IEICE TRANS. INF. & SYST., VOL.E100–D, NO.4 APRIL 2017

PAPER

LSTM-CRF Models for Named Entity Recognition

Changki LEE[†a)], ***Member***

SUMMARY Recurrent neural networks (RNNs) are a powerful model for sequential data. RNNs that use long short-term memory (LSTM) cells have proven effective in handwriting recognition, language modeling, speech recognition, and language comprehension tasks. In this study, we propose LSTM conditional random fields (LSTM-CRF); it is an LSTM-based RNN model that uses output-label dependencies with transition features and a CRF-like sequence-level objective function. We also propose variations to the LSTM-CRF model using a gate recurrent unit (GRU) and structurally constrained recurrent network (SCRN). Empirical results reveal that our proposed models attain state-of-the-art performance for named entity recognition.

key words: *LSTM-CRF, LSTM RNN, recurrent neural network, name entity recognition*

1. Introduction

Recurrent neural networks (RNNs) are a form of neural sequence model that achieves state-of-the-art performance on language modeling, speech recognition, and machine translation [1]–[3]. RNNs that use long short-term memory (LSTM) cells have proven effective in handwriting recognition, language modeling, speech recognition, and language comprehension tasks [4]–[6].

In this study, we examine the use of RNNs in named entity recognition (NER). NER is a subtask of information extraction that seeks to locate and classify elements in text into pre-defined categories such as the names of persons, organizations, locations, etc. For example, the sentence of “U.N. official Ekeus heads for Baghdad.” is tagged as “U.N./B-ORG official/O Ekeus/B-PER heads/O for/O Baghdad/B-LOC ./O” where a BIO tag indicates whether it begins (B-), is inside (I-), or is outside (O) of a named entity. The most obvious approach to this task is the use of conditional random fields (CRFs), in which an exponential model is used to compute the probability of a label sequence given input word sequences [7]. CRFs produce the globally most likely label sequence, and have been used widely in NER. Recently, Collobert et al. proposed a semantic/syntactic extraction using a neural network architecture (SENNA), which is a sentence-level neural network that generated state-of-the-art results for NER [8]. They extended the model from a linear to non-linear architecture and replaced discrete with distributional feature representations in a continuous space. This architecture consists of the following: a projection layer that extracts features for each word, a hidden layer that extracts features from a window of words, a set of output nodes, and an output graph over which tag inference is achieved using a Viterbi algorithm.

Our current research improves on the SENNA architecture [8]. Whereas the SENNA architecture uses feed-forward or convolutional networks, we propose to use LSTM-based RNN to exploit long-range dependencies in the sequence of words. In addition, our research improves on the LSTM RNN model. LSTM RNN does not explicitly model the dependencies between output labels. LSTM RNN also produces a sequence of locally normalized output distributions, one for each word position as in a maximum entropy Markov model (MEMM). Thus, LSTM RNN can suffer from the same label bias problem as in MEMM [6]. To resolve these problems, we propose LSTM conditional random fields (LSTM-CRF), an LSTM-based RNN model that uses output label dependencies with transition features and the CRF-like sequence-level objective function. In addition, we propose variations to the LSTM-CRF model using a gate recurrent unit (GRU) [3] and structurally constrained recurrent network (SCRN) [9].

Our approach is an advance over previous research on recurrent conditional random fields (R-CRF) for language comprehension [10]. R-CRF models use RNN to exploit long-range dependencies in the sequence of words. However, training standard RNNs is difficult because of the vanishing gradient problem. To solve this problem, we propose to use LSTM RNN and its variations such as GRU and SCRN.

2. Models

In this section, we describe the models used in this study: RNN, variations of RNN (i.e., LSTM RNN, BI-LSTM RNN, GRU, BI-GRU, and SCRN), LSTM-CRF, variations of LSTM-CRF (i.e., BI-LSTM CRF, GRU-CRF, BI-GRU CRF, and SCRN-CRF).

