Homework 1: Name Entity Recognition

Alessio Orlando
Sapienza University of Rome
orlando.1792394@studenti.uniroma1.it

1 Preprocessing

The dataset given does not require a lot of preprocessing. Is already divided in tokens, not lemmatized. All the tokens are in lowercase, This means that the dataset is less contextualized and, consequentially, the classification is harder. My first step was to extract the POS trough NLTK pos_tag function. Each word in also splitted in character and then each of the character is grouped by sentences. Eventually the words can be also lemmatized, but that lead to worst results overall.

2 Words + Pos embeddings

The tokenized words are encoded with GloVe vectors (Pennington et al., 2014). They represent the basic input of the LSTM layer of the Base Model. A dimension of 200 was the easiest to train but with a good tuning the embedding of 300 was ultimately the one that gived the best results. In the best results this embeddings were frozen. Fine tuning those embeddings for this task didn't lead to higher results but instead slowed the converging of the model itself. POS embeddings are also part of the basic input of the model. The most used embedding dimension during my testing is 20 but i also used 4 and 8. By using the POS we can address this different meanings of the same word,we are able to contextualize better the sentence and consequentially disambiguate better the words.

3 Char embeddings

The char embeddings are probably the part of the model that required more tuning to be completely functional. The embedding dimension that resulted in better scores is a relatively small embedding of 50. The first reason why one should use a char embedding is because the char carries also the information of the uppercase chars and a more contextualized, sadly this dataset does not have any upper case character. Despite that it is still useful for extracting information from OOV words such as the misspelled one or simply some product, location or person name that is not present in the word vocabulary.

4 Global embeddings

Another information that i decided to take into account in my model is the global contextual embedding. The main idea was to generate a contextualized embedding for all the dataset trough a weighted average using if-idf as weights or by learning them using a CNN. Sadly for time reason i didn't managed to implement and test this kind of network and the test done with Global embedding is done by averaging the embedding resulted from a BiLSTM of the words+char.

5 Model

The core of the model is based on different BiLSTMs layers stacked on top of each other followed with a feature extraction layer. Between the feature extractor layer and the classification layer there is the activation function and few regularization layer such dropout and batchnorm. The LSTM layers take as input the combined embeddings of different extraction.

5.1 CRF classifier

The LSTM tagger is most of the time sufficient for achieving modest results in NLP tasks "but a sequence model like the CRF is really essential for strong performance on NER" (Pytorch). Conditional random fields (CRFs) is a class of statistical modeling method that predicts the tokens jointly using a conditional random field (Gardner et al., 2017). This is one of the most used and most powerful methods to achieve good results in this task since the evaluate of a token need to be strictly re-

lated to what he has before and after (for example, a I-PER cannot be consecutive of anything but a B-PER).(Lafferty et al., 2001)

5.2 Activation function

Normally the activation function used are ReLu and sometimes the SiLu. This because those activation function are easy to work with and have a predictable behaviour. In my test i tried both ReLu and SiLu and i noticed no particular difference. I tried also SELU, a self normalizing activation function and strangely i achieved better results. One of the most important things to do with this activation function is to work with normalized vectors. Is suggested to create the vector with a Gaussian distribution but in my case that was not possible, the solution the worked the best was to apply a batch normalization layer just before the activation function. This layer is also used during the testing with other activation function since there was no signification difference between the two approaches.

6 Experiments

The best model is the result of different iterations and experiments that aimed to understand the behaviour of different approaches on the same task.

6.1 Iterations

The first model utilized only words and pos embeddings. The core of the model remained pretty much the same. As we can see the results for this approach are the lowest overall 1. After that i tried to use also character embeddings. As we can see with this model we achieved a slightly better result. This is due to the fact that characters are able to capture all those meanings that a word cannot retain, even more with the OOV words. This results increased my hopes to achieve even better results by adding global context embeddings.

A further step was to add multiple dataset to understand if the learning ability of my network was limited by the amount of sample in the train data. The two additional dataset used in this iteration are: WNUT 2017 (Derczynski et al., 2017) and CoNLL 2003++ (Wang et al., 2019)

I chose the WNUT 2017 dataset because it had the same exact type of labels of our given dataset. It's taken from social networks and this means that has a lot of noise. To address this I used only the sentences that had some label other than "O" and I also removed the emojis.

CoNLL 2003++ 2 does not have the same exact labeling of our dataset. To be able to use also this dataset for my task I've been forced to separate the classification part from the other part of the model. The reasoning behind this is that even tho we do not reuse those classifiers the network still learns all the embeddings and since the task is the same the embeddings are capable of retain also the information of different dataset 3. This iteration is the one that created the best F1 score, but ultimately is not the one i picked as best one. I'll talk later about the reasoning of this choice.

