RNN vs LSTM vs GRU for Sentiment Analysis

Haja Amir Rahman (p2100803)

School of Computing

Singapore Polytechnic, Singapore

amirrahman517804@gmail.com

*Abstract*—The study is focused on comparing the effectiveness of using a Simple-RNN model, LSTM model and GRU model using Embedding Layers.

# Introduction

Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are models that are typically used on Sentiment Analysis Problems.

The recurrent neural network (RNN) is a model that is typically used on Sentiment Analysis Problems. It is a type of neural network that has a hidden layer between the input and output layers. The hidden layer is used to process sequential data such as words in sentences or words in paragraphs. The output of the hidden layer is then used as the input to the next layer. The following diagram shows how an RNN works:

The long short-term memory (LSTM) is a type of RNN that has two additional layers between the input and output layers. The first additional layer is called the gate and it is used to process sequential data such as words in sentences or words in paragraphs. The second additional layer is called the forget gate and it is used to forget previous states when processing new sequential data.

# methadology

## Data Collection

For this paper, I used a [dataset](https://www.kaggle.com/datasets/cosmos98/twitter-and-reddit-sentimental-analysis-dataset?select=Reddit_Data.csv) containing reviews/comments from the Reddit App consisting of 37248 rows and 2 columns which made it large enough to run our models.

## The Reddit Dataset



From here we can see that the main data column which contains the reviews – ‘clean\_comment’ has been cleaned for us.

For the ‘category column’, we have 3 categories according to the Kaggle site:

* 0 Indicating it is a Neutral Tweet/Comment
* 1 Indicating a Positive Sentiment
* -1 Indicating a Negative Tweet/Comment

# Exploratory Data Analysis (EDA)

Exploratory Data Analysis was performed using Pandas Profiling Report.

## Column Data Types

Since this dataset is specifically meant for sentiment analysis, there are only 2 columns which are the reviews column and the category column that tell us whether the review is positive, negative or neutral. When I found the dataset, the review column data was already cleaned so I did not need to perform much data cleaning.

## Number of reviews per category

However, when I checked the count of reviews per category in the dataset, I found that the balance was not equal.

* The positive review category ‘1’ had 15830 counts of reviews
* The negative category ‘-1’ had 13142 counts of reviews
* The neutral category ‘0’ had 8277 counts of reviews.

Due to this, I performed some restricting to make the data count for each category equal and fair. Given that the lowest data count was 8277, I performed data sampling to make the positive review and negative review category’s value counts equal to 8277.

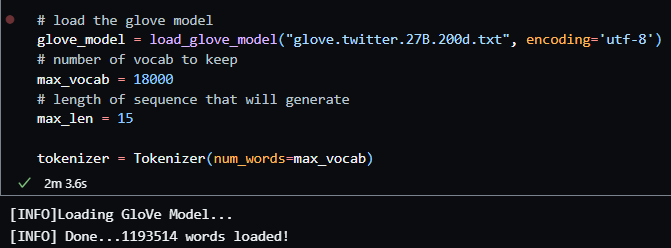
## Using glove model for embedding layers

When I did some research regarding how to use embedding layers for my models, I came across a method called glove model loading for pretrained embedding layers. I decided to use this method because, embedding layer enables us to convert each word into a fixed length vector of defined size. The resultant vector is a dense one with having real values instead of just 0’s and 1’s. The fixed length of word vectors helps us to represent words in a better way along with reduced dimensions.

This way embedding layer works like a lookup table. The words are the keys in this table, while the dense word vectors are the values. There are 2 ways of using embedding layers – one is directly implementing the existent embedding layer available in Keras and the other is using pre-trained word embeddings such as GloVe. To utitlize the pre-trained word embeddings, I created some util functions to load the pretrained embedding layers.

* load\_glove\_model load the twitter embeddings model we downloaded. This model is trained on 2 billion tweets, which contains 27 billion tokens, 1.2 million vocabs
* remove\_stopwords remove the stop words in a sentence
* lemmatize perform lemmatization on a sentence
* sent\_vectorizer convert a sentence into a vector using the glove\_model. This function may be used if we want a different type of input to the RNNs.

Then I converted the reddit review text to sequence format that will be feed into RNNs.



Next I prepared the word embeddings using the GloVe Model. The number of words is 44113 and the number of null word embeddings is 12999. The reason for using embedding layers in the model building function is because I did not one-hot encode the data as it is not a feasible embedding approach due to the large storage space required for the word vectors thus reducing model efficiency.

