Combining Discrete and Continuous Features for Deterministic Transition-based Dependency Parsing

Meishan Zhang and **Yue Zhang**
Singapore University of Technology and Design
{meishan_zhang, yue_zhang}@sutd.edu.sg

Abstract

We investigate a combination of a traditional linear sparse feature model and a multi-layer neural network model for deterministic transition-based dependency parsing, by integrating the sparse features into the neural model. Correlations are drawn between the hybrid model and previous work on integrating word embedding features into a discrete linear model. By analyzing the results of various parsers on web-domain parsing, we show that the integrated model is a better way to combine traditional and embedding features compared with previous methods.

1 Introduction

Transition-based parsing algorithms construct output syntax trees using a sequence of shift-reduce actions. They are attractive in computational efficiency, allowing linear time decoding with deterministic (Nivre, 2008) or beam-search (Zhang and Clark, 2008) algorithms. Using rich non-local features, transition-based parsers achieve state-of-the-art accuracies for dependency parsing (Zhang and Nivre, 2011; Zhang and Nivre, 2012; Bohnet and Nivre, 2012; Choi and McCallum, 2013; Zhang et al., 2014).

Deterministic transition-based parsers works by making a sequence of greedy local decisions (Nivre et al., 2004; Honnibal et al., 2013; Goldberg et al., 2014; Gómez-Rodríguez and Fernández-González, 2015). They are attractive by very fast speeds. Traditionally, a linear model has been used for the local action classifier. Recently, Chen and Manning (2014) use a neural network (*NN*) to replace linear models, and report improved accuracies.

A contrast between a neural network model and a linear model is shown in Figure 1 (a) and (b).

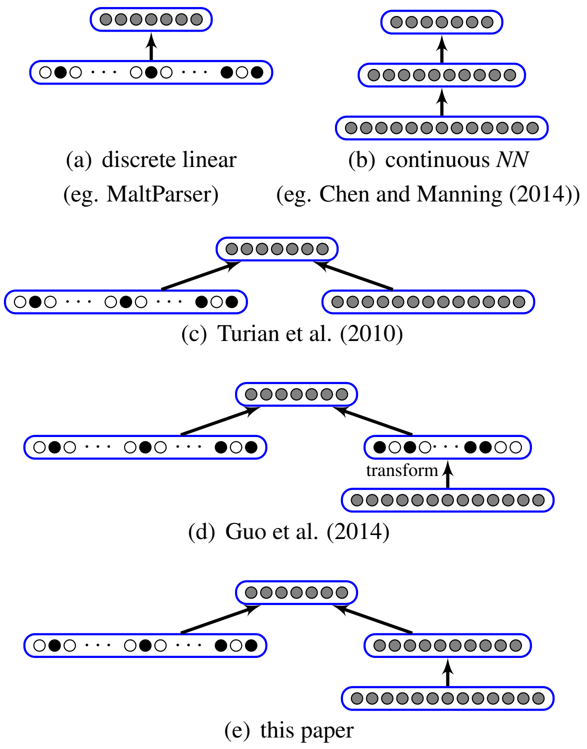

Figure 1: Five deterministic transition-based parsers with discrete and continuous features.

A neural network model takes continuous vector representations of words as inputs, which can be pre-trained using large amounts of unlabeled data, thus containing more information. In addition, using an extra hidden layer, a neural network is capable of learning non-linear relations between automatic features, achieving feature combinations automatically.

Discrete manual features and continuous features complement each other. A natural question that arises from the contrast is whether traditional discrete features and continuous neural features can be integrated for better accuracies. We study this problem by constructing the neural network shown in Figure 1 (e), which incorporates the discrete input layer of the linear model (Figure 1 (a)) into the *NN* model (Figure 1 (b)) by conjoining it with the hidden layer. This architecture is connected with previous work on incorporating word embeddings into a linear model.

