Bidirectional LSTM-CRF for Named Entity Recognition

Conference Paper · May 2019

CITATIONS
44

READS
2,175

2 authors, including:

Rrubaa Panchendrarajan
Sri Lanka Institute of Information Technology
15 PUBLICATIONS **86** CITATIONS

SEE PROFILE

Bidirectional LSTM-CRF for Named Entity Recognition

Rrubaa Panchendrarajan
School of Computing
National University of Singapore
rrubaa@comp.nus.edu.sg

Aravindh Amaresan
School of Computing
National University of Singapore
e0146310@u.nus.edu

Abstract

Named Entity Recognition (NER) is a challenging sequence labeling task which requires a deep understanding of the orthographic and distributional representation of words. In this paper, we propose a novel neural architecture that benefits from word and character level information and dependencies across adjacent labels. This model includes bidirectional LSTM (BI-LSTM) with a bidirectional Conditional Random Field (BI-CRF) layer. Our work is the first to experiment BI-CRF in neural architectures for sequence labeling task. We show that CRF can be extended to capture the dependencies between labels in both right and left directions of the sequence. This variation of CRF is referred to as BI-CRF and our results show that BI-CRF improves the performance of the NER model compare to an unidirectional CRF and backward CRF is capable of capturing most difficult entities compare to the forward CRF. Our system is competitive on the CoNLL-2003 dataset for English and outperforms most of the existing approaches which do not use any external labeled data.

1 Introduction

Named Entity Recognition is an important task in Natural Language Processing (NLP) which has drawn the attention for a few decades. NER is widely used in downstream applications of NLP and artificial intelligence such as machine translation, information retrieval, and question answering. Initially experimented sequence labeling models are linear statistical models which include Hidden Markov models (HMM) (Morwal et al., 2012), Maximum Entropy models (MEMM) (McCallum et al., 2000) and Conditional Random Fields (CRF) (Lafferty et al., 2001). Later, (Qi et al., 2009) proposed an effective neural network model using the Convolutional neural network (CNN) combined with CRF. However, this was purely based on word embedding and not capable of processing variable length input.

A well-studied solution for a neural network to take into account an effectively infinite amount of context is the BI-LSTM. (Graves et al., 2013) successfully applied BI-LSTM for speech recognition task. CNNs have also been investigated for modeling character-level information, among other NLP tasks and combination of BI-LSTM, CNN and CRF has been shown to be very successful in the field of sequence labeling task in past few years. BI-LSTM-CRF (Huang et al., 2015), BI-LSTM-CNN (Chiu and Nichols, 2016), BI-LSTM-CRF (Lample et al., 2016) and LSTM-CNN-CRF (Ma and Hovy, 2016) are few such architectures. Most recent approaches to NER have focused on multi-task learning (Liu et al., 2018a) which jointly conduct other related NLP tasks like entity linking or chunking. Moreover, very complex architectures such as stacked Recurrent Neural Network (Tran et al., 2017) and stack of classifiers (Liu et al., 2018b) are experimented in state-of-the-art models.

Conditional random field (CRF) jointly models the label decision by capturing the dependencies across adjacent labels. For example, *I-XXX* (label indicates that the given word appears inside the name

of entity type *XXX*) of an entity cannot follow the *B-YYY* (label indicates that the given word appears in the beginning of the name of entity type *YYY*) of a different entity. CRF has shown to be very effective when combining with neural architectures (Lample et al., 2016; Peters et al., 2017) for sequence labeling task. However, the models with unidirectional CRF (generally referred to as CRF) are capable of capturing the dependencies between labels in the forward direction only. This may mislead the prediction of labels for a word sequence which is highly ambiguous. For example in the sentence, "Jones Medical completes acquisition", word *Jones* is generally being used as a name of a person. Labeling the sequence by considering the word sequence in forward direction only may mislead the model to predict the word *Jones* as an entity type *Person* (*B-PER*) and *Medical* as an outside of a named entity (*O*). If the word sequence is provided in the reverse direction as well to a CRF model, identifying that the word *Medical* is part of an entity type *Organization* (*I-ORG*) will help the model to realize that the ambiguous word *Jones* is also part of *Organization* entity (*B-ORG*). Therefore, we propose a novel neural architecture with an extended version of CRF, bidirectional CRF (BI-CRF) which models the dependencies between labels in both directions. (Hsu et al., 2008) and (Murugesan et al., 2017) have successfully experimented BI-CRF in the field of medicine for NER. However, these approaches purely rely on BI-CRF, thus fail to utilize neural networks to automatically learn character and word level features. Our work is the first to apply BI-CRF in a neural architecture for NER.

