A Chinese Named Entity Recognition System with Neural Networks

[1]Hui-Kang Yi, [2]Jiu-Ming Huang, [3]Shu-Qiang Yang

[1]*National University of Defense Technology School of Computer Science, Changsha, Hunan, 410073, China*
[2]*National University of Defense Technology School of Computer Science, Changsha, Hunan, 410073, China*
[3]*National University of Defense Technology School of Computer Science Changsha, Hunan, 410073, China*
[1]*huikangyi@gmail.com*

Abstract: Named entity recognition (NER) is a typical sequential labeling problem that plays an important role in natural language processing (NLP) systems. In this paper, we discussed the details of applying a comprehensive model aggregating neural networks and conditional random field (CRF) on Chinese NER tasks, and how to discovery character level features when implement a NER system in word level. We compared the difference between Chinese and English when modeling the character embeddings. We developed a NER system based on our analysis, it works well on the ACE 2004 and SIGHAN bakeoff 2006 MSRA dataset, and doesn't rely on any gazetteers or handcraft features. We obtained F1 score of 82.3% on MSRA 2006.

1. Introduction

Named entity recognition (NER) is a subtask of Information Extraction (IE), and its performance is vital to the related tasks like entity coreference resolution and entity relationship extraction. These essential text processing tasks are import steps to deep language understanding systems.

NER task shares similar modeling approach with other sequential labeling problems (SLPs) like word segmentation, Part-of-Speech tagging (POS). a SLP system is to generate corresponding tag sequence giving the token sequences. Before the rise of Neural Networks, most popular SLP systems were modeled with probabilistic graph (statistical model), including hidden Markov modeling (HMM), maximum entropy modeling (ME), and conditional random field (CRF) [1]. For example, the widely-used Stanford NER kit [2] is trained at dataset CoNLL 2003 [3] with CRF. Most high-performance statistical models rely heavily on carefully hand-craft features, especially in Chinese files. And many of they are task-specific, so that they are difficult to adjust to new datasets or domains.

NER tasks is very challenging for two main reasons: 1) lack of well-tagged training data. the widely used NER benchmark CoNLL 2003 contains only 14,987 sentences, 204,567 tokens. 2) Comparing to other SLPs, the construct of an entity name is rather complicated. a name can be very long or very short; any word can be a name with few constraints. So NER systems normally gain lower F1 scores than word segmentation, POS tagging. And for Chinese NER task, the situation can be even more rough (see section II).

To overcome the limitation of well-tagged data, unsupervised learning methods are designed to give better generalization from small amounts of training data. As neural networks arise, SLP Modeling with neural networks draws people's attention. Collobert at al. [4] proposed a convolutional neural network (CNN) based models for SLPs. Also, recurrent neural networks (RNN) [5] and its variants such as long-short Term memory (LSTM) [6] are widely used in NLP tasks. LSTM can learn long distance dependencies for text processing. Huang et al. [7] developed a general sequential tagging model combining bidirectional Recurrent Neural Networks (RNNs) and CRFs, which achieves near state-of-the-art results on both POS and NER. Lample et al. [8] gives a more specific implementation of neural network model for NER tasks, their model introduced pretrained word embeddings from large corpus, and gives F1 scores of 90.97% on CoNLL 2003 dataset using no context features or spelling features. Ma et al. [9] combines Chiu's CNN model [10] and Lample's bidirectional LSTM-CRF model [8] for end-to-end SLPs. These models use neural networks strategies and/or pretrained word embeddings to give better generalization performance. And meanwhile, they show how far a NER system can go, without carefully designed hand-craft features.

In this paper, we mainly focus on two parts to adapt the neural network models for Chinese NER tasks: 1) discovery character level features for a word-level NER system; 2) train better word embeddings for Chinese NER tasks. We discussed details of Chinese NER systems (Section II) by comparing; how to train word embeddings (Section III) and use neural network and CRFs to model context dependencies.

2. Chinese Named Entity Recognition Tasks

A typical SLP, takes text sequence $\mathbf{X} = (x_1, x_2 \cdots, x_n)$ as input, and corresponding tag sequence $\mathbf{y} = (y_1, y_2 \cdots, y_n)$ as output. The goal of SLP is to establish the conditional probability model $P(\mathbf{y}|\mathbf{X})$. For NER tasks, the output sequence $\mathbf{y}$ are the corresponding named entity types, such as *person*, *location*, *organization* and etc.

