# Univ. of Athens, Dept of Informatics & Telecoms
# YS 19: Deep Learning for NLP
# Fall 2022 - Homework 3

Kyriakopoulos Dimitrios - sdi1900093

**Notes to the graders**: I provide a fully documented ipynb notebook with a link to Google Colaboratory. In order to run the models on the test dataset

1. Go to the "Load the dataset" section (Fourth code cell of the Notebook).
2. Change `DATA_PATH` value to the path, where the file with the data for training is located.
3. Change `TESTING_DATA_PATH` value to the path, where the file with the data for testing is located.

## 1 Introduction

In this project, I perform sentiment analysis on IMDb movies' reviews. The inputs to my model are GloVe word embeddings. I classify the reviews into two classes (Positive and Negative) using a bidirectional stacked RNNs with LSTM/GRU experimenting with several parameters.

## 2 Dataset split - Build GloVe vocabulary

I split the dataset in training, validation and testing sets saving them in csv format, so as to be ontained by the TabularDataset. I define the Field object that will store information about the way of preprocessing our data. Field is used to specify how to preprocess each data column in our dataset. LabelField is used only to define the label in classification tasks. After connecting the Field objects with the columns in the dataset I import the data with the TabularDtaset which wraps all the columns (text and labels) into a single object. Then for converting text to the appropriate form for the model I used GloVe embeddings from https://nlp.stanford.edu/projects/glove/ creating the GloVe vocabulary by mapping all the unique words (20000) in the train data to an index. Then, I used word embedding to map the index to the corresponding GloVe word embedding.

# 3 Building the model

- **Init:** The model I implemented is a bidirectional RNN Classifier with LSTM or GRU cells. Firstly I load to the model the pretrained word embeddings I found with GloVe and I update it with their weights. The first layer is the embedding layer. The next layer is a linear layer. The number of the hidden size is two times of the initial hidden size because the model is bidirectional. Above these layers I added Dropout layers. The final output is being transformed using sigmoid function.

- **Forward:** Forward function defines the forward pass of the inputs. Firstly, I pass the text through the embedding layer to get the embeddings, pass it through the dropout layer, pass it through the LSTM/GRU cells, learn from both directions, pass it through the dropout layer, pass it through the fully connected linear layer, and finally sigmoid.

- **Training:** In the training I used cross-entropy loss function and the Adam optimizer. Furthermore, I use Gradient Clipping which involves forcing the gradient values (element-wise) to a specific minimum or maximum value if the gradient exceeded an expected range.

# 4 Experimenting

I experimented with the cell type, the hidden dimension, the number of layers, the dropout value, the clip value and the number of epochs. Above are the values I experimented with for each variable:

- **cell type:** LSTM, GRU.
- **hidden dimension:** 32, 64, 128
- **number of layers:** 2, 3
- **dropout:** [0,1] (float)
- **number of epochs:** [5,15]
- **clip value:** [1,5]

In order to find the optimal hyperparameters I used optuna, an open source hyperparameter optimization framework to automate hyperparameter search. The objective function i defined returns the accuracy of the model which is the metric I maximized. Because optuna is not able to understand if our model is overfitting or underfitting, I plot the learning curves of the five best trials in order to choose the trial with the best values. The best values, I found with this method for `n_trials`=50 are:

- **cell type:** GRU

- **hidden dimension:** 32
- **number of layers:** 2
- **dropout:** 0.6212660301003445
- **number of epochs:** 13
- **clip value:** 3

In Figure 1 we can see the history of all the trials, which show that most of the trials have value (accuracy) close to 90%. In addition, plotting the high-

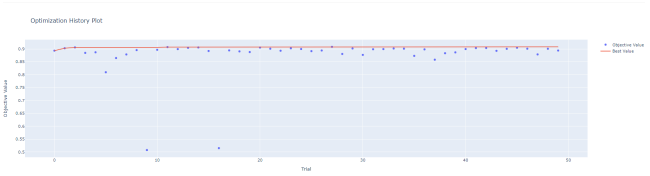


Figure 1: Trials History.

dimensional parameter relationships I got the above graph (Figure 2). Dense

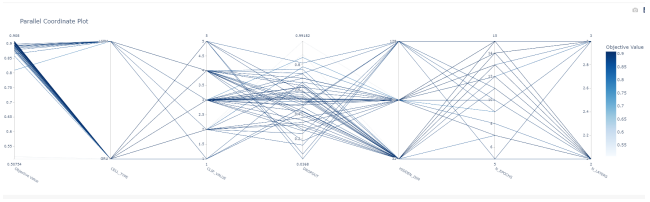


Figure 2: Parameters Relationships.

trails show the best hyperparameters, which indicate a preference for GRU cell type, hidden dimension 32 and 2 layers. As for the importance of each hyperparameter as we can see in Figure 3 dropout is the most important. Dropout is effective for regularization and preventing the co-adaptation of neurons. Fur-

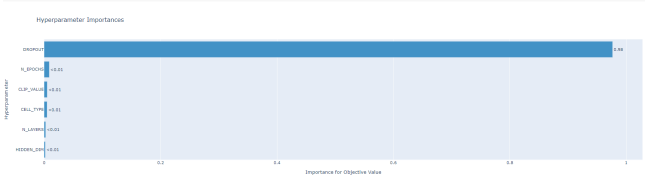


Figure 3: Hyperparameters Importance.

themore, experimenting with all the available values (50, 100, 200, 300) of the

number of features to use from GloVe emdeddings the results were similar and so I used embedding dimension 50, in order not to make the model too complex.

