Neural Language Modeling for Named Entity Recognition

Zhihong Lei[1] **Weiyue Wang**[2] **Christian Dugast**[2] **Hermann Ney**[2]
[1]Apple Inc.
[2]Human Language Technology and Pattern Recognition Group
Computer Science Department
RWTH Aachen University
zlei@apple.com
{wwang, dugast, ney}@cs.rwth-aachen.de

Abstract

Regardless of different word embedding and hidden layer structures of the neural architectures that are used in named entity recognition, a conditional random field layer is commonly used for the output. This work proposes to use a neural language model as an alternative to the conditional random field layer, which is more flexible for the size of the corpus. Experimental results show that the proposed system has a significant advantage in terms of training speed.

1 Introduction

Named entity recognition (NER) is a natural language processing (NLP) task that searches for names in a text and assigns them into several predefined categories. With the help of various neural network architectures, NER systems nowadays achieve very promising performance for tasks with a limited number of entity types (Akbik et al., 2018; Devlin et al., 2019; Jiang et al., 2019).

Besides various types of embedding techniques and long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) recurrent neural networks (RNN), another important component of those systems is the conditional random field (CRF) layer (Lafferty et al., 2001). The CRF layer is very effective for tasks such as NER and part-of-speech (POS) tagging. However, the training time of such a model increases quadratically with the vocabulary size, which is the number of different entity types in the case of NER.

To tackle this problem, we use LSTM-based neural language models (LM) on tags as an alternative to the CRF layer. With a separately trained LM (without using additional monolingual tag data), the training of the new system is about 2.5 to 4 times faster than the standard CRF model, while the performance degradation is only marginal (less than 0.3%). Thanks to its time efficiency, our system can easily be applied to corpora containing hundreds and thousands of entity types.

In addition, the LSTM-based LM potentially can perform better as it encodes more contextual information than the CRF layer. To unlock the full power of the LM, we also try to train the tagging model and LM jointly at the sequence level. In this case we lose the speed advantage, but the jointly trained system achieves comparable performance as the state-of-the-art NER model on four different corpora.

2 Background

In a bidirectional LSTM (BLSTM) CRF-based NER system, the score of a name entity sequence for a given word sequence is defined as

$$s_\theta(x_1^T, y_1^T) = \sum_{t=1}^{T} \left(A_{y_{t-1}, y_t} + \text{BLSTM}(x_1^T)_{y_t, t} \right) \tag{1}$$

The first component of the score models the transitions between the current tag at time t and its predecessor. Therefore, it is referred to as *transition score*. The second term models the dependencies of the current tag y_t on the word sequence x_1^T and it can be called *emission score*. The probability of the tag sequence can be defined as

$$P_\theta(y_1^T | x_1^T) = \frac{\exp\left(s_\theta(x_1^T, y_1^T)\right)}{\sum_{\tilde{y}_{t=1}^T} \exp\left(s_\theta(x_1^T, \tilde{y}_1^T)\right)} \tag{2}$$

[1]This work was done when Zhihong Lei was studying at RWTH Aachen University.

where θ denotes the trainable parameters of the model, including the transition matrix A and free parameters of the BLSTM model. The training objective is therefore to maximize the log likelihood of the truth tag sequence

$$\theta = \arg\max_{\hat{\theta}} \log P_{\hat{\theta}}(y_1^T | x_1^T) \tag{3}$$

It is to be noted that the denominator of the probability in Equation 2 is calculated from scores of all possible tag sequences. For example, suppose the total number of unique tags is Y, the computation complexity of the sequence probability is $\mathcal{O}(Y^T)$ (ignoring the computational complexity of BLSTM features). Since the CRF layer models only a first-order dependency, dynamic programming can be employed in training, reducing the time complexity to $\mathcal{O}(TY^2)$.

The objective of decoding is to find a tag sequence that maximizes the score for given word sequence and model parameters:

$$\tilde{y}_1^T = \arg\max_{\hat{y}_1^T} s_\theta(x_1^T, \hat{y}_1^T) \tag{4}$$

CRF based models have long been applied in sequence tagging problems. Compared to other non-CRF models, such as a pure BLSTM model, CRF models offer two advantages: first, explicit modeling of transitions between tags; second, optimization at the sequence level.

Despite the benefits, the training time of BLSTM-CRF models increases quadratically with the dimension of the tag set. On the other hand, the CRF layer models a first-order dependency of the tag sequences. Intuitively, it might be helpful to model a higher-order dependency, although this could aggravate the training problem. To this end, we propose a hybrid system, which models the tag sequence dependencies with an LSTM-based LM rather than CRF.

