A Neural parameterization

In this appendix, we describe the different components of our neural network. If unspecified, parameters are initialized with Pytorch default initialization. We report the dimensions of the building blocks of the network in Table 5. We use hyperparameters similar to Dozat and Manning (2016) and did not perform any specific hyperparameter tuning except for the BiLSTM hidden dimension (which is larger in our case).

A.1 Word-level features

We use three kind of word-level features: word embeddings, character embeddings and, for a few experiments, part-of-speech embeddings. All embeddings are concatenated to form word-level embeddings.

Word embeddings can either be pre-trained or trained end-to-end. In the case of pre-trained word embeddings, we fix them and sum them with end-to-end learned word embeddings initialized at 0. We use the Glove pre-trianed word embeddings (Pennington et al., 2014).

Character embeddings are fed to a BiLSTM. The hidden states of the two endpoints are then concatenated together. Words are truncated to 20 characters for this feature.

A.2 Sentence-level features

We follow (Kiperwasser and Goldberg, 2016) by using two stacked BiLSTM, i.e. the input of the second BiLSTM is the concatenation of the forward and backward hidden states of the first one. All LSTMs have a single layer. Projection matrices are initialized with the orthogonal approach proposed by (Saxe et al., 2013) and bias vectors are initialized to 0.

For models using Bert (Devlin et al., 2019), we learn a convex combination of the last 4 layers, in a similar spirit to ELMO. When word are tokenized in subwords by the Bert tokenizer, we use the embedding of the first sub-token. We use the pretrained models distributed by HuggingFace's Transformers library.[12]

A.3 Output weights

We have two different output layers. First, we predict part-of-speech tags with a linear projection on top of the hidden states of the first BiLSTM.

[12] https://huggingface.co/transformers/

Name	Dimension
Word embeddings	300
Character embeddings	64
Character BiLSTM	100
Character BiLSTMs layer	1
Sentence BiLSTMs	800
Sentence BiLSTMs layer	1
Sentence BiLSTMs stack	2
Span projection	500
Label projection	100

Table 5: Hyperparameters

During training, we use an auxiliary negative log-likelihood loss. Second, after the second BiLSTM we add the biaffine layers to compute span scores.

A.4 Training

The train/dev/test split follows prior work (cited in the main text) with sizes:

- 39832/1700/2416 sentences for the Discontinuous Penn Treebank;
- 40472/5000/5000 sentences for the Tiger treebank;
- 18602/1000/1000 sentences for the Negra treebank.

We optimize the parameters with the Adam variant of stochastic gradient descent descent with minibatches containing at most 5000 words for 200 epochs. We apply dropout with ratio 0.3 before the input of the character BiLSTM, before the first stack of sentence-level BiLSTM and after the second one by following the methodology of Dozat and Manning (2016).