INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

LSTM-Based NeuroCRFs for Named Entity Recognition

Marc-Antoine Rondeau[1,2], *Yi Su*[2]

[1]McGill University, Montréal, Canada
[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-grams 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 $p(\mathbf{y}|\mathbf{x})$ of an output sequence $\mathbf{y}$ given an input sequence $\mathbf{x}$ [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 $\Psi(y_t, y_{t-1}, \mathbf{x}_t)$, where $\mathbf{x}_t$ is the relevant part of the input $\mathbf{x}$ around time t.

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

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

The factor function is a weighted sum of features extracted from $\mathbf{x}$, 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".

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:

$$\Psi(y_t, y_{t-1}, \mathbf{x}_t) = G_t(\mathbf{x}_t)F(y_{t-1}, y_t) \quad (3)$$

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

$$f_i(y_{t-1}, y_t) = \begin{cases} 1, & i = Ny_{t-1} + y_t \\ 0, & i \neq Ny_{t-1} + y_t \end{cases} . \quad (5)$$

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

Equation 3 factorize the factor function into two components, G_t and F. The NN output at time t is $G_t(\mathbf{x}_t)$. 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. $F(y_t, y_{t-1})$ 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

$$h_t = a(W^{(x)}\mathbf{x}_t + W^{(h)}h_{t-1} + b), \quad (6)$$

where $a(\cdot)$ is an activation function, x_t is the layer's input vector, b is the hidden layer biases vector, $W^{(x)}$ is the weight matrix associated with the input vector and $W^{(h)}$ 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 $W^{(h)}$ is constant. It cannot be reset or retained as needed.

The equivalent of Equation 3 for recurrent NNs is

$$\Psi_t(y_t, y_{t-1}, \mathbf{x}) = G_t(\mathbf{x})F(y_{t-1}, y_t). \quad (7)$$

In this case, while the sliding window $\mathbf{x}_t$ is used in Equation 6, $G_t = h_t W^{(o)}$ is also a function of previous values $\mathbf{x}_{t-d}$ of the input sequence $\mathbf{x}$.

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 others.

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

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

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, in sentence, word and entities. The validation and test corpora were created by randomly selecting sentences. In both case, their size was set so that they both contains at least 10% of all the named entities, ignoring classes. The training corpus is the remainder of the available data and contains almost 3 millions words.

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 val-

[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 L_2-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

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

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

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

where c is the number of correct named entities retrieved, c_s 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.

Model	F_1	Std. Dev
CRF (Stanford) [20]	87.94	NA
NeuroCRF (FF)	88.75	0.2305
NeuroCRF (RNN)	88.90	0.1605
NeuroCRF (LSTM)	**89.30**	0.2432
NeuroCRF (BLSTM)	**89.23**	0.3703

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

Model	F_1	Std. Dev
CRF (Stanford)	86.09	NA
NeuroCRF (FF)	87.58	0.0739
NeuroCRF (RNN)	87.66	0.1386
NeuroCRF (LSTM)	**89.11**	0.0795
NeuroCRF (BLSTM)	**89.28**	0.1070

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

6.5. Results

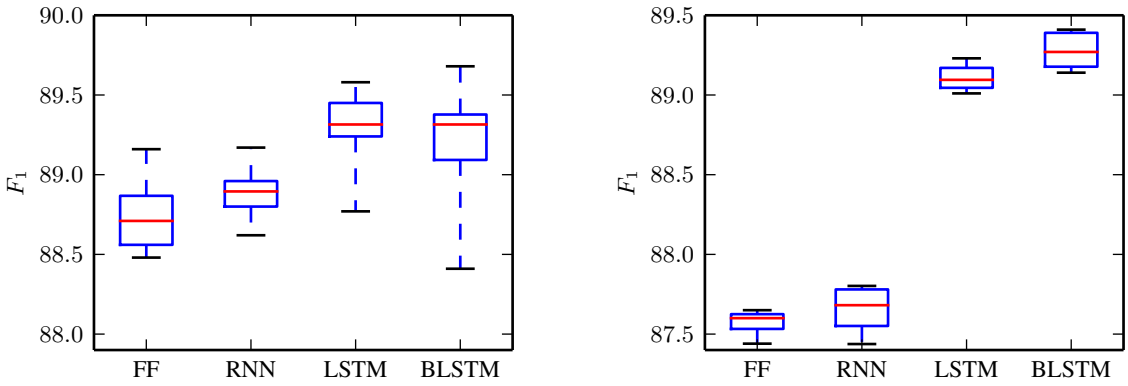
The results presented in this section are the average of 10 runs, each using the same hyper-parameters and a different random initialization. This procedure is required to compensate for the non-convexity of NNs. When comparing two systems, statistical significance is based on a two-sided T-test. We used the Stanford NER system [20] as a benchmark.[2]. 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[3] 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 shows clearly the large variance caused by the random initialization.

