Empower Sequence Labeling with Task-Aware Neural Language Model

Liyuan Liu[†] **Jingbo Shang**[†] **Xiang Ren**[♯] **Frank F. Xu**[‡] **Huan Gui**[♭] **Jian Peng**[†] **Jiawei Han**[†]

[†] University of Illinois at Urbana-Champaign, {ll2, shang7, jianpeng, hanj}@illinois.edu
[♯] University of Southern California, xiangren@usc.edu
[‡] Shanghai Jiao Tong University, frankxu@sjtu.edu.cn
[♭] Facebook, huangui@fb.com

Abstract

Linguistic sequence labeling is a general approach encompassing a variety of problems, such as part-of-speech tagging and named entity recognition. Recent advances in neural networks (NNs) make it possible to build reliable models without handcrafted features. However, in many cases, it is hard to obtain sufficient annotations to train these models. In this study, we develop a neural framework to extract knowledge from raw texts and empower the sequence labeling task. Besides word-level knowledge contained in pre-trained word embeddings, character-aware neural language models are incorporated to extract character-level knowledge. Transfer learning techniques are further adopted to mediate different components and guide the language model towards the key knowledge. Comparing to previous methods, these task-specific knowledge allows us to adopt a more concise model and conduct more efficient training. Different from most transfer learning methods, the proposed framework does not rely on any additional supervision. It extracts knowledge from self-contained order information of training sequences. Extensive experiments on benchmark datasets demonstrate the effectiveness of leveraging character-level knowledge and the efficiency of co-training. For example, on the CoNLL03 NER task, model training completes in about 6 hours on a single GPU, reaching F_1 score of 91.71$\pm$0.10 without using any extra annotations.

Introduction

Linguistic sequence labeling is a fundamental framework. It has been applied to a variety of tasks including part-of-speech (POS) tagging, noun phrase chunking and named entity recognition (NER) (Ma and Hovy 2016; Sha and Pereira 2003). These tasks play a vital role in natural language understanding and fulfill lots of downstream applications, such as relation extraction, syntactic parsing, and entity linking (Liu et al. 2017; Luo et al. 2015).

Traditional methods employed machine learning models like Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs), and have achieved relatively high performance. However, these methods have a heavy reliance on handcrafted features (e.g., whether a word is capitalized) and language-specific resources (e.g., gazetteers). Therefore, it could be difficult to apply them to new tasks or shift to new domains. To overcome this drawback, neural networks (NNs) have been proposed to automatically extract features during model learning. Nevertheless, considering the overwhelming number of parameters in NNs and the relatively small size of most sequence labeling corpus, annotations alone may not be sufficient to train complicated models. So, guiding the learning process with extra knowledge could be a wise choice.

Accordingly, transfer learning and multi-task learning have been proposed to incorporate such knowledge. For example, NER can be improved by jointly conducting other related tasks like entity linking or chunking (Luo et al. 2015; Peng and Dredze 2016). After all, these approaches would require additional supervision on related tasks, which might be hard to get, or not even existent for low-resource languages or special domains.

Alternatively, abundant knowledge can be extracted from raw texts, and enhance a variety of tasks. Word embedding techniques represent words in a continuous space (Mikolov et al. 2013; Pennington, Socher, and Manning 2014) and retain the semantic relations among words. Consequently, integrating these embeddings could be beneficial to many tasks (Liu et al. 2017; Lample et al. 2016). Nonetheless, most embedding methods take a word as a basic unit, thus only obtaining word-level knowledge, while character awareness is also crucial and highly valued in most state-of-the-art NN models.

Only recently, character-level knowledge has been leveraged and empirically verified to be helpful in numerous sequence labeling tasks (Peters et al. 2017; Rei 2017). Directly adopting pre-trained language models, character-level knowledge can be integrated as context embeddings and demonstrate its potential to achieve the state-of-the-art (Peters et al. 2017). However, the knowledge extracted through pre-training is not task-specific, thus containing a large irrelevant portion. So, this approach would require a bigger model, external corpus and longer training. For example, one of its language models was trained on 32 GPUs for more than half a month, which is unrealistic in many situations.

In this paper, we propose an effective sequence labeling framework, LM-LSTM-CRF, which leverages both word-level and character-level knowledge in an efficient way. For character-level knowledge, we incorporate a neural language

