Project 2

Neural Conditional Random Fields for Named Entity Recognition

Prerna Trushar Bhavsar PID: A59002324

Abstract

In this project, I have implemented a neural Conditional Random field (CRF) BiLSTM model for performing Named Entity Recognition (NER) on the CoNLL dataset. I also analyse the results of the same using various performance metrics. Further, I evaluate and compare my implementation's performance with the Baseline BiLSTM Tagger for the same task.

1 Introduction

Sequence Tagging in Natural Language Processing consist of various methods like chunking, POS (Part-of-speech) tagging, named entity recognition. Named Entity Recognition is a task in Natural Language Processing that deals with identifying and categorizing entities in text. Entity can be defined as any word or phrase defining the same thing. For example, Location is an entity for any word that defines a place, city, country, etc. NER can be used for information extraction. It helps us answer questions that are posed in the real world. By being able to identify and tag entities in a sentence, NER can help us analyze various high-level details of a sentence or text. It can be used to get a overview of the theme or topic of the data. NER can be used in search and recommendation engines, content classification, customer support, etc. There are various models in the literature of NLP that deal with the implementation of NER. Some of these models include Hidden Markov Model, Maximum Entropy Markov Model, and Conditional Random Fields. In this project, I implement a BiLSTM model with neural Conditional Random Field (CRF). I then compare its results with a Baseline BiLSTM model for the task of Named Entity Recognition (NER).

2 Method

2.1 BiLSTM Model for NER

The baseline model for the project uses a Recurrent Neural Network for NER. This model implements a bi-directional LSTM network for the task of named entity recognition. In a bi-directional LSTM the hidden states are computed in both the forward and backward direction. LSTM networks are a variation of RNN models which can deal with the issues of vanishing or exploding gradients. RNNs in general help capture long distance information in the input space by using a memory unit.

$$h_t = tanh(W_{ih}x_t + b_{ih} + W_{hh}h_{t-1} + b_{hh}) \quad (1)$$

The input to the network is a labeled sentence in the form $\{x,y\}$, where x is the word vector and y is the tag vector. The model predicts the tag for the time-step t by performing a logistic regression on the inputs and weights, and finding the distribution over the tags by applying a softmax.

$$f_t = W_{out}h_t + b_{out} \quad (2)$$

$$p(y_t{}^*|X;\theta) = softmax(f_t) \quad (3)$$

The model applies a negative loss likelihood loss function over the model parameters θ in order to learn the parameters than best minimize the loss.

$$loss = \sum^{x,y^*}\sum_{t=1}^{T} -logp(y_t{}^*|X;\theta) \quad (4)$$

2.2 BiLSTM CRF Model for NER

Conditional Random Fields (CRF) is one of the most commonly used approachs for NER. A Linear CRF predicts the tag for the present word depending on the tag of the previous word. Although the baseline BiLSTM model gives good feature representations, the prediction of the tags independently

without any context using these features gives sub-optimal results. In order to gain better results in sequence labelling it is essential to consider the relations between neighbouring pairs of words and then try to jointly decode the best sequence of tag predictions. Therefore, we can use CRF in the model to perform tag classification instead of independently predicting tags.

CRF is a globally normalized, discriminative model. CRF defines a conditional probabilities over all possible label sequences *y* given *x*, where *x* is the input sequence vector.

$$p(y|x, W, b) = \frac{\Pi_{i=1}^{n} \psi(y_{i-1}, y_i, x)}{\sum_{y^i \in Y(x)} \Pi_{i-1}^{n} \psi(y'_{i-1}, y'_i, x)} \quad (5)$$

where $\psi(y', y, x) = exp(W^T{}_{y',y} x_i + b_{y',y})$.

We minimize the negative loss likelihood to learn the model parameters θ i.e the weights W and biases b.

$$loss = \sum^{x^*,z} -log(p(y|x, W, b)) \quad (6)$$

For decoding, we use the viterbi algorithm to find the best label sequence by finding the sequence with highest conditional probability.

$$y^* = argmax_{y \in Y(x)}(p(y|x, W, b)) \quad (7)$$

where $Y(x)$ denotes all possible label sequences for *x*.

