically, for character sequence $s_0, s_1, \ldots, s_n$ in the i-th word, we use the last hidden states of forward and backward LSTM as the character-based representation of this word, as shown below:

$$\mathbf{c}_{w_i} = [\overrightarrow{\text{LSTM}_\text{c}}(s_0, \ldots, s_n), \overleftarrow{\text{LSTM}_\text{c}}(s_0, \ldots, s_n)] \quad (3)$$

Representation of Parser States

Generally, the buffer and action history are encoded using two vanilla LSTMs (Graves and Schmidhuber, 2005). For the stack that involves popping out top elements, we use the Stack-LSTM (Dyer et al., 2015) to efficiently encode it.

Formally, if the unprocessed word sequence in the buffer is $x_i, x_{i+1}, \ldots, x_n$ and action history sequence is $a_0, a_1, \ldots, a_{k-1}$, then we can compute buffer representation $\mathbf{b}_k$ and action history representation $\mathbf{a}_k$ at time step k as follows:

$$\mathbf{b}_k = \overleftarrow{\text{LSTM}_\text{b}}[\mathbf{e}_{x_i}, \ldots, \mathbf{e}_{x_n}] \quad (4)$$

$$\mathbf{a}_k = \overrightarrow{\text{LSTM}_\text{a}}[\mathbf{e}_{a_0}, \ldots, \mathbf{e}_{a_{k-1}}] \quad (5)$$

where each action is also mapped to a distributed representation $\mathbf{e}_a$.[6] For the state of the stack, we also use an LSTM to encode a sequence of tree elements. However, the top elements of the stack are updated frequently. Stack-LSTM provides an efficient implementation that incorporates a stack-pointer.[7] Formally, the state of the stack $\mathbf{b}_k$ at time step k is computed as:

$$\mathbf{s}_k = \text{Stack-LSTM}[\mathbf{h}_{t_m}, \ldots, \mathbf{h}_{t_0}] \quad (6)$$

where $\mathbf{h}_{t_i}$ denotes the representation of the i-th tree element from the top, which can be computed recursively similar to Recursive Neural Network (Socher et al., 2013) as follows:

$$\mathbf{h}_{parent} = \mathbf{W}_{u,l}^\top \mathbf{h}_{child} + \mathbf{b}_{u,l} \quad (7)$$

$$\mathbf{h}_{parent} = \mathbf{W}_{b,l}^\top [\mathbf{h}_{lchild}, \mathbf{h}_{rchild}] + \mathbf{b}_{u,l} \quad (8)$$

where $\mathbf{W}_{u,l}$ and $\mathbf{W}_{b,l}$ denote the weight matrices for unary(u) and binary(b) composition with parent node being label(l). Note that the composition function is distinct for each label l. Recall that the leaf nodes of each tree element are raw words. Instead of representing them with their original embeddings introduced in Section 3.3, we found that concatenating the buffer state in (5) are beneficial during our initial experiments. Formally, when a word x_i is shifted to the stack at time step k, its representation is computed as:

$$\mathbf{h}_{leaf} = \mathbf{W}_{leaf}^\top [\mathbf{e}_{x_i}, \mathbf{b}_k] + \mathbf{b}_{leaf} \quad (9)$$

Finally, the state of the system $\mathbf{p}_k$ is the concatenation of the states of buffer, stack and action history:

$$\mathbf{p}_k = [\mathbf{b}_k, \mathbf{s}_k, \mathbf{a}_k] \quad (10)$$

Training

We employ the greedy strategy to maximize the log-likelihood of the local action classifier in (1). Specifically, let z_{ik} denote the k-th action for the i-th sentence, the loss function with ℓ_2 norm is:

$$\mathcal{L}(\theta) = -\sum_i \sum_k \log p(z_{ik}) + \frac{\lambda}{2}\|\theta\|^2 \quad (11)$$

where λ is the ℓ_2 coefficient.

Models	ACE04	ACE05	GENIA	w/s
Finkel and Manning (2009)	-	-	70.3	38†
Lu and Roth (2015)	62.8	62.5	70.3	454
Muis and Lu (2017)	64.5	63.1	70.8	263
Katiyar and Cardie (2018)	72.7	70.5	73.6	-
Ju et al. (2018) [8]	-	72.2	**74.7**	-
Ours	**73.3**	**73.0**	73.9	1445
- char-level LSTM	72.3	71.9	72.1	1546
- pre-trained embeddings	71.3	71.5	72.0	1452
- dropout layer	71.7	72.0	72.7	1440

Table 1: Main results in terms of F_1 score (%). w/s: # of words decoded per second, number with † is retrieved from the original paper.

