Exploiting Mutual Benefits between Syntax and Semantic Roles using Neural Network

Peng Shi[‡*] **Zhiyang Teng**[†] **Yue Zhang**[†]
[†]Singapore University of Technology and Design (SUTD)
[‡]Zhejiang University, China
impavidity@zju.edu.cn
zhiyang_teng@mymail.sutd.edu.sg, yue_zhang@sutd.edu.sg

Abstract

We investigate mutual benefits between syntax and semantic roles using neural network models, by studying a parsing→SRL pipeline, a SRL→parsing pipeline, and a simple joint model by embedding sharing. The integration of syntactic and semantic features gives promising results in a Chinese Semantic Treebank, demonstrating large potentials of neural models for joint parsing and semantic role labeling.

1 Introduction

The correlation between syntax and semantics has been a fundamental problem in natural language processing (Steedman, 2000). As a shallow semantic task, semantic role labeling (SRL) models have traditionally been built upon syntactic parsing results (Gildea and Jurafsky, 2002; Gildea and Palmer, 2002; Punyakanok et al., 2005). It has been shown that parser output features play a crucial role for accurate SRL (Pradhan et al., 2005; Surdeanu et al., 2007).

On the reverse direction, semantic role features have been used to improve parsing (Boxwell et al., 2010). Existing methods typically use semantic features to rerank n-best lists of syntactic parsing models (Surdeanu et al., 2008; Hajič et al., 2009). There has also been attempts to learn syntactic parsing and semantic role labeling models jointly, but most such efforts have led to negative results (Sutton and McCallum, 2005; Van Den Bosch et al., 2012; Boxwell et al., 2010).

With the rise of deep learning, neural network models have been used for semantic role labeling (Collobert et al., 2011). Recently, it has been shown that a neural semantic role labeler can give state-of-the-art accuracies without using parser output features, thanks to the use of recurrent neural network structures that automatically capture syntactic information (Zhou and Xu, 2015; Wang et al., 2015). In the parsing domain, neural network models have also been shown to give state-of-the-art results recently (Dyer et al., 2015; Weiss et al., 2015; Zhou et al., 2015).

The availability of parser-independent neural SRL models allows parsing and SRL to be performed by both parsing→SRL and SRL→parsing pipelines, and gives rise to the interesting research question whether mutual benefits between syntax and semantic roles can be better exploited under the neural setting. Different from traditional models that rely on manual feature combinations for joint learning tasks (Sutton and McCallum, 2005; Zhang and Clark, 2008a; Finkel and Manning, 2009; Lewis et al., 2015), neural network models induce non-linear feature combinations automatically from input word and Part-of-Speech (POS) embeddings. This allows more complex feature sharing between multiple tasks to be achieved effectively (Collobert et al., 2011).

We take a first step[1] in such investigation by cou-

[*]Work done while the first author was visiting SUTD.

[1]Recently, Swayamdipta et al. (2016) independently proposed a similar idea to perform joint syntactic and semantic dependency parsing. Their work mainly focuses on extending actions of a greedy transition-based parser to support the joint task, achieving good performance on an English shared task, while we use a neural network for multi-task learning and we

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 968–974,
Austin, Texas, November 1-5, 2016.

pling a state-of-the-art neural semantic role labeler (Wang et al., 2015) and a state-of-the-art neural parser (Dyer et al., 2015). First, we propose a novel parsing→SRL pipeline using a tree Long Short-Term Memory (LSTM) model (Tai et al., 2015) to represent parser outputs, before feeding them to the neural SRL model as inputs. Second, we investigate a SRL→parsing pipeline, using semantic role label embeddings to enrich parser features. Third, we build a joint training model by embedding sharing, which is the most shallow level of parameter sharing between deep neural networks. This simple strategy is immune to significant differences between the network structures of the two models, which prevent direct sharing of deeper network parameters. We choose a Chinese semantic role treebank (Qiu et al., 2016) for preliminary experiments, which offers consistent dependency between syntax and semantic role representations, thereby facilitates the application of standard LSTM models. Results show that the methods give improvements to both parsing and SRL accuracies, demonstrating large potentials of neural networks for the joint task.