The last iteration was done after the decision to try adding also the Global context vector. This for give the sentence a little bit of context. This approach seems that helped a bit the model, mainly in regularize and maintain a relatively low loss.

7 Best model

As stated in the section about the third iteration the model that achieved the best result in the f1 is the one that used Words embedding with POS and char representation plus the usage of the additional data. Ultimately i chose to not utilize it due to the fact that the loss was strangely higher than the other models. Most of the other one stabilized f1 in the mid 3 low 4 point of loss 6, this model instead had a weirdly high loss of 4.5. Since the Fourth iteration gave me pretty much similar result with a lower loss i decided to use that one instead.

8 Conclusion

In conclusion most of the outcome where pretty much as expected, with more information the model started learning more and more. The information that helped the most is the char embedding, probably due to the fact that a char representation carries a lot of information also cross-words. The addition of the dataset helped the inter-class variation (see 4 vs 5). Ultimately the little information added by the global context helped to generalize more and avoid a too strict over-fitting. Over-fitting that is still a little bit present in the model but that indeed helped to obtain better results. In some tasks memorize information is the only way to learn. Said that, is still very important to generalize as much as possible, that's why in the end i chose to present the fourth model that had roughly 20% lower loss and, hopefully, lower over-fitting.

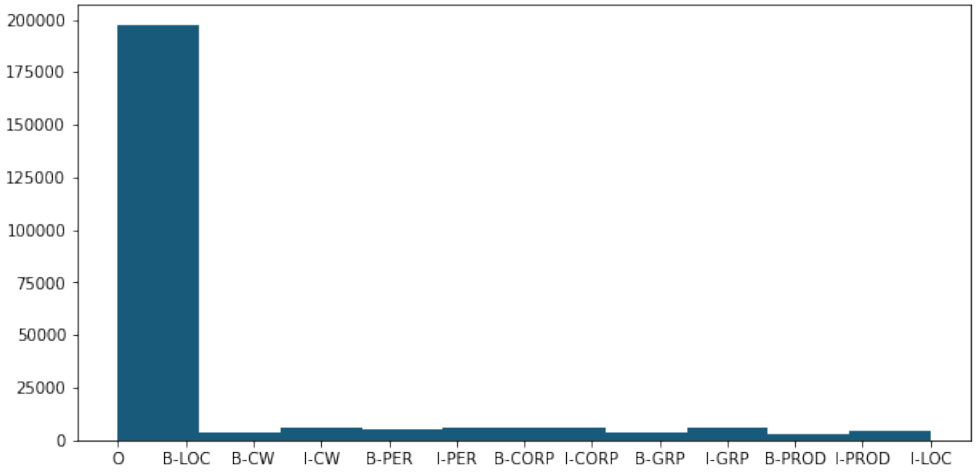

Figure 1: Class distribution: Main dataset

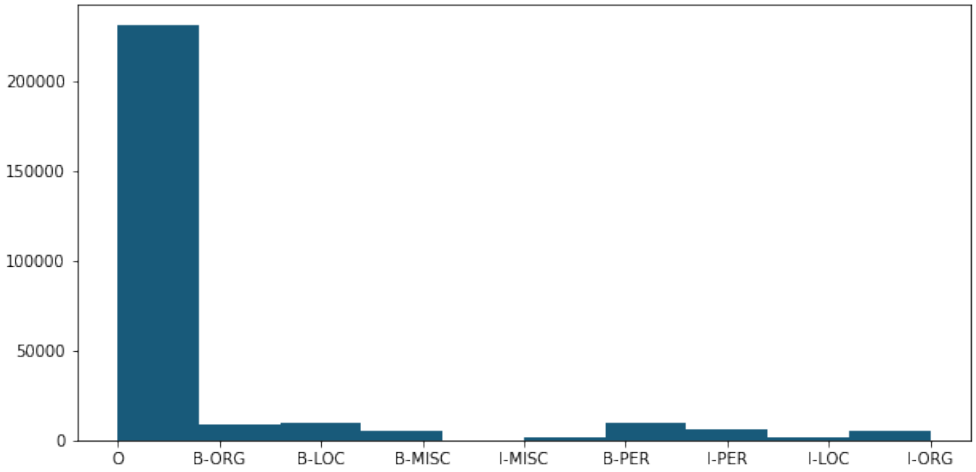

Figure 2: Class distribution: CoNLL2003++

Model	**Dev F1**	**Dev Loss**
Base Model	67.58%	3.663%
Base Model+ Char	69.53%	4.763%
Base Model + Char + Additional Datasets	**70.96%**	4.341%
Base Model+ Char+ Global+ AdditionalDatasets	70.62%	**3.544%**