In this paper, we present a neural architecture based on BI-LSTM and BI-CRF. The model consists of three components: a word embedding layer, BI-LSTM, and a BI-CRF. We use the character-based representation learning model (Lample et al., 2016) combined with pre-trained word embedding, Part-of-speech (POS) tag and casing features as the initial layer. Character-based representation learning model captures the orthographic representation (what does the word being tagged as a name look like?) of a word and pre-trained word embedding allows to represent the distributional evidence of words (where does the word being tagged tend to occur in a corpus?) in a vector space. This layer

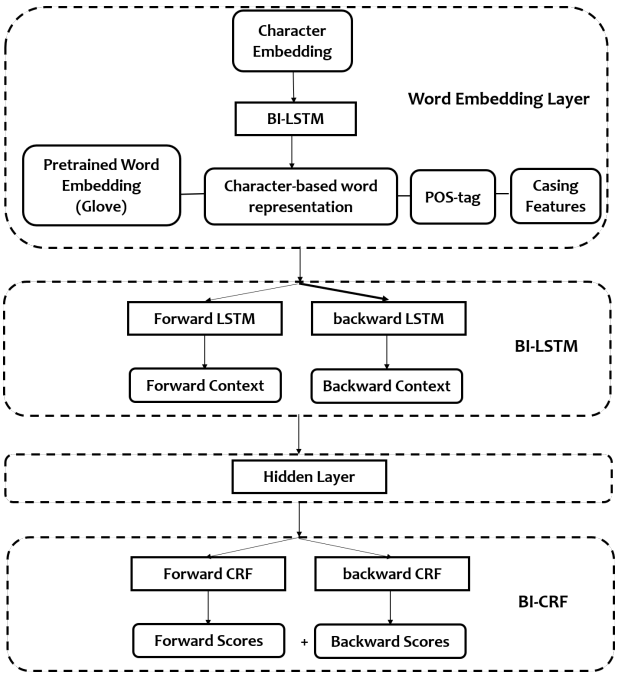

Figure 1: Architecture of the network

is coupled with a BI-LSTM on top of it to generate a context-based representation of words. Final BI-CRF layer uses the dependencies across labels in both left and right directions and the scores computed by previous layer to predict the correct label sequence. Experiments in English dataset of CoNLL-2003 (Sang and Meulder, 2003) shows, our model is competitive and outperforms most of the existing approaches that do not use any external labeled data. Moreover, we show that backward CRF is capable of capturing complex named entities thus it improves the performance of the model compare to an unidirectional CRF.

The reminder of the paper is organized as follows. Section 2 describes our model and Section 3 explains the training process. Section 4 provides the results and Section 5 discusses the performance analysis of BI-CRF.

2 Model

Our neural network is inspired by (Lample et al., 2016), where the combination of BI-LSTM and CRF is applied for a language-independent NER with a small supervised training data. Instead of a CRF, we use BI-CRF to capture the dependencies between labels in both right and left directions of a sequence.

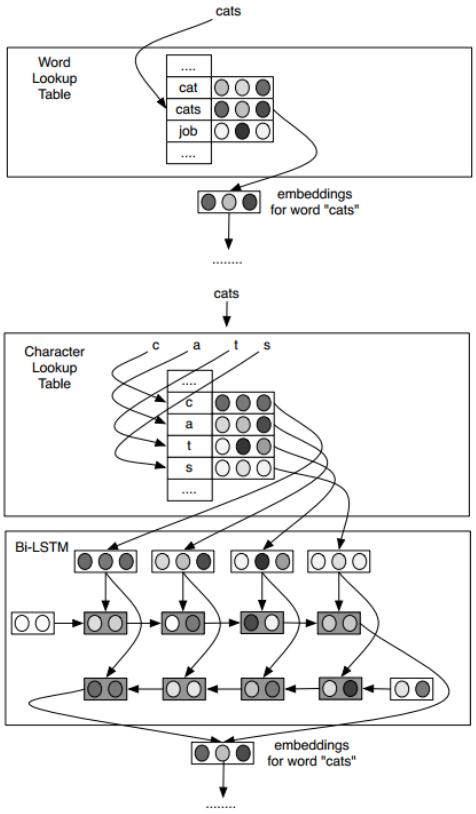

Figure 2: Illustration of character-based representation learning using BI-LSTM

Figure 1 shows the architecture of the network. We introduce the layers in our network in the following subsections.

