# Shift-Reduce CCG Parsing using Neural Network Models

**Bharat Ram Ambati** and **Tejaswini Deoskar** and **Mark Steedman**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
bharat.ambati@ed.ac.uk, {tdeoskar,steedman}@inf.ed.ac.uk

## Abstract

We present a neural network based shift-reduce CCG parser, the first neural-network based parser for CCG. We also study the impact of neural network based tagging models, and greedy versus beam-search parsing, by using a structured neural network model. Our greedy parser obtains a labeled F-score of 83.27%, the best reported result for greedy CCG parsing in the literature (an improvement of 2.5% over a perceptron based greedy parser) and is more than three times faster. With a beam, our structured neural network model gives a labeled F-score of 85.57% which is 0.6% better than the perceptron based counterpart.

## 1 Introduction

Shift-reduce parsing is interesting for practical real-world applications like parsing the web, since parsing can be achieved in linear time. Although greedy parsers are fast, accuracies of these parsers are typically much lower than graph-based parsers. Conversely, beam-search parsers achieve accuracies comparable to graph-based parsers (Zhang and Nivre, 2011) but are much slower than their greedy counterparts. Recently, Chen and Manning (2014) have showed that fast and accurate parsing can be achieved using neural network based parsers. Improving their work, Weiss et al. (2015) presented a structured neural network model which gave state-of-the-art results for English dependency parsing.

There has been increasing interest in Combinatory Categorial Grammar (CCG) (Steedman, 2000) parsing due to the simplicity of its interface between syntax and semantics. In addition to predicate-argument structure, CCG captures the unbounded dependencies found in grammatical constructions like relativization, coordination, etc. We present a neural network based shift-reduce CCG parser, the first neural network based parser for CCG. We first adapt Chen and Manning (2014)'s shift-reduce dependency parser for CCG parsing. We then develop a structured neural network model based on Weiss et al. (2015), in order to explore the impact of a beam-search on the parser. We also analyze the impact of neural network taggers (for both POS-tagging and CCG supertagging) as compared to maximum entropy taggers. Our greedy neural network parser achieves unlabeled and labeled F-scores of 89.78% and 83.27% respectively, an improvement of around 2.5% over a perceptron based greedy parser, and is more than three times faster. Due to its relevance for large-scale parsing, we make this parser available for public usage. By using a beam search, our structured neural network model gave even better results of 91.95% and 85.57% unlabeled and labeled F-scores respectively. To the best of our knowledge, ours is the first neural network based parser for CCG and also the first work on exploring neural network taggers for shift-reduce CCG parsing.

## 2 Related Work

### 2.1 CCG Parsers

Due to the availability of English CCGbank (Hockenmaier and Steedman, 2007), several wide-coverage CCG parsers have been developed (Hockenmaier and Steedman, 2002; Clark and Curran, 2007; Auli and Lopez, 2011; Zhang and Clark,



*Proceedings of NAACL-HLT 2016*, pages 447–453,
San Diego, California, June 12-17, 2016. 

2011; Lewis and Steedman, 2014a). While the majority of CCG parsers are chart-based (Clark and Curran, 2007; Lewis and Steedman, 2014a), there has been some work on shift-reduce CCG parsing (Zhang and Clark, 2011; Xu et al., 2014; Ambati et al., 2015). Zhang and Clark (2011) used a global linear model trained discriminatively with the averaged perceptron (Collins, 2002) and beam search for their shift-reduce CCG parser. A dependency model for shift-reduce CCG parsing using a dynamic oracle technique (Goldberg and Nivre, 2012) was developed by Xu et al. (2014). Ambati et al. (2015) presented an incremental algorithm for transition based CCG parsing which introduced two novel revealing actions to overcome the consequences of the greedy nature of the previous parsers.

### 2.2 Neural Network Parsers

Neural Network parsers are attracting interest for both speed and accuracy. There has been some work on neural networks for constituent based parsing (Collobert, 2011; Socher et al., 2013; Watanabe and Sumita, 2015). Chen and Manning (2014) developed a neural network architecture for dependency parsing. This parser was fast and accurate, parsing around 1000 sentences per second and achieving an unlabeled attachment score of 92.0% on the standard Penn Treebank test data for English. Chen and Manning (2014)'s parser used a feed forward neural network. Several improvements were made to this architecture in terms of using Long Short-Term Memory (LSTM) networks (Dyer et al., 2015), deep neural networks (Weiss et al., 2015) and structured neural networks (Weiss et al., 2015; Zhou et al., 2015; Alberti et al., 2015).

