Symbol	Description	Value
N	Number of layers (when not using ELMo embeddings)	8
N	Number of layers (when using ELMo embeddings)	4
d_{model}	Model dimensionality	1024
h	Number of attention heads	8
d_k	Size of attention query/key vectors	64
d_v	Size of attention value vectors	64
d_{ff}	Size of intermediate vectors in the feed-forward sublayer	2048
	Size of character embeddings (CharConcat)	32
	Size of character embeddings (CharLSTM)	64
	Attention dropout probability; see Vaswani et al. (2017)	0.2
	ReLU dropout probability in feed-forward sublayer	0.1
	Residual dropout probability (at all residual connections)	0.2
	Word embedding dropout probability	0.4
	Dropout probability for part-of-speech tag embeddings	0.2
	Dropout probability for CharConcat/CharLSTM morphological representations	0.2
	Character embedding dropout probability at the inputs to CharLSTM	0.2

Table 8: Model hyperparameters used for all of our experiments.

A Supplementary Material

A.1 Model Hyperparameters

The hyperparameters for our model are shown in Table 8. Hyperparameters were tuned on the development set for English.

A.2 Optimizer Parameters

Our model was trained using Adam with a batch size of 250 sentences. For the first 160 batches (equal to 1 epoch for English), the learning rate was increased linearly from 0 up to the base learning rate shown in Table 9. Development-set performance was evaluated four times per epoch; if it did not improve for 5 epochs in a row the learning rate was halved. The iterate that performed best on the development set was taken as the output of the training procedure.

To ensure stability of the optimizer, we found it important to use a large batch size, to warm up the learning rate over time (similar to Vaswani et al. (2017)), and to pick an appropriate learning rate.

Language	Base Learning Rate
English	0.0008
Hebrew	0.002
Polish	0.0015
Swedish	0.002
All others	0.0008

Table 9: Learning rates.

A.3 Position Embeddings

All variations of our model use learned position embeddings. Our attempts to use the sinusoidal position embeddings proposed by Vaswani et al. (2017) consistently performed worse than using learned embeddings.