# Model building

#### Next I created a custom Model Building Function as my primary purpose is to compare the validation and test accuracy results of RNN, LSTM and GRU on the same Reddit dataset.

#### The reason for using a function and not directly building the model is to use an if-else statement for the models to chronologically run from RNN to LSTM to GRU. The function also uses an if-else statement to add an Embedding Layer to the model appropriately.

#### All three main layers of the three model have a parameter of 256, followed by a 2 Dense layers with ‘relu’ activation and the last Dense layer with ‘softmax’ activation. All the models are built using the ‘categorical\_crossentropy’ loss function, ‘adam’ optimizer and the ‘accuracy’ metric.

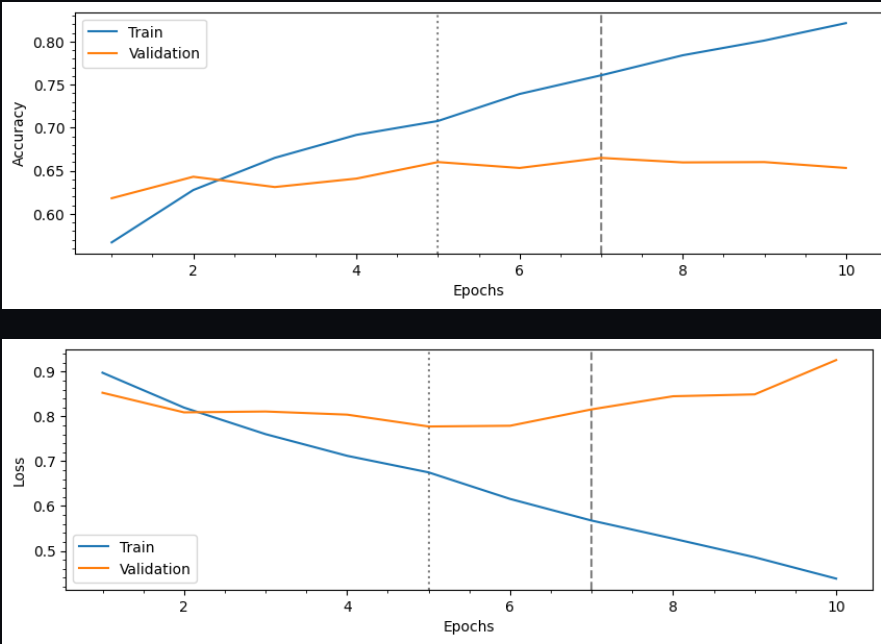
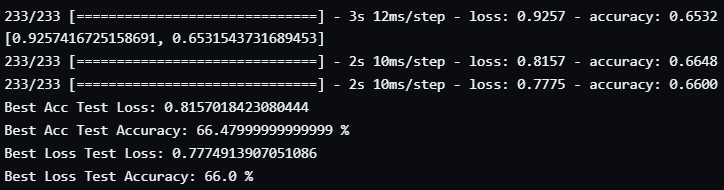
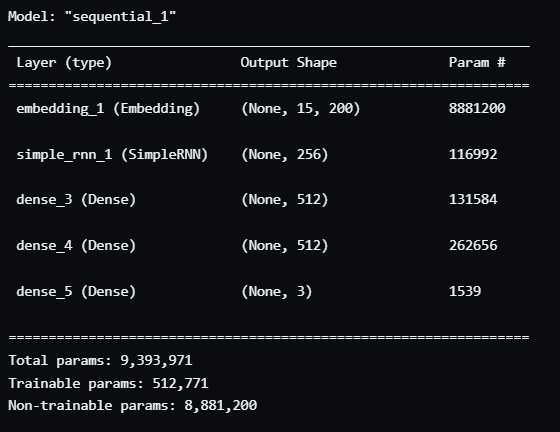
Since these are Deep Learning Models, I set the parameters for all 3 as follows:

* Run at 50 epochs
* Included Early Stopping Callback Function monitoring for max validation accuracy with a patience of 5
* Batch size of 120

