Named Entity Recognition With Parallel Recurrent Neural Networks

Andrej Žukov-Gregorič[*‡], Yoram Bachrach[*†], and Sam Coope[*]

[*]DigitalGenius, 1 Canada Square, London E14 5AB
[‡]Department of Computer Science, Royal Holloway, University of London, Egham TW20 0EX
andrej.zukovgregoric.2010@live.rhul.ac.uk
yorambac@gmail.com
sam@digitalgenius.com

Abstract

We present a new architecture for named entity recognition. Our model employs multiple independent bidirectional LSTM units across the same input and promotes diversity among them by employing an inter-model regularization term. By distributing computation across multiple smaller LSTMs we find a reduction in the total number of parameters. We find our architecture achieves state-of-the-art performance on the CoNLL 2003 NER dataset.

1 Introduction

The ability to reason about entities in text is an important element of natural language understanding. Named entity recognition (NER) concerns itself with the identification of such entities. Given a sequence of words, the task of NER is to label each word with its appropriate corresponding entity type. Examples of entity types include *Person*, *Organization*, and *Location*. A special *Other* entity type is often added to the set of all types and is used to label words which do not belong to any of the other entity types.

Recently, neural network based approaches which use no language-specific resources, apart from unlabeled corpora for training word embeddings, have emerged. There has been a shift of focus from handcrafting better features to designing better neural architectures for solving NER.

In this paper, we propose a new parallel recurrent neural network model for entity recognition. We show that rather than using a single LSTM component, as many other recent architecture have, we instead resort to using multiple smaller LSTM units. This has the benefit of reducing the total number of parameters in our model. We present results on the CoNNL 2003 English dataset and achieve the new state of the art results for models without help from an outside lexicons.

1.1 Related Work

Various approaches have been proposed to NER. Many of these approaches rely on handcrafted feature engineering or language-specific or domain-specific resources (Zhou and Su, 2002; Chieu and Ng, 2002; Florian et al., 2003; Settles, 2004; Nadeau and Sekine, 2007). While such approaches can achieve high accuracy, they may fail to generalize to new languages, new corpora or new types of entities to be identified. Thus, applying such techniques in new domains requires making a heavy engineering investment.

Over time neural methods such as (Chiu and Nichols, 2015; Ma and Hovy, 2016; Luo et al., 2015; Lample et al., 2016) emerged. More recently (Peters et al., 2017; Reimers and Gurevych, 2017; Sato et al., 2017) have set the top benchmarks in the field.

Architecturally, our model is similar to those of (Zhu et al., 2017; Hidasi et al., 2016) with the most pronounced difference being that we (1) apply our parallel RNN units across the same input (2) explore a new regularization term for promoting diversity across what features our parallel RNNs extract and (3) explicitly motivate the architecture with a discussion about parameter complexity.

The need for a wider discussion on parameter complexity in the deep learning community is being pushed by the need to make complex neural models runnable in constrained environment such as field-programmable gate arrays (FPGAs) - for a great discussion relating to running LSTMs on FPGAs see (Guan et al., 2017). Additionally, complex models have proven difficult to use in certain

[†] Now at Google DeepMind, 6 Pancras Square, London N1C 4AG.

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 69–74
Melbourne, Australia, July 15 - 20, 2018. ©2018 Association for Computational Linguistics

domains such as embedded systems or finance due to their slowness. Our architecture lends itself to parallelization and attempts to tackle this problem.

2 Named Entity Recognition

Named Entity Recognition can be posited as a standard sequence classification problem where the dataset $D = \{(\mathbf{X}_i, \mathbf{y}_i)\}_{i=1}^{k}$ consists of example label pairs where both the examples and the labels are themselves sequences of word vectors and entity types, respectively.

Specifically, an input example $\mathbf{X}_i = (\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,|X_i|})$ is a variable-length sequence of word vectors $\mathbf{x}_{i,j} \in \mathbb{R}^d$; the example's corresponding label $\mathbf{y}_i = (y_{i,1}, ..., y_{i,|X_i|})$ is a equal-length sequence of entity-type labels $y_{i,j} \in Y$ where Y is the set of all entity type labels and includes a special other 'O'-label with which all words that are not entities are labeled.

