Deep Neural Network can overcome noisy corpus in NER task

Antonio Andrea Gargiulo
Sapienza University of Rome
gargiulo.1769185@studenti.uniroma1.it

Abstract

Neural networks trained with unbalanced datasets show poor performances in recognizing minority classes. In this report, we attempt to overcome this issue by training a recurrent deep neural network with a novel noisy/decontextualized input method and show that it can overcome the noise and extract useful information to perform the NER task, with better performances in smaller classes.

1 Introduction

In this report, we want to show the advantages and disadvantages of training a Deep Neural Network for the task of **N**amed **E**ntity **R**ecognition in a **non-standard** setting of training. NER is a classification task that links a named entity with a label telling in which group it belongs, examples of groups are *PER, LOC, ORG, ...*
In our case, the task has an additional constraint: for each word composing a named entity tell if it is a starting point with *B-* (Beginning) or an Inside token with *I-* and for all the other tokens that aren't named entity tell *O* (Outside), in the literature this is commonly referred as *BIO notation*.

2 Methodology

To perform the task we choose an **RNN** architecture, since it commonly shown in the literature (Karpathy et al., 2015) that this class of *NN architectures* perform really well on sequences and some type of it like *LSTM* (Hochreiter and Schmidhuber, 1997) and *GRU* (Cho et al., 2014) can encode long and short-range dependencies among data. These architectures have also some drawbacks, they have a fixed number of memory cells, so it's difficult to learn long sequences of data without wasting some previous information (Madsen, 2019). Furthermore, some claims are that this architecture has a limited span of memorization.[1] Starting from these considerations, we want to show that a specific projected architecture can store only the important information and use this extracted knowledge in a better way for the *NER* task. The metric monitored is the *F1-score* computed only on the named entity instances.

2.1 Baseline Architecture

Stating the effectiveness of the *Bidirectional-LSTM* on various NLP tasks and the great performances of *Embeddings* in storing useful information of words in a vector space, we use this architecture as our baseline. The choice of a Bidirectional-LSTM instead of a Vanilla one is because it can encode the majority of the contextual information for a sentence.

2.2 Final Architecture

From the baseline, we performed extensive research for increasing the performance of the final model and reaching a better generalization. We construct a double recurrent layer with a *Bi-LSTM* encoding and retain useful contextual information extracted from the Embeddings input, after we select a **Bi-GRU** (Bi- for bidirectional) layer, in a fashion like Seq-to-Seq model, instead of another LSTM, because of the different effectiveness of the GRU in storing useful long-term information, with respect to LSTMs. See Figure 3.

3 Implementation

We have considered the batch size as the number of random sentences to transform in a single **decontextualized corpus** and use it as a *unique* tensor to feed our DNN. With this particular method, we want to prove if it brings new and better information to the final knowledge of our network, in

[1] https://atcold.github.io/pytorch-Deep-Learning/en/week06/06-2/

the particular task of NER. This choice is motivated since we want a more robust network, unbiased to a fixed position of a class in a sentence. The network must extract and retain information that are **invariant** to swap of location such that if $f(w_1 w_2 ... w_n) = y$ where w_i is our sentence's words and y is our prediction, we want $f(w'_1 w'_2 ... w'_n w_1) = y$ in the case w_1 don't depend on the context.
During training SGD with *momentum* (Goh, 2017) is used as our optimizer, and the learning rate is set to 0.35, automatically reduced by a factor of 0.6 by the *ReduceLROnPlateau* scheduler, monitoring an average by epoch F1-score on validation data. As our Embedding layer, we use the pre-trained **GloVe** Embeddings (Pennington et al., 2014) trained on Wikipedia data with an uncased vocabulary of 400k words and a 300 dimensions per word vector. The choice of GloVe is due to the interesting property of neighbouring words, that reveal some common peculiarities and similar words, and the fact that the vector space shows robust linear substructures.

4 Experiments

In this section, we talk about our experiments, where we show the effect of different changes on the final model. All the results of the experiments are summarized in Table 1. For all the experiments we used a *CategoricalCrossEntropy* loss function, and we used the *SGD* optimizer with momentum set at 0.94.

Baseline: we started from a Baseline model, 2-layer Bi-LSTM with a hidden dimension of 128 and a linear layer, adding a L_2 *regularization* parameter, because of the high variance on validation data.