Copyright © 2018, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

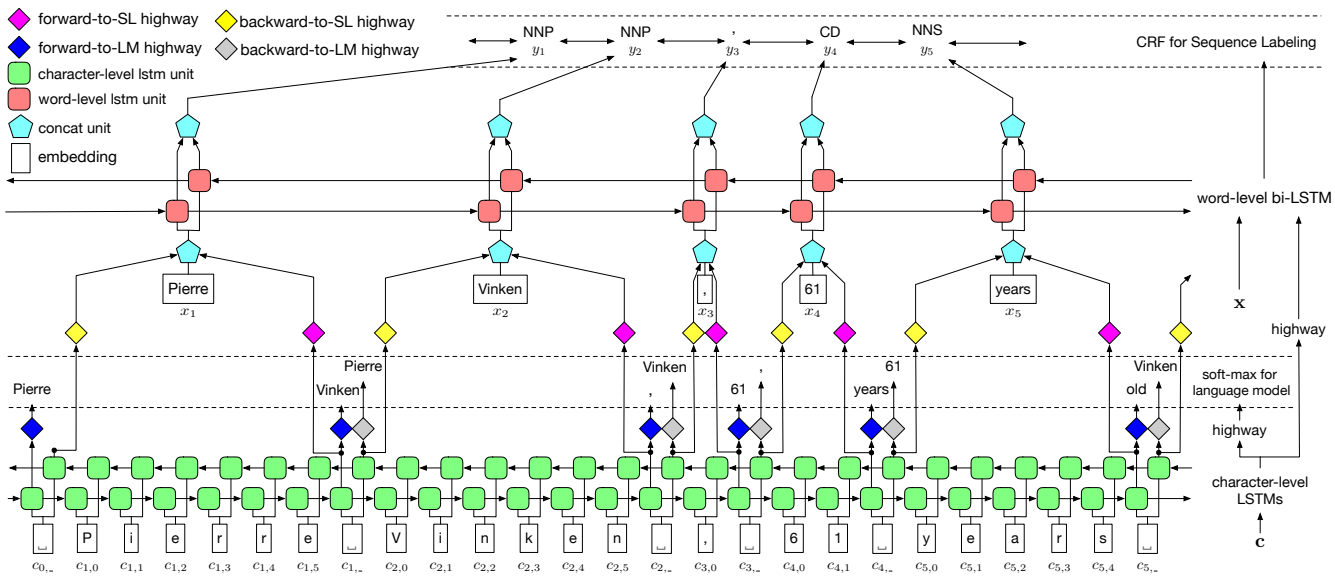

Figure 1: LM-LSTM-CRF Neural Architecture

model with the sequence labeling task and conduct multi-task learning to guide the language model towards task-specific key knowledge. Besides the potential of training a better model, this strategy also poses a new challenge. Based on our experiments, when the tasks are discrepant, language models could be harmful to sequence labeling in a naïve co-training setting. For this reason, we employ highway networks (Srivastava, Greff, and Schmidhuber 2015) to transform the output of character-level layers into different semantic spaces, thus mediating and unifying these two tasks. For word-level knowledge, we choose to fine-tune pre-trained word embeddings instead of co-training or pre-training the whole word-level layers, because the majority of parameters in word-level layers come from the embedding layer and such co-training or pre-training cost lots of time and resources.

We conduct experiments on the CoNLL 2003 NER task, the CoNLL 2000 chunking task, as well as the WSJ portion of the Penn Treebank POS tagging task. LM-LSTM-CRF achieves a significant improvement over the state-of-the-art. Also, our co-training strategy allows us to capture more useful knowledge with a smaller network, thus yielding much better efficiency without loss of effectiveness.

LM-LSTM-CRF Framework

The neural architecture of our proposed framework, LM-LSTM-CRF, is visualized in Fig. 1. For a sentence with annotations $\mathbf{y} = (y_1, \ldots, y_n)$, its word-level input is marked as $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, where x_i is the i-th word; its character-level input is recorded as $\mathbf{c} = (c_{0,_}, c_{1,1}, c_{1,2}, \ldots, c_{1,_}, c_{2,1}, \ldots, c_{n,_})$, where $c_{i,j}$ is the j-th character for word w_i and $c_{i,_}$ is the space character after w_i. These notations are also summarized in Table 1.

Now, we first discuss the multi-task learning strategy and then introduce the architecture in a bottom-up fashion.

$\mathbf{x}$	word-level input	x_i	i-th word
$\mathbf{c}$	character-level input	$c_{i,j}$	j-th char in x_i
$c_{i,_}$	space after x_i	$c_{0,_}$	space before x_1
$\mathbf{y}$	label sequence	y_i	label of x_i
$\mathbf{f}_i$	output of forward character-level LSTM at $c_{i,_}$		
$\mathbf{r}_i$	output of backward character-level LSTM at $c_{i,_}$		
$\mathbf{f^L}_i$	output of forward-to-LM highway unit		
$\mathbf{r^L}_i$	output of backward-to-LM highway unit		
$\mathbf{f^N}_i$	output of forward-to-SL highway unit		
$\mathbf{r^N}_i$	output of backward-to-SL highway unit		
$\mathbf{v}_i$	input of word-level bi-LSTM at x_i		
$\mathbf{z}_i$	output of word-level bi-LSTM at x_i		

Table 1: Notation Table.

Multi-task Learning Strategy

As shown in Fig. 1, our language model and sequence labeling model share the same character-level layer, which fits the setting of multi-task learning and transfer learning. However, different from typical models of this setting, our two tasks are not strongly related. This discordance makes our problem more challenging. E.g., although a naive co-training setting, which directly uses the output from character-level layers, could be effective in several scenarios (Yang, Salakhutdinov, and Cohen 2017), for our two tasks, it would hurt the performance. This phenomenon would be further discussed in the experiment section.

To mediate these two tasks, we transform the output of character-level layers into different semantic spaces for different objectives. This strategy allows character-level layers to focus on general feature extraction and lets the transform layers select task-specific features. Hence, our language model can provide related knowledge to the sequence labeling, without forcing it to share the whole feature space.

Character-level Layer

Character-level neural language models are trained purely on unannotated sequence data but can capture the underlying style and structure. For example, it can mimic Shakespeare's writing and generate sentences of similar styles, or even master the grammar of programming languages (e.g., XML, LaTeX, and C) and generate syntactically correct codes (Karpathy 2015). Accordingly, we adopted the character-level Long Short Term Memory (LSTM) networks to process character-level input. Aiming to capture lexical features instead of remembering words' spelling, we adjust the prediction from the next character to the next word. As in Fig. 1, the character-level LSTM would only make predictions for the next word at word boundaries (i.e., space characters or $c_{i,_}$).

Furthermore, we coupled two LSTM units to capture information in both forward and backward directions. Although it seems similar to the bi-LSTM unit, the outputs of these two units are processed and aligned differently. Specifically, we record the output of forward LSTM at $c_{i,_}$ as $\mathbf{f}_i$, and the output of backward LSTM at $c_{i,_}$ as $\mathbf{r}_i$.

