FULL-RANK LINEAR-CHAIN NEUROCRF FOR SEQUENCE LABELING

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

[1]McGill University
[2]Nuance Communications
marc-antoine.rondeaubeauchamp@mail.mcgill.ca,yi.su@nuance.com

ABSTRACT

Inspired by the success of deep neural network-hidden Markov model (DNN-HMM) in acoustic modeling for automatic speech recognition, a number of researchers from various fields have independently proposed the idea of combining DNN and conditional random fields (CRFs). Despite their subtle differences, this class of models is collectively referred to as "NeuroCRF" in this paper. We focus our attention on applying a linear-chain NeuroCRF to the fundamental and ubiquitous problem of sequence labeling in natural language processing with distributed word representations. We question the necessity of previous works' use of the neural network to learn a low-rank emission feature matrix, added to a transition feature matrix. By modeling a full-rank feature matrix directly, we show that statistically significant gains can be achieved on the CoNLL-2000 syntactic chunking task, without harming performance on tasks with low dependencies between consecutive labels, such as the CoNLL-2003 named entity recognition task.

Index Terms— Neural networks, spoken language understanding, conditional random fields

1. INTRODUCTION

Deep neural networks (NNs) and hidden Markov models (HMMs) have been successfully combined to improve automatic speech recognition (ASR) performance. Neural networks have also been successfully used in language models [1]. A number of researchers have built on those successes and proposed combination of NNs and conditional random fields (CRFs) [2, 3, 4, 5]. This vast class of models is referred as "NeuroCRFs" in this paper.

CRFs factorize a conditional distribution into simpler, usually exponential, factor functions. The factor functions rely on features extracted from an input sequence and an output sequence, with associated feature weights. CRFs are used to find the most likely output sequence, given an input sequence.

Like CRFs, NeuroCRFs have been applied to sequence labeling, a key meta-task in spoken language understanding [4, 5]. A large number of problem can be solved by assigning labels to words in a sentence and interpreting the resulting label sequence. This paper focuses on labeling using linear-chain NeuroCRFs based on distributed word representations.

The common form of this approach is to use a NN to generate emission weights, corresponding to a label being assigned to a specific word. Those are complemented by constant transition weights, corresponding to a label directly following another label. In this paper, NeuroCRFs of this form are referred as "low-rank NeuroCRFs".

The NN can be used to generate transition weights directly. This is the approach presented in this paper. We refer to the resulting models as "full-rank NeuroCRFs".

The NeuroCRFs discussed in this paper represent the words in the input sequence using a distributed word representation, also know as "word embedding" or "continuous word representation". Those representations are pre-trained on unlabeled data, such that similar words tend to have similar representations [6, 4, 7, 8].

The process used to extract information from a sentence by labeling is summarized in Section 2. Low-rank NeuroCRFs will be defined in Section 3. The approach to minimize a NeuroCRF's loss function will also be summarized in Section 3. The notion of NeuroCRF rank is explained in Section 4. Full-rank NeuroCRFs are also defined in Section 4. Finally, Section 5 will present an experimental study to evaluate the impact of its rank on a NeuroCRF's performance.

2. INFORMATION EXTRACTION BY LABELING

In this paper, we look at NeuroCRFs applied to information extraction. Those tasks consist of extracting relevant segments from a natural language sentence and classifying those segments. This is done by assigning a label to each word. Those labels correspond to the NeuroCRF's states – "label" and "state" are used interchangeably in this paper. The segments are then extracted from the resulting label sequence. Table 1 lists the types of label used.

*The author performed this work while interning at Nuance

Table 1: BIOES labeling scheme. Labels indicate the position of words in a segment, and the class of the segment.

O : Word is not part of a segment
S-*class* : Word is a single-word segment
B-*class* : First word of multi-word segment
E-*class* : Last word of multi-word segment
I-*class* : Other words of multi-word segment

3. NEUROCRF

Conditional random fields 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 $\mathbf{x}$ [9]. The distribution is factorized according to a factor graph, commonly a linear-chain. A linear-chain CRF with factors corresponding to the emission and transition probabilities of a hidden Markov model is defined as:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \exp\left(G(\mathbf{x}_t)F(y_t) + F^\top(y_{t-1})AF(y_t)\right) \tag{1}$$

$$Z(\mathbf{x}) = \sum_{\mathbf{y}' \in \text{gen}(\mathbf{x}_t)} \prod_{t=1}^{T} \exp\left(G(\mathbf{x}_t)F(y'_t) + F^\top(y'_{t-1})AF(y'_t)\right) \tag{2}$$

where $\mathbf{x}$ and $\mathbf{y}$ are a pair of input and output sequences of length T, $\text{gen}(\mathbf{x})$ is the set of possible output sequences, $G(\mathbf{x}_t)$ is a matrix of emission scores obtained from a subset of $\mathbf{x}$ centered at time t, and A is a transition matrix for a transition from output y to output y'. $F(y)$ is an indicator matrix, which selects the correct element from $G(\mathbf{x}_t)$ and A. All elements in $F(y)$ are zero, except for the element corresponding to y. With K being the number of possible states (i.e. labels), $G(\mathbf{x}_t)$ is a 1 by K matrix, $F(y)$ is a K by 1 matrix and A is a K by K matrix, with one row per source state and one column per destination state.