## 3 Our Neural Network Parser (NNPar)

The architecture of our neural network based shift-reduce CCG parser is similar to that of Chen and Manning (2014). We present the details of the network and the model settings in this section. We also discuss our structured neural network model.

### 3.1 Architecture

Figure 1 shows the architecture of our neural network parser. There are three layers in the parser: input, hidden and output layers. We first extract discrete features like words, POS-tags and CCG supertags from the parser configuration. For each of these discrete features we obtain a continuous vector representation in the form of their corresponding embeddings and use them in the input layer. Following Chen and Manning (2014), we use a cube activation function and softmax for output layer.

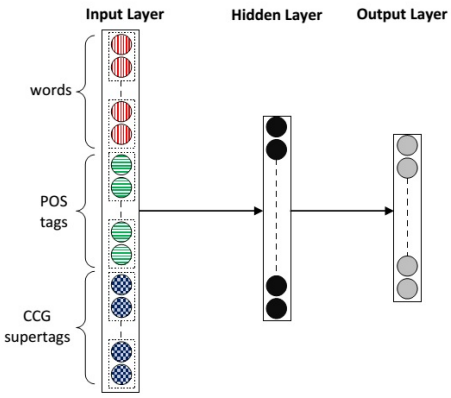


Figure 1: Our Neural Network Architecture (adapted from Chen and Manning (2014)).

### 3.2 Feature and Model Settings

We extract features from a) top four nodes in the stack, b) next four nodes in the input and c) left and right children of the top two nodes in the stack. We obtain words and POS-tags of all these 12 nodes. In case of CCG supertags, in addition to the CCG categories of the nodes in the stack (top four nodes, left and right children of top two nodes), we also obtain the lexical head categories for the top two nodes. We use a special token 'NULL' if a feature is not present in the parser configuration. So, in total we have 34 features: 12 word, 12 POS-tag and 10 CCG supertag features. For each of these 34 features we obtain their corresponding embeddings. We use Turian embeddings of dimensionality 50 (Turian-50)[1]. For the words which are not in the word embeddings dictionary, embeddings of '-UNKNOWN-' token are used as a backoff. For POS-tags and CCG supertags, the parameters are randomly initialized with values between -0.01 and 0.01.

Our input layer is a 34 (feature templates) X 50 (embedding size) dimensional vector. We use 200

[1]Lewis and Steedman (2014b) explored different publicly available word embeddings (Mnih and Hinton, 2009; Turian et al., 2010; Collobert et al., 2011; Mikolov et al., 2013) for CCG supertagging and showed that Turian-50 gave best results.

hidden units in the the hidden layer. For the output layer we compute softmax probabilities only for the actions which are possible in a particular parser configuration instead of all the actions. We use the training settings of Chen and Manning (2014) for our parser. The training objective is to minimize the cross-entropy loss with an $l_2$-regularization and the training error derivatives are backpropagated during training. For optimization we use AdaGrad (Duchi et al., 2011). $10^{-8}$ and 0.01 are the values for regularization parameter and Adagrad initial learning rate respectively. Parameters that give the best labeled F-score on the development data are used for testing data.

### 3.3 Structured Neural Network

Chen and Manning (2014)'s parser is a greedy parser and it is not straight forward to add a beam during training into their parser. As a way of introducing a beam, Weiss et al. (2015) presented a structured perceptron training for the neural network parser. They first pre-trained their neural network model. For the final layer, they trained a structured perceptron using beam search decoding which takes the neural network hidden and output layers as the input. This method, known as a structured neural network, gave the state-of-the-art results for English dependency parsing. In addition to using a softmax for the output layer, we also applied this structured neural network approach for our experiments using a beam. Unlike Weiss et al. (2015)'s neural network architecture, which consists of two hidden layers with 2048 hidden units each, we use the Chen and Manning (2014) style architecture described in the previous sections.

### 3.4 Comparison to Chen and Manning (2014)

Our neural network parser differs from Chen and Manning (2014) in a number of respects. We use CCG supertags in the input layer rather than dependency labels. For word embeddings, we use Turian embeddings (Turian et al., 2010) whereas they use Collobert et al. (2011). We have a slightly smaller set of 34 feature templates as compared to their 48 templates. Our parser has 2296 actions when instantiated by specific categorial types: 1285 `Shift`, 340 `Reduce-Left`, 593 `Reduce-Right` and 78 `Reduce-Unary` actions. In comparison, Chen and Manning (2014) have a much smaller number of actions (35 for CoNLL and 91 for Stanford dependencies). Because there are many more CCG categories ($\sim$ 500) compared to dependency labels, there are relatively more operations in a CCG parser.