Highway Layer

In computer vision, Convolutional Neural Networks (CNN) has been proved to be an effective feature extractor, but its output needs to be further transformed by fully-connected layers to achieve the state-of-the-art. Bearing this in mind, it becomes natural to stack additional layers upon the flat character-level LSTMs. More specifically, we employ highway units (Srivastava, Greff, and Schmidhuber 2015), which allow unimpeded information flowing across several layers. Typically, highway layers conduct nonlinear transformation as $\mathbf{m} = H(\mathbf{n}) = \mathbf{t} \odot g(W_H \mathbf{n} + b_H) + (1 - \mathbf{t}) \odot \mathbf{n}$, where $\odot$ is element-wise product, $g(\cdot)$ is a nonlinear transformation such as ReLU in our experiments, $\mathbf{t} = \sigma(W_T \mathbf{n} + b_T)$ is called transform gate and $(1 - \mathbf{t})$ is called carry gate.

In our final architecture, there are four highway units, named `forward-to-LM`, `forward-to-SL`, `backward-to-LM`, and `backward-to-SL`. The first two transfer $\mathbf{f}_i$ into $\mathbf{f^L}_i$ and $\mathbf{f^N}_i$, and the last two transfer $\mathbf{r}_i$ into $\mathbf{r^L}_i$ and $\mathbf{r^N}_i$. $\mathbf{f^L}_i$ and $\mathbf{r^L}_i$ are used in the language model, while $\mathbf{f^N}_i$ and $\mathbf{r^N}_i$ are used in the sequence labeling.

Word-level Layer

Bi-LSTM is adopted as the word-level structure to capture information in both directions. As shown in Fig. 1, we concatenate $\mathbf{f^N}_i$ and $\mathbf{r^N}_{i-1}$ with word embeddings and then feed them into the bi-LSTM. Note that, in the backward character-level LSTM, $\mathbf{c}_{i-1,_}$ is the space character before word x_i, therefore, $\mathbf{f^N}_i$ would be aligned and concatenated with $\mathbf{r^N}_{i-1}$ instead of $\mathbf{r^N}_i$. For example, in Fig. 1, the word embeddings of 'Pierre' will be concatenated with the output of the `forward-to-SL` over '...Pierre_' and the output of the `backward-to-SL` over '...erreiP_'.

As to word-level knowledge, we chose to fine-tune pre-trained word embeddings, instead of co-training the whole word-level layer. This is because most parameters of our word-level model come from word embeddings, and fine-tuning pre-trained word embeddings have been verified to be effective in leveraging word-level knowledge (Ma and Hovy 2016). Besides, current word embedding methods can easily scale to the large corpus; pre-trained word embeddings are available in many languages and domains (Fernandez, Yu, and Downey 2017). However, this strategy cannot be applied to character-level layers, since the embedding layer of character-level layers contains very few parameters. Based on these considerations, we applied different strategies to leverage word-level knowledge from character-level.

CRF for Sequence Labeling

Label dependencies are crucial for sequence labeling tasks. For example, in NER task with BIOES annotation, it is not only meaningless but illegal to annotate `I-PER` after `B-ORG` (i.e., mixing the person and the organization). Therefore, jointly decoding a chain of labels can ensure the resulting label sequence to be meaningful. Conditional random field (CRF) has been included in most state-of-the-art models to capture such information and further avoid generating illegal annotations. Consequently, we build a CRF layer upon the word-level LSTM.

For training instance $(\mathbf{x}_i, \mathbf{c}_i, \mathbf{y}_i)$, we suppose the output of word-level LSTM is $\mathbf{Z}_i = (\mathbf{z}_{i,1}, \mathbf{z}_{i,2}, \ldots, \mathbf{z}_{i,n})$. CRF models describe the probability of generating the whole label sequence with regard to $(\mathbf{x}_i, \mathbf{c}_i)$ or $\mathbf{Z}$. That is, $p(\hat{\mathbf{y}}|\mathbf{x}_i, \mathbf{c}_i)$ or $p(\hat{\mathbf{y}}|\mathbf{Z})$, where $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_n)$ is a generic label sequence. Similar to (Ma and Hovy 2016), we define this probability as follows.

$$p(\hat{\mathbf{y}}|\mathbf{x}_i, \mathbf{c}_i) = \frac{\prod_{j=1}^{n} \phi(\hat{y}_{j-1}, \hat{y}_j, \mathbf{z}_j)}{\sum_{\mathbf{y}' \in \mathbf{Y}(\mathbf{Z})} \prod_{j=1}^{n} \phi(y'_{j-1}, y'_j, \mathbf{z}_j)} \quad (1)$$

Here, $\mathbf{Y}(\mathbf{Z})$ is the set of all generic label sequences, $\phi(y_{j-1}, y_j, \mathbf{z}_j) = \exp(W_{y_{j-1},y_j} \mathbf{z}_i + b_{y_{j-1},y_j})$, where W_{y_{j-1},y_j} and b_{y_{j-1},y_j} are the weight and bias parameters corresponding to the label pair (y_{j-1}, y_j).

For training, we minimize the following negative log-likelihood.

$$\mathcal{J}_{CRF} = -\sum_i \log p(\mathbf{y}_i|\mathbf{Z}_i) \quad (2)$$

And for testing or decoding, we want to find the optimal sequence $\mathbf{y}^*$ that maximizes the likelihood.

$$\mathbf{y}^* = \arg \max_{\mathbf{y} \in \mathbf{Y}(\mathbf{Z})} p(\mathbf{y}|\mathbf{Z}) \quad (3)$$

Although the denominator of Eq. 1 is complicated, we can calculate Eqs. 2 and 3 efficiently by the Viterbi algorithm.

