Deep Active Learning for Named Entity Recognition

Yanyao Shen
UT Austin
Austin, TX 78712
shenyanyao@utexas.edu

Hyokun Yun
Amazon Web Services
Seattle, WA 98101
yunhyoku@amazon.com

Zachary C. Lipton
Amazon Web Services
Seattle, WA 98101
liptoz@amazon.com

Yakov Kronrod
Amazon Web Services
Seattle, WA 98101
kronrod@amazon.com

Animashree Anandkumar
Amazon Web Services
Seattle, WA 98101
anima@amazon.com

Abstract

Deep neural networks have advanced the state of the art in named entity recognition. However, under typical training procedures, advantages over classical methods emerge only with large datasets. As a result, deep learning is employed only when large public datasets or a large budget for manually labeling data is available. In this work, we show that by combining deep learning with active learning, we can outperform classical methods even with a significantly smaller amount of training data.

1 Introduction

Over the past several years, a series of papers have used deep neural networks (DNNs) to advance the state of the art in named entity recognition (NER) (Collobert et al., 2011; Huang et al., 2015; Lample et al., 2016; Chiu and Nichols, 2015; Yang et al., 2016). Historically, the advantages of deep learning have been less pronounced when working with small datasets. For instance, on the popular CoNLL-2003 English dataset, the best DNN model outperforms the best shallow model by only 0.4%, as measured by F1 score, and this is a small dataset containing only 203,621 words. On the other hand, on the OntoNotes-5.0 English dataset, which contains 1,088,503 words, a DNN model outperforms the best shallow model by 2.24% (Chiu and Nichols, 2015).

In this work, we investigate whether we can train DNNs using fewer samples under the active learning framework. Active learning is the paradigm where we actively select samples to be used during training. Intuitively, if we are able to select the most informative samples for training, we can vastly reduce the number of samples required. In practice, we can employ Mechanical Turk or other crowdsourcing platforms to label the samples actively selected by the algorithm. Reducing sample requirements for training can lower the labeling costs on these platforms.

We present positive preliminary results demonstrating the effectiveness of deep active learning. We perform incremental training of DNNs while actively selecting samples. On the standard OntoNotes-5.0 English dataset, our approach matches 99% of the F1 score achieved by the best deep models trained in a standard, supervised fashion despite using only a quarter

2 NER Model Description

We use CNN-CNN-LSTM model from Yun (2017) as a representative DNN model for NER. The model uses two convolutional neural networks (CNNs) (LeCun et al., 1995) to encode characters and words respectively, and a long short-term memory (LSTM) recurrent neural network (Hochreiter and Schmidhuber, 1997) as a decoder. This model achieves the best F1 scores on the OntoNotes-5.0 English and Chinese dataset, and its use of CNNs in encoders enables faster training as compared to previous work relying on LSTM encoders (Lample et al., 2016; Chiu and Nichols, 2015). We briefly describe the model:

Data Representation We represent each input sentence as follows. First, special `[BOS]` and `[EOS]` tokens are added at the beginning and the end of the sentence, respectively. In order to batch the computation of multiple sentences, sentences with similar length are grouped together into buckets, and `[PAD]` tokens are added at the end of sentences to make their lengths uniform inside of the

Proceedings of the 2nd Workshop on Representation Learning for NLP, pages 252–256,
Vancouver, Canada, August 3, 2017.

Formatted Sentence	[BOS]	Kate	lives	on	Mars	[EOS]	[PAD]
Tag	O	S-PER	O	O	S-LOC	O	O

Table 1: Example formatted sentence. To avoid clutter, [BOW] and [EOW] symbols are not shown.

bucket. We follow an analogous procedure to represent the characters in each word. For example, the sentence 'Kate lives on Mars' is formatted as shown in Table 1. The formatted sentence is denoted as $\{\mathbf{x}_{ij}\}$, where $\mathbf{x}_{ij}$ is the one-hot encoding of the j-th character in the i-th word.

Character-Level Encoder For each word i, we use CNNs to extract character-level features $\mathbf{w}_i^{\text{char}}$ (Figure 1). We apply ReLU nonlinearities (Nair and Hinton, 2010) and dropout (Srivastava et al., 2014) between CNN layers, and include a residual connection between input and output of each layer (He et al., 2016). So that our representation of the word is of fixed length, we apply max-pooling on the outputs of the topmost layer of the character-level encoder (Kim, 2014).