# Model evaluation and classification report

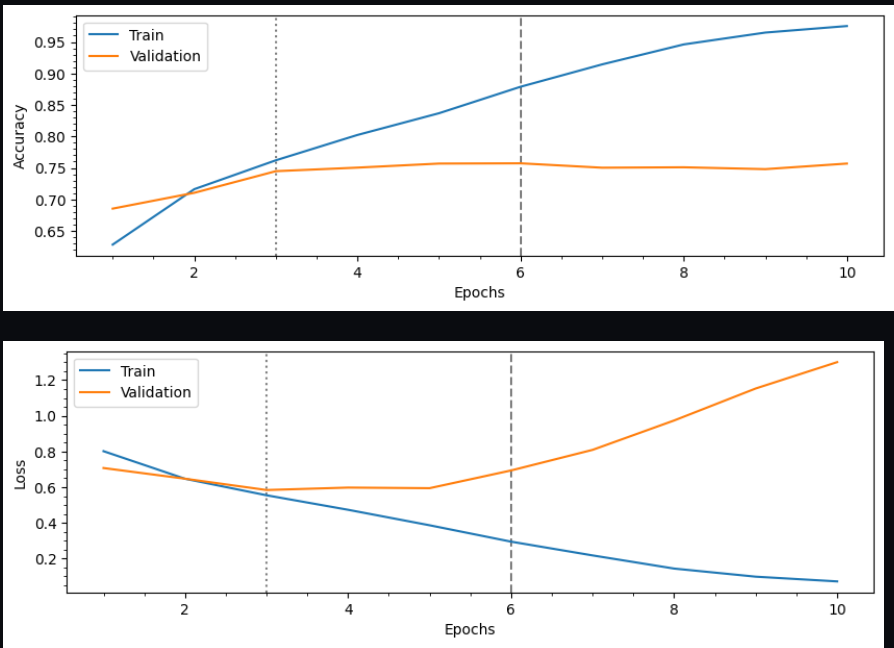
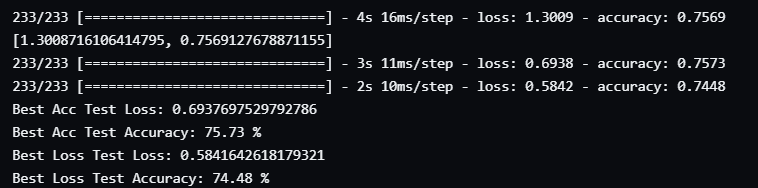
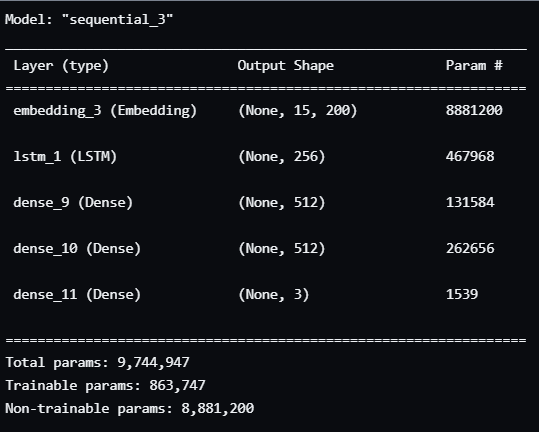
## RNN Model

The **Simple-RNN model** performed the worst among the 3 models with a test accuracy of 66.479%. I also generated a classification report for precision, accuracy and f1 score results.