Neural Language Model

The language model is a family of models describing the generation of sequences. In a neural language model, the generation probability of the sequence $\mathbf{x} = (x_1, ..., x_n)$ in the forward direction (i.e., from left to right) is defined as

$$p_f(x_1, ..., x_n) = \prod_{i=1}^{N} p_f(x_i|x_1, \ldots, x_{i-1})$$

Dataset	# of Sentences		
	Train	Dev	Test
CoNLL03 NER	14,987	3,466	3,684
CoNLL00 chunking	7,936	1,000	2,012
WSJ	38,219	5,527	5,426

Table 2: Dataset summary.

where $p_f(x_i|x_1, \ldots, x_{i-1})$ is computed by NN.

In this paper, our neural language model makes predictions for words but takes the character sequence as input. Specifically, we would calculate $p_f(x_i|c_{0,\text{-}}, \ldots, c_{i-1,1}, \ldots, c_{i-1,\text{-}})$ instead of $p_f(x_i|x_1, \ldots, x_{i-1})$. This probability is assumed as

$$p_f(x_i|c_{0,\text{-}}, \ldots, c_{i-1,\text{-}}) = \frac{\exp(\mathbf{w}_{x_i}^T \mathbf{f}^{\mathbf{N}}{}_{i-1})}{\sum_{\hat{x}_j} \exp(\mathbf{w}_{\hat{x}_j}^T \mathbf{f}^{\mathbf{N}}{}_{i-1})}$$

where $\mathbf{w}_{x_i}$ is the weight vector for predicting word x_i. In order to extract knowledge in both directions, we also adopted a reversed-order language model, which calculates the generation probability from right to left as

$$p_r(x_1, ..., x_n) = \prod_{i=1}^{N} p_r(x_i|c_{i+1,\text{-}}, \ldots, c_{n,\text{-}})$$

$$\text{where } p_r(x_i|c_{i+1,\text{-}}, \ldots, c_{n,\text{-}}) = \frac{\exp(\mathbf{w}_{x_i}^T \mathbf{r}^{\mathbf{N}}{}_i)}{\sum_{\hat{x}_j} \exp(\mathbf{w}_{\hat{x}_j}^T \mathbf{r}^{\mathbf{N}}{}_i)}$$

The following negative log likelihood is applied as the objective function of our language model.

$$\mathcal{J}_{LM} = -\sum_i \log p_f(\mathbf{x}_i) - \sum_i \log p_r(\mathbf{x}_i) \quad (4)$$

Joint Model Learning

By combining Eqs. 2 and 4, we can write the joint objective function as

$$\mathcal{J} = -\sum_i \Big(p(\mathbf{y}_i|\mathbf{Z}_i) + \lambda \big(\log p_f(\mathbf{x}_i) + \log p_r(\mathbf{x}_i) \big) \Big) \quad (5)$$

where λ is a weight parameter. In our experiments, λ is always set to 1 without any tuning.

In order to train the neural network efficiently, stochastic optimization has been adopted. And at each iteration, we sample a batch of training instances and perform an update according to the summand function of Eq. 5: $p(\mathbf{y}_i|\mathbf{Z}_i) + \lambda\big(\log p_f(\mathbf{x}_i) + \log p_r(\mathbf{x}_i)\big)$

Experiments

Here, we evaluate LM-LSTM-CRF on three benchmark datasets: the CoNLL 2003 NER dataset (Tjong Kim Sang and De Meulder 2003), the CoNLL 2000 chunking dataset (Tjong Kim Sang and Buchholz 2000), and the Wall Street Journal portion of Penn Treebank dataset (WSJ) (Marcus, Marcinkiewicz, and Santorini 1993).

- **CoNLL03 NER** contains annotations for four entity types: `PER`, `LOC`, `ORG`, and `MISC`. It has been separated into training, development and test sets.

Layer	Parameter	POS	NER	chunking
character-level embedding	dimension	30		
character-level LSTM	depth	1		
	state size	300		
Highway	depth	1		
word-level embedding	dimension	100		
word-level bi-LSTM	depth	1		
	state size	300		
Optimization	η_0	0.015	0.01	

Table 3: Hyper-parameters of LM-LSTM-CRF.

- **CoNLL00 chunking** defines eleven syntactic chunk types (e.g., `NP`, `VP`) in addition to `Other`. It only includes training and test sets. Following previous works (Peters et al. 2017), we sampled 1000 sentences from training set as a held-out development set.
- **WSJ** contains 25 sections and categorizes each word into 45 POS tags. We adopt the standard split and use sections 0-18 as training data, sections 19-21 as development data, and sections 22-24 as test data (Manning 2011).

The corpus statistics are summarized in Table 2. We report the accuracy for the WSJ dataset. And in the first two datasets, we adopt the official evaluation metric (micro-averaged F_1), and use the BIOES scheme (Ratinov and Roth 2009). Also, in all three datasets, rare words (i.e., frequency less than 5) are replaced by a special token (<UNK>).

Network Training

For a fair comparison, we didn't spend much time on tuning parameters but borrow the initialization, optimization method, and all related hyper-parameter values (except the state size of LSTM) from the previous work (Ma and Hovy 2016). For the hidden state size of LSTM, we expand it from 200 to 300, because introducing additional knowledge allows us to train a larger network. We will further discuss this change later. Table 3 summarizes some important hyper-parameters. Since the CoNLL00 is similar to the CoNLL03 NER dataset, we conduct experiments with the same parameters on both tasks.

Initialization. We use GloVe 100-dimension pre-trained word embeddings released by Stanford[1] and randomly initialize the other parameters (Glorot and Bengio 2010; Jozefowicz, Zaremba, and Sutskever 2015).