2.1 Word embedding layer

This layer computes a vector representation of a word as a combination of following four features:

- Character-based vector representation
- Pre-trained word embedding
- POS tag of the word
- Casing features of the word

Character-based word representation is learned using a BI-LSTM. A character look-up table representing the character embedding is randomly initialized with the size of all possible characters. Word vector constructed by concatenating the character embedding of all the characters appearing in a word is given as the input to a BI-LSTM. In our experiments, the hidden dimension of 24 is used for forward and backward LSTM so that a 48-dimensional character-based vector representation is learned using this BI-LSTM. Figure 2 illustrates the architecture of this component of the initial layer. Learning character level information using BI-LSTM allows to automatically capture the task-specific information at character level without using handcraft features such as prefix and suffix of a word. Moreover, this has been found to be useful to handle the out-of-vocabulary problem in NER tasks (Ling et al., 2015; Ballesteros et al., 2015).

Even though CNNs have been experimented in past (Zhang et al., 2015; Kim et al., 2016) to generate a character-based representation of words, it is designed to discover the position-invariant features of the input, especially in the field of computer vision. For example, detecting a cat appearing in an image is a position invariant task. However, the character-based information in a word such as capitalization, prefix, and suffix are position dependent so that BI-LSTM would be a better option for capturing the character-based representation of a word compare to a CNN. Character-based representation and word embedding are concatenated with POS tag and casing features to obtain the final embedding representation of a word. The following casing features are extracted for a given word as boolean representation.

- Start with capital letter
- All capital letters
- All lower case letters
- All digits
- Mix of words and digits

We observed a significant improvement in our model's performance after adding POS tag and casing features to the embedding layer (see Table 5). Following (Lample et al., 2016), dropout (Srivastava et al., 2014) is applied to the embedding layer which is a regularization technique for reducing overfitting in neural networks by randomly dropping neurons in the network with a certain probability. This encourages the next layer (BI-LSTM) to utilize all four

features of this layer to learn the contextual vector representation of a word in the next layer.

2.2 Bidirectional LSTM

Long Short-term Memory Networks (LSTM) (Hochreiter and Schmidhuber, 1997) are a special kind of Recurrent Neural Network, capable of learning long-term dependencies. The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates. Formally, the formulas to update an LSTM unit at time t are,

$$
\begin{aligned}
i_t &= \sigma(W_i[x_t; h_{t-1}] + b_i) \\
f_t &= \sigma(W_f[x_t; h_{t-1}] + b_f) \\
o_t &= \sigma(W_o[x_t; h_{t-1}] + b_o) \\
\tilde{c} &= tanh(W_c[x_t; h_{t-1}] + b_c) \\
c_t &= f_t \cdot c_{t-1} + i_t \cdot \tilde{c_t} \\
h_t &= o_t \cdot tanh(c_t)
\end{aligned}
$$

where σ is the element-wise sigmoid function and $\cdot$ is the element-wise matrix multiplication. x_t is the input vector at time t and h_t represents the hidden state vector. Weights W_i, W_f, W_o, W_c and bias b_i, b_f, b_o, b_c are the parameters to be learned. Given a sequence of input vectors $(x_1, x_2, ...x_n)$, LSTM computes a context representation vector h_t for each input x_t.

When processing a sequence of words, both past and future inputs are known for a given time thus allows to effectively utilize the features in both right and left directions. This variation of the LSTM is referred to as bidirectional LSTM (BI-LSTM) (Graves and Schmidhuber, 2005). Here, the input is given to forward and backward LSTMs to capture both left and right context of the word. Final representation of a word is obtained by concatenating the left context $\overrightarrow{h_t}$ and right context $\overleftarrow{h_t}$, $\tilde{h_t} = [\overrightarrow{h_t}; \overleftarrow{h_t}]$. We use 150-dimensional LSTM in forward and backward direction so that, a 300-dimensional vector representation is learned for each word.