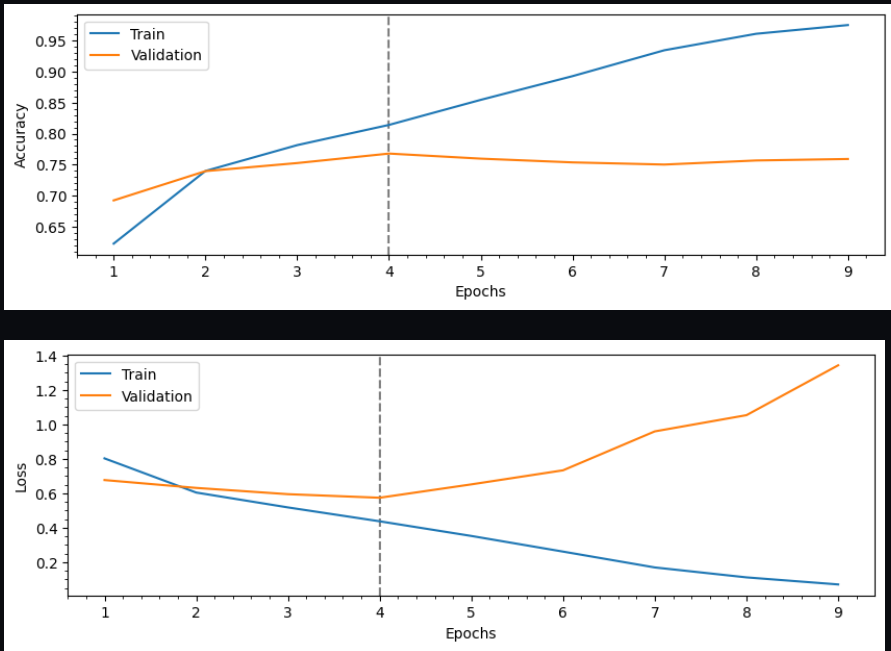
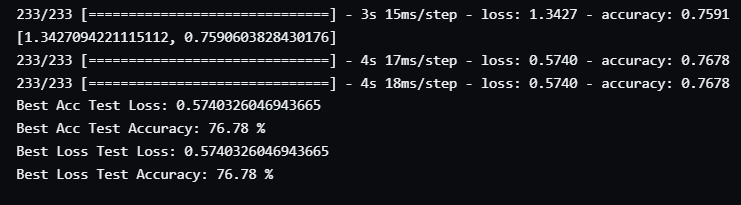
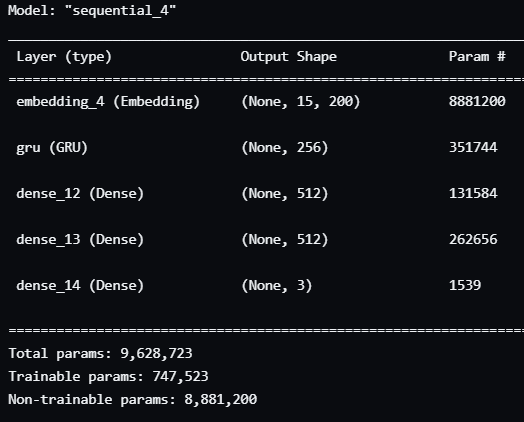
## LSTM Model

The LSTM model performed slightly better than RNN with a test accuracy of 75.73%.



## GRU Model

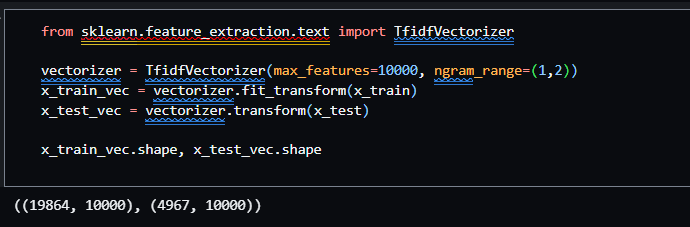
The GRU model performed the best of all the 3 models with a test accuracy of 76.78%.



# Using a non-deep learning model (Tfdif vectorization) and traditional ml models

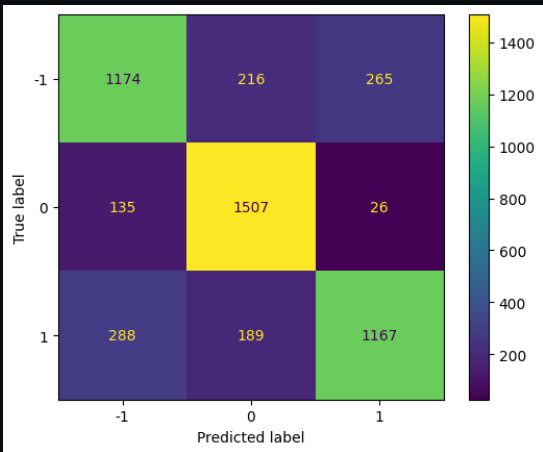
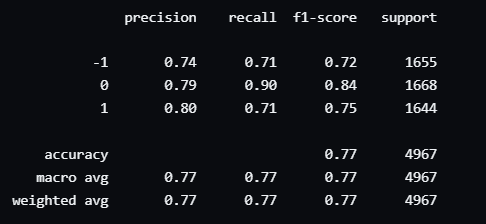
Since I only compared the model accuracy and efficiency of three deep learning models from the same family – RNN, I decided to use some traditional machine learning models to compare based on the same dataset used for the deep learning models.