4 Experiments

We mainly evaluate our models on the standard ACE-04, ACE-05 (Doddington et al., 2004), and GENIA (Kim et al., 2003) datasets with the same splits used by previous research efforts (Lu and Roth, 2015; Muis and Lu, 2017). In ACE datasets, more than 40% of the mentions form nested structures with some other mention. In GENIA, this number is 18%. Please see Lu and Roth (2015) for the full statistics.

4.1 Setup

Pre-trained embeddings GloVe (Pennington et al., 2014) of dimension 100 are used to initialize the

[6] Note that LSTM_b runs in a right-to-left order such that the output can represent the contextual information of x_i.

[7] Please refer to Dyer et al. (2015) for details.

[8] Note that in ACE2005, Ju et al. (2018) did their experiments with a different split from Lu and Roth (2015) and Muis and Lu (2017) which we follow as our split.

word vectors for all three datasets.[9] The embeddings of POS tags are initialized randomly with dimension 32. The model is trained using Adam (Kingma and Ba, 2014) and a gradient clipping of 3.0. Early stopping is used based on the performance of development sets. Dropout (Srivastava et al., 2014) is used after the input layer. The ℓ_2 coefficient λ is also tuned during development process.

4.2 Results

The main results are reported in Table 1. Our neural transition-based model achieves the best results in ACE datasets and comparable results in GENIA dataset in terms of F_1 measure. We hypothesize that the performance gain of our model compared with other methods is largely due to improved performance on the portions of nested mentions in our datasets. To verify this, we design an experiment to evaluate how well a system can recognize nested mentions.

Handling Nested Mentions

The idea is that we split the test data into two portions: sentences with and without nested mentions. The results of GENIA are listed in Table 2. We can observe that the margin of improvement is more significant in the portion of nested mentions, revealing our model's effectiveness in handling nested mentions. This observation helps explain why our model achieves greater improvement in ACE than in GENIA in Table 1 since the former has much more nested structures than the latter. Moreover, Ju et al. (2018) performs better when it comes to non-nested mentions possibly due to the CRF they used, which globally normalizes each stacked layer.

Decoding Speed

Note that Lu and Roth (2015) and Muis and Lu (2017) also feature linear-time complexity, but with a greater constant factor. To compare the decoding speed, we re-implemented their model with the same platform (PyTorch) and run them on the same machine (CPU: Intel i5 2.7GHz). Our model turns out to be around 3-5 times faster than theirs, showing its scalability.

[9]We also additionally tried using embeddings trained on PubMed for GENIA but the performance was comparable.

	GENIA					
	Nested			Non-Nested		
	P	R	F_1	P	R	F_1
Lu and Roth (2015)	76.3	60.8	67.7	73.1	70.7	71.9
Muis and Lu (2017)	76.5	60.3	67.4	74.8	71.3	73.0
Ju et al. (2018)	79.4	63.6	70.6	78.5	77.5	78.0
Ours	80.3	64.6	71.6	76.8	73.9	75.3

Table 2: Results (%) on different types of sentences on the GENIA dataset.

Ablation Study

To evaluate the contribution of neural components including pre-trained embeddings, the character-level LSTM and dropout layers, we test the performances of ablated models. The results are listed in Table 1. From the performance gap, we can conclude that these components contribute significantly to the effectiveness of our model in all three datasets.

5 Conclusion and Future Work

In this paper, we present a transition-based model for nested mention recognition using a forest representation. Coupled with Stack-LSTM for representing the system's state, our neural model can capture dependencies between nested mentions efficiently. Moreover, the character-based component helps capture letter-level patterns in words. The system achieves the state-of-the-art performance in ACE datasets.

One potential drawback of the system is the greedy training and decoding. We believe that alternatives like beam search and training with exploration (Goldberg and Nivre, 2012) could further boost the performance. Another direction that we plan to work on is to apply this model to recognizing overlapping and entities that involve discontinuous spans (Muis and Lu, 2016) which frequently exist in the biomedical domain.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments. We also thank Meizhi Ju for providing raw predictions and helpful discussions. This work was done after the first author visited Singapore University of Technology and Design. This work is supported by Singapore Ministry of Education Academic Research Fund (AcRF) Tier 2 Project MOE2017-T2-1-156.