Optimization. We employ mini-batch stochastic gradient descent with momentum. The batch size, the momentum and the learning rate are set to 10, 0.9 and $\eta_t = \frac{\eta_0}{1+\rho t}$, where η_0 is the initial learning rate and $\rho = 0.05$ is the decay ratio. Dropout is applied in our model, and its ratio is fixed to 0.5. To increase stability, we use gradient clipping of 5.0.

Network Structure. The hyper-parameters of character-level LSTM are set to the same value of word-level bi-LSTM. We fix the depth of highway layers as 1 to avoid an over-complicated model.

[1] `http://nlp.stanford.edu/projects/glove/`

Note that some baseline methods (e.g., (Chiu and Nichols 2016; Peters et al. 2017)) incorporate the development set as a part of training. However, because we are using early stopping based on the evaluation on the development set, our model is trained purely on the training set.

Compared Methods We consider three classes of baseline sequence labeling methods in our experiments.

- **Sequence Labeling Only.** Without any additional supervision or extra resources, LSTM-CRF (Lample et al. 2016) and LSTM-CNN-CRF (Ma and Hovy 2016) are the current state-of-art methods. We also list some top reported performance on each dataset (Collobert et al. 2011; Luo et al. 2015; Chiu and Nichols 2016; Yang, Salakhutdinov, and Cohen 2017; Peters et al. 2017; Manning 2011; Søgaard and Goldberg 2016; Sun 2014).
- **Joint Model with Other Supervised Tasks.** There are several attempts (Luo et al. 2015; Yang, Salakhutdinov, and Cohen 2017) to enhance sequence labeling tasks by introducing additional annotations from other related tasks (e.g., enhance NER with entity linking labels).
- **Joint Model with Language Model**: Language models have been employed by some recent works to extract knowledge from raw text and thus enhancing sequence labeling task. TagLM (Peters et al. 2017) leverages pre-trained language models and shows the effectiveness with the large external corpus, but the large model scale and long training time make it hard to re-run this model. Another work (Rei 2017) also incorporates the sequence labeling task with the language model.

For comparison, we tune the parameters of three most related baselines (Ma and Hovy 2016; Lample et al. 2016; Rei 2017)[2]. , and report the statics of the best working parameter setting. Besides, we index these models by number, and summarize the results in Tables 4, 5 and 7.

Performance Comparison

In this section, we focus on the comparisons between LM-LSTM-CRF and previous state-of-the-arts, including both effectiveness and efficiency. As demonstrated in Tables 4, 5 and 7, LM-LSTM-CRF significantly outperforms all baselines without additional resources. Moreover, even for those baselines with extra resources, LM-LSTM-CRF beats most of them and is only slightly worse than TagLM (index 4) (Peters et al. 2017).

TagLM (index 4) is equipped with both extra corpoa (about $4000X$ larger than the CoNLL03 NER dataset) and a tremendous pre-trained forward language model (`4096-8192-1024`[3]) (Jozefowicz et al. 2016). Due to the expensive resources and time required by `4096-8192-1024`, even the authors of TagLM failed to train a backward language model of the same size, instead, chose a much smaller one (`LSTM-2048-512`[4]). It is worth noting that, when either extra corpus or `4096-8192-1024` is absent, LM-LSTM-CRF shows significant improvements over TagLM (index 5, 10 and 11).

Also, LSTM-CNN-CRF outperforms LSTM-CRF in our experiments, which is different from (Reimers and Gurevych 2017). During our experiments, we discover that, when trained on CPU, LSTM-CNN-CRF only reaches 90.83 F_1 score on the NER dataset, but gets 91.37 F_1 score when trained on GPU. We conjecture that this performance gap is due to the difference of runtime environments. Therefore, we conduct all of our experiments on GPU. Additionally, we can observe that, although co-trained with language model, results of index 12 fails to outperform LSTM-CNN-CRF or LSTM-CRF. The reason of this phenomenon could be complicated and beyond the scope of this paper. However, it verified the effectiveness of our method, and demonstrated the contribution of outperforming these baselines.

Extra Resource	Index & Model	F_1 score	
		Type	Value (±std)
gazetteers	0) Collobert et al. 2011†	reported	89.59
	1) Chiu et al. 2016†	reported	91.62±0.33
AIDA dataset	2) Luo et al. 2015	reported	91.20
CoNLL 2000 / PTB-POS dataset	3) Yang et al. 2017†	reported	91.26
1B Word dataset & `4096-8192-1024`	4) Peters et al. 2017†‡	reported	91.93±0.19
1B Word dataset	5) Peters et al. 2017†‡	reported	91.62±0.23
None	6) Collobert et al. 2011†	reported	88.67
	7) Luo et al. 2015	reported	89.90
	8) Chiu et al. 2016†	reported	90.91±0.20
	9) Yang et al. 2017†	reported	91.20
	10) Peters et al. 2017†	reported	90.87±0.13
	11) Peters et al. 2017†‡	reported	90.79±0.15
	12) Rei 2017 †‡	mean	87.38±0.36
		max	87.94
		reported	86.26
	13) Lample et al. 2016†	mean	90.76±0.08
		max	91.14
		reported	90.94
	14) Ma et al. 2016†	mean	91.37±0.17
		max	91.67
		reported	91.21
	15) LM-LSTM-CRF †‡	mean	91.71±0.10
		max	91.85

Table 4: F_1 score on the CoNLL03 NER dataset. We mark models adopting pre-trained word embedding as †, and record models which leverage language models as ‡.

