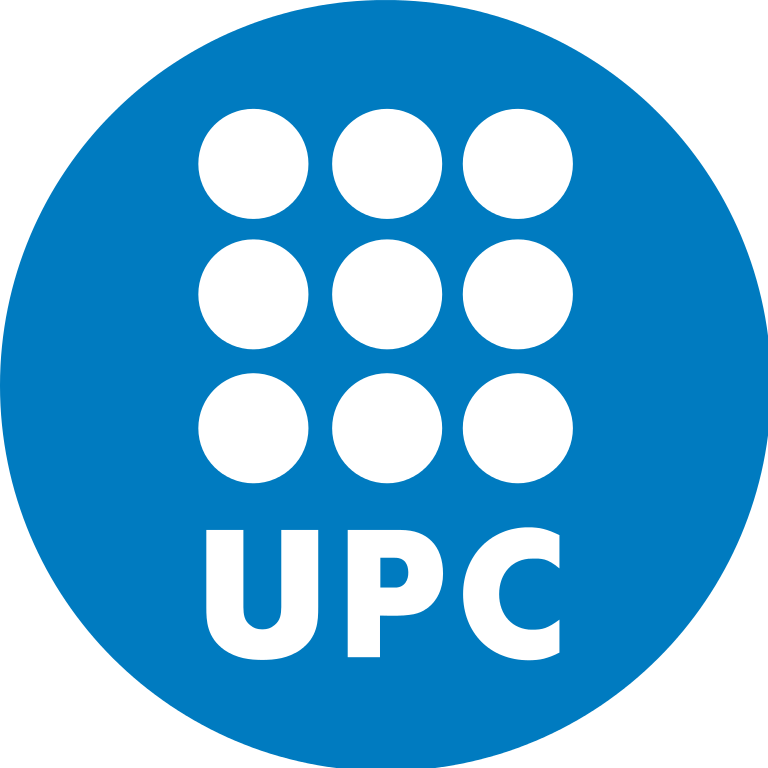
**UNIVERSITAT POLITÈCNICA DE CATALUNYA**

**BARCELONA SCHOOL OF INFORMATICS**

**Official Master on Data Science**

****

Lab 4: NN-based NERC

**Mining Unstructured Data**

**Albert Martín Garcia**

**Ximena Moure Alassio**

**Maria Paraskeva**

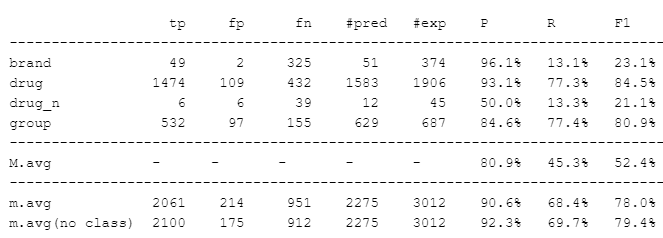
Barcelona, June 2023

### Introduction

This project entails developing a solution to the same problem as in the second task, but relying on deep learning models instead of machine learning solutions. We have described each step we have taken in order to improve the given baseline model and attempted to explain the reason behind the variation in accuracy by comparing each result to the previous ones.

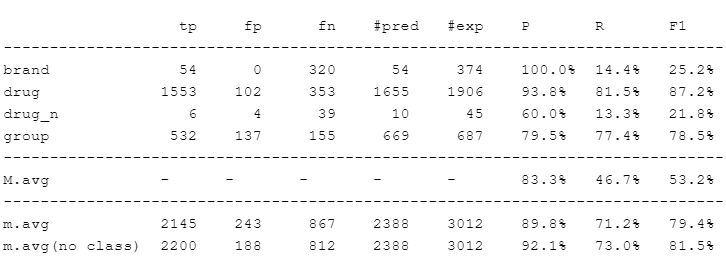
### Neural Networks NERC

For reference, executing the given notebook with our seed gives us the following initial results, which we will build upon:



**Embeddings**

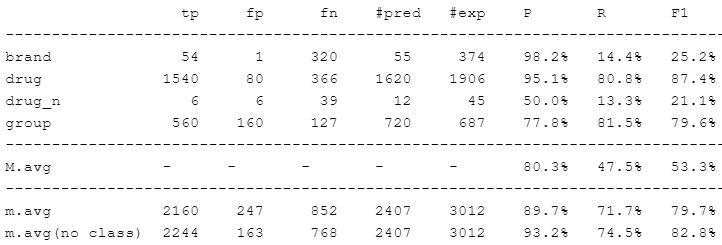
The first thing we tried was to increase the dimensionality of the embedding layer in the network by a factor of ten. We decided to perform this drastic change after a series of iterations with a different embedding size, obtaining a steady but decreasing improvement. It provided the network with more expressive power and resulted in an increased performance with a macro average score of 53.2%.