Table 1: Models results

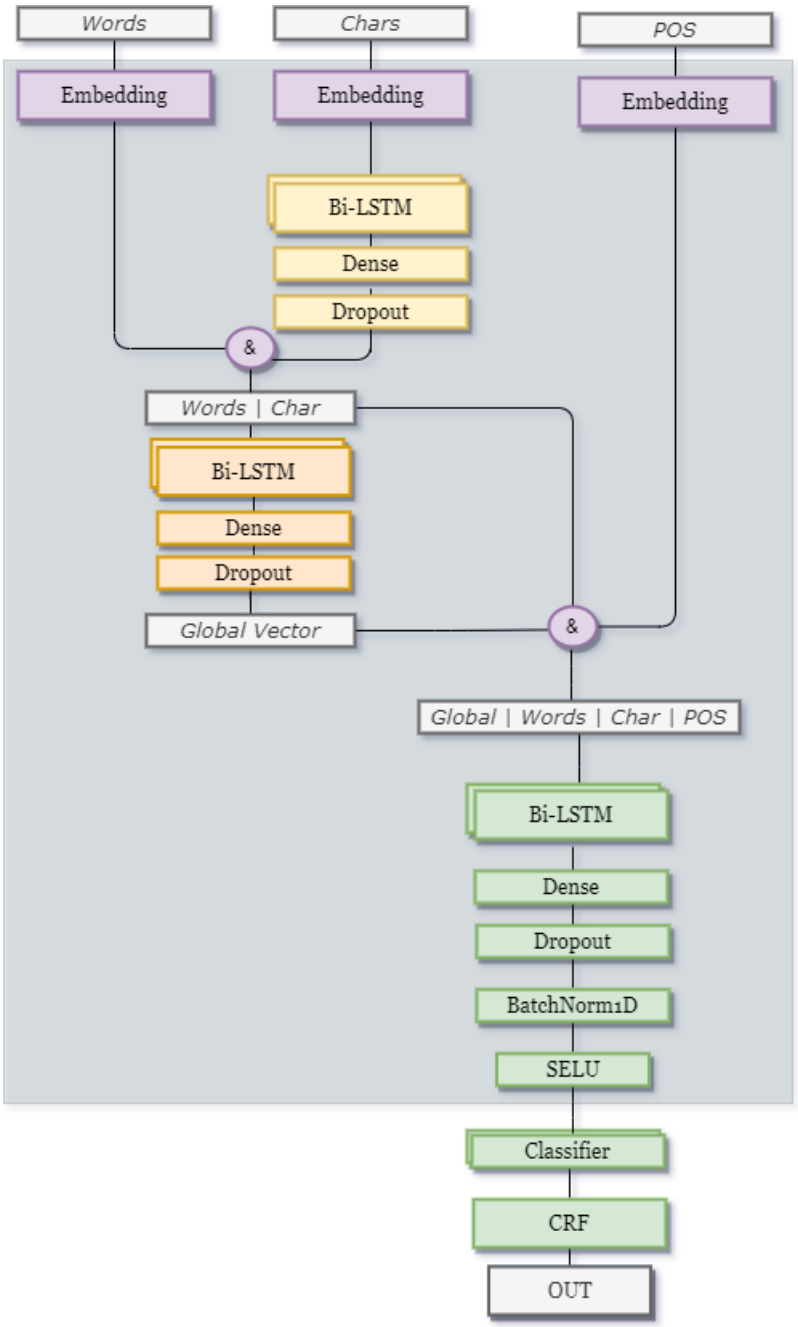

Figure 3: Model diagram: Darker box indicates the model that has been trained also on CoNLL

