## Participants

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

## Abstract

In this assignment we created and tested neural architectures for solving the **fact checking** task. We worked for finding the best way to pre-process the data, coping with class imbalance problem, setting the correct model hyper-parameters, and evaluating the different architectures we had to implement. Eventually, we evaluated the models by considering accuracy, recall, precision and f1-score metrics.

## Task description

The goal of fact checking is to classify a fact, by saying whether it is true or false. More specifically, in this assignment the positive class is labelled as "SUPPORTS", while the negative class is labelled as "REFUTES". For training the model we had at our disposal a set of claims (aka: the facts) and, for each of them, also one or more evidences. Claims and evidences together represent the input of the model.

One way for solving this task is by means of a neural architecture and, in this assignment, we created and tested different versions of it.

## Models

We substantially created a single model which contains 2 parts that can have several variations. The following image summarizes what we have implemented (highlighted in blue and green the varying parts of the model):

Diagram

Description automatically generated

The possible variations and the number of hyper-parameters is huge. Therefore, we set a baseline model just by fixing the sentence embedding and the merge blocks to their first variations, and then tuned several hyper-parameters (e.g., embedding size, number of units for hidden layers, batch size, ...).

During this phase, we discovered that the model was clearly overfitting. After some empirical testing, we eventually added regularization techniques. In particular, we added **l2** and **dropout** in the *sentence embeddings* and *classification* blocks.

**Finally**, once the best baseline model was found, we also tested all the possible model variations (those in green and blue in the above image) and their combinations.

## Training and test

We trained all the models mentioned in the previous section, and finally selected the one that was better performing on the test set.

The following image depicts how are the loss and the accuracy during the **training** of the best model we found so far:

Chart, line chart

Description automatically generated Chart, line chart

Description automatically generated

It is remarkable an **overfitting** issue. Even though we applied regularization techniques for reducing its effect, the network starts to perform badly on the validation set after just a few epochs. In this case, the higher accuracy found is 0.74, and it has been obtained at the epoch 10.

What follows are the results on the **test set**:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CLASSIFICATION REPORT | | | | | CONFUSION MATRIX |
|  | **precision** | **recall** | **f1-score** | **support** | A picture containing text, screenshot, electronics  Description automatically generated |
| **0** | 0.76 | 0.66 | 0.71 | 3583 |
| **1** | 0.70 | 0.79 | 0.74 | 3606 |
|  |  |  |  |  |
| **accuracy** |  |  | 0.73 | 7189 |
| **macro avg** | 0.73 | 0.73 | 0.72 | 7189 |
| **weighted avg** | 0.73 | 0.73 | 0.72 | 7189 |

We were able to obtain these results only after applying a **threshold translation** for deciding the final class of a prediction. We chose to do this because the training dataset is not balanced, and it is biased to the positive class, which is much more represented. Without this threshold correction the results were worse by far.

## Error analysis

We have found 2 major issues: the train set suffers from **class imbalance**, and the model **overfits** very quickly, without being able to improve so much the performances on the validation set. This is also reflected on the test set, were we observed similar score metrics.

**Concluding**, our thought is that we implemented the best model we could, and different strategies should be applied for improving the results (e.g., use different sentence embedding techniques). More details on this are on the last section of the shared notebook.