Named Entity Recognition through Learning from Experts

Martin Andrews

Red Cat Labs, Singapore
Martin.Andrews@RedCatLabs.com
http://www.RedCatLabs.com

Abstract. Named Entity Recognition (NER) is a foundational technology for systems designed to process Natural Language documents. However, many existing state-of-the-art systems are difficult to integrate into commercial settings (due their monolithic construction, licensing constraints, or need for corpuses, for example). In this work, a new NER system is described that uses the output of existing systems over large corpuses as its training set, ultimately enabling labelling with (i) better F1 scores; (ii) higher labelling speeds; and (iii) no further dependence on the external software.

Keywords: Named Entity Recognition, NER, Natural Language Processing, NLP, Recurrent Neural Network, RNN, Unstructured Data

1 Introduction

One key capability required of natural language processing (NLP) systems is to be able to identify the people, organisations and locations mentioned in a given text. These labels (plus further categories that include times, dates, and numeric quantities, for instance) are essential for understanding the facts described, yet they do not *per se* add much to the linguistic structure of the text. Therefore, building systems that can reliably perform this Named Entity Recognition (NER) has been a focus of NLP research, since it is an essential stepping-stone to exploring the other linguistic content in unstructured text.

Unfortunately, while the NER task might be considered largely conquered from a linguistic research viewpoint, building an effective system is still a challenge in a commercial setting :

1. Licenses for many existing academic systems are not conducive to being embedded within commercial systems
2. Often, existing codebases focus on 'tweaks' rather than solid engineering
3. Commercial systems may have particular task-specific requirements that are difficult to implement on a pre-built system
4. Training corpuses can be a limiting factor, since commercial uses focus on specific domains of interest, rather than domains that have well understood corpuses already available

This work describes an NER system that can be trained from the output of 'known good' systems. Since the system developed here only requires large volumes of (machine) annotated text, it essentially sidesteps several of the problems that these existing systems have in commercial settings.

Moreover, the experiments show that the new system can learn to be better than its teachers - both in the test scores obtained and labelling speed.

Importantly, the results obtained during training and testing are described here in full - the models have not been cherry-picked and tweaked for publication - which illustrates the robustness of this type of model and training process.

2 Model

2.1 Vocabulary Building

As described below, the CoNLL-2003 [1] NER datasets were chosen as the testbed for this work, and the unlabelled 'Large Corpus' was used to build the vocabulary and word-embedding features.

A vocabulary was built from the contents of the whole Large Corpus (there were 484k distinct tokens in the 1.0Gb corpus) with the following additional tokenization steps taken prior to insertion into the dictionary :

1. Convert to lower case
2. Replace each string of digits within the token with `NUMBER` (so that, for instance, '12.3456' becomes '`NUMBER.NUMBER`')

2.2 Word Embedding Layer

Skip-gram embeddings of size 100 were pre-trained over the whole large corpus and vocabulary using `word2vec`[2] as provided by the Python package `gensim`[3] (this required only 15 minutes of wall-clock time).

The token embedding was filtered so that only tokens with 10 mentions or more were included, yielding an effective vocabulary size of 118,695 distinct tokens. To cope with words not present in the embedding, a special token <`UNK`> was added to the embedding space, with a vector that corresponded to the mean vector over the rest of the known dictionary.

2.3 Additional Features

The only feature added to the vector representation of each token was an indicator {0,1} as to whether that token/word had originally contained upper-case characters. Therefore, for each token the extended vector given to the next stage was 101 elements in length.

2.4 Bi-Directional Recurrent Neural Network (RNN)

Having mapped each token to a numerical input vector, a bi-directional recurrent neural network was used to map the token embeddings to hidden states. Since each timestep corresponded to exactly one output label, it was not necessary to separate ingestion and output RNNs : a lock-step arrangement was sufficient.

The model was built using Theano using the recently announced `blocks`[4] framework, which provides many useful primatives, and is currently under active development. The sizes of the embedded parameters are given in Table 1.

In the interests of initially keeping the model as simple as possible, a very basic recurrent network was used :

$$\mathbf{h}_t^F = \tanh(\mathbf{W}^F \mathbf{h}_{t-1}^F + \mathbf{x}_t)$$
$$\mathbf{h}_t^B = \tanh(\mathbf{W}^B \mathbf{h}_{t+1}^B + \mathbf{x}_t)$$

where $\mathbf{h}_t^F$ and $\mathbf{h}_t^B$ refer the hidden states in the forward and backward chains respectively; $\mathbf{W}^F$ and $\mathbf{W}^B$ refer to independent weight matrices for each chain, and $\mathbf{x}_t$ is the extended token vector.

