Named Entity Recognition Using Convolution Neural Network and Bidirectional LSTM

Majid Laali
Department of Computer Science and Software Engineering
Concordia University, Montreal, Quebec, Canada
m_laali@encs.concordia.ca

Abstract

In this report, I explain my experiments for Named Entity Recognition (NER) using Convolution Neural Network (CNN) and Bidirectional Long-Short Term Memory (LSTM). The network achieved an F1 score of 0.9698 on the Maluuba dataset which is slightly lower that the performance of Conditional Random Field (CRF) with traditional hand-crafted features (i.e. an F1 score of 0.9722).

1 Introduction

The Named Entity Recognition (NER) task is an important task for many NLP systems such as Information Extraction systems, Question Answering systems, etc. In the NER task, given a sequence of words in a sentence with the length of n, for example $X = \{x_1, x_2, ..., x_n\}$, the goal is to label all the named entity such as person names, company names, etc by labeling input words with $Y = \{y_1, y_2, ..., y_n\}$ where each y_i follows the IOB tagging convention.

A typical approach for this task is using Conditional Random Field (CRF) with traditional hand-crafted features. However, most of these features have been designed for English and it is not easy to transform these feature to other languages such as Arabic.

In this report, I used a different approach for the NER task which has less assumption about the language, and therefore it can be potentially applied to different languages. This approach, called the CNN-LSTM Model in this report, is based on the model proposed by (Chiu and Nichols, 2015) and achieve promising results on the Maluuba dataset compared to baseline system using a CRF model. The proposed model achieved an F1 score of 0.9698 while the CRF model achieved an F1 score of 0.9722. The full implementation of the model is available at the following address: https://github.com/mjlaali/ner-exercise/

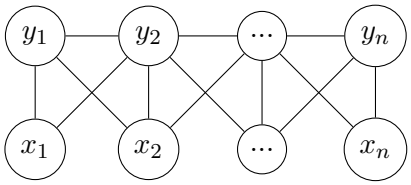

Figure 1: Caption

This report organized as follow: Section 2 describes the CRF model and the CNN-LSTM model. Then, Section 3 reports the results. Section 4 explains all the environments and tool-kits that I have used for the development of the models in details.

2 Models Architecture

In this section, I explain the baseline system and the proposed model for the NER task. First, I describe the baseline system in Section 2.1, and then explain the CNN-LSTM System in Section 2.2.

2.1 Baseline System

The baseline system is based on the Linear Chain CRF. Linear chain CRFs are discriminative models that can label a sequence with different labels. Figure 1 shows the undirected graphical model of linear chain CRFs with a context window of the radios 1.

In linear chain CRFs, $P(Y|X)$ are defined as follow:

$$P(Y|X) \propto \prod_{i=1}^{n} F_i(X) \times \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \quad (1)$$

where $F_i(X)$ is i-th feature from given input sequence.

I extracted 145 features for each x_i in dataset. Table 1 shows some of features designed for the linear CRF model. Please consult the

Feature Name	Feature Description	Example
w	String of the word	sudan → sudan
shape	Replace each character in the string of word with its group name (e.g. lowercase characters with 'L', digits with 'D', etc.)	boeing777 → LLLLLLDDD
p_i for i = 1 ... 4	the first i characters of the word	sudan → p_1 ='s', p_2 ='su', etc.
s_i for i = 1 ... 4	the last i characters of the word	sudan → s_1 ='n', s_2 ='an', etc.

Table 1: Some of features designed for the linear chain CRF

'*crf_train.py*' file at the github repository for complete list of features.

2.2 CNN-LSTML System

I adopted the neural network proposed by (Chiu and Nichols, 2015) and modified the model according to our task. Similar to (Chiu and Nichols, 2015), at the first layer, I used a CNN at the character level to embed each word to their character-level representation. More specifically, I embedded each character into a vector with 100 dimensions. Then, I applied 100 narrow convolutions with the length of three and followed by a max pooling layer over the result of the convolutions.

In contrast to (Chiu and Nichols, 2015), I concatenated the words character-level representation with pre-trained word embedding vectors to create feature vectors of words. Word embeddings are non-static, meaning that they are allowed to change during training. Next, I applied a Bidirectional Recurrent Neural Network with Long-Short Term Memory (LSTM) above the feature vectors of words. I chose the output dimension of LSTMs to be 256. Finally, the outputs of the BRNN were fed to fully connected neural network with a log-softmax activate to get log-probabilities for each tag category.

For regularization, I added a dropout layer with the probability of 0.5 between LSTM and the fully connected network. I also added L2 regularization with a weight of 0.01 for the fully connected network.

All the parameters such as the number of convolutions, the dimension of character level embedding, the output dimension of LSTM layers, etc. are the hyper parameters for the CNN-LSTM model and the proper way to set their value is to use the validation dataset. I plan to search over in the hyper parameter space to find the best value for them in future work.

In the rest of this section, I will provide the justifications for my main decisions in designing the proposed neural network.

2.2.1 Character Level CNN

The character patterns of each word have important information for the NER task. For example, most of the hand-crafted features designed for the NER task (e.g. the proposed features for the CRF model) use character patterns of words. This motivates us to use a character level CNN to extract important patterns for the NER task from words.