NER First of all, we have to point out that the results of index 1, 4, 8, 10 and 11 are not directly comparable with others since their final models are trained on both training and development set, while others are trained purely on the training set. As mentioned before, LM-LSTM-CRF outperforms all baselines except TagLM (index 4). For a thorough comparison, we also compare to its variants, TagLM (index 5), TagLM (index 10) and TagLM (index 11). Both index 10 and 11 are trained on the CoNLL03 dataset alone, while index 11 utilizes language model and index 10 doesn't. Comparing F_1 scores of these two settings, we can find that TagLM (index

[2]Implementations: `https://github.com/xuezhemax/lasagnenlp` (Ma et al. 2016), `https://github.com/glample/tagger` (Lample et al. 2016) and `https://github.com/marekrei/sequence-labeler` (Rei 2017)

[3]`4096-8192-1024` is composed of character-level CNN with 4096 filters, 2 layers of stacked LSTMs with 8192 hidden units each and a 1024-dimension projection unit.

[4]`LSTM-2048-512` is composed of a single-layer LSTM with 2048 hidden units and a 512-dimension projection unit.

Ind & Model	Accuracy	
	Type	Value (±std)
0) Collobert et al. 2011†	reported	97.29
16) Manning 2011	reported	97.28
17) Søgaard 2011	reported	97.50
18) Sun 2014	reported	97.36
12) Rei 2017†‡	mean	96.97±0.22
	max	97.14
	reported	97.43
13) Lample et al. 2016†	mean±std	97.35±0.09
	maximum	97.51
14) Ma et al. 2016†	mean±std	97.42±0.04
	maximum	97.46
	reported	97.55
15) LM-LSTM-CRF †‡	mean±std	97.53±0.03
	maximum	97.59

Table 5: Accuracy on the WSJ dataset. We mark models adopting pre-trained word embedding as †, and record models which leverage language models as ‡.

Model	CoNLL03 NER		WSJ POS		CoNLL00 Chunking	
	h	F_1Score	h	Accuracy	h	F_1Score
LSTM-CRF	46	90.76	37	97.35	26	94.37
LSTM-CNN-CRF	7	91.22	21	97.42	6	95.80
LM-LSTM-CRF	6	91.71	16	97.53	5	95.96
LSTM-CRF⋆	4	91.19	8	97.44	2	95.82
LSTM-CNN-CRF⋆	3	90.98	7	96.98	2	95.51

Table 6: Training statistics of TagLM (index 4 and 5) and LM-LSTM-CRF on the CoNLL03 NER dataset.

Extra Resource	Ind & Model	F_1 score	
		Type	Value (±std)
PTB-POS	19) Hashimoto et al. 2016†	reported	95.77
	20) Søgaard et al. 2016†	reported	95.56
CoNLL 2000 / PTB-POS dataset	3)Yang et al. 2017†	reported	95.41
1B Word dataset	4) Peters et al. 2017†‡	reported	96.37±0.05
None	21) Hashimoto et al. 2016†	reported	95.02
	22) Søgaard et al. 2016†	reported	95.28
	9) Yang et al. 2017†	reported	94.66
	12) Rei 2017†‡	mean	94.24±0.11
		max	94.33
		reported	93.88
	13) Lample et al. 2016†	mean	94.37±0.07
		maximum	94.49
	14) Ma et al. 2016†	mean	95.80±0.13
		maximum	95.93
	15) LM-LSTM-CRF †‡	mean	95.96±0.08
		maximum	96.13

Table 7: F_1 score on the CoNLL00 chunking dataset. We mark models adopting pre-trained word embedding as †, and record models which leverage language models as ‡.

Ind & Model	F_1score	Module	Time · Device	
15) LM-LSTM-CRF	91.71	total	6	h·GTX 1080
5) Peters et al. 2017	91.62	`LSTM-2048-512`	320	h·Telsa K40
		`LSTM-2048-512`	320	h·Telsa K40
4) Peters et al. 2017	91.93	`4096-8192-1024`	14112	h·Telsa K40
		`LSTM-2048-512`	320	h·Telsa K40

Table 8: Training time and performance of LSTM-CRF, LSTM-CNN-CRF and LM-LSTM-CRF on three datasets. Our re-implementations are marked with ⋆

11) even performs worse than TagLM (index 10) , which reveals that directly applying co-training might hurt the sequence labeling performance. We will also discuss this challenge later in the Highway Layers & Co-training section.

Besides, changing the forward language model from `4096-8192-1024` to `LSTM-2048-512`, TagLM (index 5) gets a lower F_1 score of 91.62±0.23. Comparing this score to ours (91.71±0.10), one can verify that pre-trained language model usually extracts a large portion of unrelated knowledge. Relieving such redundancy by guiding the language model with task-specific information, our model is able to conduct both effective and efficient learning.

POS Tagging Similar to the NER task, LM-LSTM-CRF outperforms all baselines on the WSJ portion of the PTB POS tagging task. Although the improvements over LSTM-CRF and CNN-LSTM-CRF are less obvious than those on the CoNLL03 NER dataset, considering the fact that the POS tagging task is believed to be easier than the NER task and current methods have achieved relatively high performance, this improvement could still be viewed as significant. Moreover, it is worth noting that for both NER and POS tagging tasks, LM-LSTM-CRF achieves not only higher F_1 scores, but also with smaller variances, which further verifies the superiority of our framework.

Chunking In the chunking task, LM-LSTM-CRF also achieves relatively high F_1 scores, but with slightly higher variances. Considering the fact that this corpus is much smaller than the other two (only about $1/5$ of WSJ or $1/2$ of CoNLL03 NER), we can expect more variance due to the lack of training data. Still, LM-LSTM-CRF outperforms all baselines without extra resources, and most of the baselines trained with extra resources.

