University of Thessaly
Department of Electrical & Computer Engineering

Neurofuzzy Computing

Winter Semester

Semester's Project

PROFESSOR
Katsaros Dimitrios

STUDENTS
Koukougiannis Dimitrios
AEM 2537
dkoukougiannis@uth.gr
Tsoukas Panagiotis
AEM 2650
ptsoukas@uth.gr

Volos, February 2022

Contents

Abstract

The twenty first century is without doubt the century of information. Using a device, many people are in the position to access information easily and most of the times free. Unfortunately, many people and organizations take advantage of this trend and they try to manipulate the public by publicing fake news. In this project, we will try to tackle this problem by creating a neural network which will predict, which titles correspond to fake news and which not. For this purpose we will use the python programming language and the Keras framework.

1 Used layers

We will start by mentioning the used layers of our architecture, that will be used to make the prediction.

GRU

First mentioned layer is the Gated Recurrent Unit (GRU). This specific unit is very similar with the LSTM and it consists of the reset gate and the update gate. More specifically the first one determines how many previous inputs are going to be used as memory. The advantages of this architecture is that it is trained much faster than the traditional LSTM layer. Furthermore, it is simpler and that's why it was used on this case.

Multilayer Perceptrons

This architecture is the older one. It consists of neurons which calculate a local sum by multiplying previous inputs with some variables (weights) and adding some others called biases. After that, we use an activation function which multiplies each sum using a function (e.g linear). It is common to use the vector notation and multiply each calculated value with a matrix. We use multiple layers of neurons.

2 The architecture

The architecture used can be retrieved from the file Architecture.ipynb . We use an input layer, embedding layer, GRU layer and MLP.

Input layer

Receives the inputs (sentence with the title of the examined article) as a vector of numbers mapped with each distinct word.

Embedding layer

Its input is the examined sentence and its output a 64-number vector.

GRU

A gru layer consisting of 50 units. Activation function: Tanh.

Multilayer perceptron

Four layers are added. First one has 60 neurons densely connected with the outputs of the previous layers and a Relu activation function is applied. The second layer has 30 neurons with a tanh activation functioned applied. The third layer is consisting of 10 neurons with a Relu activation function used and the last layer has 2 neurons. For this specific layer a sigmoid function is applied. If the first neuron is fired that means that the article isn't fake news and if the second neuron is fired the article is predicted fake. Use activation functions: relu, tanh, relu and sigmoid.

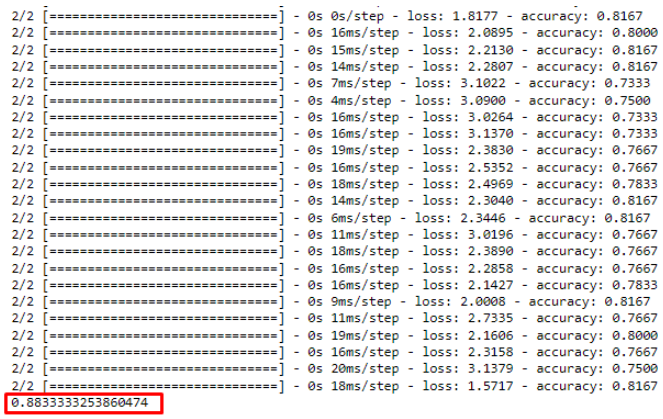

Figure 1: Final accuracy

3 Training algorithm

After trying many optimizers and losses for the training process we decided to pick the Adam optimizer and choose the binary crossentropy as our loss function. Fifty epoches are executed and during each one of them, a batch of size 40 is chosen. Note that, binary crossentropy is ideal for classification progress.

4 Final Trainable Values

Due to the fact that initial trainable parameters are assigned randomly, multiple iterations were executed and we stored the maximum accuracy for the test data. In Weights Data.txt file, the final trainable parameters are stored as they were printed by our program.

5 Measured Accuracy and Timing

After the above iterations, a graph module which plots the measured accuracy.

On figures 1 and 5 it is shown that maximum accuracy achieved was **88.33%**. Furthermore, a graph of the elapsed time for the training is depicted.

Meanwhile on figure 2 the elapsed time for each step is shown for the testing 20% of test data. For each step, the total time is calculated.
Finally on figure 3 elapsed time for each epoch is shown.

6 Measured Losses

On figure 4, the calculated loss for the training phase is depicted. We conclude that during the training process, the loss function is between 0.09 and 0.12. On figure 6, it's clearly shown that for each distinct iteration during the test phase the loss function ranges from 0.5 and 3.2 approximately. The smallest loss was detected when the accuracy was measured **88.33%**.

