Chitu Stefan-Catalin

Machine learning text classification

Colab link:
https://colab.research.google.com/drive/1O6u1s2JaaVOZ5MiFot_v6Zn7F2ujv_5n

1) Data preprocessing:

Because the data was encrypted, I decided not to change characters format. For example “A@” will be treated as different from “a@”

No non-literary character is eliminated.

2) Features extraction:

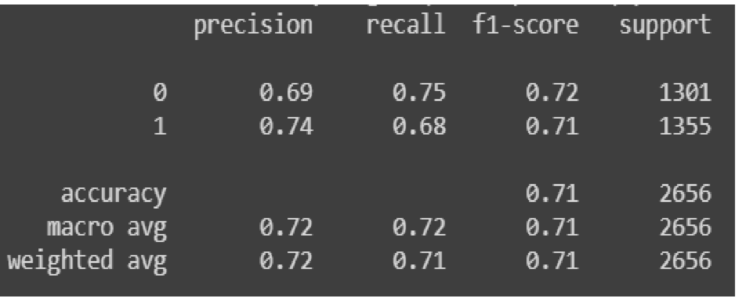
To get numeric values from the text I used tf-idf, the default version from sklearn: TfidfVectorizer.

The n-grams gave a better accuracy than the simple words, the most efficient variant was the 6-character n-grams. I used the ‘char_wb’ parameter to only get the n-grams inside words.

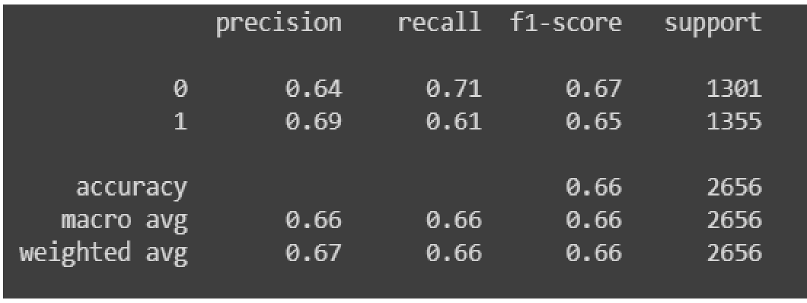
To normalize the data I used l2.

N-grams

Full words



```
              precision    recall  f1-score   support

           0       0.69      0.75      0.72      1301
           1       0.74      0.68      0.71      1355

    accuracy                           0.71      2656
   macro avg       0.72      0.72      0.71      2656
weighted avg       0.72      0.71      0.71      2656
```

