A Implementation Details

We use bi-LSTMs (Hochreiter and Schmidhuber, 1997) for feature representations both at word and sentence level. A 2-layer bi-LSTM takes input from 64-dimensional character embeddings, and encodes intra-token information into its 128 hidden units (64 for each direction). Another 2-layer bi-LSTM builds sentence-level context-sensitive features with the character LSTM encodings as inputs, and assigns a 192-dimensional vector representation to each word in the sentence. All scoring functions for the edges/transitions are in the form of deep biaffine transformation (Dozat and Manning, 2017). For feature sets with more than two vectors, we define the score to be the sum of pairwise biaffine scores. Scoring of $\{\mathbf{g}, \mathbf{h}, \mathbf{m}, \mathbf{si}\}$ in the baseline 1EC parser is defined as the sum of biaffine scores of the follow pairwise interactions: $\{\mathbf{g}, \mathbf{m}\}$, $\{\mathbf{h}, \mathbf{m}\}$, $\{\mathbf{si}, \mathbf{m}\}$. Sum of biaffine scores for $\{\mathbf{s}_1, \mathbf{s}_0\}$ and $\{\mathbf{s}_0, \mathbf{b}_0\}$ constitute the score for the three-vector feature set $\{\mathbf{s}_1, \mathbf{s}_0, \mathbf{b}_0\}$. All neural-network weight parameters are randomly initialized (Glorot and Bengio, 2010), including the word and character embeddings. We train each model using Adam optimizer (Kingma and Ba, 2015) with initial learning rate 0.002, until the dev-set performance converges. During training, dropout is applied to both multi-layer perceptrons in the deep biaffine functions and the recurrent connections (Srivastava et al., 2014; Gal and Ghahramani, 2016). We set the keep rate to be 0.7 everywhere. Our implementation is based on the DyNet library (Neubig et al., 2017). Our code, including our re-implementation of the third-order 1EC parser with neural scoring, is available at `https://github.com/tzshi/mh4-parser-acl18`.

References

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of the 5th International Conference on Learning Representations*.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1019–1027.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 249–256.

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

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 4th International Conference on Learning Representations*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.