2.1 RNN and LSTM RNN Models

Given an input vector sequence $\boldsymbol{x} = \{x_1, x_2, \ldots, x_T\}$, an RNN computes the hidden vector sequence $\boldsymbol{h} = \{h_1, h_2, \ldots, h_T\}$ and output vector sequence $\boldsymbol{y} = \{y_1, y_2, \ldots, y_T\}$ (y_t is a one-hot encoded vector) by iterating the following equations

Manuscript received April 27, 2016.
Manuscript revised December 9, 2016.
Manuscript publicized January 20, 2017.

[†]The author is with the Department of Computer Science, Kangwon National University, 192–1 Hyoja2-dong Chuncheon-si Gangwon-do, Korea.
a) E-mail: leeck@kangwon.ac.kr (Corresponding author)

DOI: 10.1587/transinf.2016EDP7179

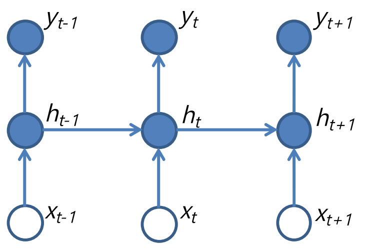

Fig. 1 RNN architecture.

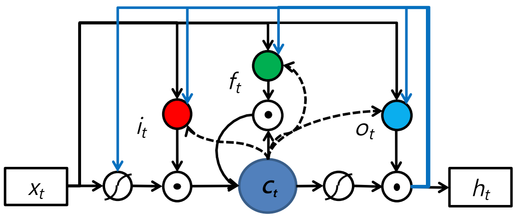

Fig. 2 LSTM memory block.

from $t = 1$ to T:

$$h_t = f(UEx_t + Vh_{t-1} + b_h) \tag{1}$$
$$P(y_t|\mathbf{x}) = {y_t}^T g(Wh_t + b_y) \tag{2}$$

where U, V, and W denote weight matrices (e.g., U is the input-hidden weight matrix), E is a weight matrix for the word and feature embedding, b_h and b_y denote bias vectors (e.g., b_h is a hidden bias vector), f is a nonlinear function (i.e., sigmoid or tanh function), and g is the softmax function. This RNN architecture is shown in Fig. 1. Figure 1 reveals that no direct connection exists between output labels. RNN also produces a sequence of locally normalized output distributions, one for each word position. Thus, it can suffer from a label bias problem [6].

LSTM RNN has been shown to perform more effectively than the standard RNN at finding and exploiting long-range dependencies in the data [6]. One difference from the standard RNN is that the LSTM uses a memory cell with gate units, which are linear activation functions. LSTM RNN is defined according to the following equations.

$$i_t = \sigma(W_{ix}Ex_t + W_{ih}h_{t-1} + W_{ic}c_{t-1} + b_i) \tag{3}$$
$$f_t = \sigma(W_{fx}Ex_t + W_{fh}h_{t-1} + W_{fc}c_{t-1} + b_f) \tag{4}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{cx}Ex_t + W_{ch}h_{t-1} + b_c) \tag{5}$$
$$o_t = \sigma(W_{ox}Ex_t + W_{oh}h_{t-1} + W_{oc}c_t + b_o) \tag{6}$$
$$h_t = o_t \odot \tanh(c_t) \tag{7}$$
$$P(y_t|\mathbf{x}) = {y_t}^T g(W_{yh}h_t + b_y) \tag{8}$$

where σ denotes the sigmoid function; g denotes the softmax function; $\odot$ denotes the element-wise product among vectors; and i, f, o, and c are the input, forget, and output gates, and memory cell, respectively. W_{ix}, W_{ih}, W_{ic}, W_{fx}, W_{fh}, W_{fc}, W_{cx}, W_{ch}, W_{ox}, W_{oh}, W_{oc}, and W_{yh} are weight matrices, E is a weight matrix for the word and feature embedding, and b_i, b_f, b_c, b_o, and b_y are bias terms. Figure 2 shows a single LSTM memory block. Each memory block contains an input gate that controls the flow of input activations into the memory cell, a forget gate that scales the state of the cell before adding it as input to the cell through a self-recurrent connection of the cell, and an output gate that controls the output flow of cell activations into the rest of the network [4].

