NLP Homework 1: Named Entity Recognition

Gioele Migno
1795826

1 Introduction

Named Entity Recognition (NER) is a NLP task in which we are interested in locate and classify named entities mentioned in a text, a named entity is a real world object that can be denoted with a proper name. There are several types of named entities, in our case we are interested in six of them: *person*, *corporation*, *location*, *product*, *group* and *creative work*. As most NLP tasks, to obtain high performance NER requires analysis of the text as a sequence of tokens, this allows the model to capture the context of each word and classify them according to their contexts.

2 Data Processing

Usually named entities start with a capital letter, this information could be useful to distinguish them to other words and could be provided as additional input to our system. However, the dataset provided to us contains all lowercase words therefore this property cannot be exploited.

The actual pre-processing steps used in the final model are word vectorization and POS tagging.

2.1 Words Vectorization

When we deal with words we have to find a way to represent them as fixed length numerical vectors. The most simple way to do this is to assign an index to each word in our training set and then use a word embedding able to transform that indexes into non-sparse vectors. A finite set of sentences cannot contain all possible words of a language, therefore external words are mapped as unknown and represented with the same vector. Words indexation has been performed as a preprocessing step considering as unknown those words that do not appear at least two times in the training set, this choice should make more robust our system for real world data. The vectorization step instead is performed by a pre-trained embedding layer.

2.2 POS Tagging

Part-Of-Speech tagging is the process of assign to each word in a text its category (part of speech) like noun, verb, adjective and so on. This kind of additional information is very useful for our task since a named entity is usually a proper noun, it is not always the case though. POS Tagging step has been performed as a preprocessing step using NLTK POS Tagger. Such tagger as default uses a non-universal POS tagset [1] that contains more than forty categories. Instead of use all of them we use only those that appear at least one-hundred times in the training set, other categories are mapped as unknown. Each part-of-speech is represented as a one-hot encoded vector, therefore by reducing the number of categories we reduce POS-tag dimension. A better solution would be to use logits of the POS-tagger but for what we know, NTLK implementation provides only the most likely tag.

3 Model Architecture

Two different model architectures have been proposed to solve our NER task. The baseline architecture is composed by a word embedding layer, a multi-layers bidirectional LSTM and a dense layer used to make the final classification step. Then the baseline architecture has been improved by adding an external POS Tagger and more dense layers for the classification step. Figure 1 shows differences between baseline and upgraded architecture.

3.1 Baseline

The baseline model has been crucial to choose several components. The following choices have been made by performing several tests, called studies,

[1] Run for more info: nltk.help.upenn_tagset()

shown in ATTACHMENTS folder provided as attachment to this report, each study is referenced by its date.

Word Embedding

The first studies performed are **2022-0411-0810** and **2022-0411-1258**. They were conducted in order to choose the word embedding and some initial training parameters. The first study uses *word2vec-google-news-300* embedding and the second *glove.6B.100d*. The performance obtained are very similar, therefore to make the decision we looked at the percentage of words contained in the training set for which embeddings provide the vector representations, the best is the second with 98%. The second couple of studies is **2022-0411-1745**, **2022-0411-1957** used to choose whether start the training with embedding layer trainable or not. Best performance have been obtained with embedding not trainable.

LSTM Parameters

For a LSTM model the main parameters are direction (mono-bidirectional), number of features in the hidden state, and number of layers. We have chosen to use a bidirectional LSTM at prior, about number of features instead we looked at the previous studies in which in a range of [100, 300], 220 hidden dimension performed better. Regarding number of layers, we tested two, three and four layers, the best performance has been obtained with three layers. [2]

3.2 BiLSTM + POS Tagger

Starting by the previous baseline model we tried to improve performance by adding POS tags in input at the model. The network received in input POS tags as concatenation in two points: at the input of the BiLSTM and at the input of last dense layers. The idea behind this choice is that by giving tags in input at the LSTM we help it to understand the context of each word, and giving in input at the dense layers we avoid lost of this information at the classification step. Due to lack of time, no experiments have been performed to confirm or deny this intuition. In this architecture, in addition to the final linear layer, there are two dense layers with dimensions (out_lstm_dim x 300), (300 x 300), dropout and relu activation function. [3]

[2]NOTE_TRAINING_BASELINE.pdf file shows tests

[3]NOTE_TRAINING_UPGRADED.pdf file shows tests

4 Results

The training for both models has been performed in Google Colab for 100 epochs using Adam optimizer with lr=0.001, dropout equals to 50% and l2 norm used as regularization technique. (l2=5e-05 baseline, l2=1e-05 BiLSTM+POS Tagger). The loss function used is the cross entropy, baseline performed better with weighted loss and the other model without. The loss weight for each class i-th has been computed as:

$$weight_i = 1 - \frac{\#label_i}{\#words\_in\_dataset}$$

During the training, the batch size used was 32 and each sentence was padded to a dimension of 50 words, since training set does not contain sentences longer than 41 words. However, to be robust to longer sentences, the final model pads input to 200 words.