State: we added a change in the hidden state h_{t_0} and cell state vector c_{t_0} of the *Bi-LSTM* layer, and we see a good improvement when the two vectors are polarized in the same direction (i.e. $h_{t_0} == c_{t_0}$). Moreover, we see a better improvement when the two vector are inizialized with *ones* instead of zeros.

GRU: we changed the LSTM with a *GRU* and we notice a drop in performance since the GRU is more dependent on the long-context and it suffers greatly from our noisy decontextualized approach.

Layers: we tested adding more layers to the LSTM and we see an improvement with *3 layers* and a drop in performances after this number.

Dimensions & Activation: we see that an increase of *hidden dimension* to 200 and the add of a *LeakyReLu* activation function before the linear layer is also beneficial to the model.

Architecture fusion: we add a *Bi-GRU* layer after the *Bi-LSTM*, and we see an overall increase in performance, stating the fact that the two RNNs architecture can extract and store different useful pieces of information, overcoming the difficulties of the GRU due to the noisy input.

Expert system & Gradient clipping: inspired by the NMS and the thresholding functions of the objects detection neural networks, we add an expert system, Figure 4, that looks at the final prediction of the model and is used to remove some of the gross errors produced by a neural network. We also add a *Gradient clipping* (Pascanu et al., 2013) procedure to avoid exploding gradients problems.

Pre-trained Embeddings: we select the **GloVe** pre-trained Embeddings as input of our well-established model and freeze them as we don't want task-dependent changes in the vector space representation, but only a more general words space representation.

5 Results

Table 1 shows the results of our experiments by an F1-score comparison. As we can see, from Figure 1, the network is able to perform well in a NER task with our novel technique of a noisy corpus. Furthermore, if we compare the confusion matrices Figure 2 of the same network trained with and without the noisy corpus approach, we can see an improvement in performance in recognising the classes present in lower quantities in the training dataset.

6 Conclusions

In this report, we empirically showed that a different approach of training Deep Neural Networks for the task of NER can improve some weaknesses of standard training in settings with an unbalanced dataset. We show that our type of training can reach comparable performances in the NER task in comparison with a standard one. Showing outperforming results in those classes that are present in smaller numbers, we prove that this novel approach can be used as a transformation on unbalanced datasets.

Model	**mom.**	$\|L\|_2$	h_{t_0}	c_{t_0}	**GRU**	**#layer**	**#hid. dim.**	**act.**	**F1-score**
Baseline	-	-	$\vec{0}$	$\vec{0}$	-	2	128	-	0.4675
State00	0.94	8e-5	$\vec{0}$	$\vec{0}$	-	2	128	-	0.5237
State01	0.94	8e-5	$\vec{0}$	$\vec{1}$	-	2	128	-	0.5085
State10	0.94	8e-5	$\vec{1}$	$\vec{0}$	-	2	128	-	0.4892
State11	0.94	8e-5	$\vec{1}$	$\vec{1}$	-	2	128	-	0.5270
GRU	0.94	8e-5	$\vec{0}$	-	$\surd$	2	128	-	0.4458
LSTM-l3	0.94	8e-5	$\vec{1}$	$\vec{1}$	-	3	128	-	0.5464
LSTM-l4	0.94	8e-5	$\vec{1}$	$\vec{1}$	-	4	128	-	0.5115
L3-200h	0.94	8e-5	$\vec{1}$	$\vec{1}$	-	3	200	-	0.5585
Lrelu	0.94	8e-5	$\vec{1}$	$\vec{1}$	-	3	200	LReLu	0.5595
BilsBigru	0.94	8e-5	$\vec{1}$	$\vec{1}$	$\surd$	2x2	2x200	LReLu	0.5699
Exp-clipg†	0.94	8e-5	$\vec{1}$	$\vec{1}$	$\surd$	2x2	2x200	LReLu	0.5740
P-GloVe*†	**0.94**	**8e-5**	$\vec{1}$	$\vec{1}$	$\surd$	**2x2**	**2x100**	**LReLu**	**0.6913**

Table 1: Ablation study in 20 epochs for architectures with non-pre-trained embeddings. The last row shows the results obtained with the pre-trained embeddings in 25 epochs. *with dropout †with expert system.

