Named Entity Recognition using Recurrent Neural Networks and Conditional Random Fields

Reyhane Askari Hemmat
askarihr@iro.umontreal.ca

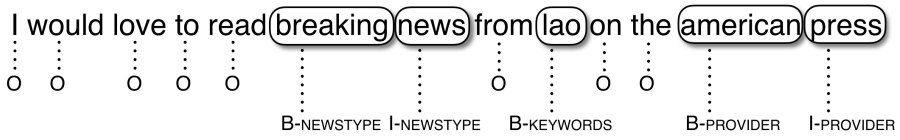
Abstract

In this technical report, we compare two sequence models to build a classifier system that is capable of labeling IOB (Inside, Outside, Beginning) tags given an input sentence. We used Simple Recurrent Neural Networks (SRNs) and Conditional Random Fields (CRFs) as the sequence models.

1 Task Definition

Named Entity Recognition (NER) is a classification problem which amounts to finding the name of each token (AKA entity) in a context such as language. In this problem we are trying to find the IOB tags. IOB which stands for inside, outside and beginning is a tagging format for the tokens inside each chunk. The B-tag points to the start of a chunk. The I-tag points to the tokens inside of each chunk and the O-tag points to the tokens that belong to no chunk. We can only use the B-tag when a tag is followed by a tag of the same type without O tokens between them.

The models are trained with the *News Tagged Dataset* which contains 1000 sentences. The longest sentence in the dataset has 29 words. Each word is tagged using one of the following tags; *I-SECTION, B-SECTION, O, I-KEYWORDS, B-KEYWORDS, B-PROVIDER, B-NEWSTYPE, I-NEWSTYPE, I-PROVIDER*. Our aim is to build a system that can predict the tags for unseen and new sentences. For example given the sentence; *news about Obama*, we would like to find the following:



2 Models

In this section we first introduce Simple Recurrent model which is the baseline model for all Recurrent Neural Network. We also briefly introduce Conditional Random Field that has been one of the common probabilistic models used for the task of NER.

2.1 Simple Recurrent Network Model

Simple Recurrent Network (SRN) introduced by [1] can be seen as an extension to normal feedforward neural networks with a recurrent weight matrix from hidden states to themselves. In our model, given an input sentence $\{x_1, ..., x_T\}$, SRN computes hidden states $\{h_1, ..., h_T\}$ and emits a sequence of

output probabilities $\{o_1, ..., o_T\}$ using the following equations:

$$\tilde{h}_t = U.x_t + b_1 \tag{1}$$

$$h_t = \text{Tanh}(W.h_{t-1} + \tilde{h}_t) \tag{2}$$

$$o_t = \text{Softmax}(V.h_t + b_2) \tag{3}$$

where $U \in \mathbb{R}^{128\times300}$, $W \in \mathbb{R}^{128\times128}$, and $V \in \mathbb{R}^{128\times9}$ are input-to-hidden, hidden-to-hidden (recurrent), and hidden-to-output weight matrices and Tanh$(.)$ is the activation.

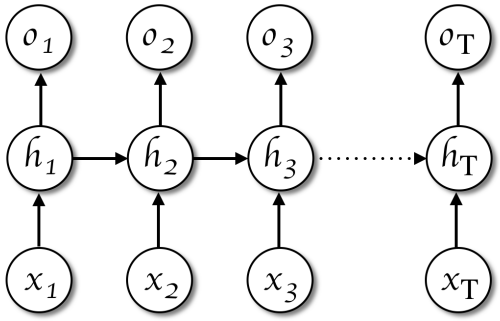


Figure 1: A Recurrent Neural Network with recurrent connection from hidden units to themselves.

2.2 Conditional Random Field Model

Conditional random fields (CRFs) [2] are discriminative undirected graphical models mostly used for labeling sequential data such as in natural language processing where it predicts sequences of labels for sequences of input samples. A regular classifier does the prediction task for a each sample without using the neighboring samples whereas the CRFs use the context for the prediction task. CRFs take a discriminative approach and model the conditional probability of *p(y|x)*.

Let *G = (V, E)* be a graph such that $Y = (Y_v)_{v\in V}$, so that Y is indexed by the vertices of G. Then (X, Y) is a conditional random field in case, when conditioned on X, the random variables Y_v obey the Markov property with respect to the graph: $p(Y_v|X, Y_w, w \neq v) = p(Y_v|X, Y_w, w \sim v)$, where $w \sim v$ means that w and v are neighbors in G.