In particular, Turian et al. (2010) incorporate word embeddings as real-valued features into a CRF model. The architecture is shown in Figure 1(c), which can be regarded as Figure 1(e) without the hidden layer. Guo et al. (2014) find that the accuracies of Turian et al can be enhanced by discretizing the embedding features before combining them with the traditional features. They use simple binarization and clustering to this end, finding that the latter works better. The architecture is shown in Figure 1(d). In contrast, Figure 1(e) directly combines discrete and continuous features, replacing the hard-coded transformation function of Guo et al. (2014) with a hidden layer, which can be tuned by supervised training.[1]

We correlate and compare all the five systems in Figure 1 empirically, using the SANCL 2012 data (Petrov and McDonald, 2012) and the standard Penn Treebank data. Results show that the method of this paper gives higher accuracies than the other methods. In addition, the method of Guo et al. (2014) gives slightly better accuracies compared to the method of Turian et al. (2010) for parsing task, consistent with Guo et al's observation on named entity recognition (NER). We make our C++ code publicly available under GPL at `https://github.com/SUTDNLP/NNTransitionParser`.

2 Parser

We take Chen and Manning (2014), which uses the arc-standard transition system (Nivre, 2008). Given an POS-tagged input sentence, it builds a projective output y by performing a sequence of state transition actions using greedy search. Chen and Manning (2014) can be viewed as a neutral alternative of MaltParser (Nivre, 2008).

Although not giving state-of-the-art accuracies, deterministic parsing is attractive for its high parsing speed (1000+ sentences per second). Our incorporation of discrete features does not harm the overall speed significantly. In addition, deterministic parsers use standard neural classifiers, which allows isolated study of feature influences.

[1]Yet another alternative structure is to directly combine the two types of inputs, and replacing the input layer of (b) using them. Wang and Manning (2013) compared this architecture with (c) using a CRF network, finding that the latter works better for NER and chunking.

3 Models

Following Chen and Manning (2014), training of all the models using a cross-entropy loss objective with a L2-regularization, and mini-batched AdaGrad (Duchi et al., 2011). We unify below the five deterministic parsing models in Figure 1.

3.1 Baseline linear (*L*)

We build a baseline linear model using logistic regression (Figure 1(a)). Given a parsing state x, a vector of discrete features $\Phi_d(x)$ is extracted according to the *arc-standard* feature templates of Ma et al. (2014a), which is based on the *arc-eager* templates of Zhang and Nivre (2011). The score of an action a is defined by

$$\text{Score}(a) = \sigma\big(\Phi_d(x) \cdot \overrightarrow{\theta}_{d,a}\big),$$

where σ represents the sigmoid activation function, $\overrightarrow{\theta}_d$ is the set of model parameters, denoting the feature weights with respect to actions, a can be SHIFT, LEFT(l) and RIGHT(l).

3.2 Baseline Neural (*NN*)

We take the Neural model of Chen and Manning (2014) as another baseline (Figure 1(b)). Given a parsing state x, the words are first mapped into continuous vectors by using a set of pre-trained word embeddings. Denote the mapping as $\Phi_e(x)$. In addition, denote the hidden layer as Φ_h, and the ith node in the hidden as $\Phi_{h,i}$ ($0 \leq i \leq |\Phi_h|$). The hidden layer is defined as

$$\Phi_{h,i} = \big(\Phi_e(x) \cdot \overrightarrow{\theta}_{h,i}\big)^3,$$

where $\overrightarrow{\theta}_h$ is the set of parameters between the input and hidden layers. The score of an action a is defined as

$$\text{Score}(a) = \sigma\big(\Phi_h(x) \cdot \overrightarrow{\theta}_{c,a}\big),$$

where $\overrightarrow{\theta}_{c,a}$ is the set of parameters between the hidden and output layers. We use the *arc-standard* features Φ_e as Chen and Manning (2014), which is also based on the *arc-eager* templates of Zhang and Nivre (2011), similar to those of the baseline model *L*.

