Project : Supervising learning

Haution Matthieu

Github link : https://github.com/Matt0408/Project-supervised-learning

1. Introduction

1.1. Dataset and task

This project will be based on the "goemotion" dataset, which is a dataset of TensorFlow where we can find comments and emotions. The different comments come from the website Reddit, and are associated to the 27 categories of emotions or neutral.

The task here will be to build a supervised learning model able to recognize emotions present into a comment.

1.2. Objectives of the project

This project have for objectives to:

- Build a model able to recognize emotions present in a comment
- Improve the model
- Tunes hyperparameters

The aim is to have a coherent model with the proposed subjects.

2. Proposed pipeline

2.1. Preprocessing

The first step is to split our dataset into a training set, a test set and a validation test in order to have all sets needed to build a model.

The second step is to do the preprocessing part, the aim of this part is to build datasets that can feed our model correctly.

For the problem we are trying to solve the preprocessing will consist to take all comments as our X-vector and all the labels as our Y-vector. For the labels we have a list with 0 or 1 depending on if the emotion is present or not into the comment.

Then we will use a tokenization process in order to break a stream of text into individual words or meaningful elements, it also help to filter out noise from dataset. We will use “OOV” to handle out-of-vocabulary words.

2.2. Model

Once those steps are in place, we can start to setup a first baseline model by defining all layers we want, then train it before fitting it and choosing a loss function.

The first model will have the following layers:

- An embedding layer as input layer
- A LSTM layer with 64 hidden units because we are trying to do a text categorization and it is a good choice when we are trying to capture the complex dependencies for understanding the meaning of the text
- A bidirectional layer to consider the full context
- A dense layer as output layer with 28 neurons as we want to categorize our dataset into 27 emotions and neutral

As loss function, we will use the categorical cross entropy because we are on a problem of multiclass classification.

We will observe what result we have with this first model and after that we can try to manipulate layers and hyperparameter to improve the model.

3. Experimental methodology

First of all, a thing we can tune is the metric we used : for our first model we used the classification accuracy because it seems to be the best way to have a representative result. We can try to use a confusion matrix in order to identify biases in the model’s predictions.

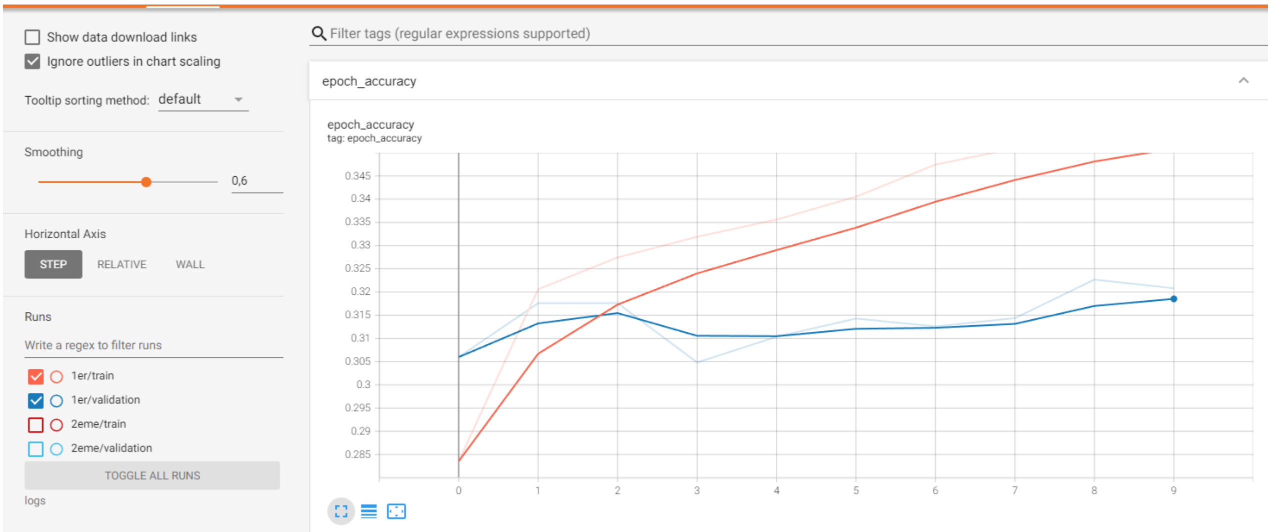
The next step to improve the model will be to use other models and modify the layers we used. We will try to add a convolutional layer as it is a good layer for processing text data.