**LSTM repetitions**

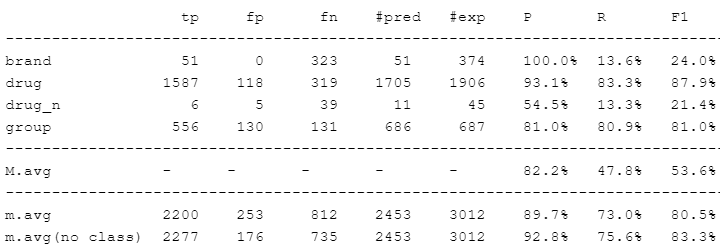
The number of repetitions in the basic model was set to 200. By lowering it to 100 we get a slightly worse performance than the previous model with the macro average score to achieve 53.1%

While expecting better results, raising the repetitions to 1000 only improved the score by 0.2% as it can be seen in the result table below. Since the dataset is not very complex nor very large, the model may have already learned most of the relevant patterns in the data with 100 repetitions, so increasing them did not lead to significant improvement in performance.



**Max length and Suffix length values**

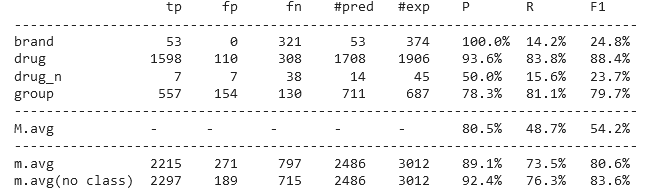
Next, we tried to play with the maximum word length and the suffix length in order to see if we can achieve different results. By setting max length to 50 and suffix length to 4, the obtained result went up to 53.6%. We then tried with the same max length but with suffix length set to 3, but the score dropped back to 53%.



**Dropout**

Next, we attempted to make the model more robust by increasing the dropout probability in both the embedding layers and in the LSTM units. As we have chosen a very large value for the embedding dimension, we felt comfortable setting this proportion to a high value such as 50%.

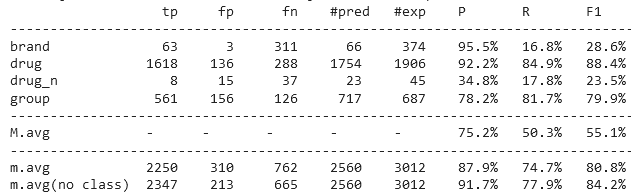
Indeed, instead of lowering the accuracy due to using less features when training, we obtained a small increase in the macro F1 due to the increased robustness resulting from having redundant features.

****

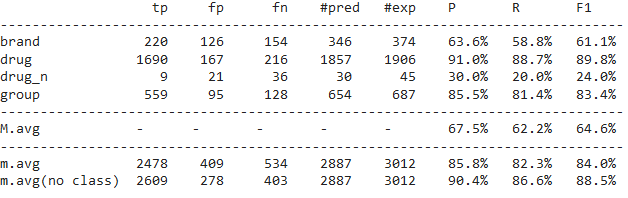
**Output layer complexity**

We hypothesised that a single linear combination of the hidden states at each timestep would not be optimal to perform predictions, as there may be more nuanced decisions to be made. Thus, we added an intermediate dense layer before the final dense+softmax, with output dimension twice the size of the output features.

Indeed, this proved to be a beneficial addition, as it increased the macro F1 score by a point, meaning that we were correct in hypothesising that more complex decisions were required in order to determine class probabilities.



**Adding features**

At this point it is clear that simply tweaking the model will not lead to significant improvement. Thus, we decided to extract more features for each token of the sequence. Initially, we chose to extract a prefix of same length as the suffix and the length of the word as a categorical feature as well. While it is true that maybe having it as numerical will allow the model to better extrapolate to longer words, we believe that using a different embedding for every word length will be better, as it offers greater distinction between words of short length, which are the majority. We also added the same embeddings used in the previous labs, them being the dash count, number of digits, and the presence of each token in the CSV files. Using 300 features for each of the embeddings, the results increase significantly:  


**Optimizer**

We first tried to change the optimizer to Nadam.This optimizer is a combination of Adam and Nesterov accelerated gradient descent. It adds Nesterov momentum to Adam, which can improve convergence speed. Unfortunately, the results were not as good as expected, with a total drop of 2.6% in the macro average F1 score.

In search of a better optimizer, we opted for Adamax. It is a variant of Adam that uses the L-infinity norm instead of the L2 norm to scale the gradient. The results were even worse with a 41.3% macro average so, compared to all three, the initial Adam optimizer was the best option.