Figure 2 and Figure 3 show respectively loss value and accuracy and F1 Score (macro) during the training. The same is shown in Figure 4 and Figure 5 for BiLSTM+POS Tagger model. As we can see in these plots, models start to overfit the training set, however F1 score computed for the validation set remains stable and slightly improve.

Figure 6 shows performance results for both of models. As we can see, upgraded model (BiLSTM+POS Tagger) performs better for precision and recall but slightly worse for precision, moreover baseline model has a higher F1 score for PER named entities.

Figure 7 shows confusion matrices of both models. Seqeval library [4] does not provide a function to compute them, therefore sklearn library has been used. As we can note, both models classify wrongly several words as named entities, this problem is mainly caused by our unbalanced dataset.

5 Conclusions

To solve the proposed NER task, a model based on BiLSTM layers has been proposed, in order to improve performance an external POS Tagger has been added, that allowed to increase performance of about +3% F1 Score (macro).

[4]https://github.com/chakki-works/seqeval

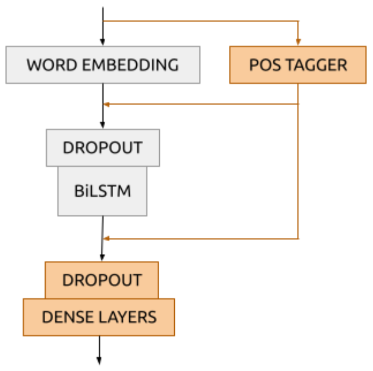

Figure 1: Model architecture. In orange components added or modified respect to the baseline model