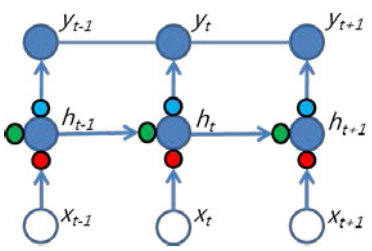

Fig. 3 LSTM-CRF architecture.

2.2 LSTM-CRF Model

To add dependence between output labels in LSTM RNN, we propose an LSTM-CRF model that extends the output layer, as given in (8), of LSTM RNN according to the following.

$$s_{word}(y_t, t) = {y_t}^T(W_{yh}h_t + b_y) \tag{9}$$
$$s_{sent}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{T}\{[A]_{y_{t-1},y_t} + s_{word}(y_t, t)\} \tag{10}$$
$$\log P(\mathbf{y}|\mathbf{x}) = s_{sent}(\mathbf{x}, \mathbf{y}) - \log \sum_{\mathbf{y}'} \exp(s_{sent}(\mathbf{x}, \mathbf{y}')) \tag{11}$$

where $s_{sent}(\mathbf{x}, \mathbf{y})$ denotes the score of output-tag sequence $\mathbf{y}$, $s_{word}(y_t, t)$ denotes the score of y_t tag at the t-th word, and the element of $[A]_{y,y'}$ of the weight matrix A denotes the transition score for jumping from y to y' output tags in successive words. The dynamic programming such as forward algorithm and Viterbi algorithm can be used efficiently to compute $P(\mathbf{y}|\mathbf{x})$ and optimal output-tag sequence for inference as in CRF. We use the standard cross-entropy as the objective function when training the proposed model. We train the model using back propagation through time (BPTT) with stochastic gradient descent. Given a sentence $\mathbf{x}$ at inference time, we use the Viterbi search algorithm to find the most probable output-tag sequence $\mathbf{y}$. Figure 3 shows the architecture of LSTM-CRF. LSTM-CRF has dependence between output labels and exploits long-range dependencies in the given sentence.

2.3 BI-LSTM RNN and BI-LSTM-CRF Models

In sequence labeling task, we can utilize a bidirectional LSTM RNN (BI-LSTM RNN) and a bidirectional LSTM-CRF (BI-LSTM CRF) to access to both past and future input words and features for a given time. Figure 4 shows the architecture of BI-LSTM CRF.

2.4 GRU, BI-GRU, GRU-CRF, and BI-GRU CRF Models

Recently, variations of LSTM RNN were proposed to address the vanishing gradients problem. Among those, we

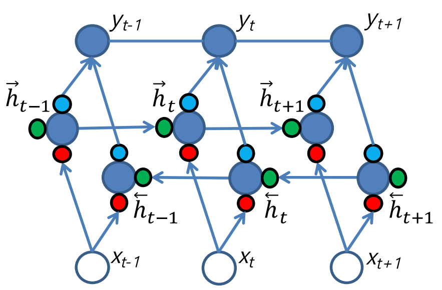

Fig. 4 BI-LSTM-CRF architecture.

use a GRU and SCRN, and we extend these models to add dependence between output labels.

Cho et al. proposed GRU, which functions effectively in sequence-based tasks with long-term dependencies [3]. Similar to the LSTM unit, GRU has gating units: reset and update gates. The update gate z selects whether the hidden state is to be updated with a new hidden state $\tilde{h}$. The reset gate r determines whether the previous hidden state is ignored. Detailed equations that describe GRU are as follows.

