Leveraging Linguistic Structures for Named Entity Recognition with Bidirectional Recursive Neural Networks

Peng-Hsuan Li
National Taiwan University
No. 1, Sec. 4, Roosevelt Rd.
Taipei 10617, Taiwan
jacobvsdanniel@gmail.com

Ruo-Ping Dong
National Tsing Hua University
No. 101, Sec. 2, Kuang-Fu Rd.
Hsinchu 30013, Taiwan
dongruoping@gmail.com

Yu-Siang Wang
National Taiwan University
No. 1, Sec. 4, Roosevelt Rd.
Taipei 10617, Taiwan
b03202047@ntu.edu.tw

Ju-Chieh Chou
National Taiwan University
No. 1, Sec. 4, Roosevelt Rd.
Taipei 10617, Taiwan
jjery2243542@gmail.com

Wei-Yun Ma
Academia Sinica
No. 128, Sec. 2, Academia Rd.
Taipei 11529, Taiwan
ma@iis.sinica.edu.tw

Appendix A CNN and Highway

Our character-level embeddings are computed by passing one-hot character vectors of each word through a series of convolutional and highway layers.

For each token, the first step is the preprocessing. If a token is longer than 18 characters, e.g. a web address, its tail is cut down to make it 18 characters long, and then a special *start* character is prepended and a special *end* character is appended. Otherwise, the token is appended with special *padding* characters to make it 18 characters long before prepending *start* and appending *end*. The unification of length is to facilitate batch (parallel) processing. The decoration of *start* and *end* is to let a window-shifting model like CNN know if it is looking at the prefix or suffix of a token.

The second step is to form a matrix from the 20-length character sequence. Every character is transformed into a one-hot vector except for *padding*, which is transformed into a zero vector. Suppose there are n distinct characters including *start* and *end* in the corpus. Then the result will be a 20-by-n matrix.

The third step is parallel convolutions. The convolution kernels might have different heights, a.k.a. window sizes, but they are all 20 in width. Each of the convolution outputs of the kernels is max-pooled to a scalar, representing if the token has the pattern captured by the corresponding kernel. As a result, if there are m kernels, the CNN outputs a m-dimensional character-level embedding for the token.

Finally, the vector is further transformed by a highway layer. This captures the potential interactions between patterns. For each vector u, the output layer computes a vector $v\prime$ with the same length.

$$t = \sigma(uW_t + b_t)$$
$$v = ReLU(uW_v + b_v)$$
$$v\prime = (1 - t) \odot u + t \odot v$$

Essentially, $v\prime$ is the weighted sum of u and its transformation v. b_t is initialized by a Gaussian distribution with negative mean so the layer initially passes its input to the output like a highway.

Appendix B Parameters

Parameters of BRNN-CNN are contained in the CNN, the highway layer, the BRNN hidden layers, and the output layer. In addition, the word embedding look-up table is also trainable.

B.1 Initialization

Except for the look-up table and b_t, all other parameters are initialized with Xavier initializer. The

Hyperparameter	Trial Range	Final Setting
Input Layer		
Pre-Trained Word Vectors	50d-Collobert, 300d-GloVe	300d-GloVe
Character Vector Dimension	ont-hot, 25	ont-hot
Maximal Kernel Height	3, 5	3
Kernels for Each Height	$h \times 20$, $h \times 40$	$h \times 40$
Lexicons (Non-Exclusive)	SENNA, DBpedia	PER, ORG, LOC of SENNA
Hidden Layers		
Number of Hidden Layers	1, 2, 3	3
Hidden Vector Dimension	300, 350, 400, 450, 500, 600, 1200	350
Optimization		
Learning Rate of Adam	1e-5, 1e-4, 1e-3	1e-5
Epsilon of Adam	1e-8, 1e-4, 1e-2, 1e-1, 1e-0	1e-2
Keep Rate of Dropout	0.65, 0.70, 0.75, 0.80, 0.90, 1	0.65

Table 1: Hyperparameter selection for OntoNotes 5.0 NER.

Layer	Weights and Biases
Word Embeddings	Look-up table
CNN	Kernels
Highway Layer	W_t, W_v, b_t, b_v
Hidden Layers	W_{bot}, W_{top}, b_{bot}, b_{top}
Output Layer	W_{out}, b_{out}

Table 2: Parameters of BRNN-CNN.

initializer tries to ensure the scale of output values for deep networks. This is desirable because BRNN-CNN might go down an indefinitely deep recursion.

For word embedding look-up table, if a pre-trained GloVe vector could be found for a word, the corresponding row is initialized by that vector. Otherwise, the Gaussian distribution with zero mean is used.

B.2 Early Stopping

After each epoch of parameter updates with the training corpus, the validation F1 score is checked. Since initialization, the best score is kept. If a record has not been broken for 20 epochs, the training stops and the parameter values that achieved the best score are restored.

B.3 Dropout

Dropout layers are added for the input layer and the hidden layers. For each node i, elements of I_i (excluding lexicon features), $H_{bot,i}$, and $H_{top,i}$ are randomly zeroed in training time.

Appendix C Hyperparameters

The hyperparameters of BRNN-CNN includes the dimensions of layers, initialization options, and parameters of optimization algorithms.

For the word-level vectors, the word vector dimension is decided alongside the pre-trained vectors used for initialization. One set of the pre-trained word vectors is the 300-dimensional vectors trained by GloVe on an 840 billion-token web corpus. The other is the 50-dimensional vectors of lower-cased tokens released by Collobert in his SENNA system.

For character-level vectors, one-hot vectors can be used. However, a trainable character embedding look-up table could also be used if the corpus is large enough. As for the kernels, there are 40, 80, and 120 kernels for each height 1, 2, and 3.

For lexicon features, which external resources are used need to be decided. The lexicons released with the SENNA system includes PER, ORG, LOC, and MISC. Besides, lexicons can be extracted from DBpedia, the ontology of Wikipedia. For example, a lexicon of person names could be constructed by including every instance falling into a type in the subtree rooted by the *person* type.

For hidden layers, the best dimension is actually dependent on the number of layers. In Table 1, 350 is selected, but 450 is selected if there is only one hidden layer per direction.

The final values of these hyperparameters are mostly searched greedily. A better configuration could very likely be found by a comprehensive grid search.