The initial conditions to the forward and backward chains, $\mathbf{h}_{-1}^F$ and $\mathbf{h}_{T+1}^B$ are set to values (that are also trainable inputs) at the beginning and end of each sentence respectively.

Labelling Output Layer One feature of the CoNLL-2003 datasets was that in addition to the basic {`PER`, `ORG`, `LOC`, `MISC`} entity labels, there were also specific 'Beginning' labels to be used to separate two entities which abutted against each other without any other intervening token. However, situations in which this actually arose were very rare (respectively {0.0%, 0.2%, 0.1%, 0.7%} of each token's occurrences). Therefore, to simplify the output stage logic, only 5 labels were learned (the entity labels, plus `O` for non-entity tokens).

The output stage consisted of a dense linear layer (with bias), with each of the 5 label outputs at a given timestep connected to all the RNN hidden units at the same timestep (both forwards and backwards chains), followed by softmax :

$$\mathbf{d}_t = \mathbf{X}^F \mathbf{h}_t^F + \mathbf{X}^B \mathbf{h}_t^B + \mathbf{b}_t$$
$$\mathbf{p}_t^i = \frac{e^{\mathbf{d}_t^i}}{\sum_k e^{\mathbf{d}_t^k}}$$

where $\mathbf{d}_t$ refers to the linear combination over the RNN outputs at that timestep; $\mathbf{X}^F$ and $\mathbf{X}^B$ refer to independent weight matrices for each chain; $\mathbf{b}_t$ is a bias term; and $\mathbf{p}_t$ is the softmax output for the assigned label.

This 'one-hot' representation was trained using Categorical Cross-Entropy for each label summed over batches of sentences as an objective function for gradient descent, which used an ADADELTA[5] step rule.

During the test phase, labels were simply read from the output stage, without post-processing.

Table 1. Model Parameters

Parameter Set	Notation	Shape	# of Float32
Word Embedding (all tokens)	$\mathbf{x}$	(118695, 100)	11,869,500
Generated features	`token_ucase()`	(..., 1)	n/a
State transition matrices	$\mathbf{W}^F$ and $\mathbf{W}^B$	(101, 101) × 2	20,402
State initialisation vectors	$\mathbf{h}^F_{-1}$ and $\mathbf{h}^B_{T+1}$	(101, 1) × 2	202
RNN outputs to label matrix	$\mathbf{X}^F$ and $\mathbf{X}^B$	(202, 5)	1,010
RNN outputs to label biases	$\mathbf{b}_t$	(1, 5)	5
Total (RNN only)			21,619
Total Model			11,891,119

2.5 External Models

As a basis for learning, the RNN was trained against the provided training set (3.3Mb) as well as the Large Corpus labelled by two external models. These models were chosen because they are both state-of-the-art, have acceptable licenses and were easiest to use off-the-shelf.

Please note that this paper's results are only possible because its RNN models are able to 'stand on the shoulders of giants' : there is no intention here to detract from the fine work that went into creating these models in the first place.

Other potential candidate models are mentioned below (in Related Work), but studying the following was sufficient for the present experiments.

In all cases, care was taken to ensure that the models all treated the given tokenization in the same way, and that the results obtained from the models alone matched the reported results.

MITIE According to its GitHub page (`https://github.com/mit-nlp/MITIE`), the MITIE project (built around `dblib` [6]) is a state-of-the-art information extraction tool, which performs named entity extraction and binary relation detection. It is available under a permissive Open Source license (which, interestingly, was one of the key objectives of the funding for the project provided by the DARPA XDATA program).

Model files specifically constructed for the CoNLL 2003 NER task are available for download. All MITIE output here was created using the 343Mb model file `english_ner_model_just_conll.dat.bz2`.

Stanford Named Entity Recognizer According to its substantial documentation page (`http://nlp.stanford.edu/software/CRF-NER.shtml`) the Stanford Named Entity Recognizer provides a general implementation of (arbitrary order) linear chain Conditional Random Field (CRF) sequence models [7], and is included in the Stanford CoreNLP suite of NLP tools.

According to Stanford's NER benchmarks, the Stanford model was used to submit results in the original CoNLL-2003 competition, and performed well. The model file used here (v3.5.2 of `english.conll.4class.distsim.crf.ser.gz`) is close to (or an improvement on, it is unclear) the original CoNLL-tuned version. The compressed model size appears to be approximately 110Mb.