Our contributions can be summarized as:

• We show that the state-of-the-art LSTM semantic role labeler of Zhou and Xu (2015), which has been shown to be able to induce syntactic features automatically, can still be improved using parser output features via tree LSTM (Tai et al., 2015);

• We show that state-of-the-art neural parsing can be improved by using semantic role features;

• We show that parameter sharing between neural parsing and SRL improves both sub tasks, which is in line with the observation of Collobert et al. (2011) between POS tagging, chunking and SRL.

• Our code and all models are released at https://github.com/ShiPeng95/ShallowJoint.

2 Models

2.1 Semantic Role Labeler

We employ the SRL model of Wang et al. (2015), which uses a bidirectional Long Short-term Memory (Hochreiter and Schmidhuber, 1997; Graves and Schmidhuber, 2005; Graves et al., 2013) for sequential labeling.

work on a Chinese dataset.

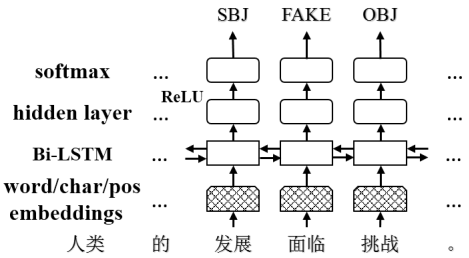

Figure 1: Bi-LSTM Semantic Role Labeler

Given the sentence "人类(human) 的(de) 发展(development) 面临(face) 挑战(challenge)", the structure of the model is shown in Figure 1. For each word w_t, the LSTM model uses a set of vectors to control information flow: an input gate i_t, a forget gate f_t, a memory cell c_t, an output gate o_t, and a hidden state h_t. The computation of each vector is as follows:

$$i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1} + V^{(i)}c_{t-1} + b^{(i)})$$
$$f_t = 1.0 - i_t$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)})$$
$$o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1} + V^{(o)}c_t + b^{(o)})$$
$$h_t = o_t \odot \tanh(c_t)$$

Here σ denotes component-wise sigmoid function and $\odot$ is component-wise multiplication.

The representation of x_t is from four sources: an embedding for the word w_t, two hidden states of the last LSTM cells in a character-level bidirectional LSTM (Ballesteros et al., 2015) (denoted as $\overrightarrow{ch_t}$ and $\overleftarrow{ch_t}$, respectively), and a learned vector Part-of-Speech (POS) representation (pos_t). A linear transformation is applied to the vector representations before feeding them into a component-wise ReLU (Nair and Hinton, 2010) function.

$$x_t = \max\{0, V^{(x)}[w_t; \overrightarrow{ch_t}; \overleftarrow{ch_t}; pos_t] + b^{(x)}\}$$

The hidden state vectors at the t-th word from both directions (denote as $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$, respectively) are passed through the ReLU function, before a softmax layer for semantic role detection.

2.2 Stack-LSTM Dependency Parser

We employ the Stack-LSTM model of Dyer et al. (2015) for dependency parsing. As shown in Figure 2, it uses a buffer (B) to order input words, a stack (S) to store partially constructed syntactic trees, and

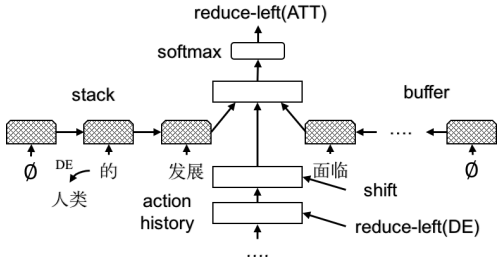

Figure 2: Stack-LSTM Parser

takes the following types of actions to build trees from input.

• **SHIFT**, which pops the top element off the buffer, pushing it into stack.

• **REDUCE-LEFT/REDUCE-RIGHT**, which pop the top two elements off the stack, pushing back the composition of the two elements with a dependent relation.

The parser is initialized by pushing input embeddings into the buffer in the reverse order. The representation of the token is same as the previous bidirectional LSTM (Bi-LSTM) model. The buffer (B), stack (S) and action history sequence (A) are all represented by LSTMs, with S being represented by a novel stack LSTM. At a time step t, the parser predicts an action according to current parser state p_t:

$$p_t = \max\{0, W^{(parser)}[s_t; b_t; a_t] + d^p\},$$
$$y_t^{(parser)} = softmax(V^{(parser)} p_t + d^y)$$

W, V and d are model parameters.