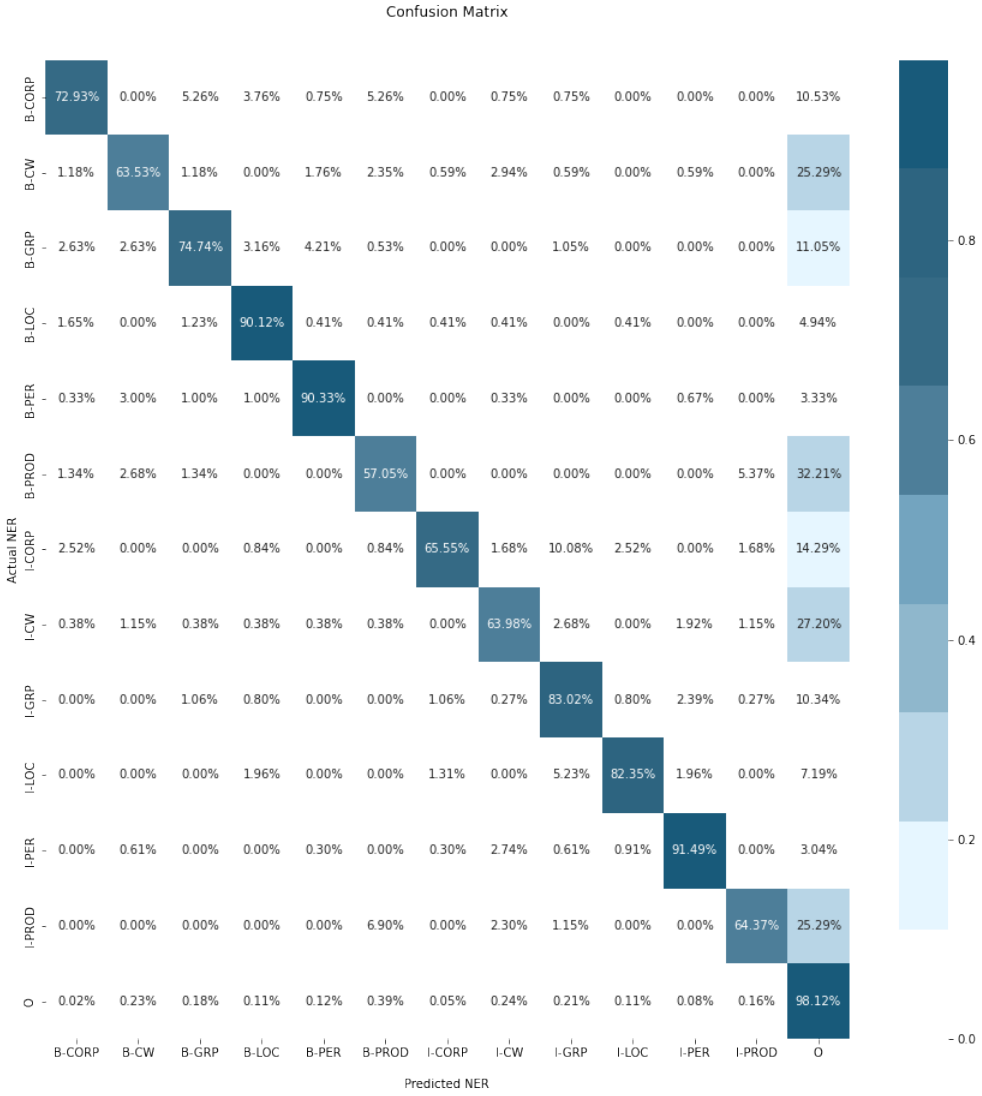

Figure 4: Confusion matrix: Third iteration (Best F1)