Following (Lample et al., 2016), a hidden layer of size, equal to the number of distinct labels is placed on top of BI-LSTM and $\tilde{h_t}$ is projected into this hidden layer. This layer produces a score matrix P, where $P_{i,j}$ represents the score of j^{th} tag of i^{th} input token. This score matrix is given to the next layer, BI-CRF.

2.3 Bidirectional Conditional Random Field

The discriminative method of classifiers model the conditional probability distribution $p(\mathbf{y} \mid \mathbf{x})$ directly. This approach is used by Conditional Random Field (CRF) (Lafferty et al., 2001) which combines the advantage of graphical modeling to predict multivariate output $\mathbf{y}$ with a large number of input features $\mathbf{x}$. CRF is a variation Markov Random Field where all the clique potentials ϕ_c, $1 \leq c \leq C$ are conditioned on input features. The log-linear representation of potential is generally assumed for the cliques. A simple case of CRF, known as Linear Chain CRF built on top of a BI-LSTM effectively models several hard constraints incorporating dependencies across the output labels. Considering general definition of CRF, let $\mathbf{x} = \{x_1, \cdots, x_T\}$ and $\mathbf{y} = \{y_1, \cdots, y_T\}$ represent observed input tokens and corresponding output labels respectively. A linear-chain CRF's distribution $p(\mathbf{y} \mid \mathbf{x})$ given by

$$
p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_c \phi_c(\mathbf{y}, \mathbf{x}) \tag{1}
$$

where $Z(\mathbf{x})$ is a normalization function

$$
Z(\mathbf{x}) = \sum_y \prod_c \phi_c(\mathbf{y}_c, \mathbf{x}) \tag{2}
$$

As the hidden layer on top of the BI-LSTM produces the score matrix P for a given sequence, the CRF layer learns only the transition probability of the output labels, $A \in \mathcal{R}^{K+2 \times K+2}$. K is the number of distinct labels and *+2* indicates one tag each for start and end marker. The input features of observed tokens $\mathbf{x} \in \mathcal{R}^K$ in the clique $\phi_c(\mathbf{x}, \mathbf{y})$ is the score matrix P_{i,y_i} learned by the hidden layer. For the given sequence of predictions $\mathbf{y} = \{y_1, \cdots, y_T\}$, the probability score including the start and end tag, y_0 and y_{T+1} introduced in the inference algorithm is defined as

$$
S(\mathbf{x}, \mathbf{y}) = \sum_{i=0}^{T} A_{y_i, y_{i+1}} + \sum_{i=1}^{T} P_{i,y_i} \tag{3}
$$

The probability for the sequence $\mathbf{y}$ is given by

$$
p(\mathbf{y} \mid \mathbf{x}) = \frac{e^{S(\mathbf{x}, \mathbf{y})}}{\sum_{y' \in \mathbf{y}} e^{S(\mathbf{x}, \mathbf{y}')}} \tag{4}
$$

The objective function is the maximum likelihood of the probability distribution denoted as,

$$\ln p(\mathbf{y} \mid \mathbf{x}) = S(\mathbf{x}, \mathbf{y}) - \ln \sum_{y' \in \mathbf{y}} e^{S(\mathbf{x}, \mathbf{y}')} \quad (5)$$

During training, maximum likelihood of the probability of correct sequences in the training are maximized. The final output tag sequence is decided based on the maximum score given by

$$\mathbf{y}^* = \underset{y' \in \mathbf{y}}{\operatorname{argmax}} \; S(\mathbf{x}, y') \quad (6)$$

(5) and (6) can be computed efficiently using dynamic programming (Lafferty et al., 2001) based algorithm known as forward and backward inference since we model only the binary interactions among the output labels.

