Named Entity Recognition

Omar Bayoumi
Sapienza University of Rome
Natural Language Processing
Homework 1
bayoumi.1747042@studenti.uniroma1.it

1 Introduction

The **Named Entity Recognition (NER)** problem belongs to the category of tagging problems. In particular, the problem to be solved is **NER with an IOB (inside, outside, beginning) scheme**. So for each word within the text the aim is to categorise them following this scheme and the following categories: **Person (PER), Location (LOC), Group (GRP), Corporation (CORP), Product (PROD), Creative Work (CW) and Other (O)**.

The models used as possible solution for this task are two Neural Networks, the first is based on a **bidirectional-Long Short-Term Memory Layer (BI-LSTM)** with a classification linear layer as output, the second, like the first, with the addition of a of **Conditional Random Field (CRF)** layer for the calculation of the loss function and the decoding of the outputs. The datasets were used respectively for their purposes: training on training set, validation on development set to avoid overfitting. The proposed model is the second one, as it achieved the highest performance on the test data.

2 Vocabulary

To create a **word embedding**, it is first necessary to read the dataset and preprocess it, and then only take certain words into account.

2.1 Preprocess Data

The sentences read from the dataset are: sentences read normally from the dataset and sentences read normally from the dataset to which a **preprocess** is applied. The preprocess consists of taking the sentence and analysing the **Part-of-speech (POS)** (**extra point**) using the **spacy** library. From this analysis, the words contained in the sentence have been replaced in their **lemmatized version**. This also led to the splitting of some words into several words (*e.g. ["eaton's"] become ["eaton", "'s"]*) and the label assigned to the split words is still the same, even though it is a B-CLASS.

The idea behind this pre-processing is to **augment the existing dataset** with "normalised" versions of words in order to give the model the ability to learn more and general word combinations. Also, the fact of leaving the same label for each subdivided word is meant to say: all these words are actually **beginnings** and so treat them as such.

The addition of this preprocess alone improved the **f1 score by 6%**.

2.2 Vocabulary Creation

After the preprocess part, it was necessary to count the **frequencies** of each word and consequently only take into account the **words occurring at least 3 times** in order to reduce the noise caused by infrequent words, thus helping the calculation of the loss function generated by the CRF.

Having this set of words, it was then easy to create the dictionaries **word2id, id2word, label2id and id2label**.

3 Dataset

The dataset was created by dividing the sentence into **windows** of fixed size and moving this window within the sentence by a fixed step, filling the excess with a **pad token**. If a word is not present in the vocabulary, the value of this word is the **unk token**. However, after several tests, it was much more efficient to create a window large enough (100 words in my case) to cover more or less all the sentences (if any sentence has more than 100 words, it is broken into several windows of course). Most probably this gave better results due to the nature of the BI-LSTM used which acquires information by reading **consecutive words and in both directions**, which would have been limited with small windows. This improved the **f1 score by 5%**.

4 BI-LSTM Model

4.1 Conditional Random Field (extra point)

A **CRF is an output layer** that encourages the neural network to produce a valid sequence of output labels, through **log-likelihood maximization** of the correct label sequence *(Guillaume et al., 2016)*. This allows adjacent labels to be considered before producing the final label. In my case, just introducing the CRF improved the **f1 score by 2%**.

A small improvement caused by the fact that the architecture is most likely improvable. As can be seen in the past work *(Guillaume et al., 2016)*, the architecture has an **embedding based also on characters** which I did not do, and in general other information could have been added to the embedding such as **upper/lower case and POS information**.

4.2 Proposed Model

The model is composed of an **Embedding layer, a BI-LSTM layer, a Linear layer and a CRF layer** *(Figure 1)*.

I used a **pretrained word embedding given by gensim, in particular *glove-twitter-200*** (Pennington et al., 2014) (**extra point**). Because of the way the vocabulary was created, some words, although few, were initialized randomly because they were not present in this embedding. As a last operation a **Log Softmax** was performed since the loss function generated by CRF is a negative log likelihood and unlike the crossentrpy loss of pytorch, the Log Softmax is not performed automatically.

5 Training

The hyperparamenter tuning is done via random search over several hyperparameters.