2.1 Challenges of Chinese NER

Comparing the English and other west languages, NER tasks for Chinese text face more challenges for some reason: 1) The concept of Chinese words is not clear and can't not be define with a universal standard. Chinese words are not separated by spaces; 2) English text have obvious features can indicate an entity name, like capitalized terms and different tense and article words, but Chinese words never change in all kinds of tense or part-of-speech; 3) Almost every Chinese character can be part of an entity name, so we can't build a vocabulary for each kind of entity type.

2.2 Word Segmentation and NER

Chinese character is the minimum lexical struct of texts. There are only 26 different English letters in total (regardless of case), but there are thousands of Chinese common characters. Chinese words, usually composed of several characters, that semantically related to the word. Take word "电脑" (computer) for example, character "电" means electricity or electronic and "脑" means brain. And the combination "电脑" (electronic brain) is used to describe the electronic machine that works like brains (computer). But for some words like transliterated words, their inner characters give mainly phonetic meaning and contribute no semantical features. Chinese word "巧克力" (chocolate) is a such typical transliterated word.

Chinese named entities are word-level. Normally, we perform word segmentation before. Word segmentation is also a SLP, and can be solved with similar model [11]. Ng et al. [12] discussed two approaches for the word segmentation issue in POS tasks: 1) performing NER tasks strictly after word segmentation (one-at-a-time); 2) performing word segmentation and NER tasks at same time (all-at-once). There are two challenges applying our model in character level regardless word segmentation: 1) an entity name can be much longer than a word in POS task; So high performance of parsing very long distance context dependencies required; 2) The ACE 2004 dataset, that we used for training, contains so many transliterated words; So character-based modeling is not suitable, instead, we considered introducing the character-level features in the word embedding layer.

3. Modeling

Our neural network modeling for named entity recognition task is similar to Lample et al. [8] and Chiu et al. [10]. We designed different heuristic function as loss of our model and compared two different word embedding model for Chinese NER dataset.

3.1 Word Embedding Layer

Pretrained word embeddings can improve model's performance comparing to random initial vectors [4]. Lample's [8] best results (F1 scores of 91.2 on CoNLL 2003 English training set) also generated with the model using pretrained word embedding from large corpus. In this part, we compared two different word embedding modeling approach: Skip-Gram [13] and CWE [14]. these two model improves performance significantly over randomly initialized vectors. CWE model is developed from the continuous Bag-of-Words model (CBOW), and it takes account the internal information of characters into Chinese words.

Chen [14] raised two challenges of character-word joint model for Chinese word embedding that could also happen in NER task: 1) A Chinese character might have various semantic meanings in different words; 2) Not all Chinese words are semantically compositional, such as transliterated words. In the area of similar tasks like POS, Ng et al. [12] found that character-based modeling slightly outperforms word-based ones and avoids biases introduced by word segmentation. But in our experiment, the character-based model didn't show its odds. We built our model in word-level and use pre-trained word embeddings trained by Skip-Gram model or CWE.

3.2 Neural Network for Encoding

Long short-term memory (LSTM), a kind of implementation RNN, is designed to combat the issue that RNN always fail to learn long dependencies in practice by introducing a memory-cell [6, 15]. Our encoding model is very close to Lample's [8]; a forward LSTM layer to encode input sequences from start to end and a backward LSTM layer to do the same in reverse. And for the concatenated outputs of LSTM layers, using another hidden layer before the next step (decoding) can improves performance obviously, see Figure. (1).

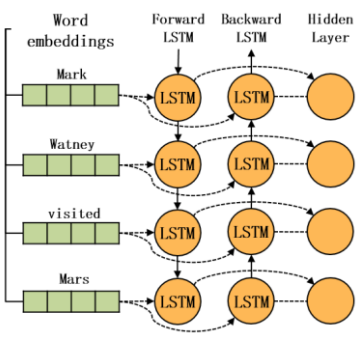

Figure 1. Bi-directional LSTMs with an extra hidden layer

3.3 Decoding Layer and Loss Function

We use Conditional Random Field (CRF) for decoding. For a input sentence

$$\mathbf{X} = (x_1, x_2 \cdots, x_n)$$

and a sequence of predictions

$$\mathbf{y} = (x_1, x_2 \cdots, x_n)$$

consider matrix $\mathbf{P}$ of size $n \times k$, where k is the number of distinct tags, $P_{i,j}$ corresponds to the score of the j^{th} tag of the i^{th} word in a sentence. And matrix $\mathbf{A}$ of size $k \times k$ is the matrix of transition scores, where $A_{i,j}$ represents the score of a transition from the tag i to tag j. We minimize the average log probability of the target words and log sum of transition costs:

$$\text{loss} = -\left(\frac{1}{N}\sum_{i=1}^{N} \ln P_{i,y_i} + c\frac{1}{N-1}\sum_{i=1}^{N-1} \ln A_{y_i,y_{i+1}}\right) \quad (1)$$