$$r_t = \sigma(W_{rx}Ex_t + W_{rh}h_{t-1} + b_r) \tag{12}$$

$$z_t = \sigma(W_{zx}Ex_t + W_{zh}h_{t-1} + b_z) \tag{13}$$

$$\tilde{h}_t = \phi(W_{hx}Ex_t + W_{hh}(r_t \odot h_{t-1}) + b_h) \tag{14}$$

$$h_t = z_t \odot h_{t-1} + (\mathbf{1} - z_t) \odot \tilde{h}_t \tag{15}$$

$$P(y_t|\mathbf{x}) = {y_t}^T g(W_{yh}h_t + b_y) \tag{16}$$

where W_{rx}, W_{rh}, W_{zx}, W_{zh}, W_{hx}, W_{hh}, and W_{yh} denote weight matrices; E denotes a weight matrix for the word and feature embedding; b_r, b_z, b_h, and b_y denote bias vectors; σ denotes the sigmoid function; ϕ is a nonlinear function (sigmoid or tanh function); and g is the softmax function. To add dependence between output labels in GRU, we propose a GRU-CRF model that extends the output layer, as shown in (16), of GRU according to the following.

$$s_{word}(y_t, t) = {y_t}^T(W_{yh}h_t + b_y) \tag{17}$$

$$s_{sent}(\mathbf{x}, \mathbf{y}) = \sum\nolimits_{t=1}^{T}\{[A]_{y_{t-1},y_t} + s_{word}(y_t, t)\} \tag{18}$$

$$\log P(\mathbf{y}|\mathbf{x}) = s_{sent}(\mathbf{x}, \mathbf{y}) - \log \sum\nolimits_{\mathbf{y}'} \exp(s_{sent}(\mathbf{x}, \mathbf{y}')) \tag{19}$$

A bidirectional GRU (BI-GRU) and a bidirectional GRU-CRF (BI-GRU CRF) are also used to access to both past and future input words and features for a given time.

2.5 SCRN and SCRN-CRF Models

Mikolov et al. proposed SCRN, which possesses both a fully connected recurrent matrix for producing a set of quickly changing hidden units and a diagonal matrix that encourages context units to change slowly [9]. The fast layer, h_t, can learn representations similar to n-gram models, whereas the slowly changing layer, z_t, can learn topic information, which is similar to the behavior of cache models. SCRN is defined according to the following.

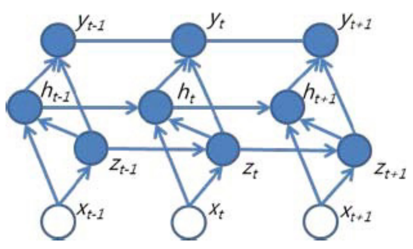

Fig. 5 SCRN-CRF architecture.

$$z_t = (1 - \alpha)BEx_t + \alpha z_{t-1} + b_z \tag{20}$$

$$h_t = \phi(Pz_t + AEx_t + Rh_{t-1} + b_h) \tag{21}$$

$$P(y_t|\mathbf{x}) = {y_t}^T g(W_{yh}h_t + W_{yz}z_t + b_y) \tag{22}$$

where α is a hyper-parameter in (0, 1), B, P, A, R, and W_{yh}, W_{yz} are weight matrices, E is a weight matrix for the word and feature embedding, b_z, b_h, and b_y are bias vectors, ϕ is a nonlinear function (sigmoid or tanh function), and g is the softmax function.

To add dependence between output labels in SCRN, we propose a SCRN-CRF model that extend the output layers, as shown in (22), of SCRN according to the following.

$$s_{word}(y_t, t) = {y_t}^T(W_{yh}h_t + W_{yz}z_t + b_y) \tag{23}$$

$$s_{sent}(\mathbf{x}, \mathbf{y}) = \sum\nolimits_{t=1}^{T}\{[A]_{y_{t-1},y_t} + s_{word}(y_t, t)\} \tag{24}$$

$$\log P(\mathbf{y}|\mathbf{x}) = s_{sent}(\mathbf{x}, \mathbf{y}) - \log \sum\nolimits_{\mathbf{y}'} \exp(s_{sent}(\mathbf{x}, \mathbf{y}')) \tag{25}$$

Figure 5 shows the architecture of SCRN-CRF.

2.6 Summary of Models

The SENNA architecture can use the dependencies between output labels. However, it can not capture long-range dependencies between faraway words, because it uses a simple feed-forward neural network.

R-CRF uses RNN to exploit long-range dependencies in the sequence of words. It can use the dependencies between output labels. However, training standard RNN is difficult because of the vanishing gradient problem.