3 Experiments

3.1 CoNLL-2003

The experimental setting chosen was the same as given in CoNLL-2003 [1]. This provided several distinct datasets (statistics for which are given in Table 2), each of which were tokenised using the CoNLL-provided scripts :

"Large Corpus" This consists of 10 months of Reuters news stories, with no labelling provided;

Training Set This is a labelled set of data that models can be trained on - with the option also available (in 2003) of using external training data too;

Development Set This is a hold-out labelled test set ('`testa`') which was set aside for validation and/or hyper-parameter selection;

Test Set This is the labelled test set ('`testb`'), with scripts provided to calculate recall/precision/F1 scores both overall and for each category label.

Table 2. Data set sizes

Data sizes	Bytes	Words	Sentences
"Large Corpus"	1.0Gb	184,717,139	11,869,032
Training Set	3.3Mb	204,567	14,987
Development Set	827Kb	51,578	3,467
Test Set	748Kb	46,666	3,685

As described earlier, no additional pre- or post- processing was applied to the data.

3.2 Models and Training

Training Initial training runs used 15 million labelled sentences (this figure was chosen to be approximately 1000 epochs on `train` - sufficient to fully learn the CoNLL-2003 provided data). For the more extensive runs, the number of labelled sentences was arbitrarily fixed at 100 million (this count does not include sentences that were excluded by the 'Consensus' technique below).

Expert Scores Included in Table 3 are results for the base scores for the two expert models. These figures agree with their previously reported scores.

RNN learning from individual Experts Test results are given for 15 and 100 million sentences of training (over the output of the respective expert labelling of the Large Corpus). These results are surprisingly close to the expert they are being trained from, despite having no knowledge of the internal workings (or tweaks, tricks, etc) being used.

In order to test the variability of models built, the 'RNN-MITIE' model was trained with 15 different random number seeds for the internal model initialisation (using, however, from the same initial word embedding data). The resulting set of `testb` F1 scores had mean 88.15% and standard deviation of 0.14%.

RNN trained on Training Set alone Although the RNN has the benefit of a word embedding derived from the Large Corpus, the results show that solely learning the labelling task from the training data set (1000 epochs) was insufficient for good performance.

RNN 'Mixer' This RNN was trained from a data source that took (in turn) one sentence from each of the Training Set, and Large Corpus sets as labelled by the MITIE and Stanford experts (i.e. 3 sources in equal measure - even though this implies considerably more epochs of Training Set data, since it is so much smaller in size).

RNN 'Consensus' These RNNs were trained from a data source that took a fixed proportion α of sentences from the Training Set (given as a percentage in Table 3), and sentences whose labelling both the MITIE and Stanford experts agreed upon *in full*. The fixed proportion α, viewed as a hyper-parameter, was chosen according to the RNN performance on the Development Set (this was the only time the `testa` dataset was used).

Ensembling Simple ensembles of the most promising 'Consensus' RNNs and the given experts were created (one of each type, doing a simple vote for each output label). In addition, RNN models trained on each expert were also tested as members of ensembles, to see whether ensembling gains could be made using solely RNN-trained models.

Table 3. F1 scores for individual and ensembled models

	Sentences (millions)	Training Set F1%	Dev. Set F1%	Test Set F1%
Individual Models				
Expert-MITIE	n/a	96.98	97.11	88.10
Expert-Stanford	n/a	97.66	91.79	88.19
RNN-MITIE	15	90.43	91.11	86.58
RNN-MITIE	100	93.08	93.25	88.08
RNN-Stanford	15	90.19	89.03	85.51
RNN-Stanford	100	91.93	90.26	86.24
RNN-TrainSet	15	**99.62**	84.47	79.50
RNN-Mixer	100	99.50	93.39	88.76
RNN-Consensus-00%	100	94.01	93.04	88.64
RNN-Consensus-05%	100	98.65	**93.66**	89.45
RNN-Consensus-10%	100	99.38	93.60	**89.51**
Ensemble Models (100 million sentences)				
Consensus-05 + RNN-MITIE + RNN-Stanford		95.85	93.64	89.52
Consensus-05 + Expert-MITIE + RNN-Stanford		97.77	94.69	89.68
Consensus-05 + RNN-MITIE + Expert-Stanford		98.22	94.08	89.92
Consensus-05 + Expert-MITIE + Expert-Stanford		98.72	95.34	**90.12**
Consensus-10 + Expert-MITIE + Expert-Stanford		99.00	95.38	**90.18**

4 Analysis

4.1 The CoNLL-2003 task

One surprising aspect of the CoNLL-2003 task was that the `testb` data set (on which final F1s are measured) appears to be significantly different from the training data given. Several features stand out :

1. There are many sports scores in `testb` (presumably because Reuters news carried a lot of these articles during that end-of-summer time period);
2. Sports score summaries contain a lot of numeric tokens, with little in the way of other linguistic structure;
3. Sometimes labelling of teams can be problematic, with 'China' being both a location and a team (organisation) name.