## 4 Experiments and Results

We first compare our neural network parser (NNPar)[2] with a perceptron based parser in the greedy settings. Then we analyze the impact of beam using neural network (NNPar) and structured neural network (Structured NNPar) models.

The perceptron based parser is a re-implementation of Zhang and Clark (2011)'s parser (Z&C*). A global linear model trained with the averaged perceptron (Collins, 2002) is used for this parser and an early-update (Collins and Roark, 2004) strategy is used during training. In the greedy setting (beam=1), when the predicted action differs from the gold action, decoding stops and weights are updated accordingly. When a beam is used (beam=16), weights are updated when the gold parse configuration falls out of the beam. For Z&C*, the feature set of Zhang and Clark (2011), which comprises of 64 feature templates is used. For NNPar, the 34 feature templates described in section 3.2 are used. We employ an arc-standard style shift-reduce algorithm for CCG parsing, similar to Zhang and Clark (2011), for all our experiments.

### 4.1 Data and Settings

We use the standard CCGbank training (sections 02 $-$ 21), development (section 00) and testing (section 23) splits for our experiments. All the experiments are performed using automatic POS-tags and CCG supertags. We compare performance using two types of taggers: maximum entropy and neural network based taggers (NNT). The C&C taggers [3] (Clark and Curran, 2004) are used for maximum entropy taggers. For neural network taggers, SENNA tagger[4] (version 3.0) (Collobert et al., 2011) is used

[2] We used Chen and Manning (2014)'s classifier for implementing our NNPar

[3] `http://svn.ask.it.usyd.edu.au/trac/candc/wiki`

[4] `http://ronan.collobert.com/senna/`

for POS-tagging and EasyCCG tagger[5] (Lewis and Steedman, 2014a) is used for supertagging. Both these taggers use a feed-forward neural network architecture with a single hidden layer similar to our NNPar architecture.

In the case of POS-tags, we consider the first best tag given by the POS tagger. For CCG supertags, we use a multitagger which gives n-best supertags for a word. Following Zhang and Clark (2011) and Xu et al. (2014), only during training, the gold CCG lexical category is added to the list of supertags for a word if it is not present in the list assigned by the multitagger.

### 4.2 Greedy Setting

In this section, we compare the performance of perceptron (Z&C*) and neural network (NNPar) parsers in the greedy setting. Table 1 presents the unlabeled F-score (UF), labeled F-score (LF) and lexical category accuracy (Cat.) for the Z&C* and NNPar on the CCGbank development data.

NNPar outperformed Z&C* on all the metrices. There is an improvement of 2.14% in UF and 2.4% in LF, when both the parsers used maximum-entropy (C&C) taggers. We also experimented with the revealing based incremental algorithm of Ambati et al. (2015). Neural network parser gave better results than the perceptron parser for the incremental algorithm as well. Using the incremental algorithm, our NNPar obtained UF and LF of 89.08% and 81.07% which is 0.3% and 1.6% respectively lower than the results with the non-incremental algorithm. So, for the rest of the experiments we use non-incremental parsing algorithm of Z&C*.

Using neural network based taggers (NNT) didn't give any improvement for Z&C* in the greedy settings. Performance of NNT is slightly lower than C&C tagger which could be the reason for this (Lewis and Steedman, 2014a). But for NNPar, NNT improved the performance over C&C by 0.7%. Lewis and Steedman (2014a) and Xu et al. (2015) showed improvements in the performance of C&C, a graph based parser, by using neural network taggers. Our result with NNPar is in line with theirs and shows that neural network taggers can improve the performance of shift-reduce CCG parsers as well. We obtained final unlabeled and labeled F-scores of 90.09% and 83.33% respectively on the development data. To the best of our knowledge these are the best reported results for greedy shift-reduce CCG parsing.

| *Model* | *Tagger* | *UF* | *LF* | *Cat.* |
|---|---|---|---|---|
| Z&C* | C&C | 87.24 | 80.25 | 91.09 |
| Our NNPar | C&C | 89.38 | 82.65 | 91.72 |
| Z&C* | NNT | 87.00 | 79.78 | 90.52 |
| Our NNPar | NNT | **90.09** | **83.33** | **92.03** |

Table 1: Performance of greedy CCG parsers on CCGbank development data (Sec. 00).

