NCRF++: An Open-source Neural Sequence Labeling Toolkit

Jie Yang, Yue Zhang
Singapore University of Technology and Design

Overview

Motivation: build a unified deep learning based sequence labeling framework which is efficient and handy.

Sequence Labeling: assign a categorical label to each member of a sequence inputs, e.g., POS tagging, NER, Chunking.

Models: **Representation** + **Inference**

- **Represent**: discrete (manual features)
  neural (LSTM, CNN)
- **Inference**: softmax
  CRF, HMM, MEMM

NCRF++: based on PyTorch, compatible with Python 2 and 3. Neural version of CRF++.

Framework: layer-wise design (Figure 1).

- **Char seq. layer**: char LSTM/GRU/CNN*
- **Word seq. layer**: word LSTM/GRU/CNN
- **Inference layer**: softmax/CRF

* manual feature embeddings are also supported.

Advantages:

- **Fully configurable**: user can customize structure with a configuration file, no code work. (Figure 2)
- **Flexible with features**: it integrates SOTA char/word neural features and also supports user-defined features.
- **Effective**: it gives comparable performance with SOTA sequence labeling models. (Table 1)
- **N-best**: it supports n-best CRF decoding which gives more candidate labels. (Figure 3)
- **Efficient**: batched implementation lead to a fast running speed. (>2000 sent/s, in Figure 4)

```
##NetworkConfiguration##
use_crf=True
word_seq_feature=LSTM
word_seq_layer=1
char_seq_feature=CNN
feature=[POS] emb_dir=None emb_size=10
feature=[Cap] emb_dir=%(cap_emb_dir)
##Hyperparameters##
...
```

Figure 2: Configuration file segment

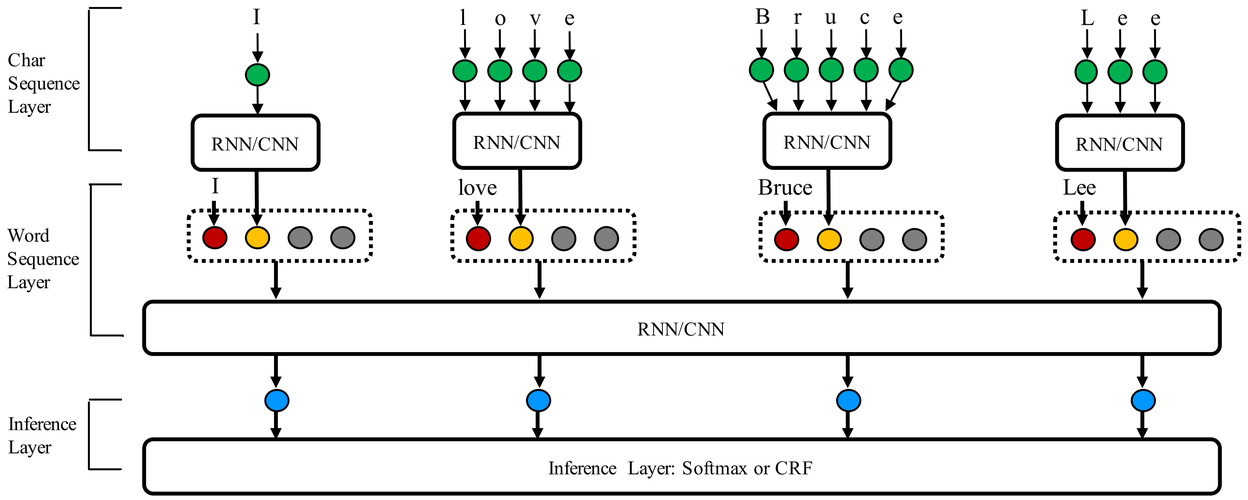

Figure 1: NCRF++ Framework. Circle meaning: Green=char embeddings; Red=word embeddings; Yellow=char sequence rep; Blue=word sequence rep; Grey=feature embeddings.

Experiments

Tasks and Datasets:

- **Named Entity Recognition (NER)**: CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003).
- **Chunking**: CoNLL 2000 share task (Tjong Kim Sang and Buchholz, 2000).
- **Part-of-speech tagging (POS)**: WSJ portion of PTB, same split with Ma and Hovy (2016).