Variations of RNN (i.e., LSTM RNN, BI-LSTM RNN, GRU, BI-GRU, and SCRN) can capture long-range dependencies in the sequence of words and solve the vanishing gradient problem in RNN. However, they do not explicitly model the dependencies between output labels.

LSTM-CRF and its variations (i.e., BI-LSTM CRF, GRU-CRF, BI-GRU CRF, and SCRN-CRF) can capture the dependencies between output labels and long-range dependencies between faraway words. They can also solve the vanishing gradient problem. However, their computational cost is high.

Table 1 summarizes the characteristics of the models proposed in this paper. #h denotes the number of hidden units, #c denotes the number of context units for SCRN and SCRN-CRF, and α is the hyper-parameter of SCRN and SCRN-CRF. Dropout rate, initial learning rate, weight decay rate, and activation function are omitted in the table because

Table 1 A brief summary of models.

Algorithm	Long-range dependency	Label dependency	Sequence-level objective func.	Hyper-parameter
SENNA	X	O	O	#h
FFNN	X	X	X	#h
RNN	Δ	X	X	#h
LSTM RNN	O	X	X	#h
BI-LSTM RNN	O	X	X	#h
GRU	O	X	X	#h
BI-GRU	O	X	X	#h
SCRN	O	X	X	#h, #c, α
R-CRF	Δ	O	O	#h
LSTM-CRF	O	O	O	#h
BI-LSTM-CRF	O	O	O	#h
GRU-CRF	O	O	O	#h
BI-GRU-CRF	O	O	O	#h
SCRN-CRF	O	O	O	#h, #c, α

they are hyperparameters that are common to all models.

3. Experiments

We evaluate CRF, feed forward neural network (FFNN), RNN, LSTM RNN, LSTM-CRF, BI-LSTM RNN, BI-LSTM CRF, GRU, GRU-CRF, BI-GRU, BI-GRU CRF, SCRN, and SCRN-CRF models on a NER task. To compare the performance of these modes with existing studies, we used the CoNLL03 data set[†], which is a NER benchmark data set based on Reuters data, and ETRI Korean NER data set [11]. We train models using training data and tune hyperparameters of models using validation data. Because ETRI Korean NER data set do not have a validation data set, we use part of training data for validation purpose. Table 2 shows the size of tokens and labels for training, validation, and test sets respectively.

We use two types of vector values for input vector x_t: word and feature embedding. For word embedding, we used the Collobert and Weston English word embedding and Korean word embedding produced by Word2vec [14]. We used randomly initialized value for feature embedding. For features, we use the capital letter and gazetteer features as in [8]. Capital letter features indicate whether each word is in lowercase, is all uppercase, has first letter capital, or has at least one non-initial capital letter. Gazetteer features indicate whether a word or chunks of words are found in the gazetteer under one of four categories (i.e., person, location, organization, and miscellaneous entities). We used the gazetteer provided by the CoNLL challenge as in [8]. For CRF, we use word, POS tag, capital letter, gazetteer, word cluster, and word embedding features.

We implemented our proposed models using Theano [12]. We use stochastic gradient descent with momentum with a learning rate of 0.1 to train modes. We divide the learning rate by 2 after each training epoch when the validation error does not decrease.

We applied the dropout [13] to the projection and hidden layers in all models except CRF (dropout probabilities are 0.2 and 0.5 for the projection and hidden layers, respectively).

In our experiments, we chose the hyperparameters of our models by trying a few different networks over the validation set. For example, the number of hidden units from {50, 100, 200, 300, 500, 700, 900, 1100, 1300} was chosen by optimization on the validation set. For each experiment, we report the average and standard deviation of 10 trials with random initial parameters.

Figure 6 shows the F1 score for each model on the CoNLL03 validation set with respect to the number of hidden units.

Table 3 shows the numbers of hidden units and context units which were chosen by optimization on the validation set, the number of parameters of different models, and the F1 score on the CoNLL03 test set. RNN and its variations

[†]See http://www.cnts.ua.ac.be/conll2003/ner.