We use a variant of CRF referred to as bidirectional CRF (BI-CRF) which dissects the entities in both forward and backward directions. In forward parsing (CRF_A), the input tokens $\mathbf{x} = \{x_1, \cdots, x_T\}$ are read and labelled in the original direction (left to right) and in backward parsing (CRF_B), the original input order is reversed and labelled (right to left), $\mathbf{x} = \{x_T, \cdots, x_1\}$ with its corresponding labels $\mathbf{y} = \{y_T, \cdots, y_1\}$. The final probability scores are computed by adding the scores of CRF_F and CRF_B. Hence, the optimal output tag sequence is given by

$$\mathbf{y}^* = \underset{y' \in \mathbf{y}}{\operatorname{argmax}} \left[S_A(\mathbf{x}, y') + S_B(\mathbf{x}, y') \right] \quad (7)$$

During training, the negative sum of log probability of the correct sequences of both forward CRF ($\ln p(\mathbf{y} \mid \mathbf{x})_F$) and backward CRF ($\ln p(\mathbf{y} \mid \mathbf{x})_B$) is minimized as indicated by equation (8). Different variations of combining the forward CRF and backward CRF results like maximum and average are experimented and it is noticed that the summation operation gives the optimal performance (refer Table 5). Figure 3 illustrates the connectivity between the last three layers, BI-LSTM, a hidden layer and BI-CRF.

$$loss = -(\ln p(\mathbf{y} \mid \mathbf{x})_F + \ln p(\mathbf{y} \mid \mathbf{x})_B) \quad (8)$$

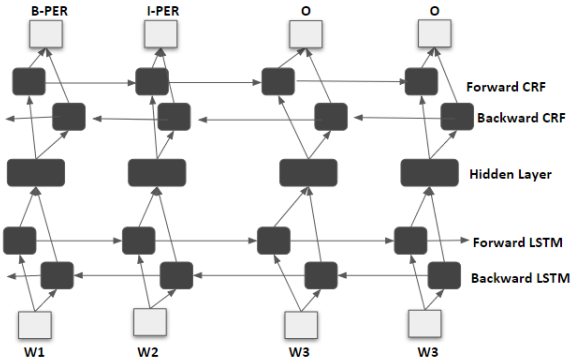

Figure 3: Illustration of Final Three Layers

Dataset	No. of Sentences	No. of tokens
Train	14,987	203,621
Dev	3,466	51,362
Test	3,684	46,435

Table 1: Number of sentences and tokens in CoNLL-2003 dataset

Dataset	LOC	MISC	ORG	PER
Train	7140	3438	6321	6600
Dev	1837	922	1341	1842
Test	1668	702	1661	1671

Table 2: Number of entities in CoNLL-2003 dataset

3 Training

3.1 Dataset

As mentioned before, we evaluate our neural model on CoNLL-2003 dataset (Sang and Meulder, 2003) for English language. The dataset contains four different types of named entities: *Person* (*PER*), *Organization* (*ORG*), *Location* (*LOC*) and *Miscellaneous* (*MISC*). Sentences in the dataset are represented in the IOB format (Inside, Outside and Beginning) where a token is labeled as *I-XXX* if it is inside a named entity type XXX, *O* if it is outside named entities and *B-XXX* if it is beginning of an entity type XXX. Therefore, the model is trained to classify a word sequence into nine different labels including one *O* label and *B-XXX* & *I-XXX* tags for each type of entity.

Table 1 and Table 2 summarize the statistics of the dataset (Sang and Meulder, 2003). As the size of the dataset is small, the final performance of the model

is reported by training the model by combining the training and validation set after tuning the hyper-parameters. Moreover, it is worth to note that the number of occurrences of *MISC* entity in the dataset is very low compared to other three types of entities. All the digits in the dataset were replaced with zero as an only pre-processing step.

3.2 Tuning Hyper-Parameters

The model is trained using back-propagation algorithm (Hecht-Nielsen, 1989) to minimize the combined loss of forward and backward CRF as indicated by equation (8). Parameter optimization is performed with stochastic gradient descent (SGD) with a learning rate of 0.01. We explored more sophisticated optimization algorithms such as Adam (Kingma and Ba, 2014). Even though, other optimization algorithms lead to faster convergence, none of them meaningfully improve upon SGD with gradient clipping in our preliminary experiments. Hyper-parameters are selected based on the performance of the development dataset. To reduce the effect of "gradient exploding", gradient clipping of 5.0 (Pascanu et al., 2012) is used. Table 3 summarizes the hyper-parameter space evaluated and the best parameters used for the final evaluation in the test set. We use early stopping (Caruana et al., 2000) based on performance on development dataset. After obtaining the optimal values for the hyper-parameters, validation set is combined with the training set and the model is trained again to evaluate the final performance of the model.

