LSTM-Based NeuroCRFs for Named Entity Recognition

Marc-Antoine Rondeau\textsuperscript{1,2}, Yi Su\textsuperscript{2}

\textsuperscript{1}McGill University, Montréal, Canada
\textsuperscript{2}Nuance Communications, Montréal, Canada

marcantoine.rondeau@gmail.com, yi.su@nuance.com

Abstract

Although NeuroCRF, an augmented Conditional Random Fields (CRF) model whose feature function is parameterized as a Feed-Forward Neural Network (FF NN) on word embeddings, has soundly outperformed traditional linear-chain CRF on many sequence labeling tasks, it is held back by the fact that FF NNs have a fixed input length and therefore cannot take advantage of the full input sentence. We propose to address this issue by replacing the FF NN with a Long Short-Term Memory (LSTM) NN, which can summarize an input of arbitrary length into a fixed dimension representation. The resulting model obtains $F_1=89.28$ on WikiNER dataset, a significant improvement over the NeuroCRF baseline’s $F_1=87.58$, which is already a highly competitive result.

Index Terms: neural networks, spoken language understanding, conditional random fields, LSTM

1. Introduction

Neural networks are a powerful tools for natural language processing. They can be used to perform features analysis for conditional random fields, removing the need for extensive feature engineering, and improving performance on a variety of tasks. In particular, recurrent networks are well suited for language modelling. They can be used to model long term dependencies between words while avoiding the difficulties of n-gram language models. This support for long term dependencies between variables is not without cost. Variables are always considered to be dependent, even if they are not. Long short-term memory (LSTM) units were developed to address this issue.

LSTM layers update an internal memory based on the current input, the previous value of the internal memory and the previous output. This update include the possibility of discarding the internal memory as needed. In effect, LSTM layers dynamically control the dependencies between variables. When operating as a feature analysis component, LSTM layers have a variable window size. When operating as a continuous state machines, they have a variable Markov order.

Those properties are useful for the feature analysis of a NeuroCRF applied to information extraction. In this case, the NeuroCRF is used to segment an input sequence and classify the segments. Obviously, the model should see the entire segment when classifying it. Without some form of recurrence, this is only possible by choosing a very large input window size. The LSTM layers’ capacity to forget as needed allows the model to change the window size dynamically during feature analysis. This enables the model to support both long and short segments.

In this paper, we compare two new NeuroCRF models to full rank feed-forward neural network (FF NN) based NeuroCRF. The first new model is a RNN-based full rank NeuroCRF and use a conventional recurrent NN. The second new model is a LSTM-based full rank NeuroCRF. Models were trained and tested on a large corpus extracted from Wikipedia and automatically labelled. This large amount of data allows us to train models with more parameters. It also reduces the performance variation caused by the model initializations, facilitating the comparison of two configurations.

Section 2 presents some existing work in this area. Section 3 summarizes NeuroCRFs. Section 4 describes RNN-based NeuroCRFs. Section 5 presents the new LSTM-based NeuroCRFs. Finally, Section 6 shows the results of an experimental study comparing those models.

2. Background

The idea of using LSTM units for NER is not new and can be traced back to [1]. This initial attempt was quite different from more recent approaches, especially in the output layer.

R-CRF [2], similar to RNN-based low-rank NeuroCRFs, achieved good performance on ATIS [3]. LSTM units without CRF layer [4] also achieved good performance on ATIS. Finally, [5] presented a study of the impact of various recurrent network architectures. Those experiments showed improvements when using a bidirectional recurrent layer. Our work is similar to [6], which was uploaded to arXiv during the writing of this paper. Like [6], we build on those existing approaches by combining a CRF layer with a bidirectional LSTM layer. Unlike [6] and [2], we used a full rank CRF layer [7]. We also used different input features, and word representations, and analyzed in more details the impact of context when a bidirectional LSTM layer is used.

3. NeuroCRF

Conditional random fields (CRFs) are a class of graphical models defining a conditional distribution $p(y|x)$ of an output sequence $y$ given an input sequence $x$ [8]. The graph factorizes the distribution into sets of simpler factor functions. Linear chain CRFs, in particular, factorize a distribution into time. The factor functions, in this case, are of the form $\Psi(y_t, y_{t-1}, x_t)$, where $x_t$ is the relevant part of the input $x$ around time $t$.

$$\log p(y|x) = \left( \sum_{t=1}^{T} \Psi(y_t, y_{t-1}, x_t) \right) - \log Z(x) \quad (1)$$

$$Z(x) = \sum_{y} \exp \left( \sum_{t=1}^{T} \Psi(y_t, y_{t-1}, x_t) \right) \quad (2)$$

The factor function is a weighted sum of features extracted from $x$, $y_t$ and $y_{t-1}$, where the weights are the model’s parameters. The set of features is created manually, by an human expert, a process known as “feature engineering”.