3 Hybrid System with Neural Language Models

By taking the logarithm of the sequence probability, we obtain

$$\log P_\theta(y_1^T | x_1^T) = s_\theta(x_1^T, y_1^T) - \log \sum_{\hat{y}_{t=1}^T} \exp\left(s_\theta(x_1^T, \hat{y}_1^T)\right) \tag{5}$$

The score of the tag sequence can be replaced by the right terms of the Equation 1. We observe that the summation of the transition scores is very similar to the log probability of the tag sequence, which is calculated from a first-order LM. If we replace the transition matrix A with an LSTM-based tag LM, the sequence score can be defined as

$$s_\theta(x_1^T, y_1^T) = \sum_{t=1}^{T} \left(\lambda \log P_{\text{LSTM}}(y_t | y_0^{t-1}) + \text{BLSTM}(x_1^T)_{y_t,t}\right) \tag{6}$$

Note that in practice, we add a scale λ to the log probabilities.

However, due to the introduction of the LSTM-based LM, sequence training with dynamic programming is no longer feasible. The following sections discuss two different approaches to train the hybrid model.

3.1 Separate Training

The training objective is to maximize the sequence score of the true tags, i.e. $s_\theta(x_1^T, \hat{y}_1^T)$. We note that the first term and the second term of the new sequence score are independent of each other, eventually leading to cross-entropy training of the BLSTM-based tagging model and the LSTM-based LM separately:

$$s_\theta(x_1^T, y_1^T) = \sum_{t=1}^{T} \left(\lambda \log P_{\text{LSTM}}(y_t | y_0^{t-1}) + \log P_{\text{BLSTM}}(y_t | x_1^T)\right) \tag{7}$$

Since no additional monolingual data is used and the tag set is usually quite small (compared to the word vocabulary), training such a tag LM is very cheap. The computational complexity is therefore dominated by the BLSTM-based tagging model, whose training is much faster than that of the CRF version. Compared to the CRF layer, the computational complexity of the softmax is $\mathcal{O}(TY)$. During decoding, we apply beam search and calculate the transition scores from the LSTM-based LM.

3.2 Joint Training

In addition to the separate training approach, the LSTM-based LM can also be trained jointly with the tagging model. In this case, a single training loss is computed and back-propagated to both LM and BLSTM-based tagging model.

To jointly train the hybrid model at the sequence level, theoretically the scores for all possible tag sequences must be calculated. The dynamic programming algorithm used for the first-order transition model is infeasible for our joint training with the LSTM-based LM.

A number of publications dealt with issues related to sequence training, particularly those related to optimizing the search process (Wiseman and Rush, 2016; Daumé III and Marcu, 2005). Inspired by them we develop a straightforward training method. Looking at the denominator in Equation 2, although it is not feasible to go through all tag sequences, we can estimate the denominator by considering only those hypotheses with the highest score and that can be generated from the beam search. The log likelihood of the true tags can be approximated by

$$\log P'_\theta(y_1^T|x_1^T) = \underbrace{s_\theta(x_1^T, y_1^T)}_{\text{gold score}} - \underbrace{\log \sum_{\hat{y}_{t=1}^T \in \text{beam}} \exp\left(s_\theta(x_1^T, \hat{y}_1^T)\right)}_{\text{beam score}} \tag{8}$$

and it can be written as the difference between the gold score and the beam score. The beam search enables us to generate the top K hypotheses and their sequence scores, from which the beam score can then be calculated.

The problem with this approximation is that maximizing the log likelihood may penalize the beam score. This is inconsistent with decoding because the best hypothesis is generated from the beam. Therefore, we want the best scored hypotheses to stay on the beam. To address this issue, we refer to Wiseman and Rush (2016), in which subsequent candidates are generated from the true token when it falls off the beam. We adopt this idea and develop our simplified method: for each time step, we replace the K-th candidate with the true tag if it is not included in the K-best list.

4 Experiments

We experiment on four benchmark NER tasks in three languages: CoNLL 2003 English/Dutch, OntoNotes English and GermEval 2014 German.

4.1 Models

As we conduct all our experiments using Flair (Akbik et al., 2019a), we adopt the recommended setup for the baseline BLSTM-CRF models. We use GloVe word embeddings (Pennington et al., 2014) for the CoNLL03 English task, while FastText embeddings (Mikolov et al., 2018) for CoNLL03 Dutch and GermEval, and FastText web crawl embeddings for OntoNotes are used. Pooled Flair embeddings (Akbik et al., 2019b) are used for all experiments except for OntoNotes, for which the non-pooled version (Akbik et al., 2018) is used. The BLSTM has a single 256-unit forward and backward LSTM layer followed by a CRF layer. For non-CRF baseline models, the CRF layers are simply removed leaving all other parameters unchanged. For each task, we train four CRF and non-CRF models respectively.