The difficulty of `testb` is noticable specifically in the F1 scores for 'RNN-TrainSet', which completely ignores `testa` during training. Note, though, that all the other training runs may have implicit dependencies on `testa` simply because the MITIE and Stanford systems may have relied on hyper-parameter selection based on `testa` performance.

4.2 Model Complexity

As mentioned in the description of the models used, an attempt was made to keep the model becoming more complicated than necessary. The results obtained indicate that the Simple Recurrent setup used is sufficient for the NER task.

However, for more complex tasks, it seems likely that Gated Recurrent Units (GRU[8]) or their highly parameterized predecessor Long Short-Term Memory (LSTM[9]) may have more expressive power (particularly since these are now commonly being stacked in layers). Fortunately, the `blocks` framework chosen here is flexible enough to accommodate these enhancements.

In the context of the NER task, it is possible that the Stanford model incorporates processes that are difficult to learn for the Simple Bi-Directional RNN used - as evidenced by the F1 scores converging more slowly during training than is the case for the MITIE model. In addition, during ensembling, using RNN-Stanford was significantly less impactful than RNN-MITIE, which is a pity, since the Stanford model is heavier computationally, as can be seen from Table 4.

Table 4. System labelling speed

	Sentences per second	Comment
Expert-MITIE	1,646	OpenBLAS / Lapack found during compilation, but the system appeared to run single-threaded
Expert-Stanford	48	This was invoked through `Stanford CoreNLP`, but only stages relevant to NER were run
RNN (all)	3,123	This implementation was GPU-based, and timings were taken during backprop training (simply labelling requires fewer operations)

4.3 Implementation Speed

The RNN implementation benefited significantly from using a consumer-grade GPU. One feature of the Theano/`blocks` framework is that the model description is coded independent of the target computing device, since Theano is capable of dynamically creating `C++`, `OpenCL`, and `CUDA` code as required.

By choosing an appropriate batch size for the training (so that multiple sentences to be trained in parallel), a speed-up of 35x was realized over the initial choice of parameters used by example code online (see Table 5).

A GPU blocksize of 256 was chosen, since higher blocksizes appeared to cause significant delays in saving `Checkpoint` data to disk (the 1Gb files saved for the 256 blocksize were deemed acceptable).

Overall, the training time on each 100 million sentence experiment was 7-8 hours. The Consensus experiments took approximately 50% longer, solely because much of the data ingested was immediately discarded (and not learned).

Table 5. Training time in seconds on 150k sentences (lower is better)

	CPU	GPU
batchsize	i7-4770 CPU @ 3.40GHz	GTX 760 (2Gb)
8	1030	455
64	254	70
256	211	29
512	n/a	23

4.4 Consensus Methods

Evidently, training a model solely on the data upon which experts agree is an effective approach. What is surprising is that it still works in the cases where the experts would disagree, because the model would not have received training from either expert in these circumstances.

Comparing the Consensus models with 'Mixer' (which is very similar in design, except that no filtering is taking place : the three sources of training data are used on a round-robin basis), it is clear that filtering the training examples is actually beneficial to learning.

There is also a sense in which the Consensus models are performing an ensembling-together of three different training datasets - with the ensemble voting taking place during the ingestion phase, rather than on the final output labels. Interestingly, this puts a heavy burden on the generalization ability of the RNN model to cases in which its supposed teachers disagree. Apparently, this is something these models are capable of doing.

4.5 Ensembling

The best results obtained in this paper were (unsurprisingly) from ensembles of models. Indeed, some of these results broke through the apparent 90% F1 score barrier. However, it was somewhat disappointing that ensembles of pure RNN models didn't reach the same levels of performance of RNN models ensembled with the original experts. This is particularly true of the Stanford model, which is the more desirable of the two models chosen to eliminate (due to speed and licensing considerations).

On the other hand, from a practical point of view, optimising out the last ounce of performance is probably less important than the overall lessons to be learned : Ensembling does work between models, but the implicit ensembling provided by the training of the Consensus models may be both more robust and easy to implement.

4.6 Further Enhancements

The commercial setting in which this work takes place is particularly focused on English-language documents sourced from the ASEAN region.