2.3 DEP→SRL Pipeline

In this pipeline model, we apply Stack-LSTM parsing first and feed the results as additional features for SRL. For each word w_t to the SRL system, the corresponding input becomes,

$$x_t^{(dep)} = \max\{0, V^{(dep)}[w_t; \overrightarrow{ch}_t; \overleftarrow{ch}_t; pos_t; \mathbf{dep_t}]\}$$

where dep_t is the t-th word's dependency information from parser output and $V^{(dep)}$ is a weight matrix. There are multiple ways to define dep_t. A simple method is to use embeddings of the dependency label at w_t. However, this input does not embody full arc information.

We propose a novel way of defining $dep_{\bar{t}}$, by using hidden vector $h_{\bar{t}}$ of a dependency tree LSTM (Tai et al., 2015) at $w_{\bar{t}}$ as $dep_{\bar{t}}$. Given a dependency tree output, we define tree LSTM inputs $x_{\bar{t}}$ in the same way as Section 2.1. The tree LSTM is a bottom-up generalization of the sequence LSTM, with a node $h_{\bar{t}}$ having multiple predecessors $h_{\bar{t}-1}^k$, which corresponding to the syntactic dependents of the word $w_{\bar{t}}$. The computation of $h_{\bar{t}}$ for each $w_{\bar{t}}$ is (unlike t, which is a left-to-right index, $\bar{t}$ is a bottom-up index, still with one $h_{\bar{t}}$ being computed for each $w_{\bar{t}}$):

$$\widetilde{h}_{\bar{t}-1} = \sum_k h_{\bar{t}-1}^k$$
$$i_{\bar{t}} = \sigma(W^{(i)} x_{\bar{t}} + U^{(i)} \widetilde{h}_{\bar{t}-1} + b^{(i)})$$
$$f_{\bar{t}^k = \sigma(W^{(f)} x_{\bar{t}} + U^{(f)} h_{\bar{t}-1}^k + b^{(f)})}$$
$$c_{\bar{t}=\sum_k f_{\bar{t}}^k \odot c_{\bar{t}-1}^k + i_{\bar{t}} \odot \tanh(W^{(u)} x_{\bar{t}} + U^{(u)} \widetilde{h}_{\bar{t}-1} + b^{(u)})}$$
$$o_{\bar{t}=\sigma(W^{(o)} x_{\bar{t}} + U^{(o)} \widetilde{h}_{\bar{t}-1} + b^{(o)})}$$
$$h_{\bar{t}=o_{\bar{t}} \odot \tanh(c_{\bar{t}})}$$

For training, we construct a corpus with all words being associated with automatic dependency labels by applying 10-fold jackknifing.

2.4 SRL→DEP Pipeline

In this pipeline model, we conduct SRL first, and feed the output semantic roles to the Stack-LSTM parser in the token level. The representation of a token becomes:

$$x_t^{(srl)} = \max\{0, V^{(srl)}[w_t; \overrightarrow{ch}_t; \overleftarrow{ch}_t; pos_t; \mathbf{srl_t}]\}$$

where srl_t is the t-th word's predicted semantic role embedding and $V^{(srl)}$ is a weight matrix.

For training, we construct a training corpus with automatically tagged semantic role labels by using 10-fold jackknifing.

2.5 Joint Model by Parameter Sharing

The structure of the joint system is shown in Figure 3. Here the parser and semantic role labeler are coupled in the embedding layer, sharing the vector lookup tables for characters, words and POS. More specifically, the Bi-LSTM model of Section 2.1 and the Stack-LSTM model of Section 2.2 are used for the SRL task and the parsing task, respectively. The Bi-LSTM labeler and Stack-LSTM parser share the embedding layer. During training, we maximize the

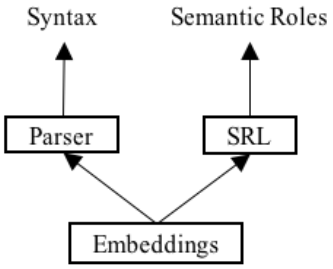

Figure 3: Joint Multi-task Model

sum of log-likelihood for the two different tasks. The loss from the semantic role labeler and the parser both propagate to the embedding layer, resulting in a better vector representation of each token, which benefits both tasks at the same time. On the other hand, due to different neural structures, there is no sharing of other parameters. The joint model offers the simplest version of *shared training* (Collobert et al., 2011), but does not employ shared decoding (Sutton and McCallum, 2005; Zhang and Clark, 2008b). Syntax and semantic roles are assigned separately, avoiding error propagation.