Hyper-parameter	**Final**	**Range**
Batch size	14	[5, 20]
Char embedding	25	[20,50]
Char LSTM size	24	[10, 35]
BI-LSTM size	150	[50, 250]
Learning rate	0.01	$[10^{-3}, 10^{-1.8}]$
Gradient clipping	5.0	[1, 10]
Dropout	0.3	[0.1, 0.8]
Epochs	80	-
Glove dimension	100	[5-300]

Table 3: Hyper-parameter search space and the optimal values chosen for the final evaluation. LSTM size indicates the hidden state size of the corresponding LSTM and Dropout indicate the dropout value applied to the word embedding layer

4 Results

Table 4 shows the F1 scores of other models and our model on the test dataset from CoNLL-2003 (Sang and Meulder, 2003). To make a fair comparison, we report the scores of other models with and without the use of external labeled data. Our model outperforms most of the existing approaches which do not use any external labeled data. (Liu et al., 2018b) uses a stack of classifiers and (Liu et al., 2018a) experiments multi-task learning to improve the scores. (Tran et al., 2017) uses a stack of RNN which is more complex compared to a simple combination of BI-LSTM and CRF. The proposed model's performance is only behind the performance of simple architectures, BI-LSTM-CRF (Lample et al., 2016) and CNN-BI-LSTM-CRF (Ma and Hovy, 2016). (Ma and Hovy, 2016) uses CNN

Model	**F1**
(Rei, 2017)*	86.26
(Qi et al., 2009)*	89.59
(Luo et al., 2015)	89.9
(Passos et al., 2014)	90.05
(Huang et al., 2015)*	90.10
(Yang et al., 2017)	90.20
(Yang et al., 2017)*	90.26
(Dernoncourt et al., 2017)*	90.54
(Chiu and Nichols, 2016)	90.69
(Chiu and Nichols, 2016)*	90.77
(Peters et al., 2017)	90.79
(Ratinov and Roth, 2009)*	90.80
(Passos et al., 2014)*	90.90
(Lample et al., 2016)	90.94
(Ratinov and Roth, 2009)*	91.20
(Ma and Hovy, 2016)	91.21
(Tran et al., 2017)	91.69
(Ghaddar and Langlais, 2018)*	91.73
(Liu et al., 2018a)	91.85
(Peters et al., 2017)*	91.93
(Liu et al., 2018b)	92.38
Our Model	90.84

Table 4: NER results for CoNLL-2003 test dataset. * indicates, the model is trained with the use of external labeled data

Variation	F1
Basic	89.763
Basic + dropout	90.398
Basic + dropout + POS tag + cas	90.84
Basic + dropout (uni-CRF)	90.218
Model + maximum	90.722
Model + average	90.716

Table 5: Results of the model using different configurations. "Basic" indicates the model without dropout or any other additional features. "cas" indicates the casing features and "uni-CRF" indicates the unidirectional CRF. *maximum* and *average* indicates operation performed to combined backward and forward CRFs

to extract a character-level representation of words. Both these approaches use IOBES tagging scheme, a variant of IOB which encodes information about singleton entities (S) and explicitly marks the end of a named entity (E). (Ratinov and Roth, 2009) has shown that using a more expressive tagging scheme like IOBES improves the model performance marginally. Therefore this could be a possible reason for why our model's performance is behind both (Lample et al., 2016) and (Ma and Hovy, 2016).

Moreover, we explored the impact of different components of the system. Table 5 shows the results of this evaluation. The basic model shows the result of the system when dropout at word embedding layer is not applied and POS tag and casing features are not provided to the model. We observed that dropout at word embedding layer encourages the model to utilize both character level representation and pre-trained word embedding for the learning. Dropout at word embedding layer improves the performance of the model by 0.635. Moreover, the language-specific features such as Part-of-speech (POS) tag and casing features further improves the performance by 0.35.