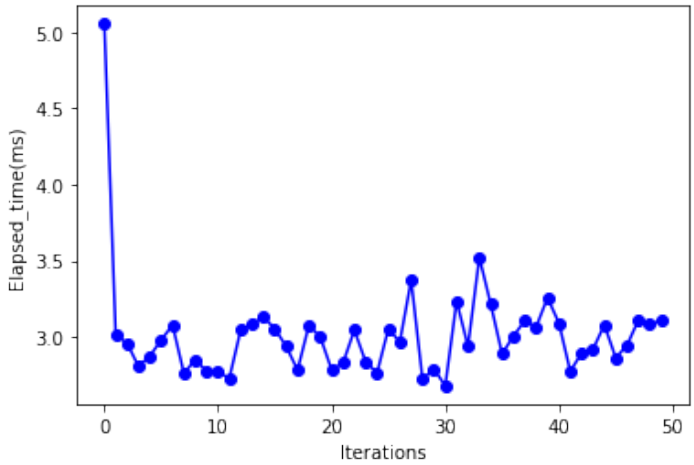

Figure 2: Elapsed time plot for many iterations

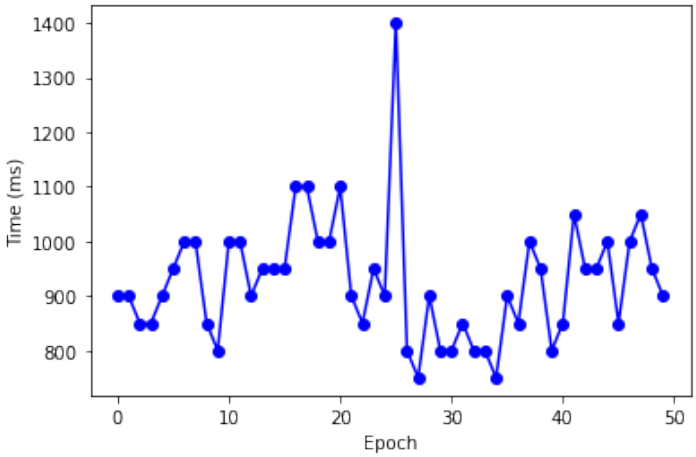

Figure 3: Elapsed time plot for each epoch

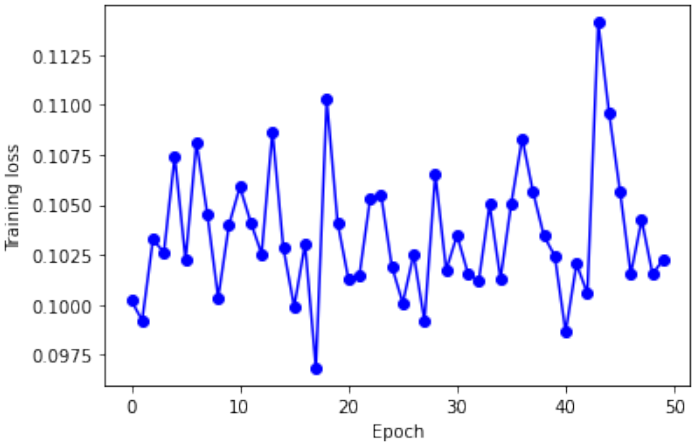

Figure 4: Training loss

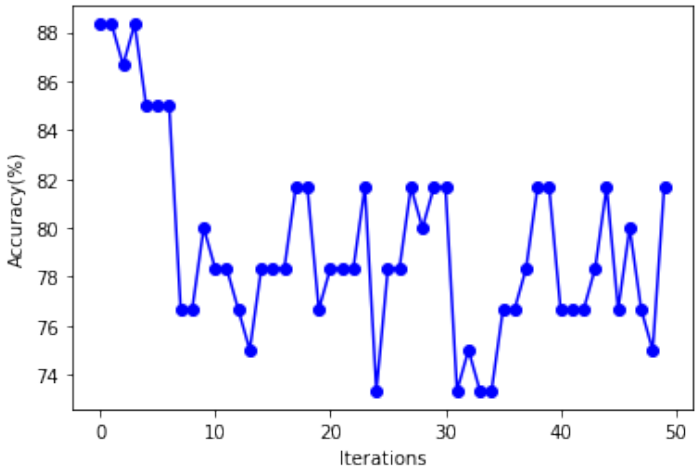

Figure 5: Accuracy plot

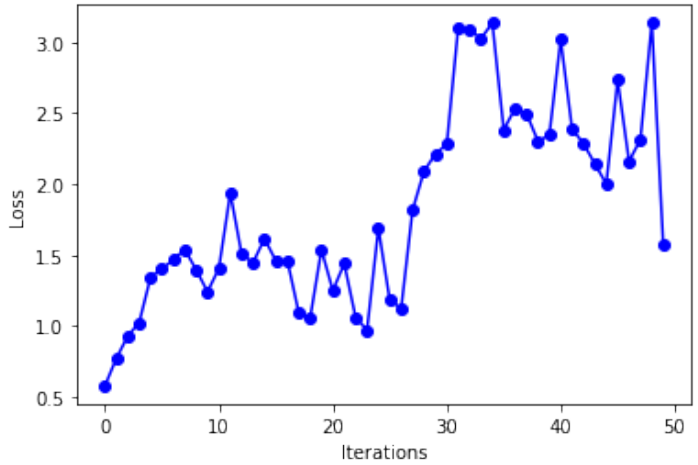

Figure 6: Loss plot

7 References

Keras: https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/

Tokenize() Function: https://www.geeksforgeeks.org/python-nltk-nltk-tokenizer-word_tokenize/

Punctuation Removal: https://www.geeksforgeeks.org/python-remove-punctuation-from-string/

Tokenizer Method: https://stackoverflow.com/questions/51956000/

Remove Stop Words: https://www.geeksforgeeks.org/removing-stop-words-nltk-python/

How to use word embedding-layers:
https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/

Python Documentation
https://docs.python.org/3/reference/datamodel.html