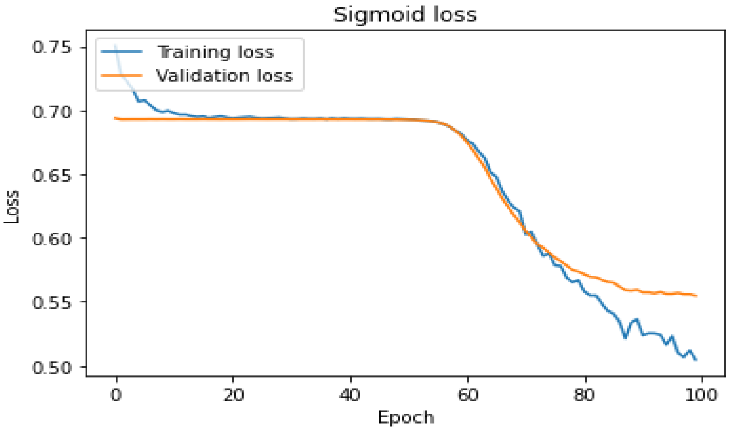


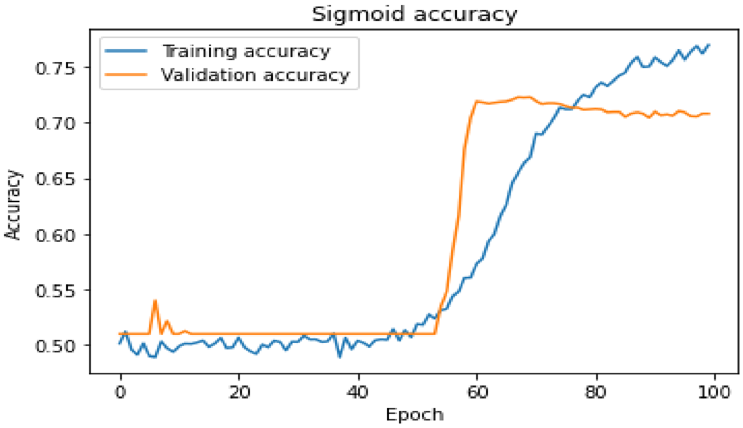
```
              precision    recall  f1-score   support

           0       0.64      0.71      0.67      1301
           1       0.69      0.61      0.65      1355

    accuracy                           0.66      2656
   macro avg       0.66      0.66      0.66      2656
weighted avg       0.67      0.66      0.66      2656
```

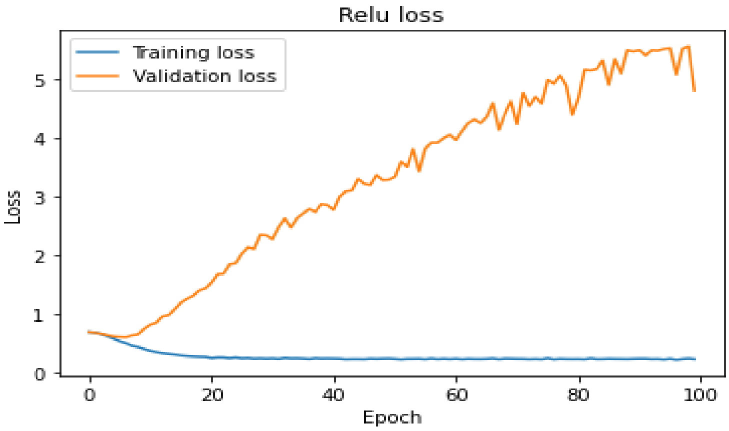
3) Neural network model:

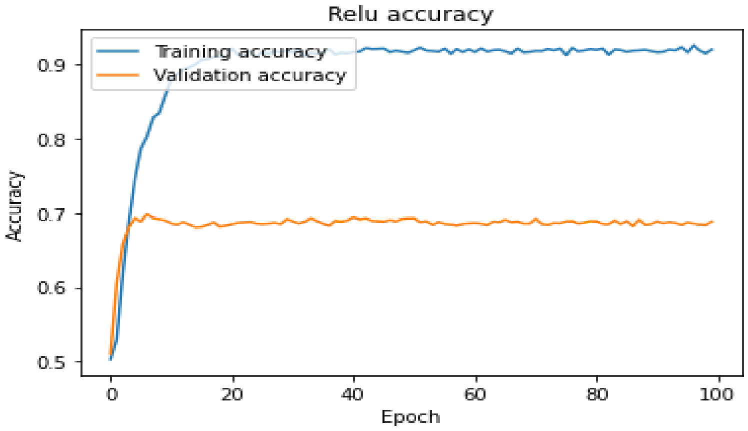
I used keras to easily implement a mini network with two hidden layers. With a small number of perceptrons per layer I had good results, otherwise the model would be overfitting very quickly. The optimal number of epochs I have found is 100.