2.2.2 Combining Word Embedding and CNN Outputs

While the character level CNN extract surface features, word embedding vectors contain valuable semantic features about the word. Therefore, I concatenated word embedding vectors with the outputs of character level CNN to create a better feature representation for each word.

2.2.3 BRNN

The CRF model will label a sequence of words with labels that maximizes the probability of $P(Y|X)$. This means that for labeling each word, the CRF model uses the information of previous words and also next words. To mimic similar behavior in the CNN-LSTM model, I used a BRNN network. BRNN network consists of two RNN network: the forward RNN and the backward RNN. The forward RNN (similar to the forward pass in CRFs) contains the information of previous words and the backward RNN (similar to the backward pass in CRFs) provides the information of next words for labeling each word.

2.2.4 Treatment of Unknown Words

To obtain the word embedding of words, I used the provided pre-trained word embedding vector. These vectors have a dimension of 300 and they were trained on the Google news corpus. However, in the dataset, there are some words, called unknown words, that did not occur in the Google

Dataset Name	# Instances
Training dataset	810
Development dataset	90
Test dataset	100

Table 2: The number of sentences of different datasets

news corpus, and therefore, the vector representation of them are not available in the pre-trained vectors. To handle unknown words, I extended the dimension of word embedding vectors to 301, where the first dimension is either 1 (known word) or 0 (unknown word), and initialize word embedding vectors to 301 zeros for unknown words. I did not fix the word embedding vectors and allowed them to be tuned during the training phase. So, the model can learn a good representation for unknown word based on the training dataset[1].

3 Experimental Results

The Maluuba dataset contains 1000 sentences with the annotation of four entities: KEYWORD, NEWSTYPE, PROVIDER and SECTION. According to my analysis the maximum length of sentences is 29 words and the maximum length of words is 19 in the dataset. I divided the dataset into three parts: Training Dataset, Development Dataset and Test Dataset. Table 2 provides the number of sentences of each dataset.

I used both Training dataset and Development dataset to train the CRF model. The CRF model contained an L2 regularization with the coefficient of 1e-3. For CNN-LSTM model, I learned the model on the training dataset and used the development dataset for early stopping. I used the cross-entropy as the loss function for the model and used the Adam optimizer with the learning rate of 0.001, $\beta_1 = 0.9$, $beta_2 = 0.999$ and $\epsilon = 1e-8$ to obtain the best parameters for the model. The batch size was 16 and the best parameters of the model have been found after 39 epochs.

I tested both models on the Testing dataset. Table 3 and Table 4 show the results of the CRF model and the CNN-LSTM model on the Maluuba dataset respectively.

[1]With respect to the dataset size, it is very hard for the model to create a meaning full word embedding vectors for unknown words

4 Environments Setup

All the code of this work is in Python 3.5 and available in the `https://github.com/mjlaali/ner-exercise/`. To train and test models, the following packages should be installed:

1. **Tensorflow 0.10:** `https://www.tensorflow.org/`
2. **Keras:** `https://keras.io/`
3. **Python-crfsuite:** `https://python-crfsuite.readthedocs.io/en/latest/`

4.1 Train models

To train the models, clone the github repository and create a folder called *'data'*. Then, put two files with the name of *'train.txt'* and *'test.txt'*. These two files contain IOB tag information of the sentences. Please also copy the *'wordvecs.txt'* file in 'data' folder. Finally, run the *'cnn_lstm_train.py'* or *'crf_train.py'* to train the CNN-LSTM model or the CRF model, respectively.

4.2 Test trained models

To test the learned model, please run either *'test_cnn_lstm_model.py'* or *'test_crf_model.py'* scripts.

References

[Chiu and Nichols2015] Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Tags	Precision	Recall	F1-score	Support
B-KEYWORDS	0.95833	0.97183	0.96503	71
I-KEYWORDS	0.98165	0.98618	0.98391	217
B-NEWSTYPE	0.97101	0.95714	0.96403	70
I-NEWSTYPE	0.98611	1.00000	0.99301	71
B-PROVIDER	0.98305	0.95082	0.96667	61
I-PROVIDER	0.97674	0.92308	0.94915	91
B-SECTION	1.00000	0.94737	0.97297	19
I-SECTION	0.94444	0.94444	0.94444	18
avg/total	0.97718	0.96764	0.97223	618

Table 3: The results of the CRF model on the Maluuba dataset

Tags	Precision	Recall	F1-score	Support
B-KEYWORDS	0.97183	0.97183	0.97183	71
I-KEYWORDS	0.96000	0.99539	0.97738	217
B-NEWSTYPE	0.98551	0.97143	0.97842	70
I-NEWSTYPE	1.00000	1.00000	1.00000	71
B-PROVIDER	0.96667	0.95082	0.95868	61
I-PROVIDER	0.96512	0.91209	0.93785	91
B-SECTION	1.00000	0.94737	0.97297	19
I-SECTION	0.94118	0.88889	0.91429	18
avg/total	0.97094	0.96926	0.96982	618

Table 4: The results of the CNN-LSTM model on the Maluuba dataset