3.3 Linear model with real-valued embeddings (*Turian*)

We apply the method of Turian et al. (2010), combining real-valued embeddings with discrete features in the linear baseline (Figure 1(c)). Given a state x, the score of an action a is defined as

$$\text{Score}(a) = \sigma\Big((\Phi_d(x) \oplus \Phi_e(x)) \cdot (\overrightarrow{\theta}_{d,a} \oplus \overrightarrow{\theta}_{c,a}) \Big),$$

where $\oplus$ is the vector concatenation operator.

3.4 Linear model with transformed embeddings (*Guo*)

We apply the method of Guo et al. (2014), combining embeddings into the linear baseline by first transforming into discrete values. Given a state x, the score of an action is defined as

$$\text{Score}(a) = \sigma\Big((\Phi_d(x) \oplus d(\Phi_e(x))) \cdot (\overrightarrow{\theta}_{d,a} \oplus \overrightarrow{\theta}_{c,a}) \Big),$$

where d is a transformation function from real-value to binary features. We use clustering of embeddings for d as it gives better performances according to Guo et al. (2014). Following Guo et al. (2014), we use compounded clusters learnt by K-means algorithm of different granularities.

3.5 Directly combining linear and neural features (*This*)

We directly combine linear and neural features (Figure 1(e)). Given a state x, the score of an action is defined as

$$\text{Score}(a) = \sigma\Big((\Phi_d(x) \oplus \Phi_h(x)) \cdot (\overrightarrow{\theta}_{d,a} \oplus \overrightarrow{\theta}_{c,a}) \Big),$$

where Φ_h is the same as the *NN* baseline. Note that like d in Guo, Φ_h is also a function that transforms embeddings Φ_e. The main difference is that it can be tuned in supervised training.

4 Web Domain Experiments

4.1 Setting

We perform experiments on the SANCL 2012 web data (Petrov and McDonald, 2012), using the Wall Street Journal (WSJ) training corpus to train the models and the WSJ development corpus to tune parameters. We clean the web domain texts following the method of Ma et al. (2014b). Automatic POS tags are produced by using a CRF model trained on the WSJ training corpus. The POS tags are assigned automatically on the training corpus by ten-fold jackknifing. Table 1 shows the corpus details.

Domain	#Sent	#Word	TA
WSJ-train	30,060	731,678	97.03
WSJ-dev	1,336	32,092	96.88
WSJ-test	1,640	35,590	97.51
answers	1,744	28,823	91.93
newsgroups	1,195	20,651	93.75
reviews	1,906	28,086	92.66

Table 1: Corpus statistics of our experiments, where TA denotes POS tagging accuracy.

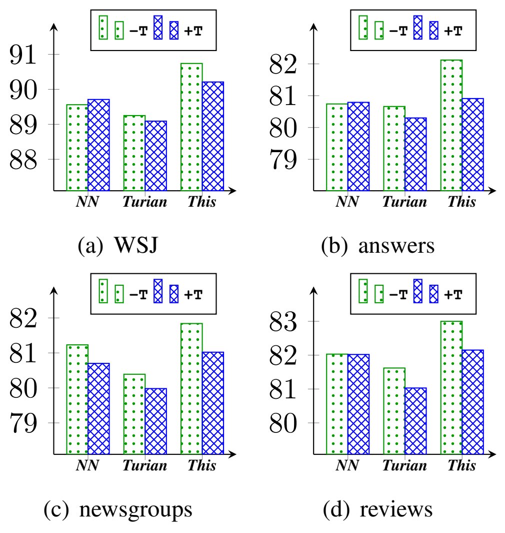

Figure 2: Dev results on fine-tuning (UAS).

Following Chen and Manning (2014), we use the pre-trained word embedding released by Collobert et al. (2011), and set $h = 200$ for the hidden layer size, $\lambda = 10^{-8}$ for L2 regularization, and $\alpha = 0.01$ for the initial learning rate of Adagrad.