Table 2 Size of tokens and labels for training, validation, and test sets.

Data set	#labels	Training set (#tokens)	Validation set (#tokens)	Test set (#tokens)
CoNLL03	9	203,621	51,362	46,435
ETRI Korean NER data set	7	2,135,374	N/A	74,114

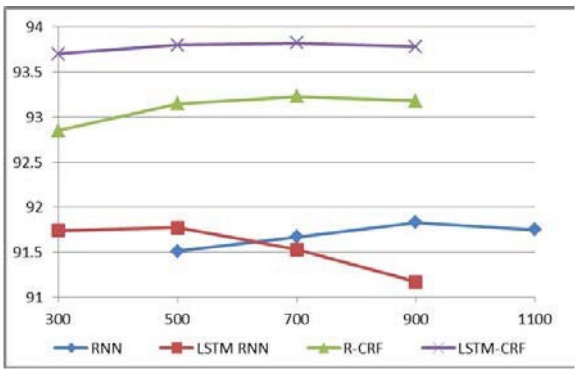

Fig. 6 F1 score on the CoNLL03 validation set (y-axis) vs. number of hidden units (x-axis).

Table 3 F1 score results with different models in CoNLL03 test set. For F1 scores, standard deviations are in parentheses.

Algorithm	#hidden units	#context units	#parameters	F1
SENNA	700	-	389k	89.59
CRF	-	-	10,788k	85.58
FFNN	900	-	471k	85.72 (± 0.13)
RNN	900	-	1,282k	86.78 (± 0.11)
LSTM RNN	500	-	1,909k	86.96 (± 0.13)
BI-LSTM RNN	500	-	3,717k	88.13 (± 0.13)
GRU	500	-	1,458k	86.74 (± 0.20)
BI-GRU	500	-	2,815k	88.11 (± 0.19)
SCRN	1300	50	2,412k	86.83 (± 0.16)
R-CRF	700	-	880k	89.54 (± 0.32)
LSTM-CRF	700	-	3,192k	89.95 (± 0.15)
BI-LSTM-CRF	900	-	13,335k	**90.24** (± 0.26)
GRU-CRF	1300	-	6,749k	89.92 (± 0.17)
BI-GRU-CRF	1100	-	10,030k	89.97 (± 0.21)
SCRN-CRF	900	100	1,413k	89.43 (± 0.24)

Table 4 An example of results of NER in CoNLL03 data set.

1. Newmont , in fact , will not benefit from the **Santa Fe** acquisition on an earnings basis …	
Answer	Newmont/B-ORG ,/O in/O fact/O ,/O will/O not/O benefit/O from/O the/O **Santa/B-ORG Fe/I-ORG** acquisition/O on/O an/O earnings/O basis/O …
SENNA	Newmont/B-ORG ,/O in/O fact/O ,/O will/O not/O benefit/O from/O the/O **Santa/B-LOC Fe/I-LOC** acquisition/O on/O an/O earnings/O basis/O …
BI-LSTM-CRF	Newmont/B-ORG ,/O in/O fact/O ,/O will/O not/O benefit/O from/O the/O **Santa/B-ORG Fe/I-ORG** acquisition/O on/O an/O earnings/O basis/O …
2. **Lazio** have injury doubts about striker …	
Answer	**Lazio/B-ORG** have/O injury/O doubts/O about/O striker/O …
SENNA	**Lazio/B-LOC** have/O injury/O doubts/O about/O striker/O …
BI-LSTM-CRF	**Lazio/B-ORG** have/O injury/O doubts/O about/O striker/O …
3. … Zywiec had its eye on **Okocim** , which has said it would start producing …	
Answer	… Zywiec/B-ORG had/O its/O eye/O on/O **Okocim/B-ORG** ,/O which/O has/O said/O it/O would/O start/O producing/O …
SENNA	… Zywiec/B-ORG had/O its/O eye/O on/O **Okocim/B-LOC** ,/O which/O has/O said/O it/O would/O start/O producing/O …
BI-LSTM-CRF	… Zywiec/B-ORG had/O its/O eye/O on/O **Okocim/B-ORG** ,/O which/O has/O said/O it/O would/O start/O producing/O …
4. The 16-year-old who attended **Sale Grammar School** in …	
Answer	The/O 16-year-old/O who/O attended/O **Sale/B-ORG Grammar/I-ORG School/I-ORG** in/O …
BI-LSTM RNN	The/O 16-year-old/O who/O attended/O **Sale/B-MISC Grammar/I-ORG School/O** in/O …
BI-LSTM-CRF	The/O 16-year-old/O who/O attended/O **Sale/B-ORG Grammar/I-ORG School/I-ORG** in/O …
5. 5-1 Draw **Real Madrid**	
Answer	5-1/O Draw/O **Real/B-ORG Madrid/I-ORG**
BI-LSTM RNN	5-1/O Draw/O **Real/O Madrid/I-ORG**
BI-LSTM-CRF	5-1/O Draw/O **Real/B-ORG Madrid/I-ORG**
6. … office in **Downing Street** …	
Answer	… office/O in/O **Downing/B-LOC Street/I-LOC** …
BI-LSTM RNN	… office/O in/O **Downing/B-LOC Street/I-ORG** …
BI-LSTM-CRF	… office/O in/O **Downing/B-LOC Street/I-LOC** …