Experiments:

- **6 CRF frameworks:** {Char LSTM (CLSTM), Char CNN (CCNN), Nochar}
  x {Word LSTM (WLSTM), Word CNN (WCNN) }
- **Settings:** Glove 100 embedding; SGD optimizer; batch_size=10; dropout=0.5
- **Evaluation:** Best result under 5 random seeds, detail and statistical results are shown in our COLING 2018 paper "Design Challenges and Misconceptions in Neural Sequence Labeling".

Performance: (Table 1)

- CLSTM+WLSTM+CRF has same structure with Lample et al (2016), gives similar results.
- CCNN+WLSTM+CRF has the same structure with Ma and Hovy (2016) and Yang et al. (2017), they have comparable performance.
- Word LSTM are generally better than word CNN under all settings.
- Character features (LSTM/CNN) are useful for all three tasks.
- Char LSTM and Char CNN can give comparable improvements.

Models	NER (F1)	Chunk (F1)	POS (Acc)
Nochar+WCNN+CRF	88.90	94.23	96.99
CLSTM+WCNN+CRF	90.70	94.76	97.38
CCNN+WCNN+CRF	90.43	94.77	97.33
Nochar+WLSTM+CRF	89.45	94.49	97.20
CLSTM+WLSTM+CRF	91.20	95.00	97.49
CCNN+WLSTM+CRF	**91.35**	**95.06**	97.46
Lample et al. (2016)	90.94	--	97.51
Ma and Hovy (2016)	91.21	--	**97.55**
Yang et al. (2017)	91.20	94.66	**97.55**
Peters et al. (2017)	90.87	95.00	--

Table 1: Model performance

Analysis

N-best decoding:

- Figure 3 is evaluated on NER test dataset.
- N-best decoding provides more candidate labels than the ordinary decoding.
- Oracle F1 increases from 91.35% to 97.47% in 10-best decoding.
- Oracle token accuracy increase from 98.00% to 99.39% in 10-best decoding.

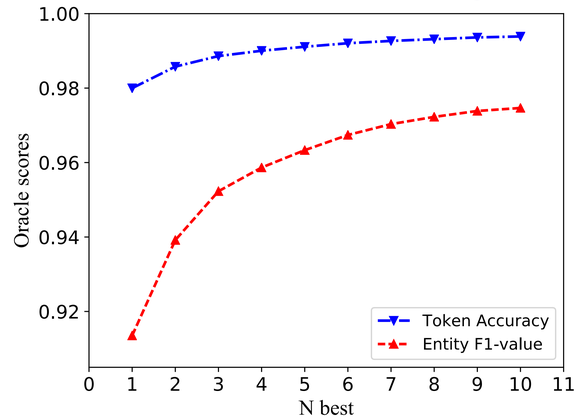

Figure 3: Oracle NER F1 in n-best decoding

Running Speed:

- Running speed increases significantly with the increment of batch size. Decoding speed starts saturating at batch_size=200 but training process doesn't.
- **Fast:** training speed reaches 1000 sents/s and decoding speed exceeds 2000 sents/s under large batch with GPU acceleration.

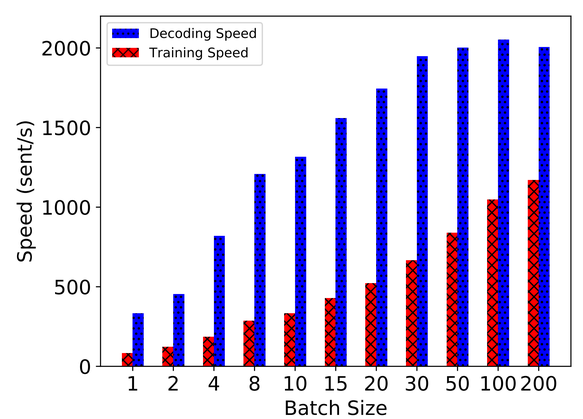

Figure 4: Speed with batch size (NER task)