\footnotesize{Copyright © 2016 ISCA http://dx.doi.org/10.21437/Interspeech.2016-288

http://dx.doi.org/10.21437/Interspeech.2016-288}
NeuroCRFs use a NN to learn relevant feature automatically, ideally removing the need for feature engineering. The experimental results presented in Section 6 were obtained using full rank NeuroCRFs [7], where the factor functions are:

\[ \Psi_t(y_t, y_{t-1}, x_t) = G_t(x_t)F(y_{t-1}, y_t) \] (3)

\[ F(y_{t-1}, y_t) = \begin{bmatrix} f_0(y_{t-1}, y_t) \\ \vdots \\ f_{N^2-1}(y_{t-1}, y_t) \end{bmatrix} \] (4)

\[ f_i(y_{t-1}, y_t) = \begin{cases} 1, & i = N_{y_{t-1}} + y_t \\ 0, & i \neq N_{y_{t-1}} + y_t \end{cases} \] (5)

where \( x \) and \( y \) are a pair of input and output sequences of length \( T \), \( G_t(x_t) \) is the neural network outputs vector obtained from a subset of \( x \) centred at time \( t \), and \( F(y_{t-1}, y_t) \) is an indicator matrix, which pick outputs from \( G_t(x_t) \). \( N \) is the number of possible labels in \( y \).

Equation 3 factorize the function factor into two components, \( G_t \) and \( F_t \). The NN output at time \( t \) is \( G_t(x_t) \). With \( N \) labels, \( G_t \) is vector with \( N^2 \) elements, one per transitions. Conceptually \( G_t \) is a \( N \)-by-\( N \) transition matrix, reshaped into a vector. \( F(y_{t-1}, y_t) \) is used to select the NN output corresponding to a transition from \( y_{t-1} \) to \( y_t \). It is a one-hot vector with \( N^2 \) elements; \( F \) could be replaced by indexing in \( G_t \). The NN’s input, \( x_t \), is a sliding window, centred around the \( t \)’th word.

3.1. Rank

A NeuroCRF’s rank [7] is the rank of its reshaped NN output. The NN output of full rank NeuroCRFs, presented above, is a \( N \)-by-\( N \) transition matrix whose rank is \( N \). The NN output of low rank NeuroCRFs is one. Full rank NeuroCRFs use their NN to model transitions. Its output, after training, should be high when the input contains evidence of a particular transition. Low rank NeuroCRFs use their NN to model label emissions. Other configurations are possible [9].

4. RNN-Based NeuroCRF

RNN-based NeuroCRFs are obtained by replacing the feed-forward hidden layer of a NeuroCRF with a recurrent layer, so that \( h_t \), the hidden layer vector at time \( t \), is

\[ h_t = a(W^{(c)}x_t + W^{(b)}h_{t-1} + b), \] (6)

where \( a(\cdot) \) is an activation function, \( x_t \) is the layer’s input vector, \( b \) is the hidden layer biases vector, \( W^{(c)} \) is the weight matrix associated with the input vector and \( W^{(b)} \) is the weight matrix associated with the previous output of the hidden layer. Excluding the effect of \( f(\cdot) \), the value of \( h_t \) can only decay at a fixed rate, since \( W^{(b)} \) is constant. It cannot be reset or retained as needed.

The equivalent of Equation 3 for recurrent NNs is

\[ \Psi_t(y_t, y_{t-1}, x_t) = G_t(x_t)F(y_{t-1}, y_t) . \] (7)

In this case, while the sliding window \( x_t \) is used in Equation 6, \( G_t \) is \( h_tW^{(c)} \) is also a function of previous values \( x_{t-d} \) of the input sequence \( x \).