3 Experiments

3.1 Experimental Settings

Datasets We choose Chinese Semantic Treebank (Qiu et al., 2016) for our experiments. Similar to the CoNLL corpora (Surdeanu et al., 2008; Hajič et al., 2009) and different from PropBank (Kingsbury and Palmer, 2002; Xue and Palmer, 2005), it is a dependency-based corpus rather than a constituent-based corpus. The corpus contains syntactic dependency arc and semantic role annotations in a consistent form, hence facilitating the joint task. We follow the standard split for the training, development and test sets, as shown in Table 1.

Training Details. There is a large number of singletons in the training set and a large number of out-of-vocabulary (OOV) words in the development set. We use the mechanism of Dyer et al. (2015) to stochastically set singletons as *UNK* token in each training iteration with a probability p_{unk}. The hyper-parameter p_{unk} is set to 0.2.

For parameters used in Stack-LSTM, we follow Dyer et al. (2015). We set the number of embeddings by intuition, and decide to have the size of word embedding twice as large as that of character embedding, and the size of character embedding larger than the size of POS embedding. More specifically, we fix the size of word embeddings n_w to 64, character embeddings n_{char} to 32, POS embeddings n_{pos} to 30, action embeddings n_{dep} to 30, and semantic role embeddings n_{srl} to 30. The LSTM input size is set to 128 and the LSTM hidden size to 128.

Dataset	Words	**Types**	**Singletons**	**OOV**
Train	280,043	24,866	12,012	-
Dev	23,724	5,492	-	1,505
Test	32,326	6,989	-	1,893

Table 1: Statistics of Chinese Semantic Treebank.

We randomly initialize each parameter to a real value in $[-\sqrt{\frac{6}{r+c}}, \sqrt{\frac{6}{r+c}}]$, where r is the number of input unit and c is the number of output unit (Glorot and Bengio, 2010). To minimize the influence of external information, we did not pretrain the embedding values. In addition, we apply a Gaussian noise $N(0, 0.2)$ to word embeddings during training to prevent overfitting.

We optimize model parameters using stochastic gradient descent with momentum. The same learning rate decay mechanism of Dyer et al. (2015) is used. The best model parameters are selected according to a score metric on the development set. For different tasks, we use different score metrics to evaluate the parameters. Since there are there metrics, F1, UAS and LAS, possibly reported at the same time, we use the weighted average to consider the effect of all metrics when choosing the best model on the dev set. In particular, we use F_1 for SRL, $0.5 \times LAS + 0.5 \times UAS$ for parsing, and $0.5 \times F_1 + 0.25 \times UAS + 0.25 \times LAS$ for the joint task.

3.2 Results

The final results are shown in Table 2, where F_1 represents the F_1-score of semantic roles, and UAS and LAS represent parsing accuracies. The **Bi-LSTM** row represents the bi-directional semantic role labeler, the **S-LSTM** row represents the Stack-LSTM parser, the **DEP→SRL** row represents the dependency parsing → SRL pipeline, the **SRL→DEP** row represents the SRL → dependency parsing pipeline, and the **Joint** row represents the parameter-shared model. For the DEP→SRL pipeline, *lab* and *lstm*

Model	F_1	UAS	LAS
Bi-LSTM	72.71	-	-
S-LSTM	-	84.33	82.10
DEP→SRL(*lab*/*lstm*)	73.00/**74.18**	84.33	82.10
SRL→DEP	72.71	84.75	82.62
Joint	73.84	**85.15**	**82.91**

Table 2: Results. Bi-LSTM and S-LSTM are two baseline models for SRL and parsing, respectively. DEP→SRL and SRL→DEP are two pipeline models. 'Joint' denotes the proposed model for joint parsing and semantic role labeling. *lab* uses only the dependency label as features, while *lstm* applies features extracted from dependency trees using tree LSTMs.

represents the use of dependency label embeddings and tree LSTM hidden vectors for the additional SRL features dep_t, respectively.