Given the variability of names in the region (specifically names of people), and their 'obvious' differences in spelling from English words, one further enhancement to the system is the creating of additional word features using a letter-based RNN trained on databases of English prose and ASEAN names (these corpuses have already been curated).

Thus, instead of embedding concrete gazetteers (as is common for more traditional systems), the plan is to train an RNN on the NER task on a character-by-character basis. The trained RNN can then be 'cut off at the output stage' so that its internal pre-output state (a 20 element vector) can be used as additional features for each token for the RNN described in this paper (name tokens that might otherwise all be assigned to `UNK`). This scheme may also offer the opportunity to further characterize names by country-of-origin, for instance.

5 Related Work

Surprisingly, an approach that used LSTM neural networks was previously undertaken for the CoNLL task in 2003 [10]. However, this was published well before importance of word-embedding was understood, so the results reported there (<75% F1 overall) are essentially from a different era.

Work by Collobert *et al* [11] published in 2011, demonstrated that a pure data-driven neural network approach to language tasks can be very effective. They made use of extensively trained word-embeddings, but did not make use of Recursive Neural Networks (their 'sentence scoring' element was performed using a max-pooling approach over a convolutional layer on top of the word embeddings). Their sofware `SENNA` is published under a No Commercial Usage license, and achieves approximately the same performance as the Consensus models created here.

Presented at ICLR (in May 2015), Oriol Vinyals et al [12] essentially repurposed Google's LSTM translation framework to learn 'Grammar as a foreign language'. This task is more difficult than the step-wise labelling performed herein, and required considerably larger computation resources. For example, their network needed to produce 100 different labels, and they made use of a 512-dimensional embedding, and large multi-layered LSTM networks with a 4000-dimensional internal state. Overall, their model included 34 million trainable parameters. That being said, their approach strongly influenced the direction of this work.

5.1 Other External Models Considered

Berkeley Entity Resolution System The Berkeley NER[13] system is also a state-of-the-art NER system, and is part of the suite of software used in the 'Grammar as a Foreign Language' work cited above. It is GPL3+ licensed, which would be acceptable for the current work, however it was not used here purely for time reasons.

Illinois Named Entity Tagger This NER system[14], created by the Cognitive Computation Group from the University of Illinois at Urbana-Champaign, reports scoring 90.8% `testb` F1 on the CoNLL-2003 task, which makes it an attractive candidate system to learn from.

However, despite the Illinois NER system being available under a broadly copyleft license to a Licensee for "its own academic and research purposes", the license includes the following explicit non-commercial usage clause :

> "No license is granted herein that would permit Licensee to incorporate the Software into a commercial product, or to otherwise commercially exploit the Software. "

This current work illustrates the type of legal questions that learning systems bring into focus : If the software is solely used to create a corpus annotation, and a model is trained from that corpus, has the Software been commercially exploited? Is the Licensor asserting come kind of usage rights over all output of the Software? This is surely a new set of challenges to be faced by software license writers, similar to how the GPL has evolved to avoid the 'Tivoization' problem.

6 Conclusions

This work has shown that it is possible to build a near state-of-the-art NER system based solely on the output of externally created software systems.

Even without ensembling (from which even better results were obtained), the resulting system was shown to have learned to exceed the capabilities of its teachers, while being significantly more amenable to usage within a commercial environment.

7 Acknowledgements

The author thanks DC Frontiers, a Singapore-based company that has created the data-centric service 'Handshakes' (`http://www.handshakes.com.sg/`), for their willingness to believe the system created herein was feasible.

DC Frontiers is the recipient of a Technology Enterprise Commercialisation Scheme grant from SPRING Singapore, under which this work took place.

8 Appendix

Working code to implement the RNN scheme outlined in this paper is available through links on : `https://github.com/mdda`

References

1. Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 142–147, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.
2. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.
3. Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.
4. Bart van Merriënboer, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, and Yoshua Bengio. Blocks and fuel: Frameworks for deep learning. *CoRR*, abs/1506.00619, 2015.
5. Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.
6. Davis E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.
7. Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.
8. Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.
9. Alex Graves. *Supervised sequence labelling with recurrent neural networks*, volume 385. Springer, 2012.
10. James 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*, CONLL '03, pages 172–175, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.
11. Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. Natural language processing (almost) from scratch. *CoRR*, abs/1103.0398, 2011.
12. Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. Grammar as a foreign language. *CoRR*, abs/1412.7449, 2014.
13. Greg Durrett and Dan Klein. A joint model for entity analysis: Coreference, typing, and linking. In *Proceedings of the Transactions of the Association for Computational Linguistics*, 2014.
14. Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, pages 147–155, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.