Starting with the optimizer, both **SGD**, and **Adam** were tested and the best was Adam, most likely because Adam maintains a per-parameter learning rate that improves performances on problems with sparse gradients like the NER task. The learning rate chosen is 1^{-3}. As a loss function, the CRF layer. For **BI-LSTM two hidden layers** have been chosen obtaining better results, most likely because we have reached a degree of information, such that there is a better generalization. As **Dropout**, after a fine tuning has been put to 0.5, to avoid overfitting.

Training was performed for a maximum of **100 epochs**, however an **early stopping algorithm based on patience** was included to avoid training too long (very often 15 epochs was sufficient). The early stopping algorithm adds one to the patience counter, and at five stops, if the current validation loss is greater than the validation loss obtained in at least half of a time window of ten epochs ago.

6 Results and comparative results (extra point)

From now I will talk about **model1** when I talk about the model without CRF, **model2** with CRF (all plots are about **extra points**).

Model1 has few differences from model2. **The hyperparameters are the same**, however the loss function is a **crossentropy** and the vocabulary was set based on another criterion: instead of taking words with a minimum frequency equal to three, **all words that are present in the word embedding of "glove-twitter-200" (Pennington et al., 2014) were taken**. This is because unlike model2, model1 is able to generalize more even with infrequent word.

The **training plots** of the two models are shown in *Figure 2*. Both models have been trained for more or less 15 epochs, and the checkpoint selected is that in which the **validation loss is smaller**, since subsequently there has been overfitting. When it comes to NER we usually have a high number of "O" instances and a lower number of instances of other classes, so simple accuracy **is not appropriate to evaluate this system**. So in addition to accuracy a list of scores have been printed (*Figure 3*) so as to have a general idea and the **most important** is the **Macro-F1 from "*seqeval*"** that is a harmonic average of precision and recall that takes into account the sequence of results.

Since this is a classification problem, another useful metric is the confusion matrix (*Figure 4*). Note that it is normalized for true positives, i.e. per row. Looking at the confusion matrix for both models we can see that **CW**, and **PROD** are the most confused, most likely due to their ambiguity not being fully recognized. Instead as expected, the class **O** is the one that has caused more noise within the predictions.

7 Conclusion

As we can see, more or less the models get the same results but the one using CRF is slightly better (about 1%) but still improvable. For example with the addition of controls based on POS, character embedding and optimal hyperparameter tuning.

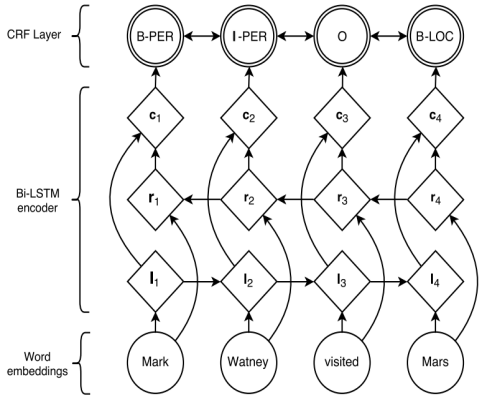
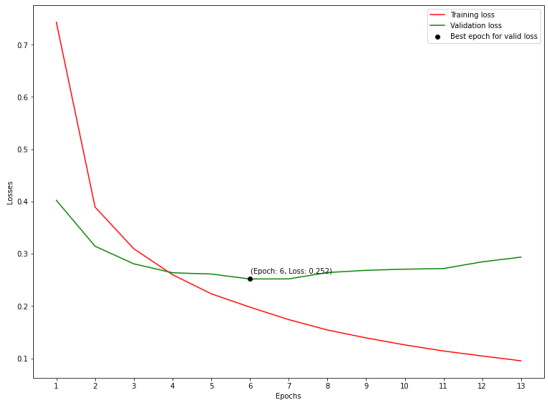
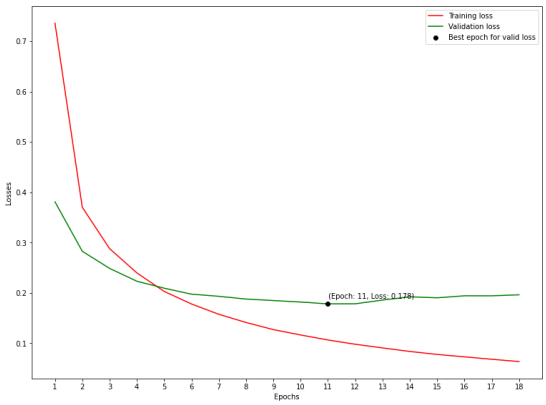


Figure 1: The architecture used.