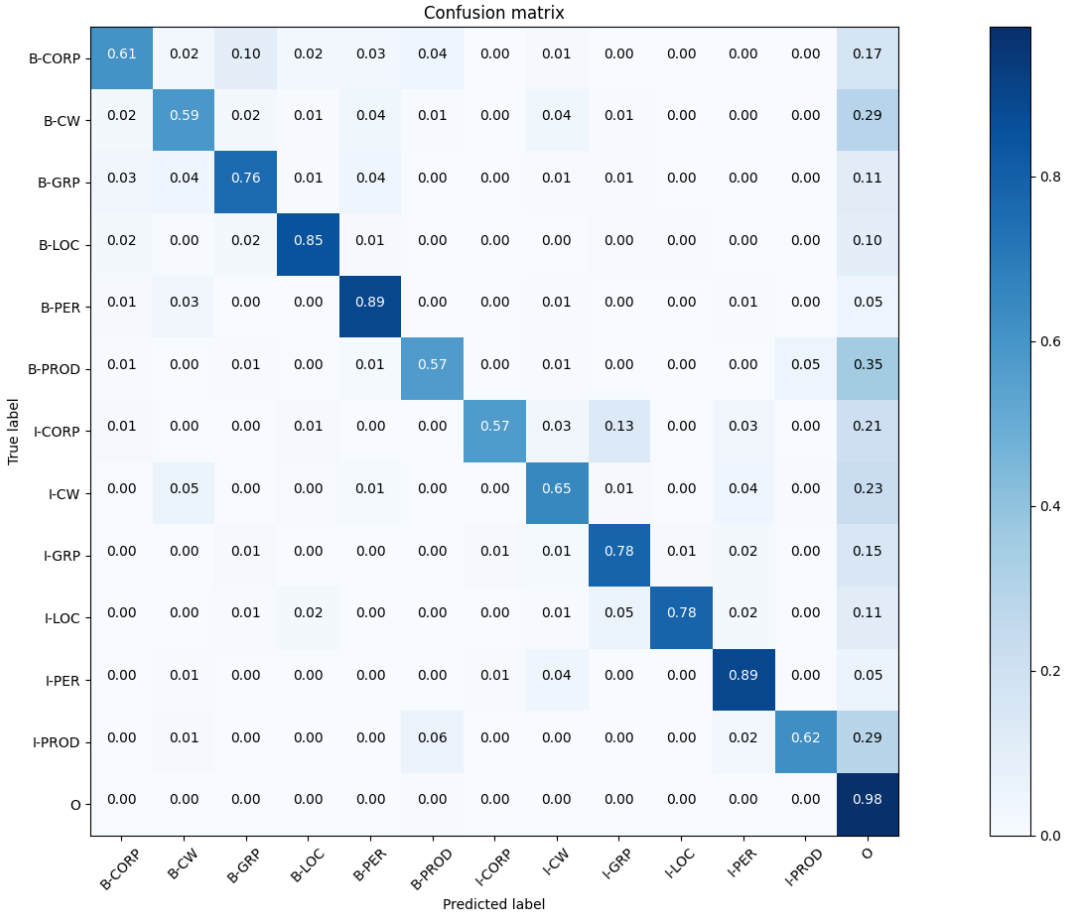

Figure 1: Confusion matrix normalized of the model trained with the pre-trained GloVe embeddings and the noisy corpus approach, on the validation dataset. We can see the better performances on the small classes CW and PROD.