4.2 Development Results

Fine-tuning of embeddings. Chen and Manning (2014) fine-tune word embeddings in supervised training, consistent with Socher et al. (2013). Intuitively, fine-tuning embeddings allows in-vocabulary words to join the parameter space, thereby giving better fitting to in-domain data. However, it also forfeits the benefit of large-scale pre-training, because out-of-vocabulary (OOV) words do not have their embeddings fine-tuned. In this sense, the method of Chen and Manning resembles a traditional supervised sparse linear model, which can be weak on OOV.

On the other hand, the semi-supervised learning methods such as Turian et al. (2010) and Guo et al. (2014), do not fine-tune the word embeddings. Embeddings are taken as inputs rather than model

Model	WSJ				answers				newsgroups				reviews			
	UAS	LAS	OOV	OOE	UAS	LAS	OOV	OOE	UAS	LAS	OOV	OOE	UAS	LAS	OOV	OOE
L	88.19	86.16	83.72	—	79.30	74.24	68.43	—	82.55	79.06	69.07	—	80.77	76.16	72.20	—
NN	89.81	87.83	84.94	84.94	79.27	74.30	69.18	**69.18**	83.71	80.35	69.60	**69.60**	81.63	77.22	72.04	**72.04**
Turian	89.17	87.21	84.13	91.35	79.57	74.57	69.60	54.21	82.89	79.65	68.48	52.63	81.33	77.04	72.30	55.03
Guo	89.33	87.21	83.82	90.83	79.32	74.22	67.36	51.76	82.75	79.31	68.18	55.06	81.87	77.25	73.03	56.80
This	**90.61**	**88.68**	**88.00**	**93.77**	**80.08**	**75.18**	**69.97**	54.21	**84.64**	**81.35**	**69.66**	53.44	**82.53**	**78.15**	**73.39**	57.20
ZPar-local	88.95	86.90	84.63	—	78.98	73.81	68.15	—	82.43	79.01	67.30	—	81.21	76.45	70.38	—
C&M(2014)	89.56	87.55	79.15	79.15	79.82	74.63	67.78	67.78	83.39	79.72	67.95	67.95	81.60	76.91	68.83	68.83

Table 2: Main results on SANCL. All systems are deterministic.

parameters. Therefore, such methods can expect better cross-domain accuracies.

We empirically compare the models *NN*, *Turian* and *This* by fine-tuning (+T) and not fine-tuning (-T) word embeddings, and the results are shown in Figure 2. As expected, the baseline NN model gives better accuracies on WSJ with fine-tuning, but worse cross-domain accuracies. Interestingly, our combined model gives consistently better accuracies with fine-tuning. We attribute this to the use of sparse discrete features, which allows the model to benefit from large-scale pre-trained embeddings without sacrificing in-domain performance. The observation on *Turian* is similar. For the final experiments, we apply fine-tuning on the *NN* model, but not to the *Turian* and *This*. Note also tat for all experiments, the POS and label embedding features of Chen and Manning (2014) are fine-tuned, consistent with their original method.

Dropout rate. We test the effect of dropout (Hinton et al., 2012) during training, using a default ratio of 0.5 according to Chen and Manning (2014). In our experiments, we find that the dense *NN* model and our combined model achieve better performances by using dropout, but the other models do not benefit from dropout.

4.3 Final Results

The final results across web domains are shown in Table 2. Our logistic regression linear parser and re-implementation of Chen and Manning (2014) give comparable accuracies to the perceptron ZPar[2] and Stanford NN Parser[3], respectively.

It can be seen from the table that both *Turian* and *Guo*[4] outperform *L* by incorporating embedding features. *Guo* gives overall higher improvements, consistent with the observation of Guo et al. (2014) on NER. Our methods gives significantly[5] better results compared with *Turian* and *Guo*, thanks to the extra hidden layer.