5. LSTM-Based NeuroCRF

We propose to use a LSTM layer [10, 11] to address FF NN based NeuroCRFs’ poor support of long term dependencies between the input and output sequences. LSTM units’ ability to forget and remember based on the input is required by the named entities variable length. LSTM units’ are based on a memory cell \( c_t \), updated by interpolating the previous value and a candidate value. This interpolation is controlled by a forget gate \( f_t \), weighting the previous value, and an input gate \( i_t \), weighting the candidate value. The hidden layer vector at time \( t \), \( h_t \), is the element-wise product of the memory cell \( c_t \) and an output gate \( o_t \). The gates and candidate values are recursive functions of the layer’s input \( x_t \) and the layer previous output \( h_{t-1} \), similar to Equation 6.

It is possible to create an anti-causal LSTM layer by reversing the direction of the recursion. It is also possible to create a bidirectional layer by concatenating causal and anti-causal units. Those two sets of units are not connected to each other.

6. Experimental study

6.1. Datasets

We used the CoNLL-2003 corpus [12] to facilitate comparison with other works. Our previous experiments suffered from two issues with this corpus. First, the training corpus is too small, which increased overfitting as we increased the number of parameters in our models. Second, the test corpus was also too small, preventing us from reaching conclusions about improvements. To address those issues, we retrieved the WikiNER corpus (wp3) from [13]. WikiNER includes the same labels as ConLL-2003, although the effective annotation directives are different. The same encoding, IOB2, is used for the CoNLL-2003 and WikiNER experiments. While we include results on CoNLL-2003 to facilitate comparisons, most of our analysis is based on the experiments using WikiNER.

Named entities are separated into 4 classes: location (LOC), person (PER), organization (ORG) and miscellaneous (MISC). Table 1 shows the size of the WikiNER training, validation and test corpus.

### Table 1: Size of WikiNER training, validation and test corpus

<table>
<thead>
<tr>
<th></th>
<th>Train</th>
<th>Val.</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Sentence</td>
<td>113,812</td>
<td>14,178</td>
<td>14,163</td>
</tr>
<tr>
<td>#Words</td>
<td>2,798,532</td>
<td>351,322</td>
<td>349,752</td>
</tr>
<tr>
<td>Entities:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#LOC</td>
<td>68,737</td>
<td>8,718</td>
<td>8,580</td>
</tr>
<tr>
<td>#MISC</td>
<td>58,826</td>
<td>7,322</td>
<td>7,462</td>
</tr>
<tr>
<td>#ORG</td>
<td>39,795</td>
<td>4,912</td>
<td>4,891</td>
</tr>
<tr>
<td>#PER</td>
<td>77,010</td>
<td>9,594</td>
<td>9,613</td>
</tr>
<tr>
<td>All</td>
<td>244,368</td>
<td>30,546</td>
<td>30,546</td>
</tr>
</tbody>
</table>

6.2. Word Representation

We pre-trained a continuous word representation, with 100 dimensions, on Wikipedia\(^1\), using word2vec [14]. Words were used without preprocessing or normalization. This word representation was used for all the following experiments. The full set of word representations is pruned when initializing a model, to remove words that do not occur in the training, test or validation sets.

\(^1\)We used a dump collected in the spring of 2014.
idation corpus. This is simply done to improve training speed and reduce the memory required. The word representation is fine-tuned when training models.

The initial word representation is obtained from a continuous bag of word (CBow) [14]. Given a window of $T = 2C + 1$ words, a CBow model is trained to predict the central word given the surrounding words. The model is a simple NN, with linear activations. The hidden layer is the sum of the word representations of the context part of the window. Once the CBow has been trained on a large corpus, those word representations are extracted from the model and used to initialize the word representation used by our NeuroCRFs.

**6.3. Configuration**

The input consists of a sliding window, using the continuous representation combined to a randomly initialized 5 dimensions capitalization embedding and a 5 dimensions part-of-speech (POS) embedding. The capitalization feature indicates that a word contains an initial upper case letter, some upper case letters, only upper case letters or only lower case letters. The POS features used were included in the datasets. Capitalization and POS features are also used by the CRF baseline system. The word representations are pre-trained from a CBow model, as described in the previous subsection.

All models consist of this input layer feeding into a hidden layer using a hardtanh activation function. The hidden layer is then connected to a full rank CRF layer, as described in Section 3.

