Neural Networks for Named Entity Recognition

Ahmed ElSheikh - 1873337

1 Introduction

Named Entity Recognition (NER) task is a word-level classification task, such that, given a sequence of tokens $x =< t_1, t_2, ..., t_T >$, the sequence is tagged and an output sequence is produced $y =< y_1, y_2, ..., y_T >$. Where every token t_i is drawn from a constructed vocabulary V and each label y_i belongs to one of the named entities **ORG**anization, **PER**son, **LOC**ation, **O**ther.

2 Network Architectures [1]

2.1 BiLSTM Model

Long Short Term Memory (LSTM) networks proved to be very effective in word-level tagging tasks due to their ability to fetch relations between tokens in a given sequence. They are known for capturing context from all preceding tokens given the current token. An extension of them are Bidirectional LSTM (BiLSTMs) networks. These read the same sequence twice, taking into account past and future tokens, hecne, capturing a richer context compared to LSTMs.

2.2 BiLSTM-CRF Model

BiLSTMs tag sequences without taking into consideration the previous label, treating the labels independently. Conditional Random Fields (CRF) (Sutton and McCallum, 2012) leverage more information by modeling the interdependence of labels, i.e. it takes into consideration the sequence of previous labels in order to predict the current one. This leads to CRFs predicting a valid sequence of tags not just predict a correct tag for a single token.

2.3 Pretrained Word Embeddings

Initialization of weights -more specifically, embeddings for networks that tackle NLP tasks- in Neural Networks can make a notable difference in performance. Consequently, it was decided to move from random initialization of word embeddings to use fastText (Bojanowski et al., 2017) pretrained word embeddings. This would help the model pick up better semantic signals, converge faster, and classify tags better compared to training our embeddings layer from scratch.

2.4 POS Embeddings

Part-of-speech (POS) tagging often increases the model's performance on different NLP tasks. Since named entities are often nouns, POS tags can offer the network this extra piece of information and consequently improve its performance. Incorporating POS tags along with good pre-trained word embeddings allows the model to take into consideration both syntactic & semantic structure of the English sentences. A pre-trained POS tagger[2] was used and the POS embeddings was concatenated with words embeddings before BiLSTM layer.

3 Experimental Setup

3.1 Data Preprocessing

All tokens in the dataset were lowered and extra whitespaces were stripped. The overall dataset was split into train-dev-test. All ware made up of only English sentences. To avoid any loss of contextual information from sentences, the data was batched and padded as per the lengthiest sequence in the batch before being fed to the model. Further insights about dataset can be found in Appendix A.

3.2 Dropout Training

A dropout layer was added after both the POS & word embedding layers, which drops random tokens from the input sequence preventing the model from depending on certain tokens or pos tags to

[1] Models discussed in this section are inspired from the work of (Lample et al., 2016)

[2] Available from NLTK

make its prediction. This approach was pioneered by (Srivastava et al., 2014) as it helps the model generalize better.

3.3 Training

Models were trained for 10 epochs where the sentences were padded per batch dynamically. A batch size of 128 was used. Since RNNs are known for exploding gradient problem, gradient clipping was deployed with a clipping value of 5.0 to combat it. RNNs are also known for being extremely prone to over-fitting so Early Stopping (ES) was deployed, where training is stopped if the model did not show any improvement in performance over the validation dataset for 5 consecutive epochs. Similarly, after 2 consecutive 'patience' epochs, the model's learning rate the learning rate is reduced by 10%. This is an implementation of the intuition that maybe the learning rate was not low enough to escape a saddle point/local minimum. [3].

4 Experiments

4.1 Effect of adding CRF layer

We started with a BiLSTM model as the baseline model for all of our experiments. This model achieved a Macro F1 score of 79.8. Adding a CRF layer on top of it pushed that F1 score to 89.7. This increase in performance is due to the fact that CRFs consider the inter-dependency of tags in a given sequence, and in NER for example, people's names can span multiple tokens and where BiLSTM will probably only tag the first name, CRFs will know how to tag both tokens as 'PER'.

4.2 Use of pretrained Embeddings

As discussed in Section 2.3, initialization of weights has a significant effect on the overall model's performance. Thus, pretrained word embeddings were used to avoid training our embeddings layer from scratch. The pretrained word embeddings contain better info about words and their context. Fasttext was used, as it breaks down words into subwords which helps in generating better word embeddings for OOV tokens. This Improved the BiLSTM CRF model's F1 score from 89.7 to 90.3.

[3]Based on the theory of GD, a minimum can only be reached when the learning rate approaches 0. Otherwise, with a large learning rate, the model's performance is expected to bounce around the minimum – not reaching it.

4.3 Hyperparameters

Increasing the model's complexity results in an increased performance up to a certain point where any further increase will lead to over-fitting on the data. The model's complexity was increased in different ways starting with using BiLSTMs instead of unidirectional LSTMs. This showed an increase from 88.5 to 89.7. Stacking BiLSTMs vertically increased performance from 90.3 to 90.9, and finally, increasing the BiLSTM's hidden dimension from 256 to 512 increased performance from 90.9 to 91.1.

Two optimizers were tested: SGD & Adam (Kingma and Ba, 2014). Adam showed faster convergence and better results as it uses Momentum and an Adaptive Learning Rate for every set of weights in the model, unlike SGD which takes more time to converge to a worse solution. This is due to the fact that SGD performs a parameter update for each training example. Optimizing our model using Adam showed a performance increase from 83.8 to 90.3.

The performance increase that resulted from the tuning of only one of the hyperpaprameters as described above can be seen as very small on its own, but, as we show in Table 4, training a model where we combined all the above tuning resulted in a significant performance increase.

4.4 Integration of PoS Embeddings

Following the same intuition from section 2.4, a POS2VEC model was trained for 30 epochs in order to generate PoS embeddings. These are used to initialize the NER model's POS embedding layer. The output of that POS BiLSTM layer is concatenated with the output of the (pretrained) word BiLSTM layer before passing it to the Classifier layer.

Contrary to the hypothesis we presented in Section 2.4, the incorporation of PoS Embeddings did not produce the best model. This might be due to the limited amount of data used to train the POS embedding vectors and the severe class imbalance found in the training dataset.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Diederik P 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*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Charles Sutton and A. McCallum. 2012. An introduction to conditional random fields. *Found. Trends Mach. Learn.*, 4:267–373.

A Data Analysis

Data set	Number of sentences
Train	$100K$
Train	$14K$
Train	$15K$

Table 1: Train-dev-test split used in the experiments

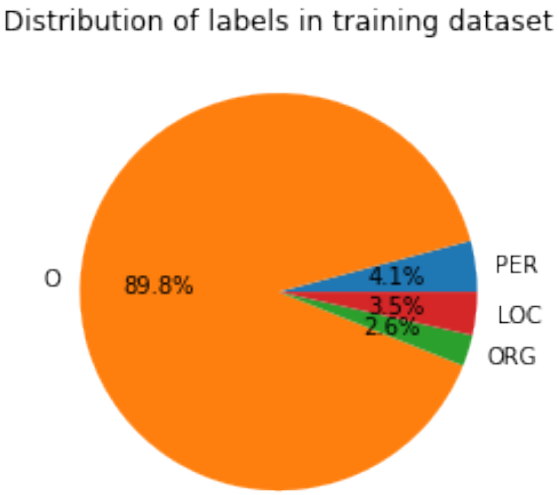

Figure 1: Distribution of labels in the training data set

Model is more likely to predict a Named Entity as the '**O**ther' class due to the massive class imbalance in the dataset, where almost 90% of the labels are of the '**O**ther' class. This effect can be seen in the confusion matrix (Figure 3) as when the model mis-classifies ORG & LOC classes, it confuses them with the 'O' class the most.