For the implmentation, we use a dynamic program to calculate the denominator or *z(x)*. This is also called a partial function and is expensive to calculate as it iterates over all possible tag sequences. In this forward algorithm dynamic program we store the values of the scores at each time-step *t* and the use this to calculate the sum of the scores by passing over the entire sequences.

For decoding, we then use viterbi algorithm which has a similar recurrence structure as the forward algorithm. This helps us find the best tag sequence and best score over all possible set of sequences.

3 Results and Discussion

3.1 Model Results

In order to evaluate and compare the results of the models, I trained both the models for *30 epochs* and *dropout = 0.3*. The Figures 1-3 show the precision, recall and F1 score curves for both the models. The precision demonstrates that the BiLSTM CRF achieved better values than those of the baseline BiLSTM model. Further, the F1 curve shows similar patterns where the F1 score for BiLSTM CRF is always higher than that of the BiLSTM model. For recall we observe that both the models show almost identical curves but the BiLSTM CRF edges over the baseline model. I also observed that the training and validation loss for the BiLSTM CRF model are lower than that of the baseline model, thus showing that it has better learning of the model parameters. The Figures 4-5 refer to the validation loss and training loss curves respectively.

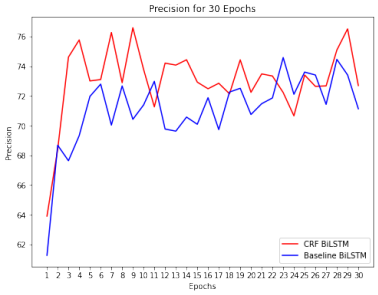

Figure 1: Precision Curve

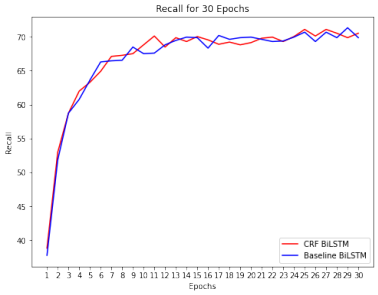

Figure 2: Recall Curve

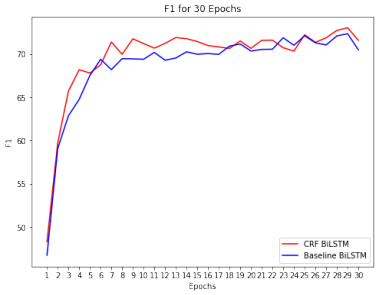

Figure 3: F1 Score Curve

From Figure 4, I analysed that the model seems to be overfitting on the training data, because the validation loss increases after a 10-15 epochs. One main reason for this could be the small dataset we

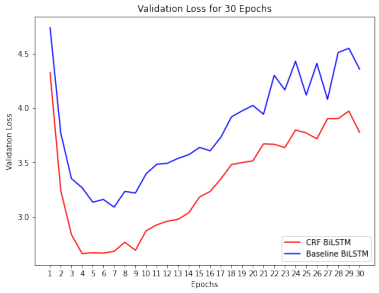

Figure 4: Validation Loss Curve

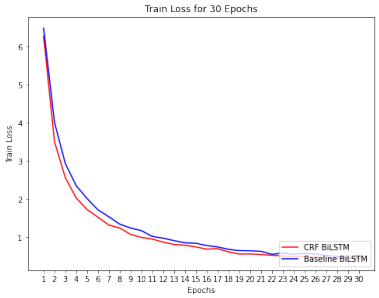

Figure 5: Train Loss Curve

are training on. Since, the size of the data is not too large, both the models overfit on the training data, thus showing similar trends in their validation loss curves. Although both the models overfit, the validation losses of BiLSTM CRF are consistently lower than those of the BiLSTM baseline model.