Model were trained using stochastic gradient descent, using a non-monotone learning rate strategy [15]. Dropout [16, 17] was used to regularize the models. We also applied L2-norm regularization to the hidden and output layers’ weights. This was found especially useful for the LSTM layer. We used a random search [18] to find the hyper-parameters, including the hidden layer size and the size of the sliding window. The random search found 200 hidden units for all WikiNER experiments. For the CoNLL experiments, it found 400 and 500 units for the FF and recurrent NeuroCRFs, respectively. The training and testing scripts were implemented using the Theano toolkit [19].

**6.4. Performance measures**

NER can be conceptually decomposed into two steps. First, segments corresponding to a named entity of some kind are extracted from the input sentence. Then, those segments are classified into a specific kind of named entity. In practise, those two steps are not separated and are performed jointly. The same labels used to segment the sentence are used to classify the segments. This conceptual split is useful when analyzing the differences between two systems.

The main performance measure used is

$$ F_1 = 2 \frac{c}{d + n} \frac{pr}{p + r} = 2 \frac{c}{c_o} \frac{c_o}{d + n} = A_c F_1^{(s)} \quad (8) $$

$$ A_c = \frac{c}{c_o} \quad (9) $$

$$ F_1^{(s)} = 2 \frac{c}{d + n} \quad (10) $$

where $c$ is the number of correct named entities retrieved, $c_o$ is the same ignoring class, $d$ is the number of entities retrieved, $n$ is the number of entities in the reference, $p = c/d$ is the precision, $r = c/n$ is the recall, $F_1^{(s)}$ is the segmental $F_1$, and $A_c$ is the classification accuracy. $F_1^{(s)}$ is the component of $F_1$ attributed to the segmentation step described above while $A_c$ is the component attributed to the classification step.

**6.5. Results**

The results presented in this section are the average of 10 runs, each using the same hyper-parameters and a different random initialization. This procedure is required to compensate for the non-convexity of NNs. When comparing two systems, statistical significance is based on a two-sided T-test. We used the Stanford NER system [20] as a benchmark. The features used by this system are a superset of the features available to the NeuroCRFs. In particular, capitalization and POS tags are used. Table 2 shows that LSTM layers improved performance compared to the FFNN baseline on the CoNLL-2003 task. In both cases, the results were found to be statistically significant ($p \leq 1\%$) using a two tailed T-Test. Because of the high variance, the reduction in $F_1$ observed when comparing the causal LSTM layer to the bidirectional layer was not statistically significant ($p \geq 60\%$). The improved performance of NeuroCRF (RNN), compared to NeuroCRF (FF), is not statistically significant ($p \geq 10\%$). All NeuroCRFs significantly outperformed the Stanford CRF benchmark. Figure 1a shows the corresponding box and whiskers plots. This figure clearly shows the large variance caused by the random initialization.

The high variance of test results using CoNLL-2003 prevents us from reaching solid conclusions. Table 3 shows the results using the much larger WikiNER task. Those results have a significantly lower variance. The LSTM layers improved performance compared to the FFNN baseline. This improvement is statistically significant ($p \leq 0.1\%$). Performance is also improved for the bidirectional LSTM layer, when compared to the causal LSTM layer. This improvement is statistically significant ($p \leq 0.1\%$). The improved performance of NeuroCRF (RNN), compared to NeuroCRF (FF), is not statistically significant ($p \geq 11\%$). Figure 1b shows the corresponding box and whiskers plots. This figure clearly shows the significant improvement attributed to the segmentation step described above while $A_c$ is the component attributed to the classification step.

**Table 2:** Experimental results for CoNLL-2003. **Bold** font indicates statistically significant improvements over NeuroCRF (FF).

<table>
<thead>
<tr>
<th>Model</th>
<th>$F_1$</th>
<th>Std. Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRF (Stanford)</td>
<td>87.94</td>
<td>NA</td>
</tr>
<tr>
<td>NeuroCRF (FF)</td>
<td>88.75</td>
<td>0.2305</td>
</tr>
<tr>
<td>NeuroCRF (RNN)</td>
<td>88.90</td>
<td>0.1605</td>
</tr>
<tr>
<td>NeuroCRF (LSTM)</td>
<td><strong>89.30</strong></td>
<td>0.2432</td>
</tr>
<tr>
<td>NeuroCRF (BLSTM)</td>
<td><strong>89.23</strong></td>
<td><strong>0.3703</strong></td>
</tr>
</tbody>
</table>

**Table 3:** Experimental results for WikiNER. **Bold** font indicates statistically significant improvements over NeuroCRF (FF).

