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

Deterministic transition-based parsers work 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).

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.\footnote{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.}

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.

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

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

where $\sigma$ represents the sigmoid activation function, $\vec{\theta}_d$ 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 $\Phi_c(x)$. In addition, denote the hidden layer as $\Phi_h$, and the $i$th node in the hidden as $\Phi_{h,i} (0 \leq i \leq |\Phi_h|)$. The hidden layer is defined as

$$\Phi_{h,i} = (\Phi_c(x) \cdot \vec{\theta}_{h,i})^3,$$

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

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

where $\vec{\theta}_{c,a}$ is the set of parameters between the hidden and output layers. We use the arc-standard features $\Phi_c$ 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

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

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

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

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

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

where $\Phi_h$ is the same as the NN baseline. Note that like $d$ in Guo, $\Phi_h$ is also a function that transforms embeddings $\Phi_e$. 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.

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

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

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...
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 that 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 and Stanford NN Parser, respectively.

It can be seen from the table that both Turian and Guo 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 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.

3https://sourceforge.net/projects/zpar/, version 0.7.

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

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

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

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

---

<table>
<thead>
<tr>
<th>Model</th>
<th>WSJ</th>
<th>newsgroups</th>
<th>reviews</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UAS</td>
<td>LAS</td>
<td>OOV</td>
</tr>
<tr>
<td>L</td>
<td>88.19</td>
<td>86.16</td>
<td>83.72</td>
</tr>
<tr>
<td>NN</td>
<td>89.81</td>
<td>87.83</td>
<td>84.94</td>
</tr>
<tr>
<td>Turian</td>
<td>89.17</td>
<td>87.21</td>
<td>84.31</td>
</tr>
<tr>
<td>Guo</td>
<td>89.33</td>
<td>87.21</td>
<td>83.82</td>
</tr>
<tr>
<td>This</td>
<td>90.61</td>
<td>88.68</td>
<td>88.00</td>
</tr>
</tbody>
</table>

Table 2: Main results on SANCL. 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.

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

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

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


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.


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