(i.e., LSTM RNN, BI-LSTM RNN, GRU, BI-GRU, and SCRN) except SCRN outperform CRF and FFNN. LSTM-CRF and its variations (i.e., BI-LSTM-CRF, GRU-CRF, BI-GRU-CRF, and SCRN-CRF) except SCRN-CRF significantly outperform SENNA, R-CRF, RNN, and variations of RNN (Paired t-test, $p < 0.005$). BI-LSTM-CRF shows the best performance, but it has substantially more parameters than other models.

Table 4 shows the example of results of NER on the CoNLL03 test set. SENNA is apt to confuse Organization with Location, but BI-LSTM-CRF classifies them correctly with faraway clue words. BI-LSTM RNN does not capture the dependencies between output labels (e.g. Sale/B-MISC Grammar/I-ORG School/O, Real/O Madrid/I-ORG, and Downing/B-LOC Street/I-ORG), but BI-LSTM-CRF captures the label dependencies.

A plausible explanation of these improvements over SENNA, R-CRF, and variations of RNN is that LSTM-CRF and its variations effectively capture the dependencies between output labels and long-range dependencies between faraway words. For example, LSTM-CRF can use past input words and features using a LSTM layer and use past and future output labels using a CRF layer.

Table 5 F1 score results with different models in ETRI Korean NER data set. For F1 scores, standard deviations are in parentheses.

Algorithm	#hidden units	#context units	#parameters	Test set
SENNA	100	-	153k	90.04 (± 0.14)
CRF	-	-	51,914k	89.03
FFNN	50	-	127k	86.39 (± 0.62)
RNN	100	-	163k	88.74 (± 0.37)
LSTM RNN	50	-	212k	89.40 (± 0.27)
BI-LSTM RNN	50	-	323k	89.38 (± 0.19)
GRU	200	-	524k	89.34 (± 0.15)
BI-GRU	100	-	464k	89.78 (± 0.12)
SCRN	300	30	279k	87.47 (± 0.34)
R-CRF	100	-	163k	90.36 (± 0.24)
LSTM-CRF	200	-	664k	**90.89** (± 0.12)
BI-LSTM-CRF	100	-	584k	90.71 (± 0.15)
GRU-CRF	200	-	524k	**90.89** (± 0.07)
BI-GRU-CRF	100	-	464k	90.78 (± 0.12)
SCRN-CRF	200	100	315k	90.07 (± 0.32)

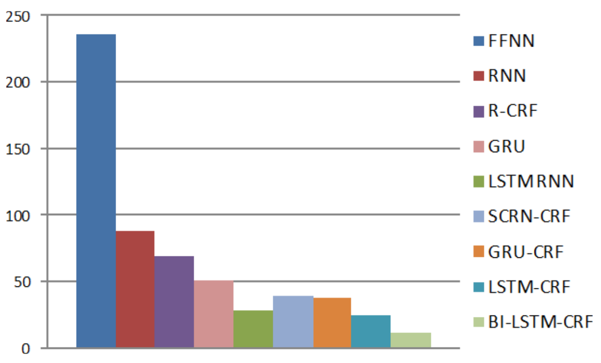

Fig. 7 Training speed (#sentence/sec.) of various models (y-axis).