(a) Model without the use of CRF. The best epoch is 6 with a validation loss of 0.252.



(b) Model that uses the CRF. Best epoch is 11 with a validation loss of 0.178.

Figure 2: Graph of the loss function of the two models

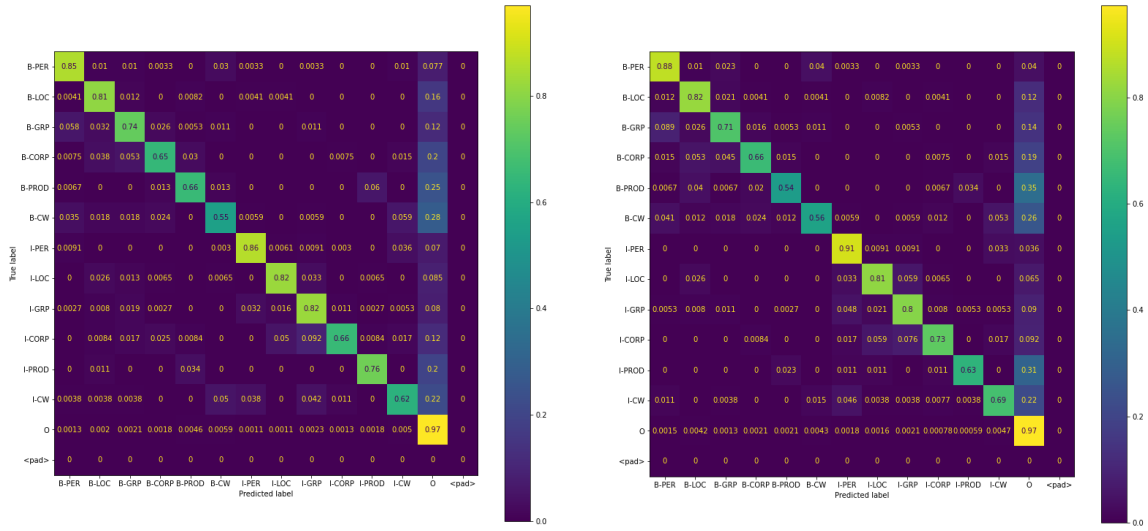
```
Micro Precision: 0.9279272213944004
Macro Precision: 0.7640977764243746
F1 score: 0.7574234829133377
Seqeval Accuracy: 0.9279272213944004
Seqeval F1 score: 0.6611607177439702
Per class Precision:
        O 0.9659632402995235
        I-PER 0.8875
        B-PER 0.8707482993197279
        I-GRP 0.842391304347826
        I-LOC 0.8289473684210527
        B-LOC 0.8073770491803278
        I-CORP 0.78
        B-GRP 0.7421052631578947
        B-CORP 0.7107438016528925
        I-PROD 0.6875
        I-CW 0.6653061224489796
        B-PROD 0.6282051282051282
        B-CW 0.5164835164835165
        <pad> 0.0
```

(a) Model without the use of CRF. The best seqeval f1 score obtained is 0.661.

```
Micro Precision: 0.9309073798133479
Macro Precision: 0.7809130948896775
F1 score: 0.760941086542142
Seqeval Accuracy: 0.9309073798133479
Seqeval F1 score: 0.6729829048219201
Per class Precision:
        O 0.9669092673459486
        I-GRP 0.8645533141210374
        B-PER 0.8407643312101911
        I-PER 0.8379888268156425
        I-CORP 0.8130841121495327
        I-PROD 0.7971014492753623
        B-GRP 0.7701149425287356
        I-LOC 0.7654320987654321
        B-LOC 0.7326007326007326
        B-CORP 0.7272727272727273
        B-PROD 0.7272727272727273
        I-CW 0.7075098814229249
        B-CW 0.6012658227848101
        <pad> 0.0
```

(b) Model that uses the CRF. The best seqeval f1 score obtained is 0.672

Figure 3: Scores obtained from the two models. You can see how the f1 score of seqeval differs by about 0.1.



(a) Model without the use of CRF.

(b) Model that uses the CRF.

Figure 4: Confusion matrix of the two models. As can be seen, the one that doesn't use CRF seems to behave better, however it has a lot of noise which could make the predictions ambiguous. Instead, the one using CRF has many more 0's indicating less ambiguity in the prediction.

References

Guillaume et al. 2016. Neural architectures for named entity recognition.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation.