Bidirectional LSTM-CRF Models for Sequence Tagging

Name : Archana Kalburgi

Course : CS 541-B Artificial Intelligence

Email ID : akalburg@stevens.edu

Table of Contents

Abstract

Named Entity Recognition (NER) is a problem in Natural Language Processing (NLP), which involves locating and classifying names (people, places, organizations, etc.) that appear in unstructured text. Many of the NLP applications that use this problem are dealing with use-cases such as machine translation, information retrieval, chatbots, etc. In a nutshell, for each training sentence, I would like to predict what "tag" each token will have.

Introduction

NLP tasks have traditionally involved tagging sequences, chunking, and recognition of named entities. These tasks have been the focus of research for several decades. It is possible to use the output of taggers for downstream applications. For example, a named entity recognizer trained on user search queries can help identify which segments of text are products, thus triggering certain product ads. As another example, such tag information can be used by a search engine to find relevant web pages

Methods

In this paper the authors have proposed a variety of **Long Short-Term Memory (LSTM)** based models for sequence tagging. These models include **LSTM networks**, **bidirectional LSTM (BI-LSTM)** networks, **LSTM with a Conditional Random Field (CRF)** layer (LSTM-CRF) and **bidirectional LSTM with a CRF layer (BI-LSTM-CRF)**.

The tagging is achieved by applying a bidirectional LSTM CRF (denoted as BI-LSTM-CRF) model to **NLP benchmark sequence tagging data sets**. The **bidirectional LSTM** component enables the BILSTM-CRF model to efficiently use both past and future input features. And the **CRF layer** enables it to use sentence level tag information. The BI-LSTMCRF model can produce state of the art accuracy of 97.55%

on POS, 94.46% on chunking and 90.10% on NER data sets. In addition, it is robust and has less dependency on word embedding as compared to previous observations.

The progress of the project and the source code can be tracked within this **git repository**.

Dataset details

The dataset used in this project is an annotated GMB(Groningen Meaning Bank) corpus for entity classification with enhanced and popular features by Natural Language Processing applied to the data set which is built specifically to train the classifier to predict named entities such as name, location, etc.

Tags

The tags used in datasets are in the form of "B-geo", "-org", "O", the **I-prefix** indicates that the tag is inside a chunk (i.e. a noun group, a verb group etc.); the **O-prefix** indicates that the token belongs to no chunk; the **B-prefix** indicates that the tag is at the beginning of a chunk that follows another chunk without O tags between the two chunks

Statistics

The dataset has 1,048,575 words and the target column is titled “tag”.

Tagged entities

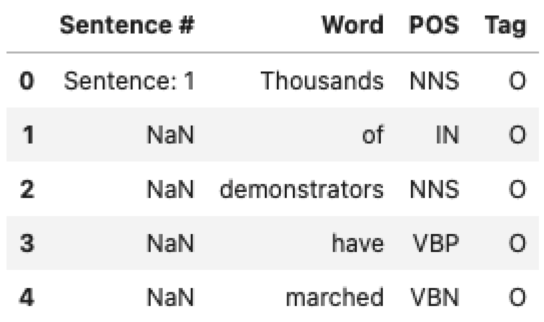
The number of tagged entities are as detailed below:
'O': 1146068', geo-nam': 58388, 'org-nam': 48034, 'per-nam': 23790, 'gpe-nam': 20680, 'tim-dat': 12786, 'tim-dow': 11404, 'per-tit': 9800, 'per-fam': 8152, 'tim-yoc': 5290, 'tim-moy': 4262, 'per-giv': 2413, 'tim-clo': 891, 'art-nam': 866, 'eve-nam': 602, 'nat-nam': 300, 'tim-nam': 146, 'eve-ord': 107, 'per-ini': 60, 'org-leg': 60, 'per-ord': 38, 'tim-dom': 10, 'per-mid': 1, 'art-add': 1

The entity tag used in this dataset is as follows:

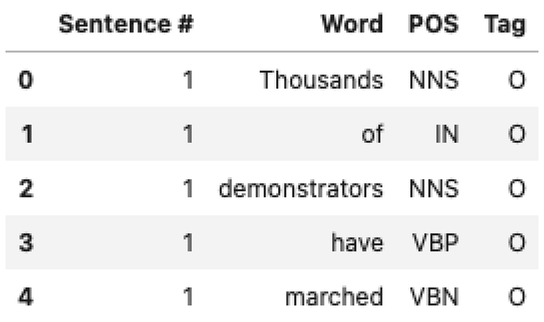
Tag	Meaning	Example
geo	Geography	Britain
org	Organization	IAEA
per	Person	Thomas
gpe	Geopolitical entity	Indian
tim	Time	Wednesday
art	Artifact	Pentastar
eve	Event	Armistice
nat	Natural phenomenon	H5N1

Data preprocessing and visualization

Majority of the data was clean. As a part of data preprocessing I have replaced the missing values with the last valid observation by propagating it forward to the next valid backfill. Following screenshots depict the difference between before and after replacing null values.

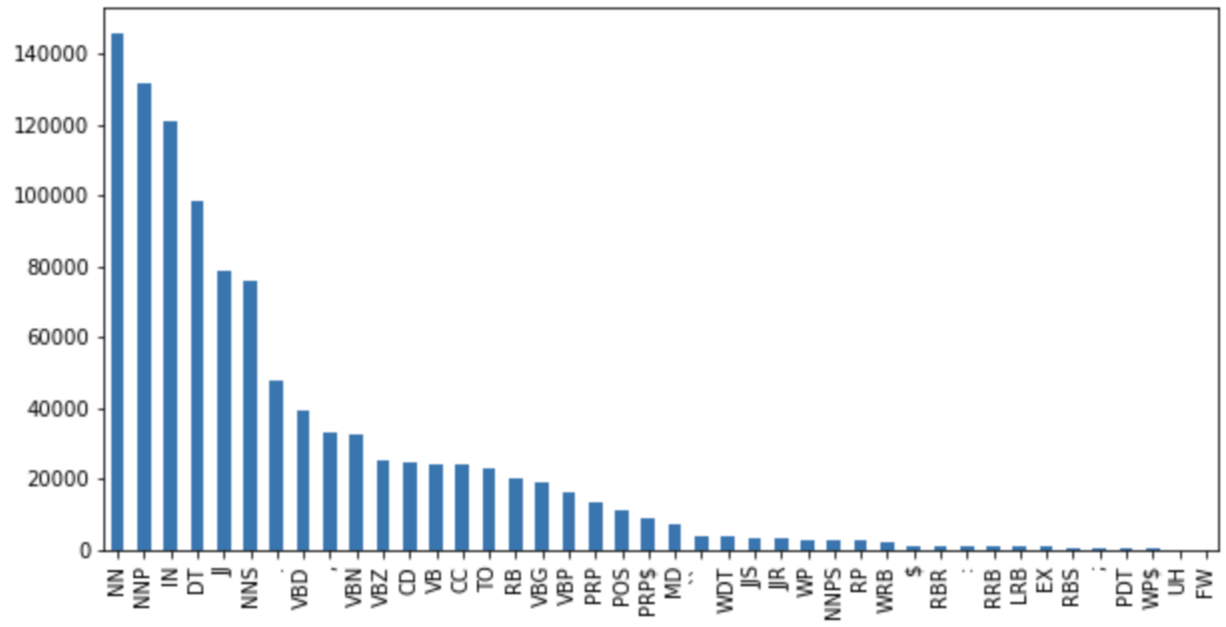


	Sentence #	Word	POS	Tag
0	Sentence: 1	Thousands	NNS	O
1	NaN	of	IN	O
2	NaN	demonstrators	NNS	O
3	NaN	have	VBP	O
4	NaN	marched	VBN	O

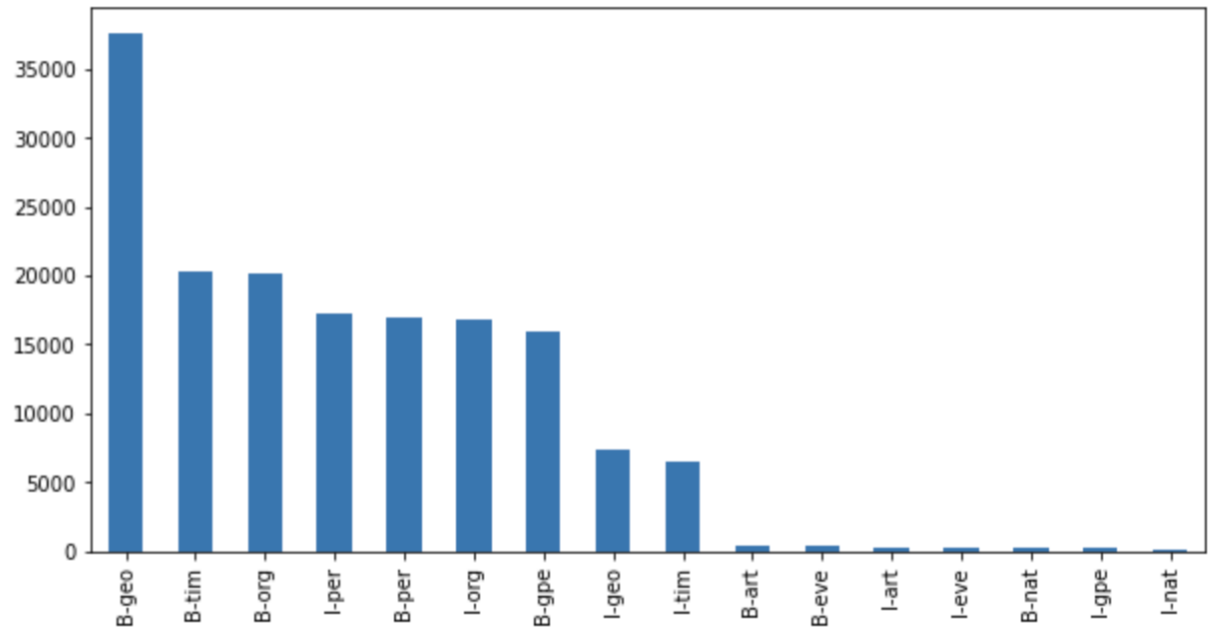


	Sentence #	Word	POS	Tag
0	1	Thousands	NNS	O
1	1	of	IN	O
2	1	demonstrators	NNS	O
3	1	have	VBP	O
4	1	marched	VBN	O

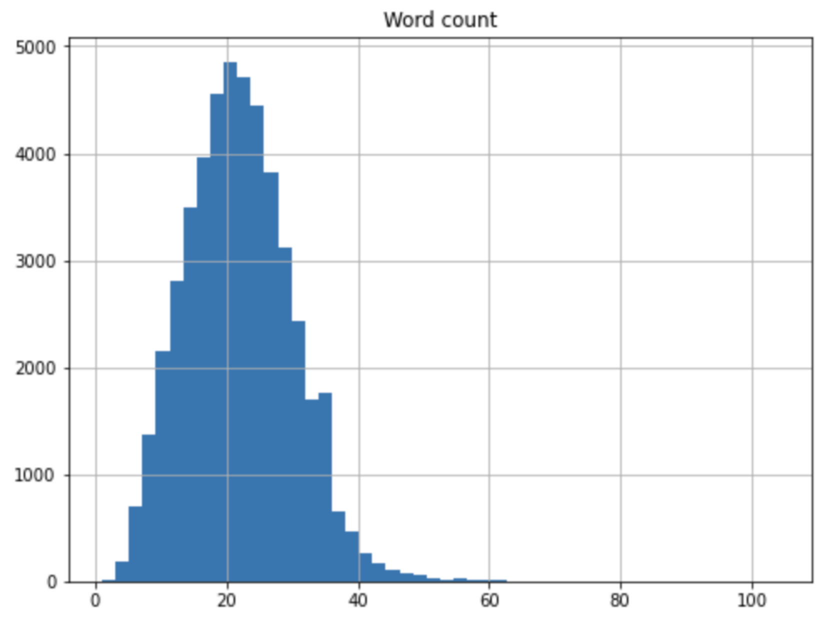
In the dataset there are 42 unique POS tags and 17 unique NER tags. The following plots depicts the distribution of the tags



Distribution of POS tags



Distribution of NER tags

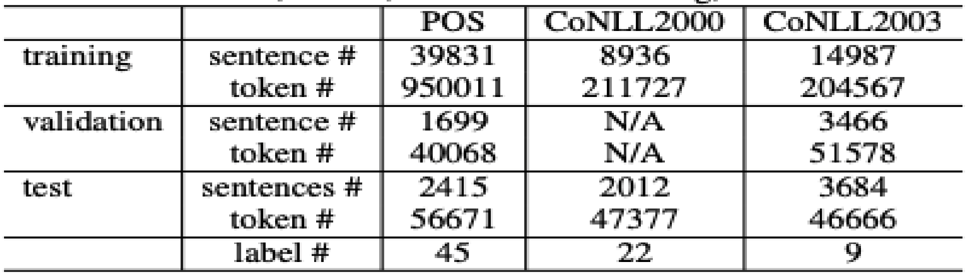


Distribution of the words

Train, test and validation sets

The following table shows the size of the sentences, tokens, and labels for training, validation and test sets respectively.

Table 1: Size of sentences, tokens, and labels for training, validation and test sets.



		POS	CoNLL2000	CoNLL2003
training	sentence #	39831	8936	14987
	token #	950011	211727	204567
validation	sentence #	1699	N/A	3466
	token #	40068	N/A	51578
test	sentences #	2415	2012	3684
	token #	56671	47377	46666
	label #	45	22	9

More details about the data can be found here.

Tools and technologies

Software/ library packages

This project is implemented using python 3.9.1. And I will be making use of the following library package.

1. Keras 2.2.4
2. Pickle
3. Regular expression: re
4. Pandas library
5. Numpy library
6. Matplot library for visualization
7. Scikit library
8. Tensorflow

Hardware setup

In this project I am using GPU 1.13.1. The models will be run on my local machine where the data to train the model are stored.

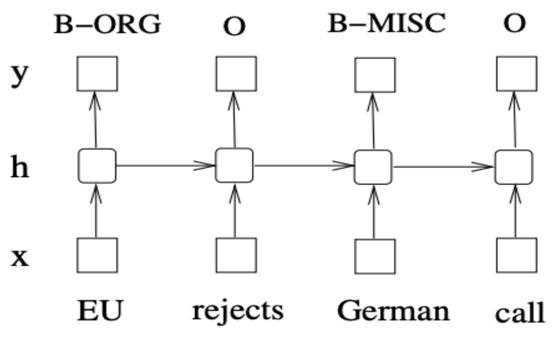
Experiments

The models used in this paper are LSTM, BI-LSTM, CRF, LSTM-CRF and BI-LSTM-CRF. The model performance and implementation details are discussed below in short.

Neural network architecture

RNN

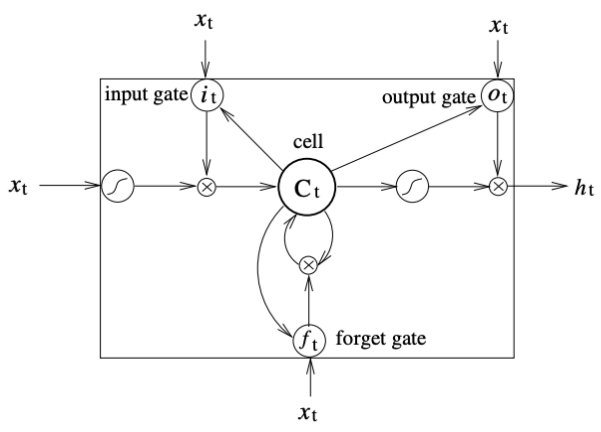
Following figure shows the RNN structure. which has an input layer *x*, hidden layer *h* and output layer *y*. In the named entity tagging context, x represents input features and y represents tags. Figure 1 illustrates a named entity recognition system in which each word is tagged with another (O) or one of four entity types: Person (PER), Location (LOC), Organization (ORG), and Miscellaneous (MISC). The sentence *"EU rejects German call to boycott British lamb ."* is tagged as B-ORG O B-MISC O O O B-MISC O O, where B-, I- tags indicate beginning and intermediate positions of entities.



A simple RNN network

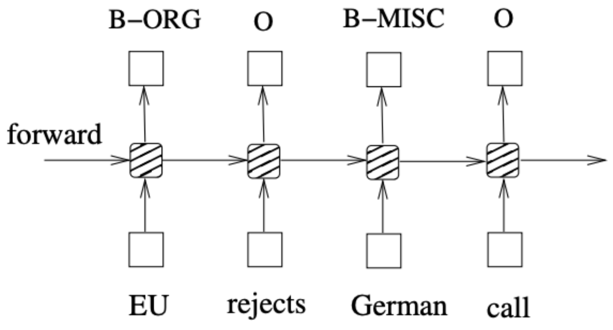
LSTM

In this paper Long Short Term Memory is being applied for sequence tagging. Long ShortTerm Memory networks are the same as RNNs, except that the hidden layer updates are replaced by purpose-built memory cells. As a result, they may be better at finding and exploiting long range dependencies in the data. Following figure shows a single LSTM memory cell.



Long-short term memory cell

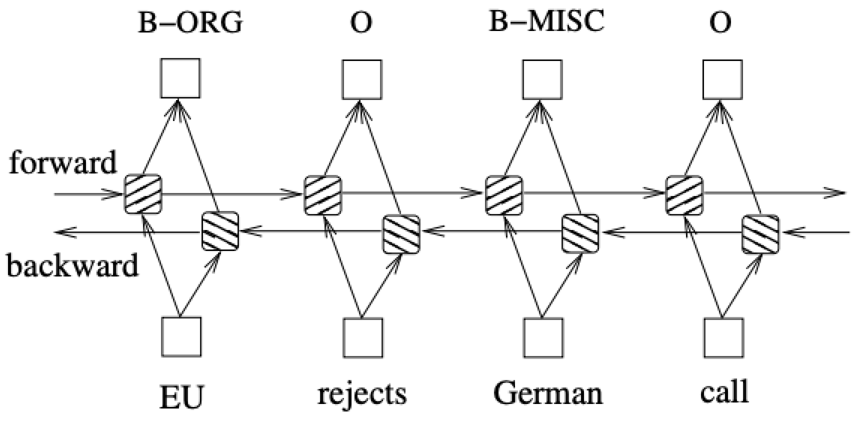
Following is a LSTM sequence tagging model which employs the above LSTM memory cells.



A LSTM network

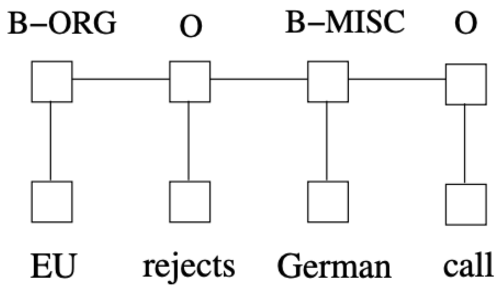
Bidirectional LSTM

A bidirectional LSTM network is trained using backward propagation through time. The forward and backward passes over the unfolded network over time are carried out in a similar way to regular network forward and backward passes, except that the hidden states are unfolded for all time steps.



Bidirectional LSTM network

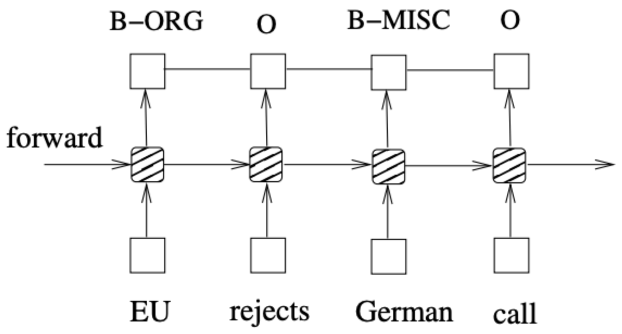
CRF networks



A CRF network

LSTM-CRF networks

The LSTM network and a CRF network are combined to form a LSTM-CRF model as shown in the following figure.

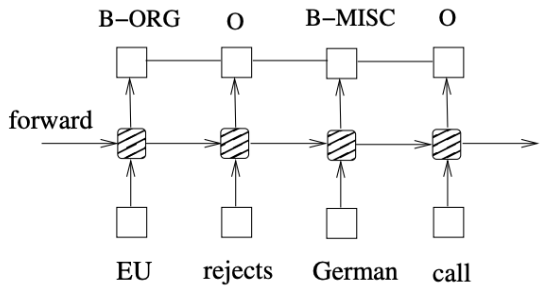


A LSTM-CRF model

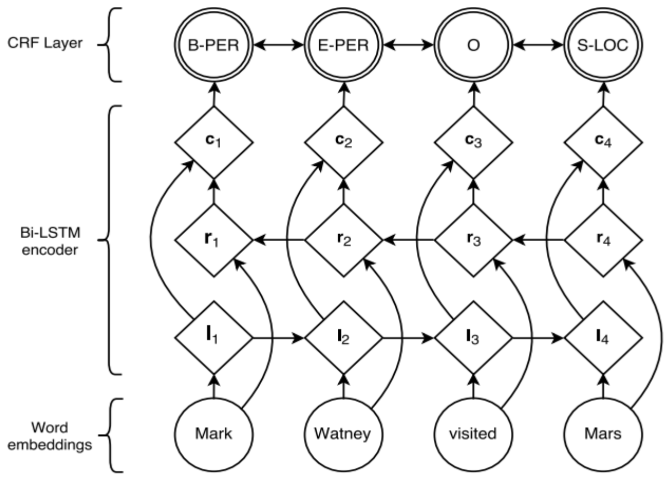
This network can efficiently use past input features via a LSTM layer and sentence level tag information via a CRF layer. A CRF layer is represented by lines which connect consecutive output layers. A CRF layer has a state transition matrix as parameters. This layer can be effectively used to the past and the future tags to predict the current tags similar to bidirectional LSTM networks where past and the future features are used to make current predictions.

BI-LSTM-CRF networks

Similar to a LSTM-CRF network, a bidirectional LSTM network in a CRF network to form a BI-LSTM-CRF network (Fig. 7). In addition to the past input features and sentence level tag information used in a LSTM-CRF model, a BILSTM-CRF model can use the future input features. The extra features can boost tagging accuracy as we will show in experiments.



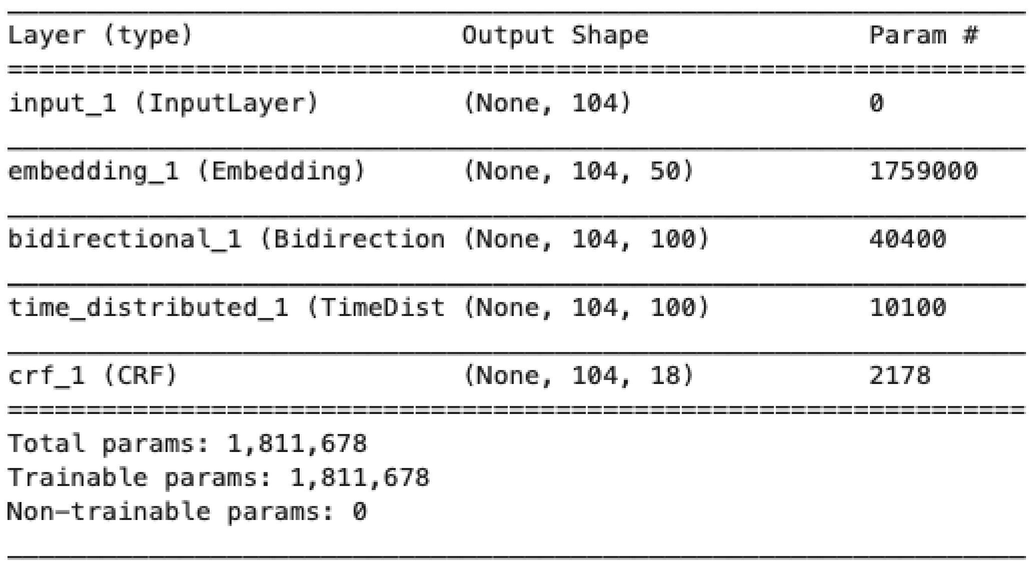
A BI-LSTM-CRF model.



Main architecture of the network. Word embeddings are given to a bidirectional LSTM. *li* represents the word *i* and its left context, *ri* represents the word i and its right context. Concatenating these two vectors yields a representation of the word *i* in its context, *ci*.

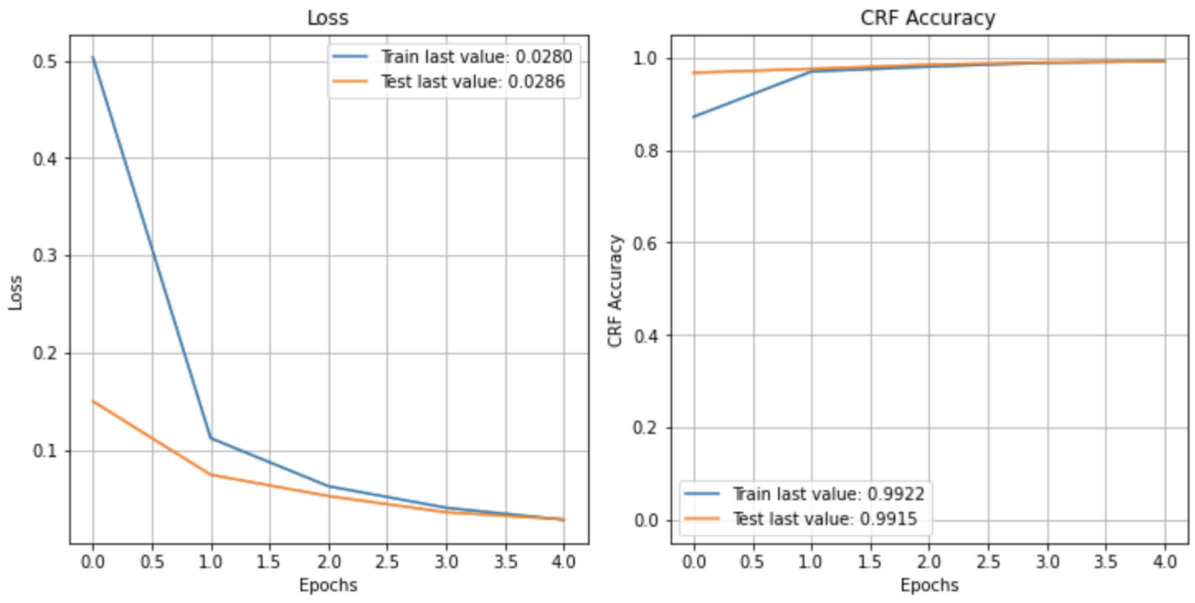
Results

A BI-LSTM-CRF model was trained with a total of 1,811,678 parameters. Following is the model summary.

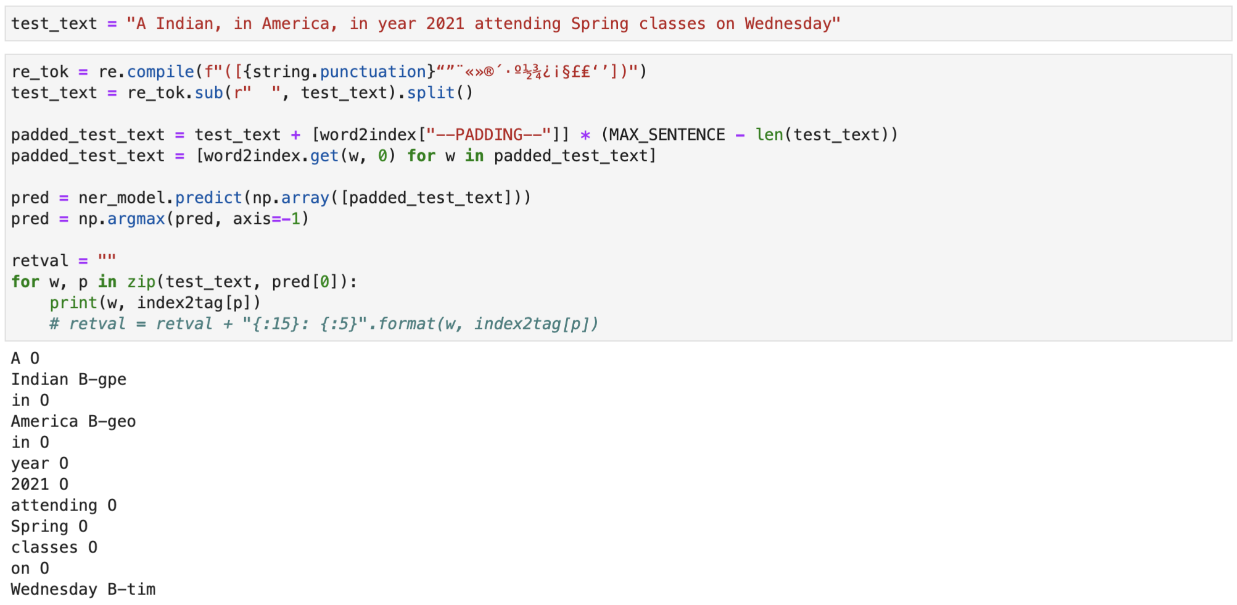


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 104)               0
_________________________________________________________________
embedding_1 (Embedding)      (None, 104, 50)           1759000
_________________________________________________________________
bidirectional_1 (Bidirection (None, 104, 100)          40400
_________________________________________________________________
time_distributed_1 (TimeDist (None, 104, 100)          10100
_________________________________________________________________
crf_1 (CRF)                  (None, 104, 18)           2178
=================================================================
Total params: 1,811,678
Trainable params: 1,811,678
Non-trainable params: 0
_________________________________________________________________
```

After trying various combinations of parameters with the final model I was able to achieve an accuracy of 99.15% on the validation data at the 5th epoch. The train, test loss and accuracy can be studied in the following plot.



The model was able to predict the tags on the the test data as can be seen in the following snapshot



```
test_text = "A Indian, in America, in year 2021 attending Spring classes on Wednesday"
```

```
re_tok = re.compile(f"([{string.punctuation}“”¨«»®´·º½¾¿¡§£₤‘’])")
test_text = re_tok.sub(r"  ", test_text).split()

padded_test_text = test_text + [word2index["--PADDING--"]] * (MAX_SENTENCE - len(test_text))
padded_test_text = [word2index.get(w, 0) for w in padded_test_text]

pred = ner_model.predict(np.array([padded_test_text]))
pred = np.argmax(pred, axis=-1)

retval = ""
for w, p in zip(test_text, pred[0]):
    print(w, index2tag[p])
    # retval = retval + "{:15}: {:5}".format(w, index2tag[p])
```

```
A O
Indian B-gpe
in O
America B-geo
in O
year O
2021 O
attending O
Spring O
classes O
on O
Wednesday B-tim
```

Conclusion

With regard to POS, chunking and NER data sets, the model was able to provide close to state of the art accuracy (or close to it). Additionally, the model is robust and does not rely heavily on word embedding. Without resorting to word embedding, it can achieve accurate tagging.

References

1. [Hochreiter and Schmidhuber1997] S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. Neural Computation, 9(8):1735-1780.
2. [Graves et al.2013] A. Graves, A. Mohamed, and G. Hinton. 2013. Speech Recognition with Deep Recurrent Neural Networks. Arxiv.
3. [Finkel et al.2005] J. R. Finkel, T. Grenager, and C. Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. Proceedings of ACL.