In order to evaluate the contribution of the BI-CRF, we report the performance of the model with unidirectional CRF as well. The architecture of this model is same as (Lample et al., 2016). It can be observed that the performance of the model with unidirectional CRF is behind its corresponding BI-CRF. This indicates that BI-CRF helps the model to identify more complex entities which the unidirectional CRF fails to capture. Moreover, the variation of BI-CRF is experimented by either taking the maximum or average of the forward and backward CRFs. However, there is no performance improvement observed compared to the summation mechanism explained in Section 2.3.

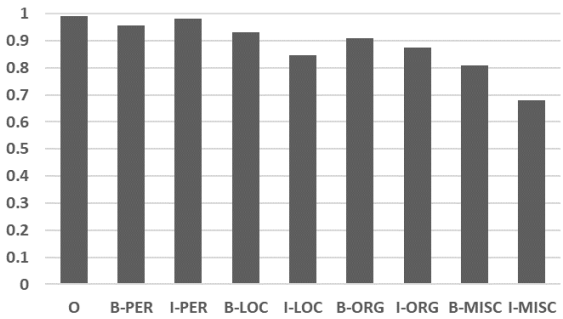

Figure 4: Average F1 scores of individual labels

Label	CRF_F	CRF_B	NOC
O	0.99443	0.0.99432	38356
B-PER	0.95481	0.95398	1617
I-PER	0.97923	**0.97945**	1156
B-LOC	0.931265	0.930925	1668
I-LOC	0.85455	**0.85534**	257
B-ORG	0.900925	0.900615	1661
I-ORG	0.86239	**0.862645**	835
B-MISC	0.81522	**0.81625**	702
I-MISC	0.68087	**0.68231**	216

Table 6: Average F1-Score of forward and backward CRF. NOC indicates the number of observations of a label type in the test dataset

Finally, we evaluated the F1 score of different labels individually to identify which type of entity is more difficult for the model to identify. Figure 4 shows the results of this evaluation and it can be observed from the results that name of *Miscellaneous* entities and words inside the name of *Location* entities are difficult for the model to identify. Possible reasons for this observation could be, lack of training observation for *Miscellaneous* entity (refer Table 2) and the absence of common patterns in *Miscellaneous* entities and words inside the name of *Location* entities. Therefore we can conclude that identifying *Miscellaneous* entities is much more challenging compared to other type of named entities.

Token	Label	CRF_F	CRF_B
...	...	...	...
a	O	O	O
brozen	O	O	O
medal	O	O	O
in	O	O	O
downhill	O	O	O
at	O	O	O
the	O	O	O
1991	O	B-MISC	O
World	B-MISC	I-MISC	B-MISC
Champion ship	I-MISC	I-MISC	I-MISC
.	O	O	O

Table 7: Example sentence from the test dataset, where backward CRF correctly identifies the beginning of a miscellaneous entity

Token	Label	CRF_F	CRF_B
ALPINE	O	B-MISC	B-MISC
SKIING	O	I-MISC	I-MISC
-WOMEN S	O	O	O
WORLD	B-MISC	B-MISC	B-MISC
CUP	I-MISC	I-MISC	I-MISC
DOWNHILL	O	I-MISC	O
RESULTS	O	O	O
.	O	O	O

Table 8: Example sentence from the test dataset, where backward CRF correctly identifies the end of a miscellaneous entity

5 Performance Analysis of BI-CRF

In order to highlight the contribution of BI-CRF, we evaluated the performance of the model on the presence of forward and back CRF individually in the final layer and an analysis of patterns of words that backward CRF is excellent in correctly classifying the label sequence compared to a forward CRF. Table 6 shows the result of individual performance evaluation of forward and backward CRF and surprisingly, the performance of backward CRF was slightly higher when classifying labels which are inside named entities and *Miscellaneous* entities. It is worth to note that the occurrence of *Miscellaneous* entities and words appearing inside a named entity is lower compared to the beginning of a named entity in the all three set of dataset (training, development test) which makes the learning more difficult. Therefore this could be a possible reason for the overall improvement in the model when unidirectional CRF is replaced with BI-CRF in the final layer. Table 7 and 8 show examples from the test dataset where the backward CRF correctly identifies the beginning and end of *Miscellaneous* entities.