Our OOV performance is higher than *NN*, because the embeddings of OOV words are not tuned, and hence the model can handle them effectively. Interestingly, *NN* gives higher accuracies on web domain out-of-embedding-vocabulary (OOE) words, out of which 54% are in-vocabulary.

Note that the accuracies of our parsers are lower than the best systems in the SANCL shared task, which use ensemble models. Our parser enjoys the fast speed of deterministic parsers, and in particular the baseline *NN* parser (Chen and Manning, 2014).

5 WSJ Experiments

For comparison with related work, we conduct experiments on Penn Treebank corpus also. We use the WSJ sections 2-21 for training, section 22 for development and section 23 for testing. WSJ constituent trees are converted to dependency trees using Penn2Malt[6]. We use auto POS tags consistent with previous work. The ZPar POS-tagger is used to assign POS tags. Ten-fold jackknifing is performed on the training data to assign POS automatically. For this set of experiments, the parser hyper-parameters are taken directly from the best settings in the Web Domain experiments.

The results are shown in Table 3, together with some state-of-the-art deterministic parsers. Comparing the *L*, *NN* and *This* models, the observations are consistent with the web domain.

[2]https://sourceforge.net/projects/zpar/, version 0.7.

[3]http://nlp.stanford.edu/software/nndep.shtml

[4]We compound six clusters of granularities 500, 1000, 1500, 2000, 2500, 3000.

[5]The p-values are below 0.01 using pairwise t-test.

[6]http://stp.lingfil.uu.se/ nivre/research/Penn2Malt.html

System	UAS	LAS
L	89.36	88.33
NN	91.15	90.04
This	**91.80**	**90.68**
ZPar-local	89.94	88.92
Ma et al. (2014a)	90.38	–
Chen and Manning (2014)	91.17	89.99
Honnibal et al. (2013)	91.30	90.00
Ma et al. (2014a)*	91.32	–

Table 3: Main results on WSJ. All systems are deterministic.

Our combined parser gives accuracies competitive to state-of-the-art deterministic parsers in the literature. In particular, the method of Chen and Manning (2014) is the same as our *NN* baseline. Note that Zhou et al. (2015) reports a UAS of 91.47% by this parser, which is higher than the results we obtained. The main results include the use of different batch size during, while Zhou et al. (2015) used a batch size of 100,000, we used a batch size of 10,000 in all experiments. Honnibal et al. (2013) applies dynamic oracle to the deterministic transition-based parsing, giving a UAS of 91.30%. Ma et al. (2014a) is similar to ZPar local, except that they use the *arc-standard* transitions, while ZPar-local is based on *arc-eager* transitions. Ma et al. (2014a)* uses a special method to process punctuations, leading to about 1% UAS improvements over the vanilla system.

Recently, Dyer et al. (2015) proposed a deterministic transition-based parser using LSTM, which gives a UAS of 93.1% on Stanford conversion of the Penn Treebank. Their work shows that more sophisticated neural network structures with long term memories can significantly improve the accuracy over local classifiers. Their work is orthogonal to ours.

6 Related Work

As discussed in the introduction, our work is related to previous work on integrating word embeddings into discrete models (Turian et al., 2010; Yu et al., 2013; Guo et al., 2014). Along this line, there has also been work that uses a neural network to automatically vectorize the structures of a sentence, and then taking the resulting vector as features in a linear NLP model (Socher et al., 2012; Tang et al., 2014; Yu et al., 2015). Our results show that the use of a hidden neural layer gives superior results compared with both direct integration and integration via a hard-coded transformation function (e.g binarization or clustering).

There has been recent work integrating continuous and discrete features for the task of POS tagging (Ma et al., 2014b; Tsuboi, 2014). Both models have essentially the same structure as our model. In contrast to their work, we systematically compare various ways to integrate discrete and continuous features, for the dependency parsing task. Our model is also different from Ma et al. (2014b) in the hidden layer. While they use a form of restricted Boltzmann machine to pre-train the embeddings and hidden layer from large-scale ngrams, we fully rely on supervised learning to train complex feature combinations.