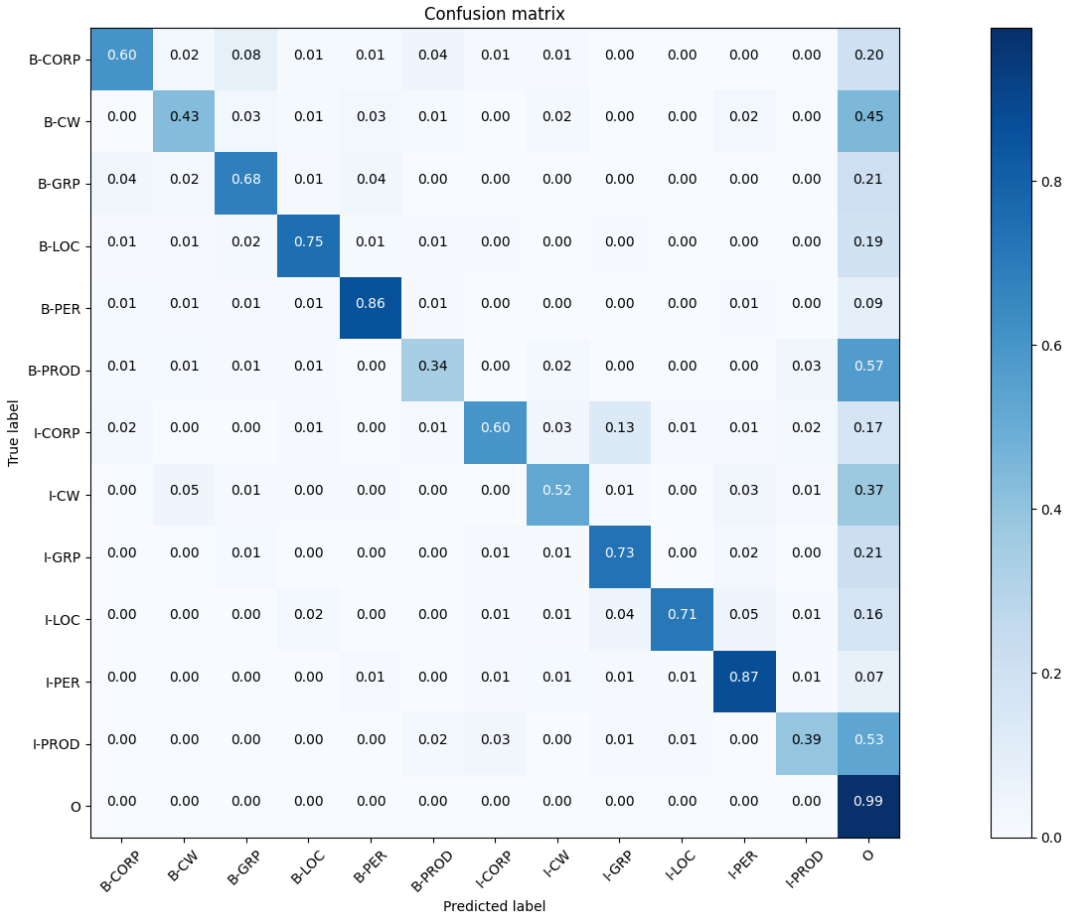

Figure 2: Confusion matrix normalized of the model trained with the pre-trained GloVe embeddings with a standard approach, on the validation dataset. We can see comparable overall results, but worst performances on small classes CW and PROD.

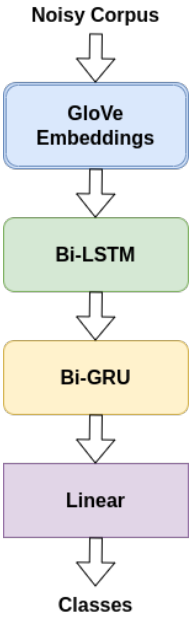

Figure 3: Final architecture of the model trained with the pre-trained GloVe embeddings and the noisy corpus approach.

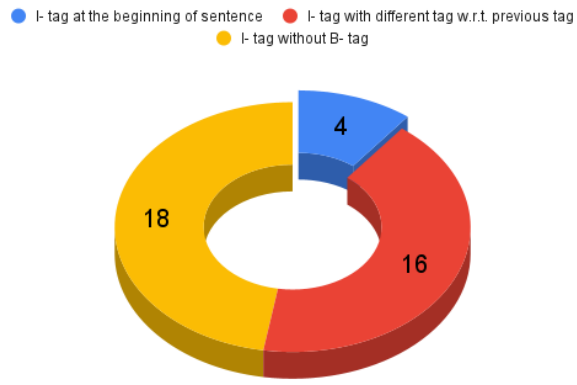

Figure 4: Impact of the rule based expert system on the final model, tested with validation data. We see that the NN still make some gross errors.

References

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259.

Gabriel Goh. 2017. Why momentum really works. *Distill*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. *CoRR*, abs/1506.02078.

Andreas Madsen. 2019. Visualizing memorization in rnns. *Distill*. Https://distill.pub/2019/memorization-in-rnns.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, page III–1310–III–1318. JMLR.org.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.