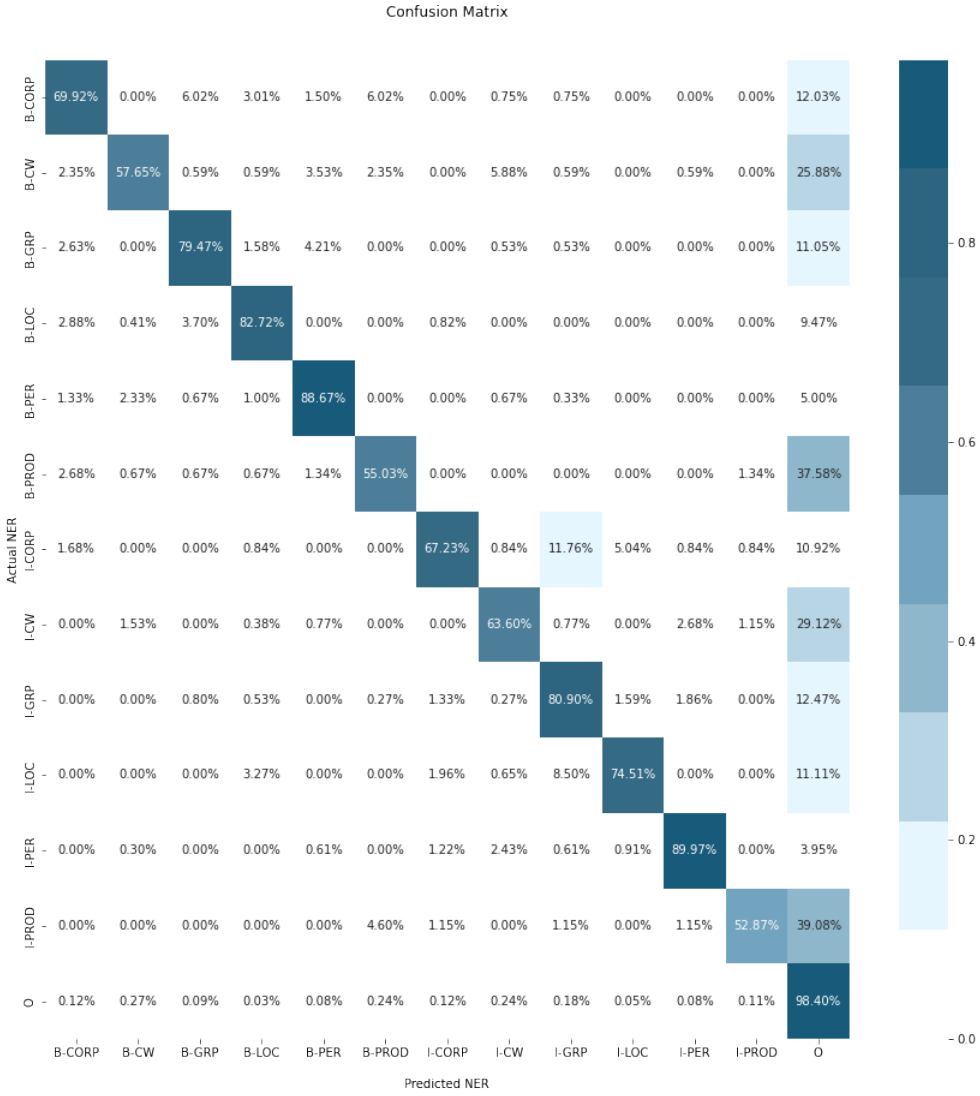

Figure 5: Confusion matrix: Fourth iteration (Best overall)

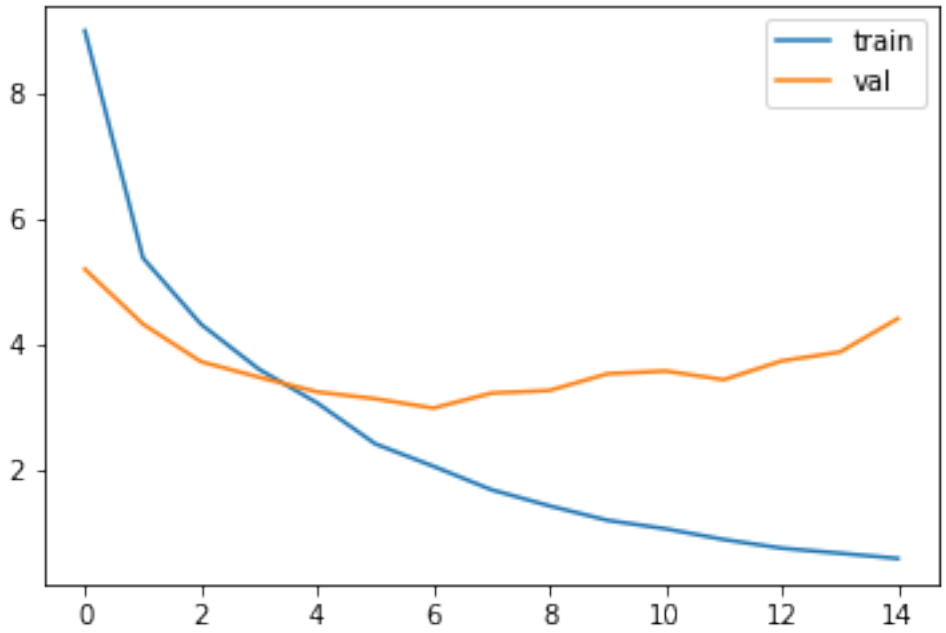

Figure 6: Loss: Fourth iteration

References

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the WNUT2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 140–147, Copenhagen, Denmark. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

John D. Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Robert Guthrie Pytorch. Making dynamic decisions and the bi-lstm crf.

Zihan Wang, Jingbo Shang, Liyuan Liu, Lihao Lu, Jiacheng Liu, and Jiawei Han. 2019. Crossweigh: Training named entity tagger from imperfect annotations. *arXiv preprint arXiv:1909.01441*.