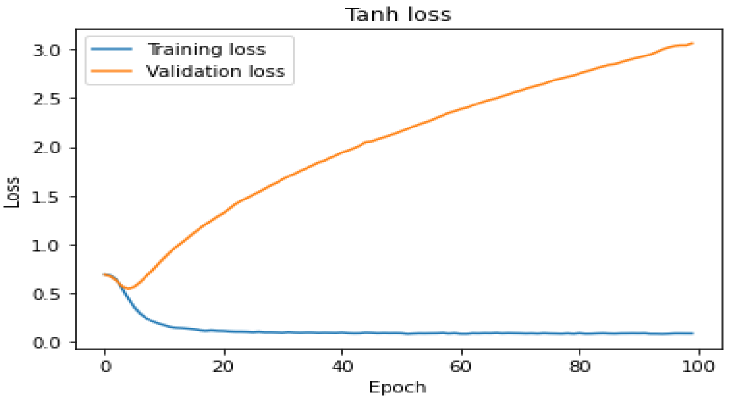
For the last layer I used the sigmoid activation function, for the rest I tried several variants.

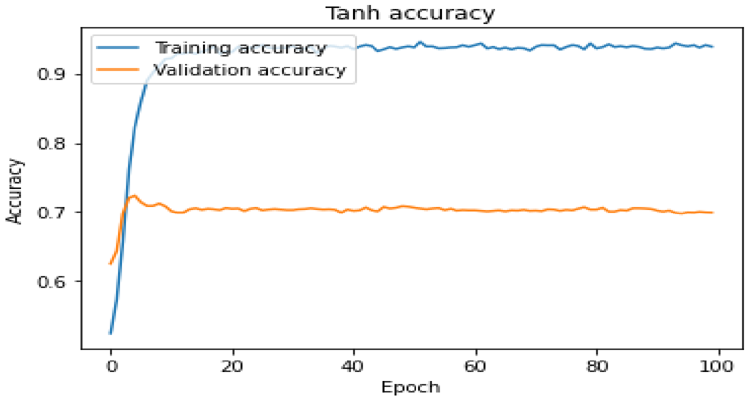










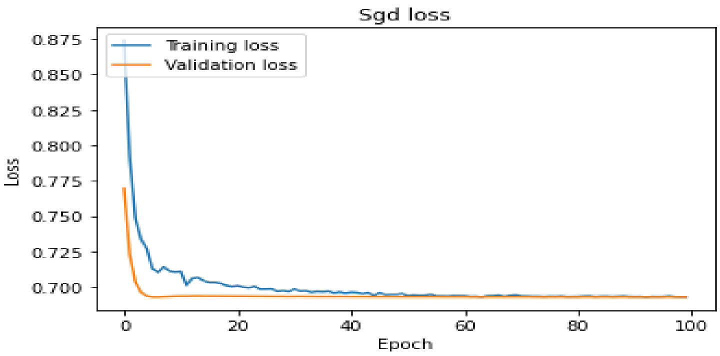


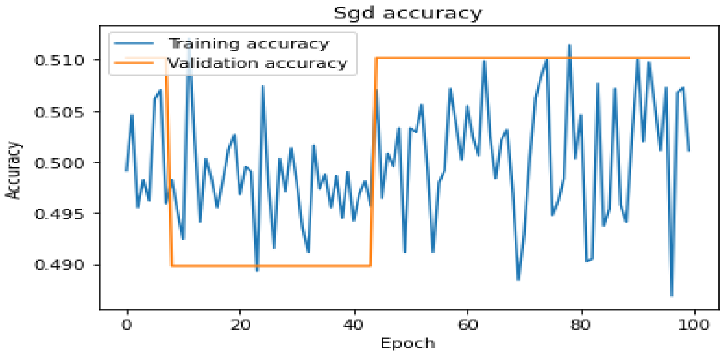
As can be seen from the graphs, the best model was the one with sigmoid.