### 4.3 Beam Search

We next analyze the impact of beam-search on the various parsers. For Z&C* and Structured NNPar, we use a beam of size 16 both during training and testing; for NNPar, a beam (of 16) can be used only during testing. Table 2 presents the results using a beam size of 16. Results are presented with a beam of size 16 to enable direct comparison with Zhang and Clark (2011), since our parsing algorithm is similar to theirs.

The top 3 rows of the table show the results of our experiments and the last 2 rows contain published results of Zhang and Clark (2011) and Xu et al. (2014). Using a beam improved the performance of both the perceptron and neural network parsers. Since NNPar uses a beam only during testing, there is only slight improvement in the f-score. Using a structured neural network gave a significant boost in performance. Structured NNPar is better than NNPar on all the metrices which shows that Structured NNPar is a stronger model than NNPar. We obtained a final LF of 85.69% on the development data which is 1.3% better than the Z&C*, the structured perceptron counter part, and 1.1% better than NNPar. This is the best published result on the development data for shift-reduce CCG parsing.

### 4.4 Final Test Results

Table 3 presents the results for the final test data. The top 2 rows of the table present the results in the greedy settings. The middle 3 rows of the table show the results with a beam. The last 2 rows give the published results of Zhang and Clark (2011)

[5] http://homepages.inf.ed.ac.uk/s1049478/easyccg.html

| *Model* | *Beam* | *UF* | *LF* | *Cat.* |
|---|---|---|---|---|
| Z&C* | 1 | 87.28 | 80.78 | 91.44 |
| Our NNPar | 1 | 89.78 | 83.27 | 91.89 |
| Z&C* | 16 | 91.28 | 85.00 | 92.79 |
| Our NNPar | 16 | 91.14 | 84.44 | 92.22 |
| Our Structured NNPar | 16 | **91.95** | 85.57 | **92.86** |
| Zhang and Clark (2011) | 16 | - | 85.48 | 92.77 |
| Xu et al. (2014) | 128 | - | **86.00** | 92.75 |

Table 3: Results on CCGbank test data (Sec. 23).

| *Model* | *UF* | *LF* | *Cat.* |
|---|---|---|---|
| Z&C* | 91.17 | 84.34 | 92.42 |
| Our NNPar | 91.46 | 84.55 | 92.35 |
| Our Structured NNPar | **92.19** | **85.69** | **93.02** |
| Zhang and Clark (2011) | | 85.00 | 92.77 |
| Xu et al. (2014) | | 85.18 | 92.75 |

Table 2: Impact of the beam on CCGbank development data (Sec. 00).

and Xu et al. (2014). With the greedy setting, our NNPar outperformed Z&C* by around 2.5%, obtaining 89.78% and 83.27% UF and LF respectively. These are the best reported result for greedy shift-reduce CCG parsing.

In the case of the beam search parsers, we achieved final best scores of 91.95% in UF and 85.57% in LF with our Structured NNPar. Structured NNPar gave improvements of 1.1% over the NNPar and 0.6% over the structured perceptron model, Z&C*. Structured NNPar gets better category accuracy, but lower LF than Xu et al.(2014). Note however that we use a much smaller beam size of 16 (similar to Z&C) as compared to theirs (128). Increasing the beam size improved the accuracy but significantly reduced the parsing speed. Testing with a beam of size 128 gave 0.2% improvement in labelled F-score but slowed the parser by ten times.

### 4.5 Speed

Beam-search parsers are more accurate than greedy parsers but are very slow. With neural network models we can build parsers which give a nice trade-off between speed and accuracy. Table 4 present the speed comparison for both Z&C* and our NNPar in greedy settings. NNPar is much faster, parsing 350 sentences per second compared to Z&C* which parses 110 sentences per second. Parsers with a beam of size 16 parse around 10 sentences per second and parsers with a beam of size 128 parse around 1 sentence per second. These numbers don't include POS tagging and supertagging time.

| *Model* | *Sentences/Second* |
|---|---|
| Z&C* | 110 |
| NNPar | 350 |

Table 4: Speed comparison of perceptron and neural network based greedy parsers.