Wang and Manning (2013) consider integrating embeddings and discrete features into a neural CRF. They show that combined neural and discrete features work better without a hidden layer (i.e. Turian et al. (2010)). They argue that non-linear structures do not work well with high dimensional features. We find that using a hidden layer specifically for embedding features gives better results compared with using no hidden layers.

7 Conclusion

We studied the combination of discrete and continuous features for deterministic transition-based dependency parsing, comparing several methods to incorporate word embeddings and traditional sparse features in the same model. Experiments on both in-domain and cross-domain parsing show that directly adding sparse features into a neural network gives higher accuracies compared with all previous methods to incorporate word embeddings into a traditional sparse linear model.

Acknowledgments

We thank the anonymous reviewers for their constructive comments, which helped to improve the paper. This work is supported by the Singapore Ministry of Education (MOE) AcRF Tier 2 grant T2MOE201301 and SRG ISTD 2012 038 from Singapore University of Technology and Design.

References

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *EMNLP-CONLL*, pages 1455–1465.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*, pages 740–750.

Jinho D. Choi and Andrew McCallum. 2013. Transition-based dependency parsing with selectional branching. In *ACL*, pages 1052–1062.

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.

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 *ACL*, pages 334–343.

Yoav Goldberg, Francesco Sartorio, and Giorgio Satta. 2014. A tabular method for dynamic oracles in transition-based parsing. *Transactions of the Association for Computational Linguistics*, 2:119–130.

Carlos Gómez-Rodríguez and Daniel Fernández-González. 2015. An efficient dynamic oracle for unrestricted non-projective parsing. In *ACL-IJCNLP*, pages 256–261.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Revisiting embedding features for simple semi-supervised learning. In *Proceedings of the 2014 Conference on EMNLP*, pages 110–120.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Matthew Honnibal, Yoav Goldberg, and Mark Johnson. 2013. A non-monotonic arc-eager transition system for dependency parsing. In *Proceedings of the 17th CONLL*, pages 163–172.

Ji Ma, Yue Zhang, and Jingbo Zhu. 2014a. Punctuation processing for projective dependency parsing. In *ACL (Volume 2: Short Papers)*, pages 791–796.

Ji Ma, Yue Zhang, and Jingbo Zhu. 2014b. Tagging the web: Building a robust web tagger with neural network. In *ACL*, pages 144–154.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. In *CoNLL*.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on EMNLP-CONLL*, pages 1201–1211.

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with compositional vector grammars. In *ACL*, pages 455–465.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL*, pages 1555–1565.

Yuta Tsuboi. 2014. Neural networks leverage corpus-wide information for part-of-speech tagging. In *Proceedings of the 2014 EMNLP*, pages 938–950.

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 ACL*, pages 384–394.

Mengqiu Wang and Christopher D. Manning. 2013. Effect of non-linear deep architecture in sequence labeling. In *IJCNLP*, pages 1285–1291.

Mo Yu, Tiejun Zhao, Daxiang Dong, Hao Tian, and Dianhai Yu. 2013. Compound embedding features for semi-supervised learning. In *NAACL*, pages 563–568.

Mo Yu, Matthew R Gormley, and Mark Dredze. 2015. Combining word embeddings and feature embeddings for fine-grained relation extraction. In *NAACL*.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *EMNLP*, pages 562–571.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th ACL*, pages 188–193.

Yue Zhang and Joakim Nivre. 2012. Analyzing the effect of global learning and beam-search on transition-based dependency parsing. In *COLING 2012: Posters*, pages 1391–1400.

Yuan Zhang, Tao Lei, Regina Barzilay, Tommi Jaakkola, and Amir Globerson. 2014. Steps to excellence: Simple inference with refined scoring of dependency trees. In *ACL*, pages 197–207.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *ACL*, pages 1213–1222.