In a NeuroCRF, the emission matrix is the output of a neural network. The NN's input, $\mathbf{x}_t$ is a sliding window, centred on the tth words. Words are replaced by a continuous word representation [6, 4, 7, 8]. The representations are pre-trained on a large amount of unlabeled data. Complementary features, such as gazetteer matches, can be included in $\mathbf{x}_t$.

3.1. Training

The transition features and the NN are trained simultaneously, by minimizing the loss function $L(\mathbf{y}|\mathbf{x}) = -\log p(\mathbf{y}|\mathbf{x})$ using stochastic gradient descent. The gradient $\frac{\partial L(\mathbf{y}|\mathbf{x})}{\partial n_y}$ of this function with respect to the NN's outputs is back-propagated through the NN, updating its weights and biases. As shown in [9], this is done using dynamic programming, which removes the need for an explicit enumeration of all possible $\mathbf{y}$.

3.2. Regularization and Dropout

Models were regularized using norm regularization and dropout [10, 11]. When training with dropout, hidden units are randomly removed from the network. This should prevent co-adaptation of the hidden units to the training data. Without dropout, multiple hidden units could detect a group of features than tend to co-occur in the training data. Those same co-occurrences are unlikely to be present in the test data, which will prevent the detection of the features. With dropout, the model will tend to create individual feature detectors, which are less affected by the co-occurrence of their feature in the training data.

4. FULL-RANK NEUROCRF

Equation 1, which has been used in previous work [4], can be re-written as:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \exp\left(F^\top(y_{t-1})\left(RG(\mathbf{x}_t) + A\right)F(y_t)\right), \tag{3}$$

where R is a constant column of K ones, one per state. The effective NN output matrix $\hat{G}(\mathbf{x}_t) = RG(\mathbf{x_t})$ is a K by K matrix whose rank is 1. We propose to replace this low-rank NeuroCRF with a full-rank NeuroCRF by modeling $\hat{G}(\mathbf{x}_t)$ directly. The NN of a full-rank NeuroCRF has K^2 outputs, one per state transition. This removes the need for a separate transition matrix, which is redundant with the biases of the NN's output layer. A full-rank NeuroCRF is defined as:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \exp\left(F^\top(y_{t-1})G(\mathbf{x}_t)F(y_t)\right). \tag{4}$$

Its NN output matrix has a rank of K.

4.1. Motivation

A low-rank NeuroCRF's NN is used to model emission weights as a function of the input, but not of the previous state. The transition weights are fixed after training, and are not a function of the input. This structure is inherited from HMMs. HMMs assume that the visible output (e.g. sound) are generated by a hidden state (e.g. phoneme segment), and that it is independent of the previous state.

Removing this assumption enables the NN to learn more complex relations between the visible input (the NN input window) and the state sequence $\mathbf{y}$. The NN becomes free to assign probability mass not only to a specific state but to a specific state in a specific context.

Finally, it should be noted that a full-rank NeuroCRF can learn parameters equivalent to a low-rank NeuroCRF. When

Table 2: Test, validation and training size in sentences, words and segments.

	# Sentences	# Words	# Segments
Chunking (CoNLL-2000)			
Test	2,012	47,377	23,852
Validation	1,000	23,615	11,958
Training	7,936	188,112	95,020
NER (CoNLL-2003)			
Test	3,453	46,435	5,648
Validation	3,250	51,362	5,942
Training	14,041	203,621	23,499

Table 3: Training sets' details.

	Chunking	NER
# Words inside segment	163,700	34,600
Average segment length	1.77	1.45
% Segments length 1	55.98%	63.11%
Entropy (labels)	3.36	1.24
Conditional entropy	1.52	0.87
Mutual information	1.84	0.37

Table 4: Experimental results with and without dropout.

	Chunking		NER	
	Without	With	Without	With
Benchmark [4]	94.32	N/A	88.67	N/A
Low-Rank	94.45	94.33	88.53	88.63
Full-Rank	94.61	94.57	87.92	88.65

the added degrees of freedom are not necessary, a full-rank NeuroCRF should learn parameters equivalent to those of a low-rank NeuroCRF.