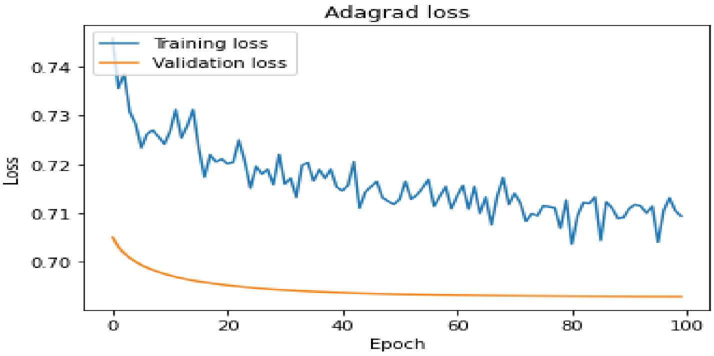
I’ve also used a dropout layer (with a value of 0.5) to prevent overfitting.

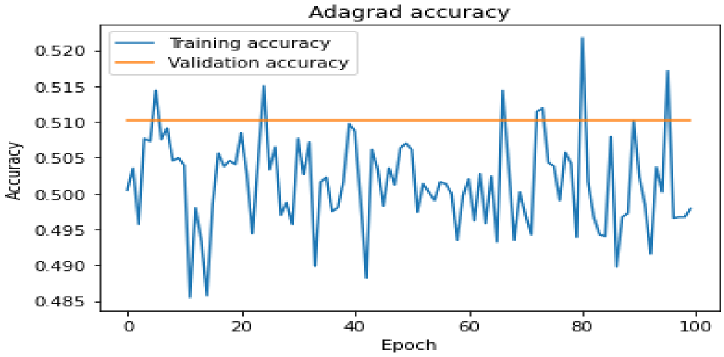
As for the loss function most of those offered by keras had similar performance, but the best was binary_crossentropy.

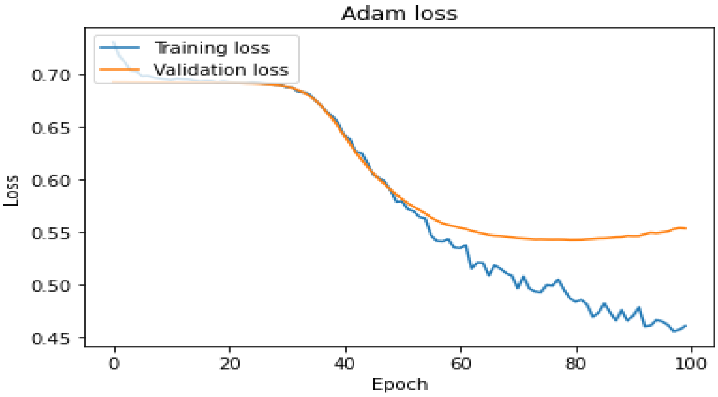
The situation is different for gradient descent functions. Sgd and adagrad both are having a poor performance both on training and on validation sets, on the other hand adam gives good results.