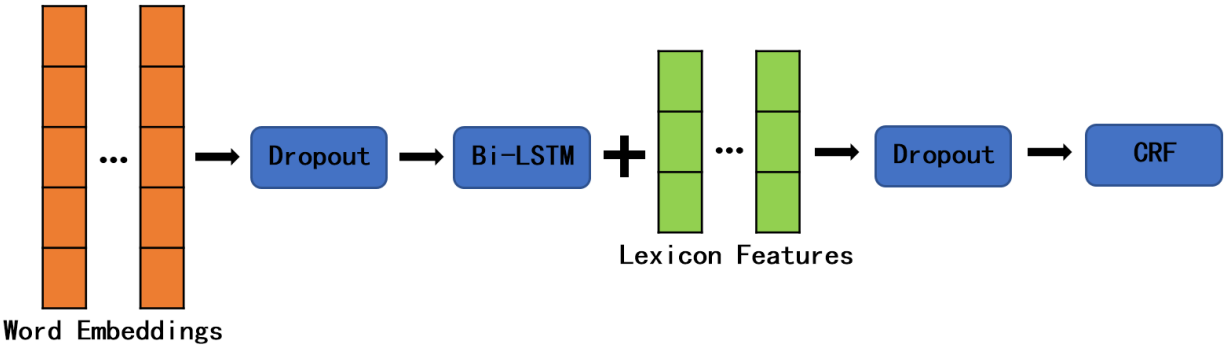

Figure 2. Order of Modeling Parts. Blue box represents process on the vectors; "+" means append a vector to another

In the experiment, we compared our loss function to the log
transition costs:

$$\text{loss} = -\left(\frac{1}{N}\sum_{i=1}^{N} \ln P_{i,y_i} + c\frac{1}{N-1}\sum_{i=1}^{N-1} \ln A_{y_i,y_{i+1}}\right) \quad (1)$$

In the experiment, we compared our loss function to the log likelihood function [1] Lample [8] used, our function gives better convergence efficiency. We also tested softmax function for decoding. in this way, only unary features are modeled, the loss function (equation. (1) becomes

$$\text{loss} = -\frac{1}{N}\sum_{i=1}^{N} \ln P_{i,y_i}$$

which is same as Mikolov et al. [13]. we use a simple greedy algorithm to get our prediction

$$y_i^* = arg\max_j P_{i,j}$$

and use viterbi algorithm to get prediction when using CRF to decode.

3.4 Lexicon Features

We didn't use any gazetteer to construct the lexicon features. Instead, we used the character-level statistics in training set only. We denote a Chinese character as c, a Chinese word as w, and w^j represents the j^{th} character in the word w, function $Tag(w)$ returns the entity type of word w. We count the how often every character that appears in different entity types k

$$Count_{k,j} = \sum_{c}\sum_{w,Tag(w)=k} Appear(c,w,j)$$

Where

$$Appear(c,w,j) = \begin{cases} 1, & if\ c = w^j \\ 0, & if\ c \neq w^j \end{cases}$$

then we use the sum of frequency of characters appearing in each tag as the lexicon features of words. We specified the offset of characters in a word, to expect giving more position related inner information to the model.

3.5 Order of Layers and Dropout

Technically, the word embeddings layer, bi-directional LSTM layer, lexicon features, are steps of feature modeling and they work independently. Word embedding gives semantic features; LSTM layer is used to modeling features of context dependency based on word embeddings; lexicon features is complement for character-level features. few experiments show that appending lexicon features between LSTM layer and decoding layer, out-performs other order of the feature processing layer. To avoid the model biases on one type of features over another, we apply a dropout mask before the LSTM layer and before the decoding layer. see Figure. (2)

3.6 Tagging Scheme

An entity mention refers to a single word or several continuous words in texts, so an informative tagging schemes is needed. Ratinov et al. [16] and Dai et al. [17] shows that, the schemes who give more information (BILOU for example), significantly outperforms BIO tagging scheme. But recent published NER systems include lamples' [8], did not observe a significant improvement from BIO to BILOU scheme. We accepted BILOU as our tagging scheme though all the experiments.