5. EXPERIMENTAL STUDY

5.1. Datasets

NeuroCRFs were applied to two shared tasks, CoNLL-2000 and CoNLL-2003 [12, 13] Segments were extracted, from natural language sentences, and classified. The nature of those segments differs. CoNLL-2000 is a syntactic chunking task, where the segments are defined by their syntactic role in the sentence. CoNLL-2003 is a named entity recognition task (NER), where the segments are named entities. Following [4], the data were relabeled to use the BIOES labeling scheme described in Section 2.

For CoNLL-2000, a validation set was created by selecting 1,000 random sentences from the training set. This validation set was added to the training set when training the final models. CoNLL-2000 uses 11 classes, resulting in 45 labels. In the training set, 87% of the words are part of a segment, with an average length of 1.77 words. CoNLL-2003 uses 4 classes, resulting in 17 labels. In the training set, 17% of the words are part of a segment, with an average length of 1.45 words. The validation set was *not* added to the training set when training the final models. Table 2 shows the size of the test, validation and training sets in sentences, words and segments.

5.2. Configuration

All the neural-networks used in this study have a linear output layer and a single hidden layer, with a hard hyperbolic tangent activation function, following [4]. The input to those networks is based on a pre-trained continuous word representation augmented by a capitalization representation and a part-of-speech representation. The capitalization and part-of-speech representations are initialized randomly and are not pre-trained. The pre-trained word representation is the "LM2" word representation used in [4]. The input vector of the network is a sliding window containing the central word, the previous 2 words and the following 2 words.

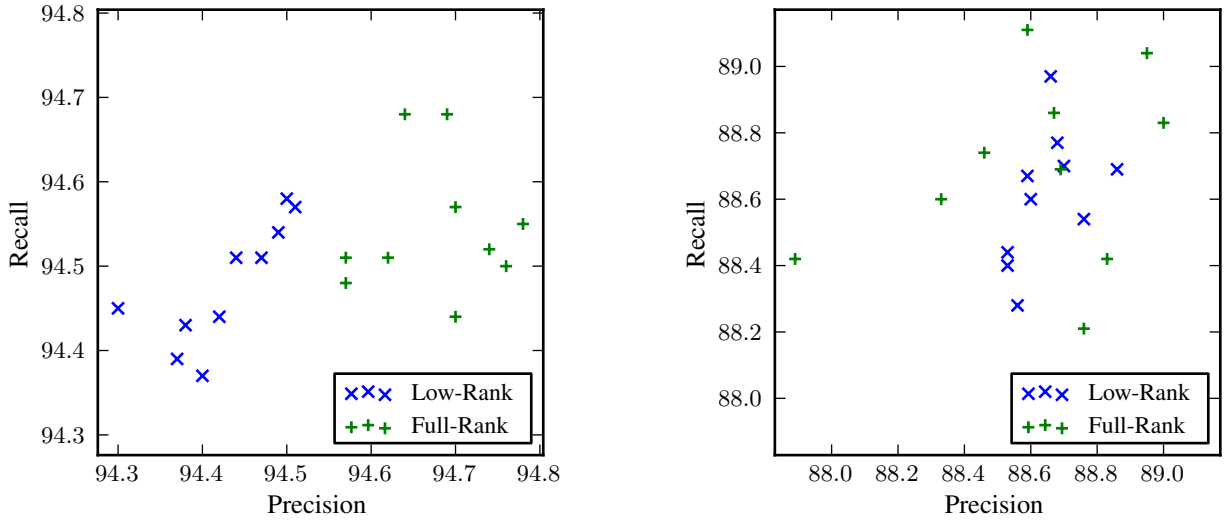
Random search [14] was used to find the learning rate, the size of the hidden layer, regularizations coefficients and dropout rate (when used). We did this search for low-rank and full-rank NeuroCRFs.

5.3. Results

Table 4 shows average F_1 for 10 random initializations per configuration. We used results from [4], which is equivalent to a low-rank NeuroCRF, as a benchmark. We report the average F_1 from ten random initializations, with and without dropout. As expected, the full-rank NeuroCRF significantly improved performance for chunking ($p \leq 1\%$ with a Welsh's t-test). Unfortunately, we cannot report a similar results for NER, where we obtained a statistically significant degradation.

Tables 2 and 3 indicate two significant difference between those two tasks. First, while both tasks are trained on a similar number of words, the chunking task (CoNLL-2000) contains more segments, and most of the words are in a segment. For NER (CoNLL-2003), most of the words are not in a segment. Low-rank NeuroCRFs are simpler models and can be trained with less data. More data is required when training full-rank NeuroCRFs.