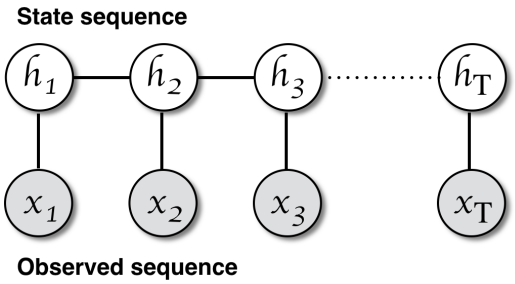


Figure 2: A graphical illustration of Conditional Random Fields. x's are the observed variables corresponding to tokens. h's are also the hidden variables that correspond to IOB labels.

Thus, the joint distribution of $p_\theta(y|x)$ is :

$$p_\theta(y|x) \propto exp(\sum_{e\in E,k} \lambda_k f_k(e, y|_e, x) + \sum_{v\in V,k} \mu_k g_k(v, y|_v, x)) \tag{4}$$

where x is a data sequence, y a label sequence, and $y|_S$ is the set of components of y associated with the vertices in subgraph S.

3 Features

For each word in each sentence, we used a 300 dimensional vector representation produced using the well-known `word2vec` toolkit. `word2vec` uses distributed representations of text produced using a neural network that captures the meaning of each word.

There exists some unseen words for which no vector representation is provided. By taking a quick look we find out that these words belong to two groups. The first group is the conjunction words and the second group mostly contains names. To handle the first group (the conjunctions such as 'a', 'an', 'and', 'of', 'to') we are using the same vector that has been specified for the word *"from"* in the word2vec toolkit. For the second group which mostly contained the names such as 'trenton', 'iurie', 'atmore', we decided to use a general vector representation produced by taking a mean over all vectors inside the word2vec corpus.

4 Environments and Toolkits

In order to implement the models introduced in section 2, we use the following libraries in python 2.7. In the following subsections, we provide the needed information for installations.

4.1 Theano

Theano [3] is a python library that allows you to define computational graphs in a symbolic format. Theano also provides automatic differentiation which is so useful for training neural networks.

```
git clone git://github.com/Theano/Theano.git
cd Theano
python setup.py install
```

4.2 Blocks

Blocks [4] is another python library built on the top of Theano that enables us to use predefined modules, stack them, and train them in a straightforward way. In this technical report, we only use Blocks to define our `Linear` and `SimpleRecurrent` modules.

```
pip install git+git://github.com/mila-udem/blocks.git@stable \
-r https://raw.githubusercontent.com/mila-udem/blocks/stable/requirements.txt
```

4.3 PyCRFSuite

Among different implementations of CRF models, `CRFSuite` [5] is a fast and efficient implementation in C++ that is largely used for labeling sequential data. `PyCRFSuite` is a python wrapper over `CRFSuite` that allows us to used it using python.

```
git clone https://github.com/jakevdp/pyCRFsuite.git
cd pyCRFsuite
python setup.py install
```

5 Results

The mean cost for the SRN model on the validation set is 0.1922. We trained the model in 300 epochs with 10 batches. The simple recurrent block has 128 nodes. The model overfits during the 300 epochs. Since we split the dataset into training and validation set, we can find where the model overfits and using the parameters with the minimum cost on the validation set. These parameters are saved in the `RNN_best_params.npy` which is then used for the test.

The CRF model was set for 5000 iterations but it was overfitted after 30 epochs. Our loss is 11712.419 and we also used L2 regularization. and `CRFmodel`.

6 References

[1] WU, XINYU, MICHAEL MCTEAR, and PIYUSH OJHA. "simple recurrent network." New Methods In Language Processing. Routledge, 2013.

[2] Lafferty, John, Andrew McCallum, and Fernando Pereira. "Conditional random fields: Probabilistic models for segmenting and labeling sequence data." Proceedings of the eighteenth international conference on machine learning, ICML. Vol. 1. 2001.

[3] Team, The Theano Development, et al. "Theano: A Python framework for fast computation of mathematical expressions." arXiv preprint arXiv:1605.02688 (2016).

[4] Van Merriënboer, Bart, et al. "Blocks and fuel: Frameworks for deep learning." arXiv preprint arXiv:1506.00619 (2015).

[5] N. Okazaki. Crfsuite: a fast implementation of conditional random fields (crfs), 2007