References

Steven Abney, Michael Collins, and Amit Singhal. 2000. Answer extraction. In *Proc. of the sixth conference on applied natural language processing*.

Beatrice Alex, Barry Haddow, and Claire Grover. 2007. Recognising nested named entities in biomedical text. In *Proc. of BioNLP*.

Kai-Wei Chang, Rajhans Samdani, and Dan Roth. 2013. A constrained latent variable model for coreference resolution. In *Proc. of EMNLP*.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *TACL*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Proc. of LREC*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. of ACL*.

Jenny Rose Finkel and Christopher D Manning. 2009. Nested named entity recognition. In *Proc. of EMNLP*.

R. Florian, H. Hassan, A. Ittycheriah, H. Jing, N. Kambhatla, X. Luo, N. Nicolov, and S. Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *Proc. of HLT-NAACL*.

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

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*.

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

Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. 2018. A neural layered model for nested named entity recognition. In *Proc. of NAACL-HLT*.

Arzoo Katiyar and Claire Cardie. 2018. Nested named entity recognition revisited. In *Proc. of NAACL-HLT*.

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proc. of NAACL-HLT*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proc. of ACL*.

Liyuan Liu, Xiang Ren, Qi Zhu, Shi Zhi, Huan Gui, Heng Ji, and Jiawei Han. 2017. Heterogeneous supervision for relation extraction: A representation learning approach. In *Proc. of EMNLP*.

Yinxia Lou, Yue Zhang, Tao Qian, Fei Li, Shufeng Xiong, and Donghong Ji. 2017. A transition-based joint model for disease named entity recognition and normalization. *Bioinformatics*.

Wei Lu and Dan Roth. 2015. Joint mention extraction and classification with mention hypergraphs. In *Proc. of EMNLP*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proc. of ACL*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Flexible text segmentation with structured multilabel classification. In *Proc. of HLT-EMNLP*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proc. of ACL-IJCNLP*.

Aldrian Obaja Muis and Wei Lu. 2016. Learning to recognize discontiguous entities. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 75–84.

Aldrian Obaja Muis and Wei Lu. 2017. Labeling gaps between words: Recognizing overlapping mentions with mention separators. In *Proc. of EMNLP*.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proc. of ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proc. of EMNLP*.

S. Riedel and A. McCallum. 2011. Fast and robust joint models for biomedical event extraction. In *Proc. of EMNLP*.

Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *IWPT*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP*.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational linguistics*, 27(4):521–544.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*.

Taro Watanabe and Eiichiro Sumita. 2015. Transition-based neural constituent parsing. In *Proc. of ACL*.

Jie Zhang, Dan Shen, Guodong Zhou, Jian Su, and Chew-Lim Tan. 2004. Enhancing hmm-based biomedical named entity recognition by studying special phenomena. *Journal of Biomedical Informatics*, 37(6):411–422.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the chinese treebank using a global discriminative model. In *Proc. of IWPT*.

Guodong Zhou. 2006. Recognizing names in biomedical texts using mutual information independence model and svm plus sigmoid. *International Journal of Medical Informatics*, 75(6):456–467.

Guodong Zhou, Jie Zhang, Jian Su, Dan Shen, and Chewlim Tan. 2004. Recognizing names in biomedical texts: a machine learning approach. *Bioinformatics*, 20(7):1178–1190.

Appendix

The action constraints are listed as follows:

- The SHIFT action is valid only when the buffer is not empty.
- The UNARY-X actions are valid only when the stack is not empty.
- The REDUCE-X actions are valid only when the stack has two or more elements.
- If the top element of the stack is labeled, then unary actions are not valid. That is, $\{\text{X1} \rightarrow \text{X2}\}$ is not allowed.
- If only one of the top two elements of the stack is temporary, say X*, then among all reduce actions, only REDUCE-X* and REDUCE-X are valid.
- If the top two elements of the stack are both temporary, then all reduce actions are not allowed.
- If one of the elements in the stack is temporary, say X*, which means it is not finished, then last terminal symbol \$ cannot be shifted until it is reduced to X.