Secondly, the advantages of full-rank NeuroCRFs depend on the assumption that a label is dependent on the previous label. If this assumption does not hold, and labels are effectively independent of each others, full-rank NeuroCRFs



(a) Chunking (CoNLL-2000)

(b) NER (CoNLL-2003)

Fig. 1: Precision vs recall

should not outperform low-rank NeuroCRFs. Table 3 shows that for both tasks, knowing the previous label reduces the uncertainty, confirming that there is a dependency between consecutive labels. This dependency is stronger, in relative and absolute terms, for the chunking task than it is for the NER task.

Ideally, full-rank NeuroCRFs should be able to learn parameters equivalent to those of a low-rank NeuroCRF in those situations. We trained NeuroCRFs with dropout, after a new hyper-parameters search, and compared the results. Table 4 shows the results of those experiments. For NER, low-rank and full-rank NeuroCRFs' average F_1 are improved by dropout. For low-rank NeuroCRF, the difference is not statistically significant. The dropout rates found by the random search are different: 29% for low-ranks and 62% for full-rank. This confirms that full-rank NeuroCRFs require more regularization to prevent overfitting. The lack of improvement confirms that the low level of inter-label dependency indicated by the conditional entropy is not sufficient for a full-rank NeuroCRF. In this case, with proper regularization, the model learns parameters equivalent to those of a low-rank NeuroCRF.

For chunking, low-rank and full-rank NeuroCRFs' average F_1 are reduced by dropout. Full-rank NeuroCRFs are still significantly better. Those models were already regularized by early stopping and norm regularization. The addition of dropout was unnecessary and over-regularized the models. The degradation is not statistically significant for the full-rank NeuroCRFs, but *is* significant for the low-rank NeuroCRFs. This confirms that full-rank NeuroCRFs need more regularization and that the effective training set size is important.

Figure 1 shows the precisions and recalls obtained for chunking and NER. For chunking, the low-rank and full-rank NeuroCRFs do not occupy the same space. Full-rank NeuroCRFs have a significantly higher precision, and a higher average recall. For NER, the low-rank and full-rank NeuroCRFs occupy the same space, confirming that the difference between the average F_1 is not significant.

6. CONCLUSIONS

Neural networks are commonly used to model emission features for CRFs. This combination of neural networks and CRFs can be expressed as a rank-deficient matrix computed by the neural network added to a constant transition matrix. We proposed to replace the rank-deficient matrix by a full rank matrix, where the neural network models emission and transition features simultaneously. We refer to this combination as a full-rank NeuroCRF.

Full-rank NeuroCRFs outperformed low-rank NeuroCRFs on a syntactic chunking task. Those two classes of models were equivalent on a named entities recognition task. The syntactic chunking task includes significant dependencies between consecutive labels, which are well modeled by full-rank NeuroCRFs. The named entities recognition task does not have such dependencies. In this case, with proper regularization, the full-rank NeuroCRFs were equivalent to low-rank NeuroCRFs.

7. ACKNOWLEDGEMENTS

The authors would like to thank David Huggins-Daines, Paul Vozila, Ding Liu, and Joumana Ghosn from Nuance for their helpful suggestions and comments.

8. REFERENCES

[1] Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain, "Neural probabilistic language models," in *Innovations in Machine Learning*, pp. 137–186. Springer, 2003.

[2] Jian Peng, Liefeng Bo, and Jinbo Xu, "Conditional neural fields," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J.D. Lafferty, C.K.I. Williams, and A. Culotta, Eds., pp. 1419–1427. Curran Associates, Inc., 2009.

[3] Trinh Do, Thierry Arti, et al., "Neural conditional random fields," in *International Conference on Artificial Intelligence and Statistics*, 2010, pp. 177–184.

[4] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa, "Natural language processing (almost) from scratch," *The Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[5] Kaisheng Yao, Geoffrey Zweig, Mei-Yuh Hwang, Yangyang Shi, and Dong Yu, "Recurrent neural networks for language understanding," August 2013, Interspeech.

[6] Ronan Collobert and Jason Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 160–167.

[7] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean, "Distributed representations of words and phrases and their compositionality," *Neural Information Processing Systems conference*, vol. abs/1310.4546, 2013.

[8] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, "Efficient estimation of word representations in vector space," *Proceedings of Workshop at ICLR*, 2013.

[9] Charles Sutton and Andrew McCallum, *An introduction to conditional random fields for relational learning*, Introduction to statistical relational learning. MIT Press, 2006.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[11] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[12] Erik F. Tjong Kim Sang and Sabine Buchholz, "Introduction to the conll-2000 shared task: Chunking," in *Proceedings of the 2Nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning - Volume 7*, Stroudsburg, PA, USA, 2000, ConLL '00, pp. 127–132, Association for Computational Linguistics.

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

[14] James Bergstra and Yoshua Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.

[15] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio, "Theano: a CPU and GPU math expression compiler," in *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010, Oral Presentation.

[16] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio, "Theano: new features and speed improvements," Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.