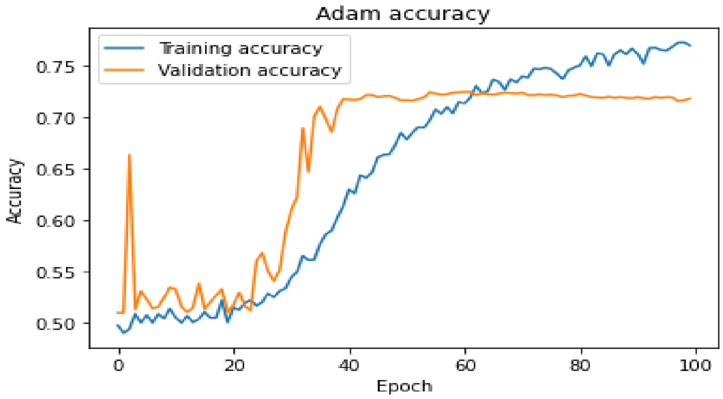






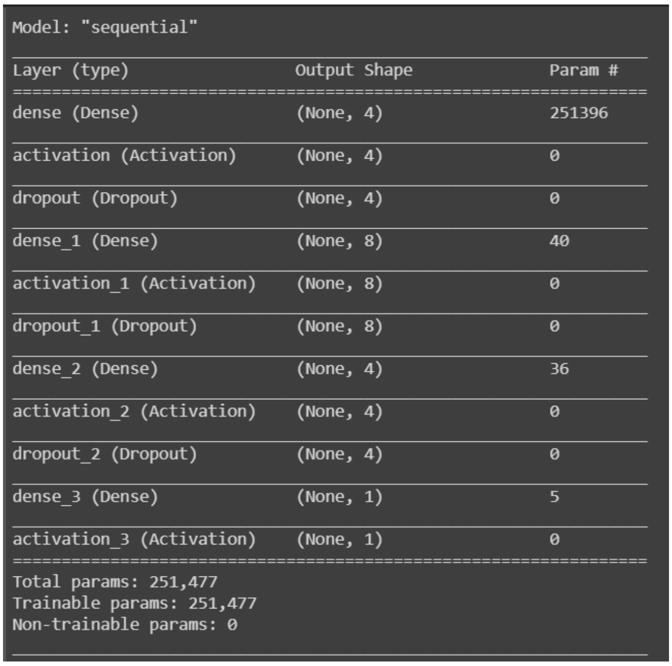






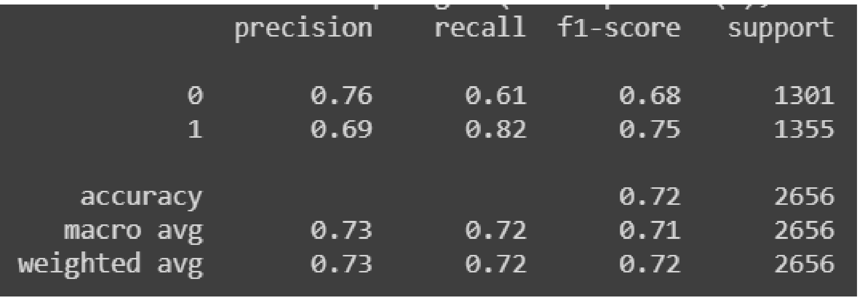
Chitu Stefan-Catalin

Model Summary



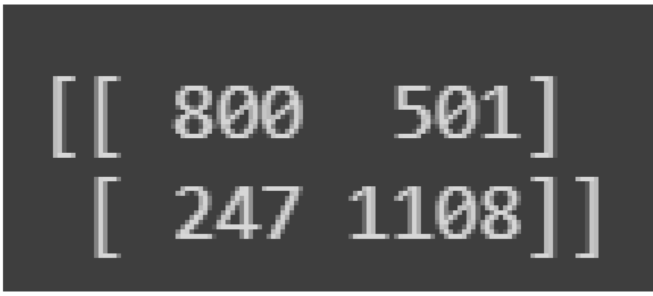
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 4)                 251396
_________________________________________________________________
activation (Activation)      (None, 4)                 0
_________________________________________________________________
dropout (Dropout)            (None, 4)                 0
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 40
_________________________________________________________________
activation_1 (Activation)    (None, 8)                 0
_________________________________________________________________
dropout_1 (Dropout)          (None, 8)                 0
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 36
_________________________________________________________________
activation_2 (Activation)    (None, 4)                 0
_________________________________________________________________
dropout_2 (Dropout)          (None, 4)                 0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 5
_________________________________________________________________
activation_3 (Activation)    (None, 1)                 0
=================================================================
Total params: 251,477
Trainable params: 251,477
Non-trainable params: 0
_________________________________________________________________
```

Classification Report



```
              precision    recall  f1-score   support

           0       0.76      0.61      0.68      1301
           1       0.69      0.82      0.75      1355

    accuracy                           0.72      2656
   macro avg       0.73      0.72      0.71      2656
weighted avg       0.73      0.72      0.72      2656
```

Confusion Matrix



```
[[ 800  501]
 [ 247 1108]]
```