# 5 Results

As metrics for the performance of the model I use precision, recall and F-measure whose values are shown below (Figure ). The train ratio of the dataset is 80%,

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.90 | 0.91 | 0.90 | 4452 |
| 1.0 | 0.91 | 0.90 | 0.90 | 4550 |
| accuracy | | | 0.90 | 9002 |
| macro avg | 0.90 | 0.90 | 0.90 | 9002 |
| weighted avg | 0.90 | 0.90 | 0.90 | 9002 |

Figure 4: Scores.

while the 20% of the dataset is used for testing. The above figure shows the learning curves. Training loss and validation loss are close to each other with

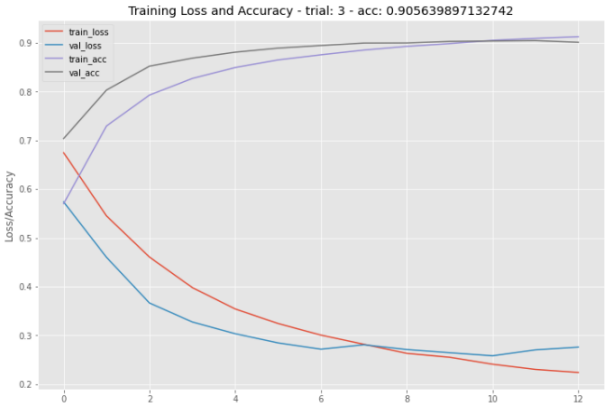


Figure 5: Learning Curves.

validation loss being slightly greater than the training loss after 6th epoch, which shows a little overfitting.I also plot ROC curve (Figure 4) which is a graph showing the performance of a classification model at all classification thresholds, which shows that the model is quite efficient to distinguish between positive class and negative class.

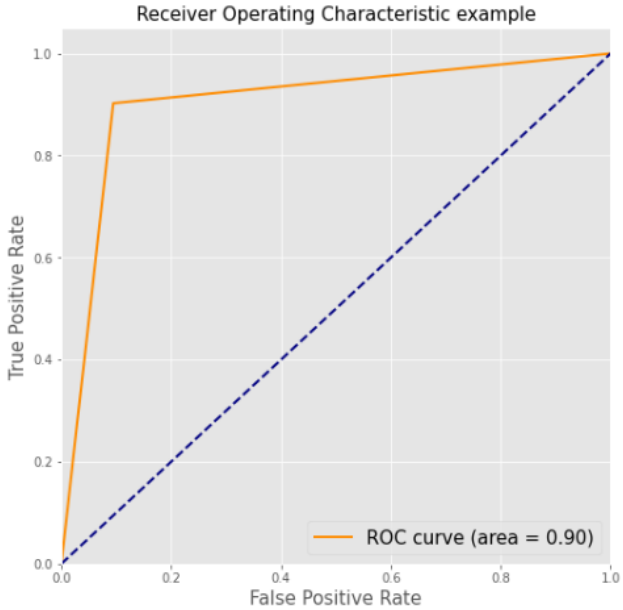


Figure 6: ROC curve.

# 6 Comparison between models

The models we are going to compare are the best models using logistic regression with TF-IDF, FFNN (Feed Forward Neural Networks) and bidirectional RNN with GloVe word embeddings as inputs from project 1, 2 and 3 respectively. The comparison is based in their scores on the test datasets and the duration of training.

- **Logistic regression** is a simple and efficient method for binary and linear classification problems. It is a classification model, which is very easy to realize and achieves very good performance with linearly separable classes
- **Feed Forward Neural Network** is an artificial neural network where there is no feedback from output to input. One can also treat it as a network with no cyclic connection between nodes
- **RNN** is Recurrent Neural Network which is again a class of artificial neural network where there is feedback from output to input. One can also define it as a network where connection between nodes (these are present in the input layer, hidden layer and output layer) form a directed graph

| Models' scores | | | |
|---|---|---|---|
| Model | Precision | Recall | F1 score |
| Regression | 0.85 | 0.85 | 0.85 |
| FFNN | 0.81 | 0.81 | 0.81 |
| BRNN | 0.90 | 0.90 | 0.90 |

Logistic regression is by far faster that the other two methods due to its simplicity, while BRNN needed only 13 epochs in contrast to FFNN which needed 100 epochs. Although FFNN needs more epochs for the training than the RNN, it is faster due it its low complexity. Logistic regression is quite accurate because the data are obviously highly linearly separated, while FFNN would be much more effective in a more complex task. Their scores are very close though. The superiotity of the RNN model over the FFNN model is due to the dynamic memory that RNN uses, while FFNN uses static memory.