We can also try to use a second bidirectional layer and try to use more neurons. Let’s see if it improve the model and then test it to see if we are not going to overfit our data.

The initial hyperparameters can also be tuned such as the vocabulary size, the max length of sequencies and embedding dimension.

4. Result and discussion

4.1. First baseline model



☐ Show data download links
☑ Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing 0,6
Horizontal Axis: STEP

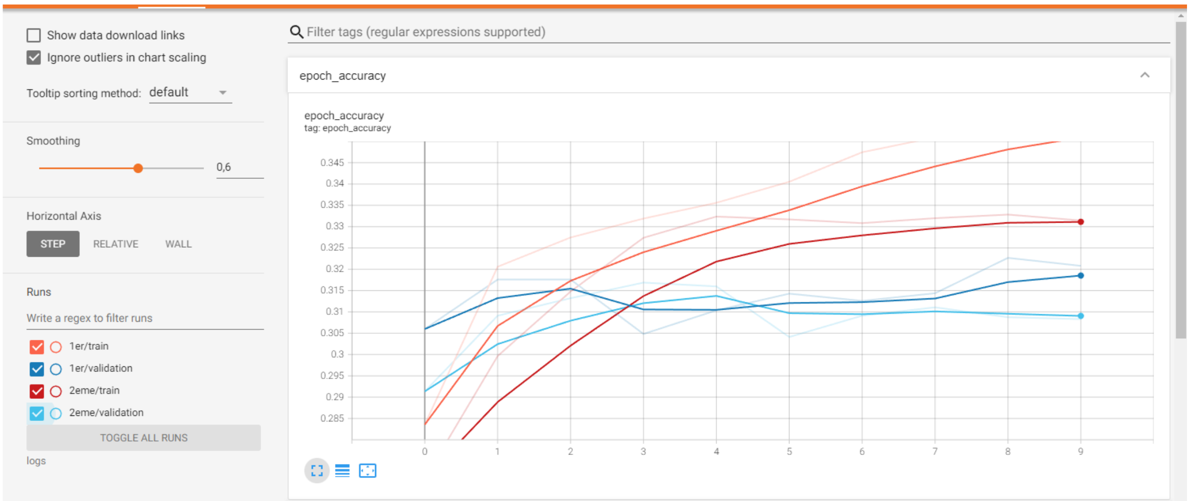
Runs
☑ 1er/train
☑ 1er/validation
☐ 2eme/train
☐ 2eme/validation

Here we have a maximum accuracy 35%, which is not the expected result. And the gap with the validation set is a bit to high.

4.2. Layer and hyperparameter tuning

As described before, I added a convolutional layer and modify the neurons size in order to try to improve the model.

The following graph shows the accuracy of this second model (few random investigation about the layer size have been made but it is not relevant to show every single model that have been made).



☐ Show data download links
☑ Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing 0,6
Horizontal Axis: STEP

Runs
☑ 1er/train
☑ 1er/validation
☑ 2eme/train
☑ 2eme/validation

Here we have a closer gap to validation set but the accuracy is also a bit lower.

About hyperparameter, let's try to augment the max length of sequencies, the vocabulary size don't have a big impact as it is a relevant parameter of the language and 100 000 cover almost the entire English language.

The impact is negligeable so we will keep the initial ones.

On the github, you can also see the model called number3 with a second bidirectional layer but the score doesn't improve much and we are overfitting a little bit so the test score is lower.

4.3. Main difficulties and chosen model

The main difficulties encountered to train this model was about the choice of hyperparameters and layers size. In fact when we tunes hyperparameter and change the width and depth of our neural network we see that there is or overfitting or no major improvement, only the gap to validation set is lower. The maximum accuracy achieved on test set is : 33%

So we will take the 2nd model to do our task as it is the one giving the best test score.

5. Conclusion

A model with a convolutional layer and a bidirectional layer seems to be the most appropriate pour our task even if we didn't succeed to reach a high score with this model, other models are worse.

As other improvement we could also try to implement modification of weight of neurons and neurons removing in order to have a better accuracy, putting in place other batching method with mini-batch can also be a solution, this are the next step if we want to have a better score