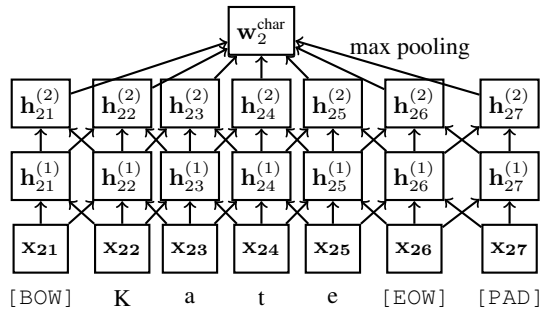


Figure 1: Example CNN architecture for Character-level Encoder with two layers.

Word-Level Encoder To complete our representation of each word, we concatenate its character-level features with $\mathbf{w}_i^{\text{emb}}$, a latent word embedding corresponding to that word:

$$\mathbf{w}_i^{\text{full}} := \left(\mathbf{w}_i^{\text{char}}, \mathbf{w}_i^{\text{emb}}\right).$$

In order to generalize to words unseen in the training data, we replace each word with a special [UNK] (unknown) token with 50% probability during training, an approach that resembles the word-drop method due to Lample et al. (2016).

Given the sequence of word-level input features $\mathbf{w}_1^{\text{full}}, \mathbf{w}_2^{\text{full}}, \ldots, \mathbf{w}_n^{\text{full}}$, we extract word-level representations $\mathbf{h}_1^{\text{Enc}}, \mathbf{h}_2^{\text{Enc}}, \ldots, \mathbf{h}_n^{\text{Enc}}$ for each word position in the sentence using a CNN. In Figure 2, we depict an instance of our architecture with two convolutional layers and kernels of width 3. We concatenate the representation at the l-th convolutional layer $\mathbf{h}_i^{(l)}$, with the input features $\mathbf{w}_i^{\text{full}}$:

$$\mathbf{h}_i^{\text{Enc}} = \left(\mathbf{h}_i^{(l)}, \mathbf{w}_i^{\text{full}}\right)$$

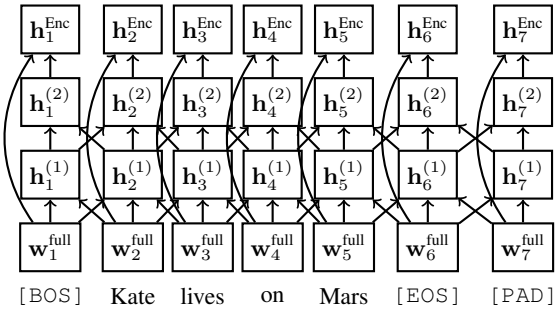


Figure 2: Example CNN architecture for Word-level Encoder with two layers.

Tag Decoder The tag decoder induces a probability distribution over sequences of tags, conditioned on the word-level encoder features: $\mathbb{P}\left[y_2, y_3, \ldots, y_{n-1} \mid \left\{\mathbf{h}_i^{\text{Enc}}\right\}\right]$[1]. We use an LSTM RNN for the tag decoder, as depicted in Figure 3. At the first time step, the [GO]-symbol is provided as y_1 to the decoder LSTM. At each time step i, the LSTM decoder computes $\mathbf{h}_{i+1}^{\text{Dec}}$, the hidden state for decoding word $i+1$, using the last tag y_i, the current decoder hidden state $\mathbf{h}_i^{\text{Dec}}$, and the learned representation of next word $\mathbf{h}_{i+1}^{\text{Enc}}$. Using a softmax loss function, y_{i+1} is decoded; this is further fed as an input to the next time step.

While it is computationally intractable to find the best sequence of tags with an LSTM decoder, Yun (2017) reports that greedily decoding tags from left to right often yields performance superior to chain CRF decoder (Lafferty et al., 2001), for which exact inference is tractable.

[1] y_1 and y_n are ignored because they correspond to auxiliary words [BOS] and [EOS]. If [PAD] words are introduced, they are ignored as well.

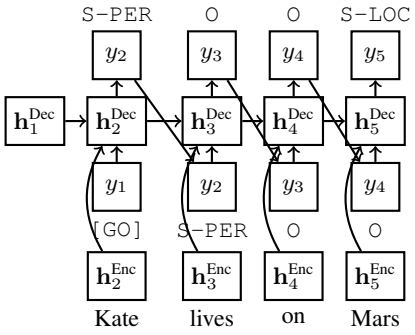


Figure 3: LSTM architecture for Tag Decoder.