## 5 Conclusion

We presented the first neural network based shift-reduce parsers for CCG, a greedy and a beam-search parser. On the standard CCGbank test data, we achieved a labeled F-score of 85.57% with our structured neural network parser, an improvement of 0.6% over the structured perceptron parser (Z&C*). Our greedy parser gets UF and LF of 89.78% and 83.27% respectively, the best reported results for a greedy CCG parser, and is more than three times faster. In future we plan to explore more sophisticated tagging and parsing models like deep neural networks (Weiss et al., 2015), recurrent neural networks (Dyer et al., 2015), and bi-directional LSTMs (Lewis et al., 2016) for shift-reduce CCG parsing.

The parser code can be downloaded at `https://github.com/bharatambati/tranccg`.

## Acknowledgments

We thank Mike Lewis, Greg Coppola and Siva Reddy for helpful discussions. We also thank the three anonymous reviewers for their useful suggestions. This work was supported by ERC Advanced

Fellowship 249520 GRAMPLUS and EU IST Cognitive Systems IP Xperience.

## References

Chris Alberti, David Weiss, Greg Coppola, and Slav Petrov. 2015. Improved Transition-Based Parsing and Tagging with Neural Networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1354–1359, Lisbon, Portugal, September.

Bharat Ram Ambati, Tejaswini Deoskar, Mark Johnson, and Mark Steedman. 2015. An Incremental Algorithm for Transition-based CCG Parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 53–63, Denver, Colorado, May–June.

Michael Auli and Adam Lopez. 2011. A Comparison of Loopy Belief Propagation and Dual Decomposition for Integrated CCG Supertagging and Parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 470–480, Portland, Oregon, USA, June.

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.

Stephen Clark and James R. Curran. 2004. The Importance of Supertagging for Wide-Coverage CCG Parsing. In *Proceedings of Coling 2004*, pages 282–288, Geneva, Switzerland, Aug 23–Aug 27. COLING.

Stephen Clark and James R. Curran. 2007. Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33:493–552.

Michael Collins and Brian Roark. 2004. Incremental Parsing with the Perceptron Algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 111–118, Barcelona, Spain, July.

Michael Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8, July.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Ronan Collobert. 2011. Deep learning for efficient discriminative parsing. In *International Conference on Artificial Intelligence and Statistics*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-Based Dependency Parsing with Stack Long Short-Term Memory. 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 334–343, Beijing, China, July.

Yoav Goldberg and Joakim Nivre. 2012. A Dynamic Oracle for Arc-Eager Dependency Parsing. In *Proceedings of COLING 2012*, pages 959–976, Mumbai, India, December. The COLING 2012 Organizing Committee.

Julia Hockenmaier and Mark Steedman. 2002. Generative Models for Statistical Parsing with Combinatory Categorial Grammar. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 335–342, Philadelphia, Pennsylvania, USA, July.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Mike Lewis and Mark Steedman. 2014a. A* CCG Parsing with a Supertag-factored Model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar, October.

Mike Lewis and Mark Steedman. 2014b. Improved CCG parsing with Semi-supervised Supertagging. *Transactions of the Association for Computational Linguistics (TACL)*, 2:327–338.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. LSTM CCG Parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, California, June.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of Workshop at ICLR*.

Andriy Mnih and Geoffrey Hinton. 2009. A Scalable Hierarchical Distributed Language Model. In *Advances in Neural Information Processing Systems*, volume 21, pages 1081–1088.

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with Compositional

Vector Grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465, Sofia, Bulgaria, August.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA, USA.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word Representations: A Simple and General Method for Semi-Supervised Learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden, July.

Taro Watanabe and Eiichiro Sumita. 2015. Transition-based Neural Constituent 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 1169–1179, Beijing, China, July.

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.

Wenduan Xu, Stephen Clark, and Yue Zhang. 2014. Shift-Reduce CCG Parsing with a Dependency Model. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 218–227, Baltimore, Maryland, June.

Wenduan Xu, Michael Auli, and Stephen Clark. 2015. CCG Supertagging with a Recurrent Neural Network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 250–255, Beijing, China, July.

Yue Zhang and Stephen Clark. 2011. Shift-Reduce CCG Parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 683–692, Portland, Oregon, USA, June.

Yue Zhang and Joakim Nivre. 2011. Transition-based Dependency Parsing with Rich Non-local Features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA.

Hao Zhou, Yue Zhang, Shujian Huang, 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 and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1213–1222, Beijing, China, July.