The goal is then to learn a parametrized mapping $f_\theta : \mathbf{X} \rightarrow \mathbf{y}$ from input words to output entity labels. One of the most commonly used class of models that handle this mapping are recurrent neural networks.

2.1 LSTM complexity

Long short term memory (LSTM) models belong to the family of recurrent neural network (RNN) models. They are often used as a component of much larger models, particularly in many NLP tasks including NER.

Classically, an LSTM cell is defined as follows (biases excluded for brevity):

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\boldsymbol{W}_i \mathbf{h}_{t-1} + \boldsymbol{U}_i \mathbf{x}_t) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_f \mathbf{h}_{t-1} + \boldsymbol{U}_f \mathbf{x}_t) \\
\mathbf{o}_t &= \sigma(\boldsymbol{W}_o \mathbf{h}_{t-1} + \boldsymbol{U}_o \mathbf{x}_t) \\
\tilde{\mathbf{c}}_t &= \tanh(\boldsymbol{W}_c \mathbf{h}_{t-1} + \boldsymbol{U}_c \mathbf{x}_t) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned}$$

One way of measuring the complexity of a model is through its total number of parameters. Looking at the above, we note there are two parameter matrices, $\mathbf{W}$ and $\mathbf{U}$, for each of the three input gates and during cell update. If we let $\mathbf{W} \in \mathbb{R}^{n \times n}$ and $\mathbf{U} \in \mathbb{R}^{n \times m}$ then the total number of parameters in the model (excluding the bias terms) is $4(nm+n^2)$ which grows quadratically as n grows. Thus, increases in LSTM size can substantially increase the number of parameters.

3 Parallel RNNs

To reduce the total number of parameters we split a single LSTM into multiple equally-sized smaller ones:

$$h_{k,t} = \text{LSTM}_k(h_{k,t-1}, \mathbf{x})$$

where $k \in \{1, ..., K\}$. This has the effect of dividing the total number of parameters by a constant factor. The final hidden state h_t is then a concatenation of the hidden states of the smaller LSTMS:

$$h_t = [h_{1,t}; h_{2,t}; ...; h_{K,t}]$$

3.1 Promoting Diversity

To promote diversity amongst the constituent smaller LSTMs we add a orthogonality penalty *across* the smaller LSTMs. Recent research has used similar methods but applied to single LSTMs (Vorontsov et al., 2017).

We take the cell update recurrence parameters $\mathbf{W}_i$ across LSTMs (we omit the c in the subscript for brevity; the index i runs across the smaller LSTMs) and for any pair we wish the following to be true:

$$\langle \text{vec}(W_c^{(i)}), \text{vec}(W_c^{(j)}) \rangle \approx 0$$

.

To achieve this we pack the vectorized parameters into a matrix:

$$\Phi = \begin{pmatrix} \text{vec}(W_c^{(1)}) \\ \text{vec}(W_c^{(2)}) \\ \vdots \\ \text{vec}(W_c^{(N)}) \end{pmatrix}$$

and apply the following regularization term to our final loss:

$$\lambda \sum_i \|\Phi\Phi^\top - I\|_F^2 \qquad (1)$$

3.2 Output and Loss

The concatenated output h_t is passed through a fully connected layer with bias before being passed through a final softmax layer:

$$o_t = \text{softmax}(\mathbf{W}_{\text{out}} \hat{h}_t + b_{\text{out}})$$

To extract a predicted entity type $\hat{y}_t$ at time t, we select the entity type corresponding to the most probable output:

$$\hat{y}_t = argmax(o_t)$$

The loss is defined as the sum of the softmax cross-entropy losses along the words in the input sequence. More precisely, we denote by $y_t^j \in 0, 1$ a binary indicator variable indicating whether word x_t truly is an entity of type j. The loss at time t is then defined to be $\mathcal{L}_t = -\sum_j y_t^j \log(o_t^j)$. Thus the overall loss is:

$$\mathcal{L} = -\sum_t \sum_j y_t^j \log(o_t^j)$$

3.3 Implementation Details

We use bidirectional LSTMs as our base recurrent unit and use pretrained word embeddings of size 100. These are the same embeddings used in (Lample et al., 2016). We concatenate to our word embeddings character-level embeddings similar to (Lample et al., 2016) but with a max pooling layer instead. Unlike with the parallel LSTMs, we only use a single character embedding LSTM.

Parameters are initialized using the method described by Glorot and Bengio (Glorot and Bengio, 2010). This approach scales the variance of a uniform distribution with regard to the root of the number of parameters in a layer. This approach has been found to speed up convergence compared to using a unit normal distribution for initialization.

Our model uses variational dropout (Gal and Ghahramani, 2016) between the hidden states of the parallel LSTMs. Recent work has shown this to be very effective at training LSTMs for language models (Merity et al., 2017). In our experiments, we use $p = 0.1$ as our dropping probability.

We experiment with different values of the regularization term parameter but settled on $\lambda = 0.01$.

Although vanilla stochastic gradient descent has been effective at training RNNs on language problems (Merity et al., 2017), we found that using the ADAM optimizer (Kingma and Ba, 2014) to be more effective at training our model. We experimented with different values for the learning rate α, increasing α from 10^{-3} to as high as 5×10^{-3} and still obtained good results.

Similarly, we kept a constant size for the character-level embeddings, using a unit bidirectional LSTM output size of $\dim(e^{\text{char}}) = 50$. As previously discussed, we trained the network parameters using stochastic gradient descent (Werbos, 1990), augmented with the Adam optimizer (Kingma and Ba, 2014).

3.4 Relation to Ensemble Methods

Our model bears some resemblance to ensemble methods (Freund et al., 1996; Dietterich et al., 2000), which combine multiple "weak learners" into a single "strong learner"; One may view each of the parallel recurrent units of our model as a single "weak" neural network, and may consider our architecture as a way of combining these into a single "strong" network.

Despite the similarities, our model is very different from ensemble methods. First, as opposed to many boosting algorithms (Freund et al., 1996; Schapire and Singer, 1999; Dietterich et al., 2000) we do not "reweigh" training instances based on the loss incurred on them by a previous iteration. Second, unlike ensemble methods, our model is trained *end-to-end*, as a single large neural network. All the subcomponents are co-trained, so different subparts of the network may focus on different aspects of the input. This avoids redundant repeated computations across the units (and indeed, we encourage diversity between the units using our inter-module regularization). Finally, we note that our architecture does not simply combine the *prediction* of multiple classifiers; rather, we take the final *hidden layer* of each of the LSTM units (which contains more information than merely the entity class prediction), and combine this information using a feedforward network. This allows our architecture to examine inter-dependencies between pieces of information computed by the various components.

4 Experiments

We achieve state-of-the-art results on the CoNNL 2003 English NER dataset (see Table 1). Although we do not employ additional external resources (language specific dictionaries or gazetteers), our model is competitive even with some of the models that do.

To gain a better understanding of the performance of our model including how its various components affect performance we prepared four additional tables of runs.

Table 2 shows performance as a function of the number of RNN units with a fixed unit size. The

Model	F1
(Chieu and Ng, 2002)	88.31
(Florian et al., 2003)	88.76
(Ando and Zhang, 2005)	89.31
(Collobert et al., 2011)‡	89.59
(Huang et al., 2015)‡	90.10
(Chiu and Nichols, 2015)‡	90.77
(Ratinov and Roth, 2009)	90.80
(Lin and Wu, 2009)	90.90
(Passos et al., 2014)‡*	90.90
(Lample et al., 2016)‡	90.94
(Luo et al., 2015)‡	91.20
(Ma and Hovy, 2016)‡	91.21
(Sato et al., 2017)	91.28
(Chiu and Nichols, 2015)‡*	91.62
(Peters et al., 2017)‡*	91.93
This paper‡	**91.48** ±0.22

Table 1: English NER F1 score of our model on the test set of CoNLL-2003 (English). During training we optimize for the development set and report test set results for our best performing development set model. The bounded F1 results we report (±0.22) are taken after 10 runs. For the purpose of comparison, we also list F1 scores of previous top-performance systems. ‡ marks the neural models. * marks model which use external resources.

number of units is clearly a hyperparameter which must be optimized for. We find good performance across the board (there is no catastrophic collapse in results) however when using 16 units we do outperform other models substantially. Even with very small unit sizes of 8 (Table 3) our models performs relatively well without a significant degradation in results. Table 4 shows and 5 show additional results for unit size and component impact on our best performing model.

5 Conclusion

We achieve state-of-the-art results on the CoNLL 2003 English dataset and introduce a new model motivated primarily by its ability to be easily distributable and reduce the total number of parameters. Further work should be done on evaluating it across different classification and sequence classification tasks to study its performance. Additionally, a run-time analysis show be conducted to compare speedups if the model is parallelized across CPU cores.

# RNN units	F_1
1	90.53 ±0.31
2	90.79 ±0.18
4	90.64 ±0.24
8	91.09 ±0.28
16	91.48 ±0.22
32	90.68 ±0.18

Table 2: Performance as a function of the number of RNN units with a fixed unit size of 64; averaged across 5 runs apart from the 16 unit (average across 10 runs).

# RNN units	Unit size	F_1
1	1024	87.54
2	512	91.25
4	256	91.29
8	128	91.31
16	64	91.48 ±0.22
32	32	90.60
64	16	90.79
128	8	90.41

Table 3: Performance of our model with various unit sizes resulting in a fixed final output size h_t. Single runs apart from 16 unit.

Unit size	F_1
8	89.78
16	89.77
32	90.26
64	91.48 ±0.22
128	89.28

Table 4: Performance as a function of the unit size for our best performing model (16 biLSTM units). Single runs apart from with size 64.

Component	F_1
No character embeddings	90.39
No orthogonal regularization	90.79
No Xavier initialization	91.09
No variational dropout	91.03
Mean pool instead of concat	90.49

Table 5: Impact of various architectural decisions on our best performing model (16 biLSTM units, 64 unit size). Single runs.

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research* 6(Nov):1817–1853.

Hai Leong Chieu and Hwee Tou Ng. 2002. Named entity recognition: a maximum entropy approach using global information. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, pages 1–7.

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308* .

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Thomas G Dietterich et al. 2000. Ensemble methods in machine learning. *Multiple classifier systems* 1857:1–15.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics, pages 168–171.

Yoav Freund, Robert E Schapire, et al. 1996. Experiments with a new boosting algorithm. In *Icml*. volume 96, pages 148–156.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*. PMLR, New York, New York, USA, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059. http://proceedings.mlr.press/v48/gal16.html.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. pages 249–256.

Yijin Guan, Zhihang Yuan, Guangyu Sun, and Jason Cong. 2017. Fpga-based accelerator for long short-term memory recurrent neural networks. In *Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific*. IEEE, pages 629–634.

Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, pages 241–248.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991* .

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360* .

Dekang Lin and Xiaoyun Wu. 2009. 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, pages 1030–1038.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint named entity recognition and disambiguation. In *Proc. EMNLP*. pages 879–880.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354* .

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182* .

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes* 30(1):3–26.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. *arXiv preprint arXiv:1404.5367* .

Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108* .

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pages 147–155.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. *arXiv preprint arXiv:1707.09861* .

Motoki Sato, Hiroyuki Shindo, Ikuya Yamada, and Yuji Matsumoto. 2017. Segment-level neural conditional random fields for named entity recognition. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. volume 2, pages 97–102.

Robert E Schapire and Yoram Singer. 1999. Improved boosting algorithms using confidence-rated predictions. *Machine learning* 37(3):297–336.

Burr Settles. 2004. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*. Association for Computational Linguistics, pages 104–107.

Eugene Vorontsov, Chiheb Trabelsi, Samuel Kadoury, and Chris Pal. 2017. On orthogonality and learning recurrent networks with long term dependencies. *arXiv preprint arXiv:1702.00071* .

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE* 78(10):1550–1560.

GuoDong Zhou and Jian Su. 2002. Named entity recognition using an hmm-based chunk tagger. In *proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 473–480.

Danhao Zhu, Si Shen, Xin-Yu Dai, and Jiajun Chen. 2017. Going wider: Recurrent neural network with parallel cells. *arXiv preprint arXiv:1705.01346* .