Table 5 shows the numbers of hidden units and context units which were chosen by a minimal validation, the number of parameters of different models, and the F1 score on the ETRI Korean NER test set. We re-implemented SENNA model as a baseline for comparison of Korean NER data set. Similar to Table 3, LSTM-CRF and its variations except SCRN-CRF outperform CRF, FFNN, SENNA, R-CRF, RNN, and variations of RNN.

Figure 7 shows the training speed (i.e., #sentence/sec.) of various models. We can see a trade-off between the speed and accuracy. For example, BI-LSTM-CRF and LSTM-CRF, which archive the best performance in Table 3 show the worst training speed. However, GRU-CRF which performs similarly to LSTM-CRF, show faster training speed than does LSTM-CRF.

4. Conclusion

In this study, we proposed a LSTM-CRF model that specializes in sequence labeling as it adds output label dependency to the LSTM RNN model. We also propose variations to the LSTM-CRF model using a GRU and SCRN. Experiments show that our models outperform the existing models such as CRF, FFNN, SENNA, RNN, variations of RNN, and R-CRF.

For future research, we will apply the proposed model to problems similar to NER such as chunking, semantic role labeling, and natural language comprehension.

Acknowledgments

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (no. R0101-16-0062, Development of Knowledge Evolutionary WiseQA Platform Technology for Human Knowledge Augmented Services).

References

[1] T. Mikolov, S. Kombrink, L. Burget, J.H. Černocký, and S. Khudanpur, "Extensions of recurrent neural network language model," 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp.5528–5531, 2011.

[2] A. Graves, A.R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," Proc. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp.6645–6649, 2013.

[3] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," arXiv preprint arXiv:1406.1078, 2014.

[4] A. Graves, "Generating sequences with recurrent neural networks," arXiv preprint arXiv:1308.0850, 2013.

[5] H. Sak, A.W. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," Proc. INTERSPEECH. 2014.

[6] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi, "Spoken language understanding using long short-term memory neural networks," Proc. 2014 IEEE Spoken Language Technology Workshop (SLT), IEEE, pp.189–194, 2014.

[7] J. Lafferty, A. McCallum, and F.C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," Proc. 18th International Conference on Machine Learning 2001 (ICML 2001), pp.282–289. 2001.

[8] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," J. Machine Learning Research, vol.12, pp.2493–2537, 2011.

[9] T. Mikolov, A. Joulin, S. Chopra, M. Mathieu, and M.A. Ranzato, "Learning longer memory in recurrent neural networks," arXiv preprint arXiv:1412.7753, 2014.

[10] K. Yao, B. Peng, G. Zweig, D. Yu, X. Li, and F. Gao, "Recurrent conditional random field for language understanding," Proc. 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp.4077–4081, 2014.

[11] C. Lee, P.M. Ryu, and H. Kim, "Named entity recognition using a modified Pegasos algorithm," Proc. 20th ACM international conference on Information and knowledge management, ACM, p.2337, 2011.

[12] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, and Y. Bengio, "Theano: new features and speed improvements," arXiv preprint arXiv:1211.5590, 2012.

[13] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R.R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," arXiv preprint arXiv:1207.0580, 2012.

[14] T. Mikolov, W.T. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," Proc. HLT-NAACL, 2013.

Changki Lee received the B.S. in computer science from KAIST, Rep. of Korea, in 1999. He received the M.S. and PhD in computer engineering from POSTECH, Rep. of Korea, in 2001 and 2004, respectively. During 2004-2012, he had been with ETRI, Rep. of Korea, as a senior member of research staff. Since 2012, he has been with Kangwon National University, Rep. of Korean.