[1] provided by tensorflow 0.12.0

4. Experiment

4.1 Pretrained Word Embeddings

Based on the analysis in Section III, we use Wikipedia [18] as our backend corpus for training word embeddings. the original Wikipedia of articles include both simplified Chinese and traditional Chinese, we use *opencc* [2] to convert all the traditional Chinese text to simplified and deleted all the non-Chinese characters (most of they are reserved words of xml-format files). the Wikipedia dataset we used contains 260,969,652 characters, we filtered low-frequency words (mincount=5) and got a vocabulary of size 722,226. the embedding we used has dimension of 100.

the CWE model for word embedding requires a word list about transliterated words manually, we use the pre-trained embeddings (dimension=300) they provide instead.

4.2 Dataset Processing

The ACE 2004 dataset [19] consists news texts of three different languages: *Arabic*, *Chinese*, *English*. We use the Chinese data to evaluate our model. the original data (Chinese part) contains 314 files of broadcast news, 226 files of newswire and 106 files of Chinese Treebank. ACE 2004 dataset is for Automatic Content Extraction (ACE) technology evaluation. There are 6 types of named entities in ACE 2004: *FAC*, *GPE*, *LOC*, *ORG*, *PER*, *VEH*. We reorganized the data for NER tasks through few steps: 1) 1. get texts from *.xml* files and entity mentions and their offset; 2) use word segmentation tool [3] to get token sequence and get corresponding tag sequence; 3) let first 80% files be the training data, following 10% be the validation set, and last 10% be the test. By now, we obtained three different datasets *(bnews*, *Chinese Treebank*, *nwire*) for NER tasks.

We also tested our model on SIGHAN bakeoff 2006 Named Entity Recognition MSRA (MSRA 2006) task dataset. There are three types of named entities: *person*, *location*, *organization*. the training data of MSRA 2006 is well segmented but the test data is plain text. We rearrange the training data to plain text and applying *jieba* to segment the text (both training dataset and test dataset) into word sequence. So that our model would not biases by different word segmentation methods, since *jieba* tends to give slightly longer tokens.

And For all the data we used, we replaced all digits with zeros.

4.3 Traning Setting

For input data, we add two masks $< \text{BOS} >$ (Beginning of sentence) and $< \text{EOS} >$(end of sentence) to every sentence, so that $\mathbf{X} = (< \text{BOS} >, x_1, x_2 \cdots, x_n, < \text{EOS} >)$. the corresponding tag of the two mask are *U-BOS*, *U-EOS*. Then we concatenate these sentences to form our input data (A sequence of words) and cut the input sequence into lines with fixed length (=35). Our model was trained by these lines with a batch size of 20.

We compared two kinds of LSTM cells in the bi-directional LSTM layers: GRU [20], BasicLSTM [21]. Model with these two different cells give very close results, our conclusion results are all generated with a BasicLSTM cell. Increase number of layers in bidirectional LSTM doesn't improve the performance and cost more time, we use a single layer in bi-LSTMs. Before feeding word embeddings into LSTMs, we apply a dropout mask with a keep probability of 0.5.

Out model was built on Tensor flow (version=0.12.0)[4]. For all the data we used, we use *conlleval*[5] to evaluate our model in a word level.

The time for training a model is limited to 4 hours for ACE 2004 datasets and 6 hours for MSRA 2006, since we didn't observed improvement through long hours training. In fact, we found training with the loss function 1 gives same scores (statistically) comparing to log likelihood function [8] with much faster convergence efficiency.

4.4 Results

As a final experiment, we have trained our system on 4 different datasets for Chinese NER task.

Table I shows the results of different model on MSRA 2006. A single CRF parsing with only word embeddings gives poorly scores; But the CRF layer gives an increase of +3.13, while using bi-directional LSTM encoding to encode the word embeddings; Our lexicon features are extracted within the training data, it's boost to the performance is very limited. Using dropout resulted in a difference of +1.51. In fact, one dropout layer before the bi-directional LSTM layer can improve the scores significantly, we applied a dropout layer when join the lexicon features and outputs of LSTMs.

Table II shows our model's performance on different dataset. All three ACE 2004 datasets are from international political news. Our model could barely recognize the *WEA*s (weapon), especially on the Chinese treebank dataset whose size is smallest). but it achieved a F1 score of 89.87 recognizing *person* names on ACE 2004 *bnews* dataset.