B Grid Search Space

Parameter	Space
Bidirectional	False, True
Hidden Neurons	128, 256, 512
LSTM Layers #	1, 2
PreTrained Embeddings	False, True
POS PreTrained Embeddings	False, True
CRF	False, True
Optimizers	SGD, Adam

Table 2: Grid Search Space

Candidate values used during grid search for each hyperparameter. FastText embeddings were used. Approximately $71K$ embedding vectors were loaded from Fasttext while the remaining $\approx 19K$ were randomly instantiated.

C Best Hyper-parameters

Hyper-parameters	Value
Batch Size	256
Embeddings Dim	300
Bidirectional	True
Hidden Neurons	512
Dropout	0.5
Num of Layer	2
PreTrained Embeddings	FastText
CRF	True

Table 3: Best Model's Hyperparameters

D Model Summary

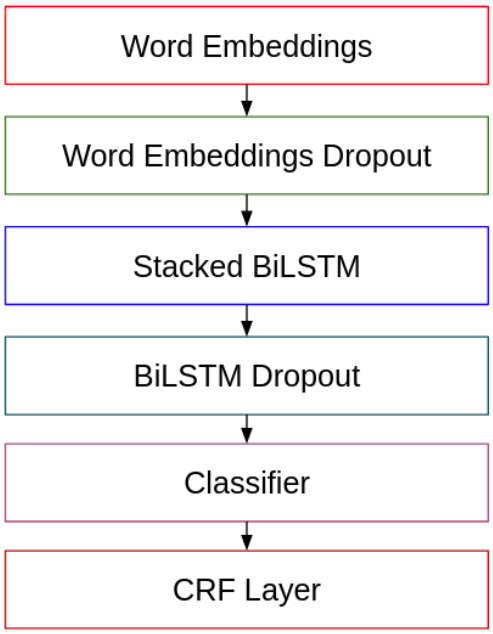

Figure 2: Best Model's Architecture

E Confusion Matrix

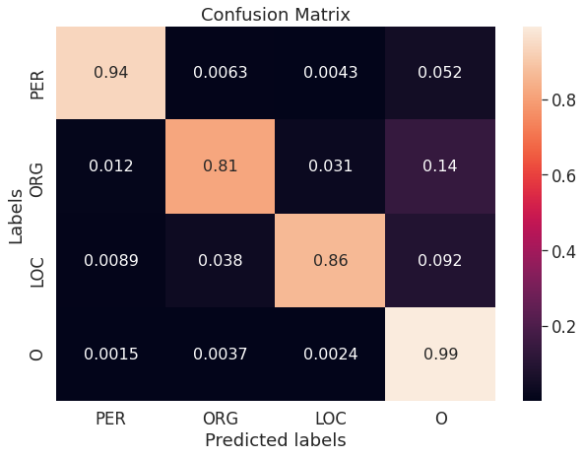

Figure 3: Best Model Confusion Matrix

Model struggles the most in classifing 'Organization' entities compared to other labels. This is due to that most of organizations contain Location names in them e.g. Washington Times-Herald, Boston Daily Advertiser, New York Times. This clearly confuses the model

F Models' Performances

Models	Score
UnStacked Architecture	Macro F1
$BiLSTM(baseline)$	79.8
$BiLSTM\_CRF\_Fasttext\_SGD$	83.8
$LSTM\_CRF$	88.5
$BiLSTM\_CRF$	89.7
$BiLSTM\_CRF\_Fasttext\_128\_Hidden$	89.9
$BiLSTM\_CRF\_Fasttext$	90.3
Stacked Architecture	
$LSTM\_CRF\_Fasttext$	89.1
$BiLSTM\_CRF\_Fasttext\_512\_Hidden\_POS$	90.7
$BiLSTM\_CRF\_Fasttext$	90.9
$BiLSTM\_CRF\_Fasttext\_512\_Hidden$	**91.1**

Table 4: Macro F1_scores (in percentage)

F1 Score	Macro Precision Per class			
_	LOC	ORG	PER	O
0.911	0.902	0.847	0.962	0.989

Table 5: Stacked BiLSTM CRF Fasttext 512 Hidden Dim (best model) Macro F1 scores, Marco Precision Per class