Efficiency We implement LM-LSTM-CRF[5] based on the PyTorch library[6]. Models has been trained on one GeForce GTX 1080 GPU, with training time recorded in Table 8.

In terms of efficiency, the language model component in LM-LSTM-CRF only introduces a small number of parameters in two highway units and a soft-max layer, which may not have a very large impact on the efficiency. To control variables like infrastructures, we further re-implemented both baselines, and report their performance together with original implementations. From the results, these re-implementations achieve better efficiency comparing to the original ones, but yield relative worse performance. Also, LM-LSTM-CRF achieves the best performance, and takes twice the training time of the most efficient model, LSTM-CNN-CRF⋆. Empirically, considering the difference among the implementations of these models, we think these methods have roughly the same efficiency.

Besides, we list the required time and resources for pre-training model index 4 and 5 on the NER task in Table 6 (Jozefowicz et al. 2016). Comparing to these language models pre-trained on external corpus, our model has no such reliance on extensive corpus, and can achieve similar performance with much more concise model and effi-

[5] `https://github.com/LiyuanLucasLiu/LM-LSTM-CRF`

[6] `http://pytorch.org/`

Model	State Size	F_1score±std	Recall±std	Precision±std
LM-LSTM-CRF	300	91.71±0.10	92.14±0.12	91.30±0.13
	200	91.63±0.23	92.07±0.22	91.19±0.30
	100	91.13±0.32	91.60±0.37	90.67±0.32
LSTM-CRF	300	90.76±0.08	90.82±0.08	90.69±0.08
	200	90.41±0.07	90.63±0.07	90.20±0.07
	100	90.74±0.22	91.08±0.50	90.42±0.17
LSTM-CNN-CRF	300	91.22±0.19	91.70±0.16	90.74±0.27
	200	91.37±0.17	91.08±0.53	90.58±0.11
	100	91.18±0.10	91.56±0.16	90.81±0.15

Table 9: Effect of hidden state size of LSTM

State Size	Model	F_1score±std	Recall±std	Precision±std
300	LM-LSTM-CRF	91.71±0.10	92.14±0.12	91.30±0.13
	LM-LSTM-CRF_NL	91.43±0.09	91.85±0.18	91.01±0.19
	LM-LSTM-CRF_NH	91.16±0.22	91.67±0.28	90.66±0.23
200	LM-LSTM-CRF	91.63±0.23	92.07±0.22	91.19±0.30
	LM-LSTM-CRF_NL	91.44±0.10	91.95±0.16	90.94±0.16
	LM-LSTM-CRF_NH	91.34±0.28	91.79±0.18	90.89±0.30
100	LM-LSTM-CRF	91.13±0.32	91.60±0.37	90.67±0.32
	LM-LSTM-CRF_NL	91.17±0.11	91.72±0.14	90.61±0.21
	LM-LSTM-CRF_NH	91.01±0.19	91.50±0.21	90.53±0.30

Table 10: Effect of language model and highway

cient training. It verifies that our LM-LSTM-CRF model can effectively leverage the language model to extract task-specific knowledge to empower sequence labeling.

Analysis

To analyze the performance of LM-LSTM-CRF, we conduct additional experiments on the CoNLL03 NER dataset.

Hidden State Size To explore the effect of model size, we train our model with different hidden state sizes. For comparison, we also apply the same hidden state sizes to LSTM-CRF and LSTM-CNN-CRF. From Table 9, one can easily observe that the F_1 score of LM-LSTM-CRF keeps increasing when the hidden state size grows, while LSTM-CNN-CRF has a peak at state size 200 and LSTM-CRF has a drop at state size 200. This phenomenon further verified our intuition of employing the language model to extract knowledge and prevent overfitting.

Highway Layers & Co-training To elucidate the effect of language model[7] and highway units, we compare LM-LSTM-CRF with its two variants, LM-LSTM-CRF_NL and LM-LSTM-CRF_NH. The first keeps highway units, but optimizes $\mathcal{J}_{CRF}$ alone; the second jointly optimizes $\mathcal{J}_{CRF}$ and $\mathcal{J}_{LM}$, but without highway units. As shown in Table 10, LM-LSTM-CRF_NH yields worse performance than LM-LSTM-CRF_NL. This observation accords with previous comparison between TagLM (index 10) and TagLM (index 11) on the CoNLL03 NER dataset. We conjecture that it is because the NER task and the language model is not strongly related to each other. In summary, our proposed co-training strategy is effective and introducing the highway layers is necessary.

Related Work

There exist two threads of related work regarding the topics in this paper, which are sequence labeling and how to improve it with additional information.

Sequence Labeling. As one of the fundamental tasks in NLP, linguistic sequence labeling, including POS tagging, chunking, and NER, has been studied for years. Handcrafted features were widely used in traditional methods like CRFs, HMMs, and maximum entropy classifiers (Lafferty, McCallum, and Pereira 2001; McCallum and Li 2003; Florian et al. 2003; Chieu and Ng 2002), but also make it hard to apply them to new tasks or domains. Recently, getting rid of handcrafted features, there are attempts to build end-to-end systems for sequence labeling tasks, such as BiLSTM-CNN (Chiu and Nichols 2016), LSTM-CRF (Lample et al. 2016), and the current state-of-the-art method in NER and POS tagging tasks, LSTM-CNN-CRF (Ma and Hovy 2016). These models all incorporate character-level structure, and report meaningful improvement over pure word-level model. Also, CRF layer has also been demonstrated to be effective in capturing the dependency among labels. Our model is based on the success of LSTM-CRF model and is further modified to better capture the char-level information in a language model manner.