3 Active Learning

As with most tasks, labeling data for NER usually requires manual annotations by human experts, which are costly to acquire at scale. Active learning seeks to ameliorate this problem by strategically choosing which examples to annotate, in the hope of getting greater performance with fewer annotations. To this end, we consider the following setup for interactively acquiring annotations. The learning process consists of multiple rounds: At the beginning of each round, the active learning algorithm chooses sentences to be annotated up to the predefined budget. After receiving annotations, we update the model parameters by training on the augmented dataset, and proceeds to the next round. We assume that the cost of annotating a sentence is proportional to the number of words in the sentence, and that every word in the selected sentence needs to be annotated; the algorithm cannot ask workers to partially annotate the sentence.

While various existing active learning strategies suit this setup (Settles, 2010), we explore the uncertainty sampling strategy, which ranks unlabeled examples in terms of current model's uncertainty on them, due to its simplicity and popularity. We consider three ranking methods, each of which can be easily implemented in the CNN-CNN-LSTM model as well as most common models for NER.

Least Confidence (LC): This method sorts examples in descending order by the probability of *not* predicting the most confident sequence from the current model (Lewis and Gale, 1994; Culotta and McCallum, 2005):

$$1 - \max_{y_1,\ldots,y_n} \mathbb{P}\left[y_1,\ldots,y_n \mid \{\mathbf{x}_{ij}\}\right]. \quad (1)$$

Since exactly computing (1) is not feasible with the LSTM decoder, we approximate it with the probability of a greedily decoded sequence.

Maximum Normalized Log-Probability (MNLP): Our preliminary analysis revealed that the LC method disproportionately selects longer sentences. Note that sorting unlabeled examples in descending order by (1) is equivalent to sorting in ascending order by the following scores:

$$\begin{aligned}
&\max_{y_1,\ldots,y_n} \mathbb{P}\left[y_1,\ldots,y_n \mid \{\mathbf{x}_{ij}\}\right] \\
\Leftrightarrow &\max_{y_1,\ldots,y_n} \prod_{i=1}^{n} \mathbb{P}\left[y_i \mid y_1,\ldots,y_{n-1},\{\mathbf{x}_{ij}\}\right] \\
\Leftrightarrow &\max_{y_1,\ldots,y_n} \sum_{i=1}^{n} \log \mathbb{P}\left[y_i \,|\, y_1,\ldots,y_{n-1},\{\mathbf{x}_{ij}\}\right]. \quad (2)
\end{aligned}$$

Since (2) contains summation over words, LC method naturally favors longer sentences. Because longer sentences require more labor for annotation, however, we find this undesirable, and propose to normalize (2) as follows, which we call Maximum Normalized Log-Probability method:

$$\max_{y_1,\ldots,y_n} \frac{1}{n} \sum_{i=1}^{n} \log \mathbb{P}\left[y_i \mid y_1,\ldots,y_{n-1},\{\mathbf{x}_{ij}\}\right].$$

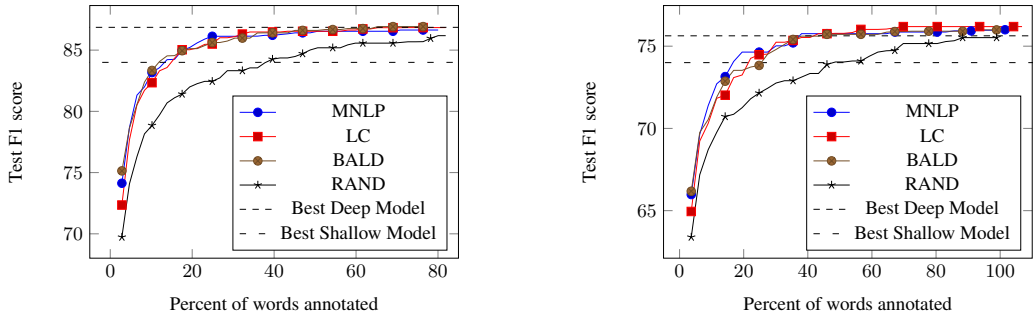
Bayesian Active Learning by Disagreement (BALD): We also consider the Bayesian metric proposed by Gal et al. (2017). Denote $\mathbb{P}^1, \mathbb{P}^2, \ldots \mathbb{P}^M$ as models sampled from the posterior. Then, one measure of our uncertainty on the ith word is f_i, the fraction of models which disagreed with the most popular choice:

$$f_i = 1 - \frac{\max_y \left|\left\{m : \text{argmax}_{y'}\, \mathbb{P}^m\left[y_i = y'\right] = y\right\}\right|}{M},$$

where $|\cdot|$ denotes cardinality of a set. We normalize this by the number of words as $\frac{1}{n}\sum_{j=1}^{n} f_j$, and sort sentences in decreasing order by this score. Following Gal et al. (2017), we used Monte Carlo dropout (Gal and Ghahramani, 2016) to sample from the posterior, and set M as 100.

4 Experiments

We use OntoNotes-5.0 English and Chinese data (Pradhan et al., 2013) for our experiments. The training datasets contain 1,088,503 words and 756,063 words respectively. State-of-the-art models trained the full training sets achieve F1 scores of 86.86 and 75.63 on the test sets (Yun, 2017).



(a) OntoNotes-5.0 English

(b) OntoNotes-5.0 Chinese

Figure 4: F1 score on the test dataset, in terms of the number of words labeled.

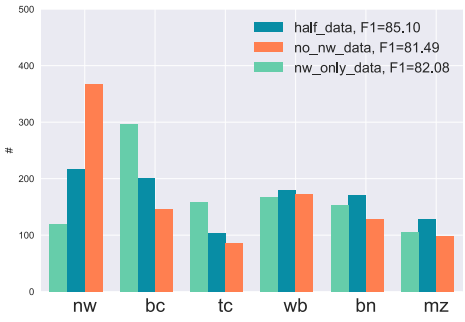


Figure 5: Genre distribution of top 1,000 sentences chosen by an active learning algorithm

Comparisons of selection algorithms We empirically compare selection algorithms proposed in Section 3, as well as uniformly random baseline (**RAND**). All algorithms start with an identical 1% of original training data and a randomly initialized model. In each round, every algorithm chooses sentences from the rest of the training data until 20,000 words have been selected, adding this data to its training set. Then, the algorithm updates its model parameters by stochastic gradient descent on its augmented training dataset for 50 passes. We evaluate the performance of each algorithm by its F1 score on the test dataset.

Figure 4 shows results. All active learning algorithms perform significantly better than the random baseline. Among active learners, **MNLP** slightly outperformed others in early rounds. Impressively, active learning algorithms achieve 99% performance of the best deep model trained on full data using only 24.9% of the training data on the English dataset and 30.1% on Chinese. Also, 12.0% and 16.9% of training data were enough for deep active learning algorithms to surpass the performance of the shallow models from Pradhan et al. (2013) trained on the full training data.

Detection of under-explored genres To better understand how active learning algorithms choose informative examples, we designed the following experiment. The OntoNotes datasets consist of six genres: broadcast conversation (bc), braodcast news (bn), magazine genre (mz), newswire (nw), telephone conversation (tc), weblogs (wb). We created three training datasets: **half-data**, which contains random 50% of the original training data, **nw-data**, which contains sentences only from newswire (51.5% of words in the original data), and **no-nw-data**, which is the complement of **nw-data**. Then, we trained CNN-CNN-LSTM model on each dataset. The model trained on **half-data** achieved 85.10 F1, significantly outperforming others trained on biased datasets (**no-nw-data**: 81.49, **nw-only-data**: 82.08). This showed the importance of good genre coverage in training data. Then, we analyzed the genre distribution of 1,000 sentences **MNLP** chose for each model (see Figure 5). For **no-nw-data**, the algorithm chose many more newswire (nw) sentences than it did for unbiased **half-data** (367 vs. 217). On the other hand, it undersampled newswire sentences for **nw-only-data** and increased the proportion of broadcast news and telephone conversation, which are genres distant from newswire. Impressively, although we did not provide the genre of sentences to the algorithm, it was able to automatically detect underexplored genres.

5 Conclusion

We proposed deep active learning algorithms for NER and empirically demonstrated that they achieve state-of-the-art performance with much less data than models trained in the standard supervised fashion.

References

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.

Aron Culotta and Andrew McCallum. 2005. Reducing labeling effort for structured prediction tasks. In *AAAI*. volume 5, pages 746–51.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*. pages 1019–1027.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017. Deep bayesian active learning with image data. *arXiv preprint arXiv:1703.02910* .

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

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

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* .

John Lafferty, Andrew McCallum, Fernando Pereira, et al. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*. volume 1, pages 282–289.

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

Yann LeCun, Yoshua Bengio, et al. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks* 3361(10):1995.

David D Lewis and William A Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*. Springer-Verlag New York, Inc., pages 3–12.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. pages 807–814.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. In *CoNLL*. pages 143–152.

Burr Settles. 2010. Active learning literature survey. *University of Wisconsin, Madison* 52(55-66):11.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Zhilin Yang, Ruslan Salakhutdinov, and William Cohen. 2016. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270* .

Hyokun Yun. 2017. Design choices for named entity recognition. Manuscript in preparation.