6 Conclusion

In this paper, we propose a neural architecture for NER based on BI-LSTM and BI-CRF. It is truly an end-to-end model not relying on any other additional labeled data. This model is the first work to experiment BI-CRF in neural architecture in sequence modeling task. The results show that our model is competitive and outperforms most of the existing approaches which do not use any external labeled data. Moreover, the evaluations conclude that backward CRF is more capable of identifying complex labels such as words appearing inside a named entity and name of *Miscellaneous* entities with a small amount of training dataset thus improves the overall performance of the model compared to a unidirectional CRF architecture.

There are several potential directions for future work. First, the performance of the model can be further enhanced by using converting the dataset from IOB to IOBES tagging scheme. Moreover, it can be explored in multi-task learning approaches to combine more useful and correlated information among different NLP tasks.

References

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based dependency parsing by modeling characters instead of words with lstms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 349–359.

Rich Caruana, Steve Lawrence, and Lee Giles. 2000. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, pages 381–387.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. In *Proceed-*

ings of the 54th Annual Meeting of the Association for Computational Linguistics, volume 4, pages 357–370.

Franck Dernoncourt, Ji Young Lee, and Peter Szolovits. 2017. Neuroner: an easy-to-use program for named-entity recognition based on neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 97–102.

Abbas Ghaddar and Phillippe Langlais. 2018. Robust lexical features for improved neural network named-entity recognition. In *Proceedings of the 27th International Conference on Computational Linguistics*.

Alex Graves and Jurgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm networks. In *Proceedings of the International Joint Conference on Neural Networks*.

Alan Graves, Abdel rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649.

Robert Hecht-Nielsen. 1989. Theory of the back-propagation neural network. In *Proceedings of the International Joint Conference on Neural Networks*, pages 593–605.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Chun-Nan Hsu, Yu-Ming Chang, Cheng-Ju Kuo, Yu-Shi Lin, Han-Shen Huang, and I-Fang Chung2. 2008. *Integrating high dimensional bi-directional parsing models for gene mention tagging*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. In *Proceedings of the 21st International Conference on Asian Language Processing*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 2741–2749.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of 3rd International Conference for Learning Representations*.

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 18th International Conference on Machine Learning*, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270.

Wang Ling, Tiago Lus, Lus Marujo, Ramon Fernandez, Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Liyuan Liu, Jingbo Shang, Frank Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2018a. Empower sequence labeling with task-aware neural language model. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

Shifeng Liu, Yifang SunWei, and WangXiaoling Zhou. 2018b. A crf-based stacking model with meta-features for named entity recognition. In *Proceedings of 2nd Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 54–66.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint named entity recognition and disambiguation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 879–888.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1064–1074.

Andrew McCallum, Dayne Freitag, and Fernando Pereira. 2000. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the 17th International Conference on Machine Learning*, pages 591–598.

Sudha Morwal, Nusrat Jahan, and Deepti Chopra. 2012. Named entity recognition using hidden markov model. *International Journal on Natural Language Computing (IJNLC)*.

Gurusamy Murugesan, Sabenabanu Abdulkadhar, Balu Bhasuran, and Jeyakumar Natarajan. 2017. Bcc-ner: bidirectional, contextual clues named entity tagger for gene/protein mention recognition. *EURASIP Journal on Bioinformatics and Systems Biology*.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning*, pages 1310–1318.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *Proceedings of the*

18th Conference on Computational Natural Language Learning, pages 78–86.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1756–1765.

Yanjun Qi, Ronan Collobert, Pavel Kuksa, Koray Kavukcuoglu, and Jason Weston. 2009. Combining labeled and unlabeled data with word-class distribution learning. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1737–1740. ACM.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the 13th Conference on Computational Natural Language Learning9*, pages 147–155.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 2121–2130.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. 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*, pages 142–147.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, pages 1929–1958.

Quan Tran, Andrew MacKinlay, and Antonio Jimeno Yepes. 2017. Named entity recognition with stack residual lstm and trainable bias decoding. In *Proceedings of the 8th International Joint Conference on Natural Language Processing*.

Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. In *Proceedings of the 2017 Conference on International Conference on Learning Representations*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657.