## Participants

Tiberio **Marras**, Gaetano **Signorelli**, Daniele **Sirocchi**

## Abstract

In this assignment we created and tested neural architectures for solving the **POS tagging** task. We worked for finding the best way to pre-process the data, managing OOV words, setting the correct model hyper-parameters, and evaluating the different architectures we had to implement. Due to a dataset heavily affected by class unbalance, we used f1 score as evaluation metric of our models.

## Task description

The goal of POS tagging is to classify each word of a sentence/document, in order to say which is its tag. One way for solving this task with a neural network is to use a RNN that returns all the hidden states, and eventually apply a classification head on each returned hidden state.

## Models

We created 4 models: we started from a baseline, and then created variants of it. The following image shows the different models (highlighted in red the variations from the baseline model):

Diagram, text

Description automatically generated

After some empirical tuning, we decided to fix some hyper-parameters so as to focus our attention on the architecture performances.

In particular, we fixed the **batch size**, the **number of units** of each layer and the **embedding size**. The number of **time steps** and the **number of classes** are obtained dynamically while processing the dataset (further details can be found in the code).

Last but not least, it important to remark that we created the train, validation and test set incrementally, managing OOV words as in a real-world situation.

## Validation and test

We trained the four models descripted in the previous section, selected the two with the highest **accuracy** value on the **validation set**, and then tested them on the **test set**.

All the models were able to reach a very good accuracy level (~90%), but the **variation 2** and the **variation 3** models were faster to train with respect to the baseline model, and also obtained a slightly higher accuracy with respect to all the other models.

The following are the loss and accuracy curves of the **variation 3** model recorded during the training:

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

The loss and accuracy curves of the other models have a very similar trend, and they can be found in the shared notebook.

For making the training faster and having higher performances, we discovered that it was important to **split the dataset into sentences**. Moreover, we also applied the **early stopping** technique to avoid overfitting.

Finally, we run the **variation 2** and **variation 3** models on the test set for assessing their performances, and we obtained very similar results:

|  |  |
| --- | --- |
| **Variation 2** F1 SCORES:   * macro: **0.73** * micro: **0.52** * weighted: **0.82** | **Variation 3** F1 SCORES:   * macro: **0.70** * micro: **0.52** * weighted: **0.83** |

Due to the class imbalance problem, we considered as the most relevant score the **micro f1**, and it depicts an important performance difference from what we have obtained during the training on the validation set.

In the shared notebook there are a **classification report** and a **confusion matrix**. Those are very useful for better understanding how the models are performing on the test set, and for comprehending on which classes they are working correctly or incorrectly. Furthermore, they were fundamental for the error analysis made in the following section.

## Error analysis

The main problems we have found are due to **class imbalance** and a relatively **small dataset**, which cause the model to be easily fooled by similar tags (e.g., their position in an English sentence could be easily exchanged), and it strains to correctly classify tags which do not have many samples in the training set.

The image below shows only the distribution of the classes on the train set. However, the validation and the test sets have a very similar trend.

The *Discussion and Error Analysis* section of the shared notebook has many details about these problems.

Chart, histogram

Description automatically generated