Leveraging Additional Information. Integrating word-level and character-level knowledge has been proved to be helpful to sequence labeling tasks. For example, word embeddings (Mikolov et al. 2013; Pennington, Socher, and Manning 2014) can be utilized by co-training or pre-training strategies (Liu et al. 2017; Lample et al. 2016). However, none of these models utilizes the character-level knowledge. Although directly adopting character-level pre-trained language models could be helpful (Peters et al. 2017). Such pre-trained knowledge is not task-specific and requires a larger neural network, external corpus, and longer training. Our model leverages both word-level and character-level knowledge through a co-training strategy, which leads to a concise, effective, and efficient neural network. Besides, unlike other multi-task learning methods, our model has no reliance on any extra annotation (Peters et al. 2017) or any knowledge base (Shang et al. 2017). Instead, it extracts knowledge from the self-contained order information.

Conclusion

In this paper, we proposed a sequence labeling framework, LM-LSTM-CRF, which effectively leverages the language model to extract character-level knowledge from the self-contained order information. Highway layers are incorporated to overcome the discordance issue of the naive co-training Benefited from the effectively captured such task-specific knowledge, we can build a much more concise model, thus yielding much better efficiency without loss of effectiveness (achieved the state-of-the-art on three benchmark datasets) . In the future, we plan to further extract and incorporate knowledge from other "unsupervised" learning principles and empower more sequence labeling tasks.

Acknowledgments

We thank Junliang Guo, Cheng Cheng and all reviewers for comments on earlier drafts that led to substantial improve-

[7]the perplexities of the forward language model on CoNLL03 NER's training / development / test sets are 52.87 / 55.03 / 50.22.

ments in the final version. Research was sponsored in part by U.S. Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), DARPA under Agreement No. W911NF-17-C-0099, National Science Foundation IIS 16-18481, IIS 17-04532, and IIS-17-41317, and grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov).

References

Chieu, H. L., and Ng, H. T. 2002. Named entity recognition: A maximum entropy approach using global information. In *COLING*.

Chiu, J. P. C., and Nichols, E. 2016. Named entity recognition with bidirectional lstm-cnns. *TACL*.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. P. 2011. Natural language processing (almost) from scratch. *JMLR*.

Fernandez, J.; Yu, Z.; and Downey, D. 2017. Vecshare: A framework for sharing word representation vectors.

Florian, R.; Ittycheriah, A.; Jing, H.; and Zhang, T. 2003. Named entity recognition through classifier combination. In *CoNLL*.

Glorot, X., and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*.

Hashimoto, K.; Xiong, C.; Tsuruoka, Y.; and Socher, R. 2016. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv:1611.01587*.

Jozefowicz, R.; Vinyals, O.; Schuster, M.; Shazeer, N.; and Wu, Y. 2016. Exploring the limits of language modeling. *arXiv:1602.02410*.

Jozefowicz, R.; Zaremba, W.; and Sutskever, I. 2015. An empirical exploration of recurrent network architectures. In *ICML*.

Karpathy, A. 2015. The unreasonable effectiveness of recurrent neural networks. `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`. Accessed: 2017-08-22.

Lafferty, J. D.; McCallum, A.; and Pereira, F. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Lample, G.; Ballesteros, M.; Kawakami, K.; Subramanian, S.; and Dyer, C. 2016. Neural architectures for named entity recognition. In *NAACL-HLT*.

Liu, L.; Ren, X.; Zhu, Q.; Zhi, S.; Gui, H.; Ji, H.; and Han, J. 2017. Heterogeneous Supervision for Relation Extraction: A Representation Learning Approach. *Proc. EMNLP*.

Luo, G.; Huang, X.; Lin, C.-Y.; and Nie, Z. 2015. Joint named entity recognition and disambiguation. In *EMNLP*.

Ma, X., and Hovy, E. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *ACL*.

Manning, C. D. 2011. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer.

Marcus, M. P.; Marcinkiewicz, M. A.; and Santorini, B. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*.

McCallum, A., and Li, W. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *CoNLL*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Peng, N., and Dredze, M. 2016. Improving named entity recognition for chinese social media with word segmentation representation learning. In *ACL*.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Peters, M. E.; Ammar, W.; Bhagavatula, C.; and Power, R. 2017. Semi-supervised sequence tagging with bidirectional language models. *arXiv:1705.00108*.

Ratinov, L., and Roth, D. 2009. Design challenges and misconceptions in named entity recognition. In *CoNLL*.

Rei, M. 2017. Semi-supervised multitask learning for sequence labeling. In *ACL*.

Reimers, N., and Gurevych, I. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. *arXiv preprint arXiv:1707.09861*.

Sha, F., and Pereira, F. 2003. Shallow parsing with conditional random fields. In *NAACL-HLT*.

Shang, J.; Liu, J.; Jiang, M.; Ren, X.; Voss, C. R.; and Han, J. 2017. Automated phrase mining from massive text corpora. *arXiv:1702.04457*.

Søgaard, A., and Goldberg, Y. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *ACL*.

Søgaard, A. 2011. Semisupervised condensed nearest neighbor for part-of-speech tagging. In *NAACL-HLT*.

Srivastava, R. K.; Greff, K.; and Schmidhuber, J. 2015. Highway networks. *arXiv:1505.00387*.

Sun, X. 2014. Structure regularization for structured prediction. In *NIPS*.

Tjong Kim Sang, E. F., and Buchholz, S. 2000. Introduction to the conll-2000 shared task: Chunking. In *Learning language in logic and CoNLL*.

Tjong Kim Sang, E. F., and De Meulder, F. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Natural language learning at NAACL-HLT*.

Yang, Z.; Salakhutdinov, R.; and Cohen, W. W. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. *arXiv:1703.06345*.