I made use of TFIDF Vectorization to fit and transform the x\_train data after running a fresh train-test-split.



## Random Forest Classifier

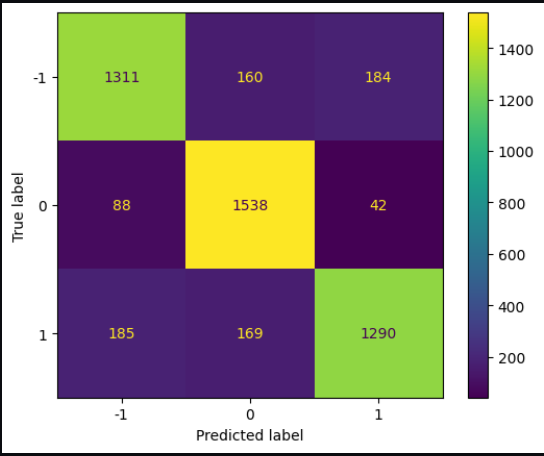
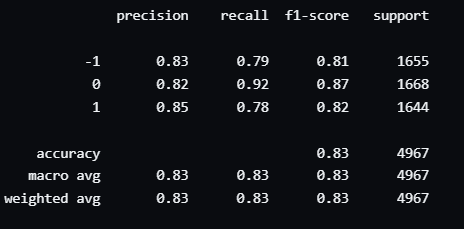
For the Random Forest Classifier Model, I did a simple model.fit, model.score and model.predict to find out how it did. Below are the classification report results and the confusion matrix respectively:



As seen from the classification report, this model has done better than all 3 deep learning models with a precision score of 77%.

## Logistic Regression Model

For the Logistic Regression Model, I did the same as Random Forest and below are the results



From the above results, we can see that Logistic Regression performed the best out of all the models run with a test score of 0.83

# Model Evaluation

So, to summarize what has been done so far, I compared the test accuracy of three different models – RNN, LSTM and GRU based on the same dataset, parameters, and layers. The GRU model performed the best with a test accuracy of 76.78% while LSTM got a test accuracy score of 75.73% and RNN got a test accuracy score of 66.479%. The Random Forest Classifier model got a precision score of 77% while the Logistic Regression model got a precision score of 83%.

# VIII.    Conclusion

However, since the reddit dataset was already cleaned and labelled with nearly 40000 rows of data, the traditional machine learning models used for comparison did better after going through TFDIF Vectorization. When comparing the 3 Deep Learning Models, the GRU model performed best and when comparing Deep Learning Models with Traditional Machine Learning Models, the Logistic Regression Model did the best with a precision of 83%.

# REFERENCES

[1] [Chen, E. (2020) *Sentiment Analysis using SimpleRNN, LSTM and gru*, *Eric Chen's Blog*. Available at: https://haochen23.github.io/2020/01/nlp-rnn-sentiment.html#.Y3sIcXZByUk (Accessed: November 23, 2022).](https://haochen23.github.io/2020/01/nlp-rnn-sentiment.html#.Y3sIcXZByUk)

[2] [Yadav, T. (2018) *Glove.twitter.27b.200d.TXT*, *Kaggle*. Available at: https://www.kaggle.com/datasets/fullmetal26/glovetwitter27b100dtxt (Accessed: November 23, 2022).](https://www.kaggle.com/datasets/fullmetal26/glovetwitter27b100dtxt)

[3] [Saxena, S. (2021) *Understanding embedding layer in Keras*, *Medium*. Analytics Vidhya. Available at: https://medium.com/analytics-vidhya/understanding-embedding-layer-in-keras-bbe3ff1327ce (Accessed: November 23, 2022).](https://medium.com/analytics-vidhya/understanding-embedding-layer-in-keras-bbe3ff1327ce)

[4] [Kapoor, A. (2020) *Deep Learning vs Machine Learning: A simple explanation*, *HackerNoon*. Available at: https://hackernoon.com/deep-learning-vs-machine-learning-a-simple-explanation-47405b3eef08 (Accessed: November 23, 2022).](https://hackernoon.com/deep-learning-vs-machine-learning-a-simple-explanation-47405b3eef08)

[5] [*Recurrent neural networks (RNN) with Keras  :   Tensorflow Core* (no date) *TensorFlow*. Available at: https://www.tensorflow.org/guide/keras/rnn (Accessed: November 23, 2022).](https://www.tensorflow.org/guide/keras/rnn)