The high variance of test results using CoNLL-2003 prevent 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 im-

[2]CoNLL-2003 results from `http://nlp.stanford.edu/projects/project-ner.shtml`

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



(a) CoNLL-2003 (b) WikiNER

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.

Model	$F_1^{(s)}$	A_c
NeuroCRF (FF)	93.75	94.66%
NeuroCRF (LSTM)	94.17	94.83%
NeuroCRF (BLSTM)	94.10	94.83%

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

Model	$F_1^{(s)}$	A_c
NeuroCRF (FF)	93.56	93.61%
NeuroCRF (RNN)	93.50	93.76%
NeuroCRF (LSTM)	94.23	94.57%
NeuroCRF (BLSTM)	94.29	94.69%

Table 5: Segmentation ($F_1^{(s)}$) and classification (A_c) performance for WikiNER

provement observed with LSTM layers.

Table 4 shows the experimental results in term of segmental F_1 and classification accuracy for CoNLL-2003. The LSTM layers improved both compared to the FFNN baseline. This is confirmed by the equivalent results for WikiNER, shown in Table 5. Those results show that having access to long term memory can facilitate the classification of a named entity. They also show that the same long term memory can help the segmentation of named entities.

6.6. Impact of Context

The results presented in the previous subsection included the context size in the hyper-parameters. We investigated the importance of this parameter with a set of experiment where it is fixed to zero. The input window for those experiments contains only features extracted from the current word. In practice, computational complexity for a LSTM layer is dominated by the input's dimensionality. Reducing the context size would therefore reduce the computational requirement of the LSTM based system. Models were trained and tested on WikiNER.

Table 6 shows that the performance of a full rank NeuroCRF using a FFNN is significantly degraded with the context is removed. The segmentation performance is reduced. More interestingly, so is the classification accuracy, confirming that 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.

Model	$F_1^{(s)}$	A_c	F_1
NeuroCRF (FF)	93.56	93.61%	87.58
Without Context	91.98	91.99%	84.61
NeuroCRF (BLSTM)	94.29	94.69%	89.28
Without Context	94.23	94.70%	89.23

Table 6: Impact of context component of input for WikiNER

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

[1] J. Hammerton, "Named entity recognition with long short-term memory," in *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics, 2003, pp. 172–175.

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

[3] P. J. Price, "Evaluation of spoken language systems: The atis domain," in *Proceedings of the Workshop on Speech and Natural Language*, ser. HLT '90. Stroudsburg, PA, USA: Association for Computational Linguistics, 1990, pp. 91–95. [Online]. Available: http://dx.doi.org/10.3115/116580.116612

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

[5] G. Mesnil, X. He, L. Deng, and Y. Bengio, "Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding." in *INTERSPEECH*, 2013, pp. 3771–3775.

[6] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF Models for Sequence Tagging," *ArXiv e-prints*, Aug. 2015.

[7] M.-A. Rondeau and Y. Su, "Full-rank linear-chain neurocrf for sequence labeling," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, April 2015.

[8] C. Sutton and A. McCallum, *An introduction to conditional random fields for relational learning*. Introduction to statistical relational learning. MIT Press, 2006.

[9] M. A. Rondeau and Y. Su, "Recent improvements to neurocrfs for named entity recognition," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*, Dec 2015, pp. 390–396.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[12] E. F. Tjong Kim Sang and F. De Meulder, "Introduction to the conll-2003 shared task: Language-independent named entity recognition," in *Proceedings of CoNLL-2003*, W. Daelemans and M. Osborne, Eds. Edmonton, Canada, 2003, pp. 142–147.

[13] J. Nothman, N. Ringland, W. Radford, T. Murphy, and J. R. Curran, "Learning multilingual named entity recognition from Wikipedia," *Artificial Intelligence*, vol. 194, pp. 151–175, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.artint.2012.03.006

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *Proceedings of Workshop at ICLR*, 2013.

[15] N. S. Keskar and G. Saon, "A nonmonotone learning rate strategy for sgd training of deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, April 2015.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[17] 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.

[18] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=2188385.2188395

[19] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio, "Theano: new features and speed improvements," Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[20] J. R. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by gibbs sampling," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2005, pp. 363–370.

[21] A. Passos, V. Kumar, and A. McCallum, "Lexicon infused phrase embeddings for named entity resolution," *CoNLL-2014*, p. 78, 2014.

[22] D. Lin and X. Wu, "Phrase clustering for discriminative learning," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*. Association for Computational Linguistics, 2009, pp. 1030–1038.

[23] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: a CPU and GPU math expression compiler," in *Proceedings of the Python for Scientific Computing Conference (SciPy)*, Jun. 2010, oral Presentation.