Table1. Different Models on Sighan Bakeoff Msra 2006

Model	F1/dev	F1/test
CRF	57.33	56.22
Bi-LSTM+Softmax	77.81	76.94
Bi-LSTM+CRF	81.56	80.07
Bi-LSTM+Lexicon+CRF	81.13	80.79
Bi-LSTM+Lexicon+CRF+Dropout	82.42	82.30

[2] http://opencc.byvoid.com/

[3] Jieba: https://github.com/fxsjy/jieba. In this step, we applied word segmentation after we got the boundaries of entity mentions, so jieba won't bring any boundary error for the next NER task

[4] https://www.tensorflow.org

[5] http://lcg-www.uia.ac.be/conll2000/chunking/

Table2. Full Model on Different Dataset

Dataset	Vocab	Tokens	F1
ACE2004 *bnews*	12,824	62,671	79.55
ACE2004 *Chinese Treebank*	5,857	24,688	78.79
ACE2004 *nwire*	12,662	61,070	79.48
MSRA 2006	72,821	1,240,945	82.30

5. Conclusion

In this Paper, we discussed details of applying a neural architecture for Chinese named entity recognition tasks. Comparing to English NER tasks, we dressed the issue of extracting Chinese character level features and compared with POS tasks. It is meaningful to model the characteristics and contexts with appropriate model and gradually abandon the handcraft features for NLP tasks such as named entity recognition. We presented a simple model for Chinese NER that combines recently published neural network models. And it doesn't rely on any hand-craft features. Several popular NER system for English [4, 8, 9, 10] uses CNNs or LSTMs to model character level features. In our next step, we will continue our study, from a linguistic point of view, on modeling Chinese characteristics with neural networks to improve our model's performance.

Acknowledgement

The work described in this paper is partially supported by The National Key Research and Development Program of China (2016QY03D0601, 2016QY03D0603) and National Natural Science Foundation of China (No.61502517, No.61672020, No.61662069).

References

[1] John Lafferty, Andrew McCallum, Fernando Pereira, et al. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proceedings of the eighteenth international conference on machine learning, ICML, volume 1, pages 282–289, 2001.

[2] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In Proceedings of the 43rd annual meeting on association for computational linguistics, pages 363–370. Association for Computational Linguistics, 2005.

[3] Erik F Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4, pages 142–147. Association for Computational Linguistics, 2003.

[4] Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. Journal of Machine Learning Research, 12(Aug):2493– 2537, 2011.

[5] Christoph Goller and Andreas Kuchler. Learning task-dependent distributed representations by backpropagation through structure. In Neural Networks, 1996., IEEE International Conference on, volume 1, pages 347–352. IEEE, 1996.

[6] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. ¨ Neural computation, 9(8):1735–1780, 1997.

[7] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. arXiv preprint arXiv:1508.01991, 2015.

[8] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. arXiv preprint arXiv:1603.01360, 2016.

[9] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bidirectional lstm-cnns-crf. arXiv preprint arXiv:1603.01354, 2016.

[10] Jason PC Chiu and Eric Nichols. Named entity recognition with bidirectional lstm-cnns. arXiv preprint arXiv:1511.08308, 2015.

[11] Koth. kcws. https://github.com/koth/kcws, 2016.

[12] Hwee Tou Ng and Jin Kiat Low. Chinese part-of-speech tagging: Oneat-a-time or all-at-once? word-based or character-based? In EMNLP, pages 277–284, 2004.

[13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013.

[14] Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huan-Bo Luan. Joint learning of character and word embeddings. In IJCAI, pages 1236–1242, 2015.

[15] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning longterm dependencies with gradient descent is difficult. IEEE transactions on neural networks, 5(2):157–166, 1994.

[16] Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In Proceedings of the Thirteenth Conference on Computational Natural Language Learning, pages 147–155. Association for Computational Linguistics, 2009.

[17] Hong-Jie Dai, Po-Ting Lai, Yung-Chun Chang, and Richard Tzong-Han Tsai. Enhancing of chemical compound and drug name recognition using representative tag scheme and fine-grained tokenization. Journal of cheminformatics, 7(1):S14, 2015.

[18] Wikipedia:Database download, 2017 (accessed April 4, 2017). https://dumps.wikimedia.org/zhwiki/latest/zhwiki-latest-pages-articles. xml.bz2.

[19] Shudong Huang Ramez Zakhary Alexis Mitchell, Stephanie Strassel. Ace 2004 multilingual training corpus ldc2005t09. Web Download., 2005.

[20] Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bah- ¨ danau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078, 2014.

[21] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. arXiv preprint arXiv:1409.2329, 2014.