In all our hybrid model experiments, the BLSTM tagger component has the same configuration as the non-CRF baselines, while the LSTM-based tag LM has an embedding matrix of size $|\text{Tags}| \times 10$, and an LSTM layer of 50 units.[1]

4.2 Results

The F1 scores of all different models are shown in Table 1. The baseline BLSTM models with or without the CRF layer are indicated by *CRF* and *Non-CRF*. Hybrid models are denoted by *Hybrid*, and *separate* and *joint* indicate the used training strategy. The joint training method with forcing true tags in the beam is indicated by *force*.

[1]Code is available at `https://github.com/ZhihongLei/flair`

Model	CoNLL03 en	CoNLL03 nl	GermEval14	OntoNotes en
CRF	93.02	89.48	84.49	88.88
Non-CRF	92.53	88.55	82.95	87.39
Hybrid separate	92.89	89.44	84.61	88.49
Hybrid joint	93.26	89.82	84.50	88.82
Hybrid joint force	93.17	89.88	84.57	89.01

Table 1: Experimental results on F1 scores. The baseline scores are an average of four different runs.

As shown in Table 1, the separately trained hybrid models significantly outperform the non-CRF baselines and performs slightly worse than the CRF models with respect to the F1 score. The jointly trained hybrid models perform comparably to the baseline CRF models. The modified sequence training method with forcing true tags in the beam does not help, probably because the impact of the problem described in Subsection 3.2 decreases with increasing beam size.

4.3 Training Time

To evaluate the training time, we perform all relevant experiments with one NVIDIA GeForce 1080 Ti. When using pre-trained embeddings, training time is dominated by the CRF layer due to low GPU utilization and high complexity, and CRF models require much more training time than non-CRF ones. Therefore, the separately trained hybrid model, which is intrinsically a non-CRF NER model and an LSTM-based tag LM, has an advantage over the CRF models in terms of training time. Table 2 shows the total training time of the baseline BLSTM-CRF-based NER model and our separately trained hybrid model. Both models are trained to converge with a similar number of epochs. We can see that the training of the hybrid model is about 2.5 to 4 times faster than that of the CRF model. Also, the more entity types the corpora contain, the greater the speed advantage.

Model	CoNLL03	OntoNotes
Baseline	3.9	47.3
Hybrid separate	1.6	11.1

Table 2: Total training time in hours. For separately trained hybrid models, the training time is calculated by including both the tagging model and the LM.

5 Discussion and Future Work

In this work, we attempt to use an LSTM-based tag LM as an alternative to the CRF layer, and to build the BLSTM-based NER / LSTM-based LM hybrid model. The LM can either be trained in advance and used only for decoding or jointly trained with the NER model: The separately trained hybrid system speeds up the training significantly with a marginal performance degradation; And the jointly trained system, which has no speed advantage, can provide comparable performance with state-of-the-art baselines. In future work, the hybrid model can be modified so that the transition score contains more information. And the current joint training approach is straightforward, but primitive. A more sophisticated training method could also be helpful. In addition, we will test the hybrid system on larger corpora with more entity types.

Acknowledgements

This work has received funding from the European Research Council (ERC) (under the European Union's Horizon 2020 research and innovation programme, grant agreement No 694537, project "SEQCLAS") and the Deutsche Forschungsgemeinschaft (DFG; grant agreement NE 572/8-1, project "CoreTec"). The GPU computing cluster was supported by DFG (Deutsche Forschungsgemeinschaft) under grant INST 222/1168-1 FUGG. The work reflects only the authors' views and none of the funding agencies is responsible for any use that may be made of the information it contains.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pages 1638–1649, Santa Fe, NM, USA.

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019a. Flair: An easy-to-use framework for state-of-the-art nlp. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 54–59, Minneapolis, MN, USA.

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019b. Pooled contextualized embeddings for named entity recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, page 724–728, Minneapolis, MN, USA.

Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the 22nd international conference on Machine learning (ICML)*, pages 169–176, Bonn, Germany.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4171–4186, Minneapolis, MN, USA.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Yufan Jiang, Chi Hu, Tong Xiao, Chunliang Zhang, and Jingbo Zhu. 2019. Improved differentiable architecture search for language modeling and named entity recognition. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3576–3581, Hong Kong, China.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*, pages 282–289, Williamstown, MA, USA.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, Miyazaki, Japan.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, Doha, Qatar.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1296–1306, Austin, TX, USA.