Comparison between Bi-LSTM and DEP→SRL shows that slight improvement is brought by introducing dependency label features to the semantic role labeler (72.71→73.00). By introducing full tree information, the *lstm* integration leads to much higher improvements (72.71→74.18). This demonstrates that the LSTM SRL model of Zhou and Xu (2015) can still benefit from parser outputs, despite that it can learn syntactic information independently.

In the reverse direction, comparison between S-LSTM and SRL→DEP shows improvement to UAS/LAS by integrating semantic role features (82.10→82.62). This demonstrates the usefulness of semantic roles to parsing and is consistent with observations on discrete models (Boxwell et al., 2010). To our knowledge, we are the first to report results using a SRL → Parsing pipeline, which is enabled by the neural SRL model.

Using shared embeddings, the joint model gives improvements on both SRL and parsing. The most salient difference between the joint model and the two pipelines is the shared parameter space.

These results are consistent with the finds of Collobert et al. (2011) who show that POS, chunking and semantic role information can bring benefit to each other in joint neural training. In contrast to their results (SRL 74.15→74.29, POS 97.12→97.22, CHUNK 93.37→93.75), we find that parsing and SRL benefit relatively more from each other (SRL 72.72→73.84, DEP 84.33→85.15). This is intuitive because parsing offers deeper syntactic information compared to POS and shallow syntactic chunking.

4 Conclusion

We investigated the mutual benefits between dependency syntax and semantic roles using two state-of-the-art LSTM models, finding that both can be further improved. In addition, simple multitask learning is also effective. These results demonstrate potentials for deeper joint neural models between these tasks.

Acknowledgments

Yue Zhang is the corresponding author. This research is supported by NSFC61572245 and T2MOE201301 from Singapore Ministry of Education. We appreciate anonymous reviewers for their insightful comments.

References

Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. *arXiv preprint arXiv:1508.00657*.

Stephen A Boxwell, Dennis N Mehay, and Chris Brew. 2010. What a parser can learn from a semantic role labeler and vice versa. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 736–744. Association for Computational Linguistics.

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

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. ACL*.

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.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288.

Daniel Gildea and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 239–246. Association for Computational Linguistics.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256.

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

A. Graves, A. Mohamed, and G. Hinton. 2013. Speech recognition with deep recurrent neural networks.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–18. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *LREC*. Citeseer.

Mike Lewis, Luheng He, and Luke Zettlemoyer. 2015. Joint a* ccg parsing and semantic role labelling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1444–1454, Lisbon, Portugal, September. Association for Computational Linguistics.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814.

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H Martin, and Daniel Jurafsky. 2005. Semantic role chunking combining complementary syntactic views. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 217–220. Association for Computational Linguistics.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2005. The necessity of syntactic parsing for semantic role labeling. In *IJCAI*, volume 5, pages 1117–1123.

Likun Qiu, Yue Zhang, and Meishan Zhang. 2016. Dependency tree representations of predicate-argument structures. In *Proc. AAAI*.

Mark Steedman. 2000. *The syntactic process*, volume 24. MIT Press.

Mihai Surdeanu, Lluís Màrquez, Xavier Carreras, and Pere R Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research*, pages 105–151.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177. Association for Computational Linguistics.

Charles Sutton and Andrew McCallum. 2005. Joint parsing and semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 225–228. Association for Computational Linguistics.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Greedy, joint syntactic-semantic parsing with stack lstms. *arXiv preprint arXiv:1606.08954*.

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. Association for Computational Linguistics.

Antal Van Den Bosch, Roser Morante, and Sander Canisius. 2012. Joint learning of dependency parsing and semantic role labeling. *Computational Linguistics in the Netherlands Journal*, 2:97–117.

Zhen Wang, Tingsong Jiang, Baobao Chang, and Zhifang Sui. 2015. Chinese semantic role labeling with bidirectional recurrent neural networks. In *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1626–1631.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. 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 323–333, Beijing, China, July. Association for Computational Linguistics.

Nianwen Xue and Martha Palmer. 2005. Automatic semantic role labeling for chinese verbs. In *IJCAI*, volume 5, pages 1160–1165. Citeseer.

Yue Zhang and Stephen Clark. 2008a. Joint word segmentation and pos tagging using a single perceptron. In *ACL*, pages 888–896.

Yue Zhang and Stephen Clark. 2008b. A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 562–571. Association for Computational Linguistics.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Hao Zhou, Yue Zhang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1213–1222.