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 \{g, h, m, si\} in the baseline 1EC parser is defined as the sum of biaffine scores of the follow pairwise interactions: \{g, m\}, \{h, m\}, \{si, m\}. Sum of biaffine scores for \{s_1, s_0\} and \{s_0, b_0\} constitute the score for the three-vector feature set \{s_1, s_0, 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