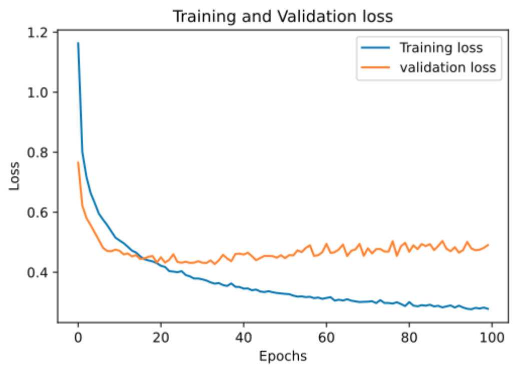

Figure 2: Loss value during baseline model training

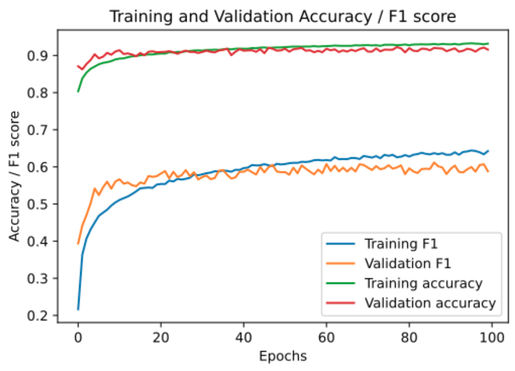

Figure 3: Accuracy and F1 Score (micro) during baseline model training

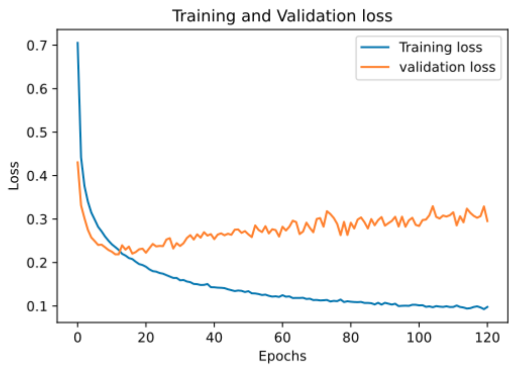

Figure 4: Loss value during BiLSTM+POS Tag model training

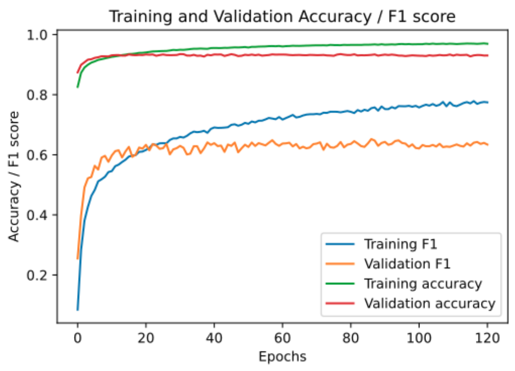

Figure 5: Accuracy and F1 Score (micro) during BiLSTM+POS Tag model training

```
-----------------------------------------------------------------
Baseline

                    precision    recall  f1-score   support

        CORP           0.51      0.61      0.56       133
          CW           0.40      0.59      0.48       170
         GRP           0.64      0.67      0.65       190
         LOC           0.67      0.81      0.74       243
         PER           0.82      0.86      0.84       300
        PROD           0.35      0.56      0.43       149

   micro avg           0.58      0.71      0.64      1185
   macro avg           0.57      0.68      0.62      1185
weighted avg           0.61      0.71      0.65      1185

-----------------------------------------------------------------
BiLSTM + POS Tagger

                    precision    recall  f1-score   support

        CORP           0.56      0.59      0.58       133
          CW           0.55      0.49      0.52       170
         GRP           0.69      0.69      0.69       190
         LOC           0.81      0.80      0.81       243
         PER           0.78      0.87      0.82       300
        PROD           0.51      0.52      0.51       149

   micro avg           0.68      0.70      0.69      1185
   macro avg           0.65      0.66      0.65      1185
weighted avg           0.68      0.70      0.69      1185
-----------------------------------------------------------------
```

Figure 6: Performance obtained with baseline and upgraded models for validation set.

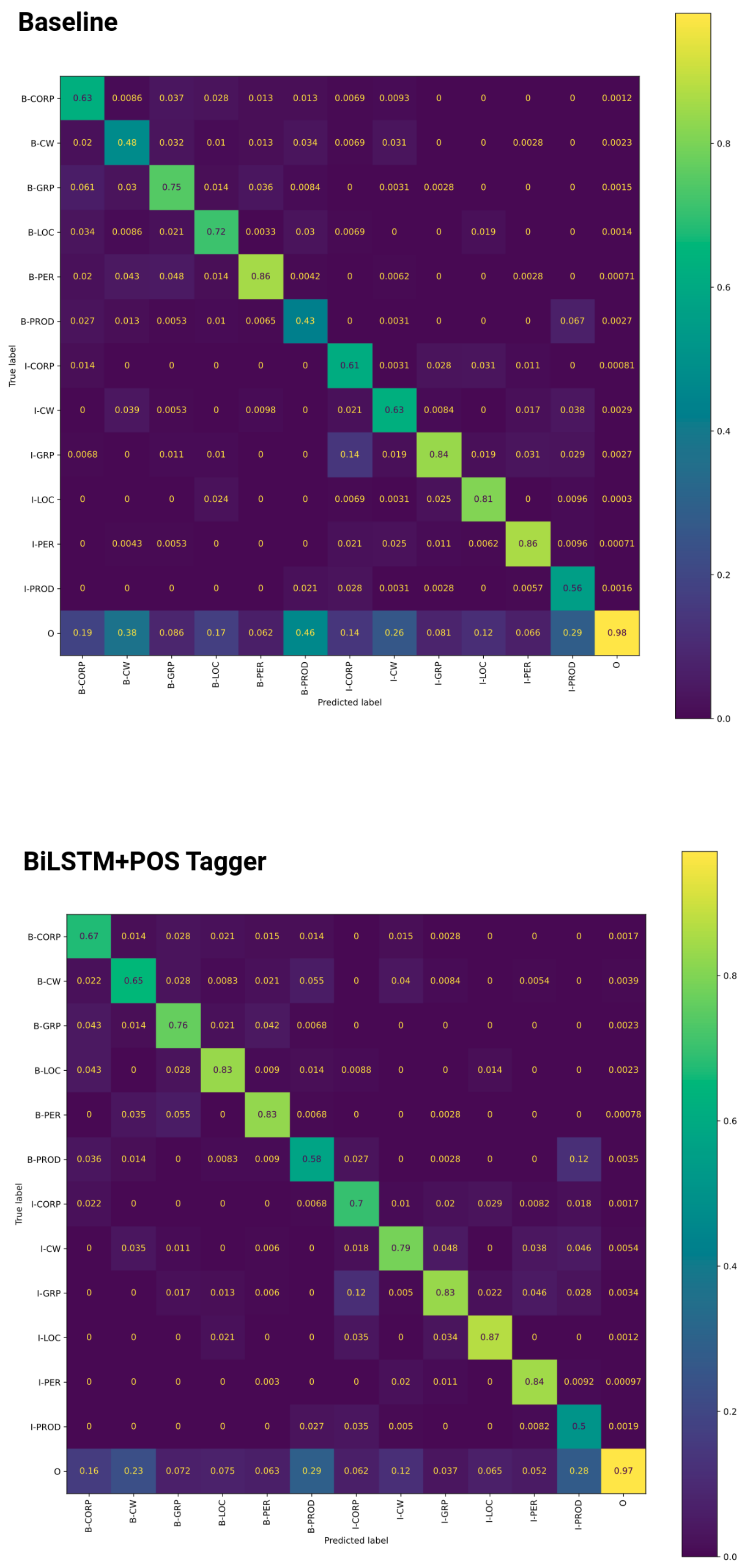

Figure 7: Confusion matrix of both models computed on validation set. All values are normalized over the predicted conditions (e.g. columns)