The BiLSTM CRF model achieved an accuracy of 75.83% calculated on the non-O tags. The baseline BiLSTM model gave an accuracy of 75.88% (non-O). Here we see that although the precision, recall, and F1 scores of the BiLSTM CRF Model has a considerable higher values than the baseline model, the accuracy does not increase even after implementing CRF.

Model	Data	Precision	Recall	F1 Score
BiLSTM Tagger	Valid	71.82	70.19.	70.99
BiLSTM CRF Tagger	Valid	**73.76**	**71.24**	**72.48**
BiLSTM Tagger	Train	97.63	97.46	97.54
BiLSTM CRF Tagger	Train	**97.93**	**97.56**	**97.74**

Table 1: Model Evaluation Results

Table 1 compares the values for both the models on their train and validation sets. The values in bold show the higher value in the comparison. It is observed that average precision achieved by BiLSTM CRF is *1.49%* higher on validation set and *0.2%* higher on the train set, than that of BiLSTM Tagger. Table 1 illustrates that BiLSTM CRF outperforms the BiLSTM Tagger, on both the datasets.

3.2 Dropout Analysis

In order to investigate the model performance, I also did analysis to see how the use of dropout might affect the training. Table 2 shows the comparison between the models when they are trained without dropout. All the other model parameters remain the same.

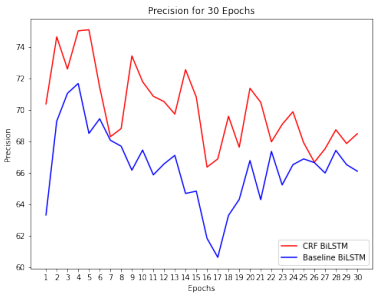

Figure 6: Precision Curve (No Dropout)

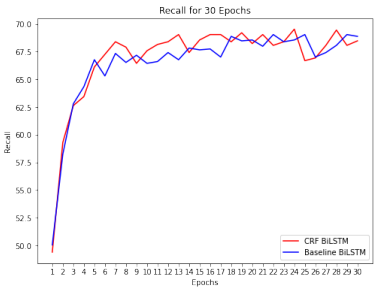

Figure 7: Recall Curve (No Dropout)

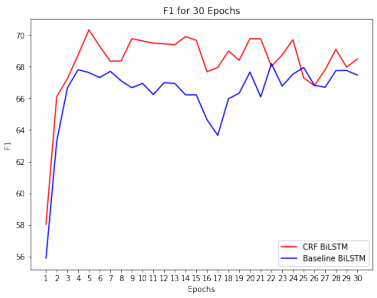

Figure 8: F1 Score Curve (No Dropout)

Model	Data	Precision	Recall	F1 Score
BiLSTM Tagger	Valid	66.85	68.81	67.81
BiLSTM CRF Tagger	Valid	**72.24**	**70.19**	**71.20**
BiLSTM Tagger	Train	**98.75**	**98.86**	**98.81**
BiLSTM CRF Tagger	Train	98.02	97.58	97.80

Table 2: Model Evaluation Results with No Dropout

The table 2 scores demonstrate that even without dropout the BiLSTM CRF model outperforms the

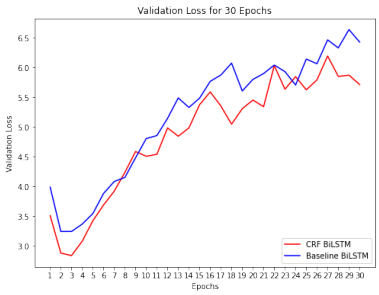

Figure 9: Validation Loss Curve (No Dropout)

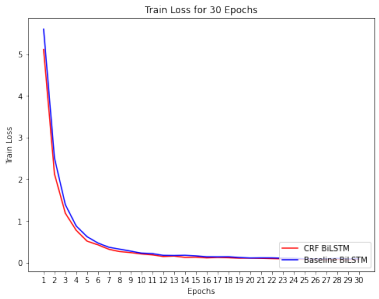

Figure 10: Train Loss Curve (No Dropout)

baseline model on validation dataset but the baseline seems to give marginally better results on the training data. The Figures 6-8 show the precision, recall, F1 curves for the models without dropout.

It is observed that both the models have a validation loss curve, (Figure 9) that has higher values than when the models were trained with *dropout = 0.3*. This illustrates that the effect of dropout on model training is that it helps reduce overfitting. Consequently, the precision, recall, F1 values of the models have comparatively lower range than those models which were trained with a dropout. This essentially shows that adding a dropout into model training works well in avoiding or reducing overfitting and hence gives better results.

3.3 Out-of-vocabulary (OOV) data analysis

To better understand the behaviour of the BiLSTM CRF model, I investigate the model performance using the evaluation metrics by performing the evaluation on Out-of-Vocabulary (OOV) sequences. To do this analysis, I created a new OOV validation set from the entire validation dataset. This OOV validation dataset consists of those sequences which contain atleast one $< unk >$ word. The $< unk >$ tag represents words which are not part of the word vocabulary of our model.

After creating this OOV validation set, I calculated the precision, recall, F1 scores on this dataset as described in Table 3. I also considered the models with and without the dropout parameter to demonstrate the results. It is observed that the BiLSTM CRF Tagger with dropout gives an F1 score of 70.25 which is *1.54%* higher than the baseline. Further, the BiLSTM CRF Tagger without dropout also outperforms the baseline model with no dropout by *4.18%*.

Model	Dropout	Precision	Recall	F1 Score
BiLSTM Tagger	Yes	69.64	67.79	68.71
BiLSTM CRF Tagger	Yes	**71.51**	**69.03**	**70.25**
BiLSTM Tagger	No	63.59	66.02	64.78
BiLSTM CRF Tagger	No	**70.26**	**67.70**	**68.96**

Table 3: OOV words Evaluation Results

These results emphasize and provide proof that adding a CRF layer, along with viterbi decoding to get a tag sequence not only performs better on in-vocabulary words but also seem to give good results with OOV dataset.

3.4 Span-level Tags Evaluation Analysis

It is important to observe how the model behaves on each NER tag, to know if the model can consistently behave better than the baseline results. Table 4 shows the comparison of various NER classes, and the associated evaluation scores for both the BiLSTM and BiLSTM-CRF model. The evaluation is performed on the best model parameters that is with *dropout = 0.3*, *epochs = 30*.

Tag	Metrics	BiLSTM CRF	BiLSTM
LOC	Precision	84.36	**86.49**
	Recall	**83.20**	79.34
	F1 Score	**83.77**	82.76
MISC	Precision	**77.24**	74.65
	Recall	**58.33**	55.21
	F1 Score	**66.47**	63.47
ORG	Precision	63.95	**64.69**
	Recall	**61.24**	60.26
	F1 Score	**62.56**	62.39
PER	Precision	**70.15**	64.48
	Recall	74.53	**77.24**
	F1 Score	**72.27**	70.28

Table 4: Span-level Tags Evaluation Results

From the Table 4 we see that the F1 scores for the BiLSTM CRF Tagger always outperform the BiLSTM Tagger. Thus showing that even at the tag-level CRF's global normalization and then decoding helps the model predict better tags. I observed that precision for LOC and ORG is higher for the

BiLSTM tagger by approximately 1.5%. In general, the evaluations show an improvement in F1 by 1-4%. Overall, this analysis demonstrates that BiLSTM CRF works well at tag-level as well and thus gives higher results than those of the baseline BiLSTM Tagger.

4 Conclusion and Future Work

In this project, I implemented BiLSTM CRF Tagger for the task of Named Entity recognition. After comparing and providing details using various performance metrics, it can be concluded that the BiLSTM CRF Tagger outperforms the Baseline BiLSTM Tagger in most of the cases. Further, this explains and provides analysis to show that using global normalization and joint decoding like in the case of CRF gives much better performance for the task of NER. For further understanding and experimentation, we could implement the BiLSTM CRF model on character level sequencing rather than the word level being performed here. Further, we could investigate how the model performs on Tag-level understanding by considering each tag and checking samples to evaluate the performance on the same.