<table>
<thead>
<tr>
<th>Model</th>
<th>$F_1$</th>
<th>Std. Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRF (Stanford)</td>
<td>86.09</td>
<td>NA</td>
</tr>
<tr>
<td>NeuroCRF (FF)</td>
<td>87.58</td>
<td>0.0739</td>
</tr>
<tr>
<td>NeuroCRF (RNN)</td>
<td>87.66</td>
<td>0.1386</td>
</tr>
<tr>
<td>NeuroCRF (LSTM)</td>
<td><strong>89.11</strong></td>
<td>0.0795</td>
</tr>
<tr>
<td>NeuroCRF (BLSTM)</td>
<td><strong>89.28</strong></td>
<td><strong>0.1070</strong></td>
</tr>
</tbody>
</table>


3Complex CRF models, requiring significant feature engineering, have reached $F_1 = 90.90$ on this task [21, 22].
668

Figure 1: Box and whiskers plots of experimental results. The boxes indicate the two central quartiles, separated by the median, while the whiskers indicate the full range of the results.

<table>
<thead>
<tr>
<th>Model</th>
<th>$F_1$ (%)</th>
<th>$A_c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>NeuroCRF (FF)</td>
<td>93.75</td>
<td>94.66%</td>
</tr>
<tr>
<td>NeuroCRF (LSTM)</td>
<td>94.17</td>
<td>94.83%</td>
</tr>
<tr>
<td>NeuroCRF (BLSTM)</td>
<td>94.10</td>
<td>94.83%</td>
</tr>
</tbody>
</table>

Table 4: Segmentation ($F_1$) and classification ($A_c$) performance for CoNLL-2003

<table>
<thead>
<tr>
<th>Model</th>
<th>$F_1$ (%)</th>
<th>$A_c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>NeuroCRF (FF)</td>
<td>93.56</td>
<td>93.61%</td>
</tr>
<tr>
<td>NeuroCRF (RNN)</td>
<td>93.50</td>
<td>93.76%</td>
</tr>
<tr>
<td>NeuroCRF (LSTM)</td>
<td>94.23</td>
<td>94.57%</td>
</tr>
<tr>
<td>NeuroCRF (BLSTM)</td>
<td>94.29</td>
<td>94.69%</td>
</tr>
</tbody>
</table>

Table 5: Segmentation ($F_1$) and classification ($A_c$) performance for WikiNER

<table>
<thead>
<tr>
<th>Model</th>
<th>$F_1$ (%)</th>
<th>$A_c$</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>NeuroCRF (FF)</td>
<td>93.56</td>
<td>93.61%</td>
<td>87.58</td>
</tr>
<tr>
<td>Without Context</td>
<td>91.98</td>
<td>91.99%</td>
<td>84.61</td>
</tr>
<tr>
<td>NeuroCRF (BLSTM)</td>
<td>94.29</td>
<td>94.69%</td>
<td>89.28</td>
</tr>
<tr>
<td>Without Context</td>
<td>94.23</td>
<td>94.70%</td>
<td>89.23</td>
</tr>
</tbody>
</table>

Table 6: Impact of context component of input for WikiNER

Having access to the entire named entity is required in order to classify it correctly. Those results confirm that NeuroCRFs using a FFNN cannot learn essential features, as those features require the context.

Table 6 also shows that a bidirectional LSTM layer can reconstruct this context. While the segmentation performance and therefore the $F_1$ were lower, the degradation is not statistically significant ($p \geq 37\%$). The classification performance was not significantly affected. Those results show that NeuroCRFs using a BLSTM NN can learn to reproduce the essential parts of the context, by using their internal memory as a substitute.

7. Conclusion

We addressed the issue of long term dependencies that limits performance of NeuroCRF models. We found that replacing the FF NN used for feature analysis by a recurrent NN improved performance on two NER tasks. While the improvement was limited with conventional recurrent layers, we obtained significant improvements when using LSTM layers. Further improvements were obtained using a bidirectional LSTM layer, where the layer is divided into causal and anti-causal sections.

The success of those bidirectional LSTM layers motivated a second set of experiments, where the preceding and following words were removed from the input. As expected, removing this context severely degraded the performance of the baseline FF based NeuroCRF. While performance were affected, the degradation when using a bidirectional LSTM based NeuroCRF was minimal, and was not statistically significant. Those results show that the computational complexity of a LSTM based system can be reduced by removing the context, without significantly affecting performance.
8. References


