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.

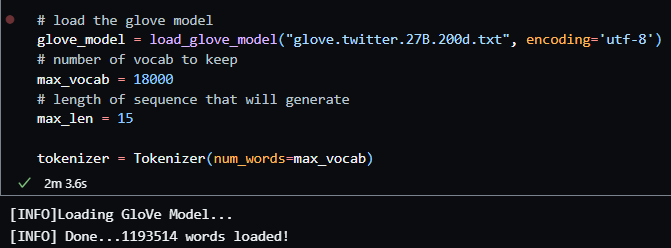
Due to this, I performed some restricting to make the data count for each category equal and fair. Given that the lowest data count was 8277, I performed data sampling to make the positive review and negative review category’s value counts equal to 8277.

## 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. To perform this, 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.

# 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.

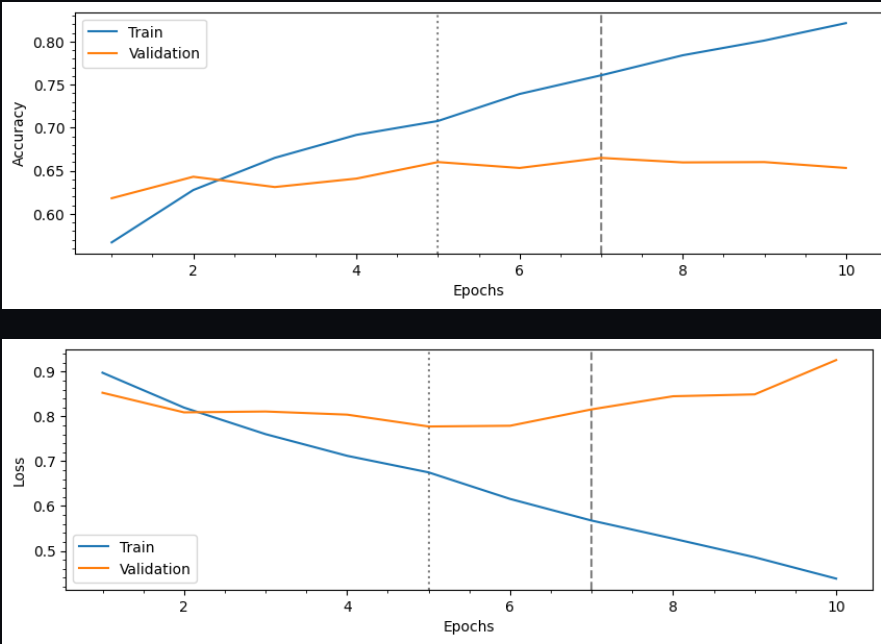
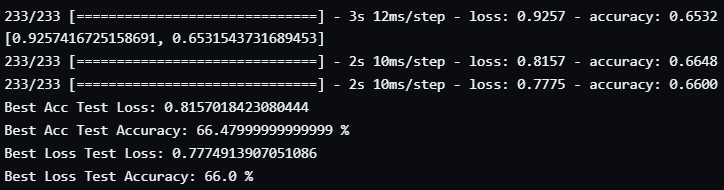
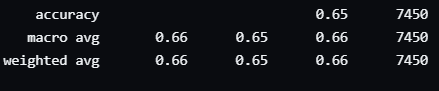
#### All the models are built using the ‘categorical\_crossentropy’ loss function, ‘adam’ optimizer and the ‘accuracy’ metric.

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

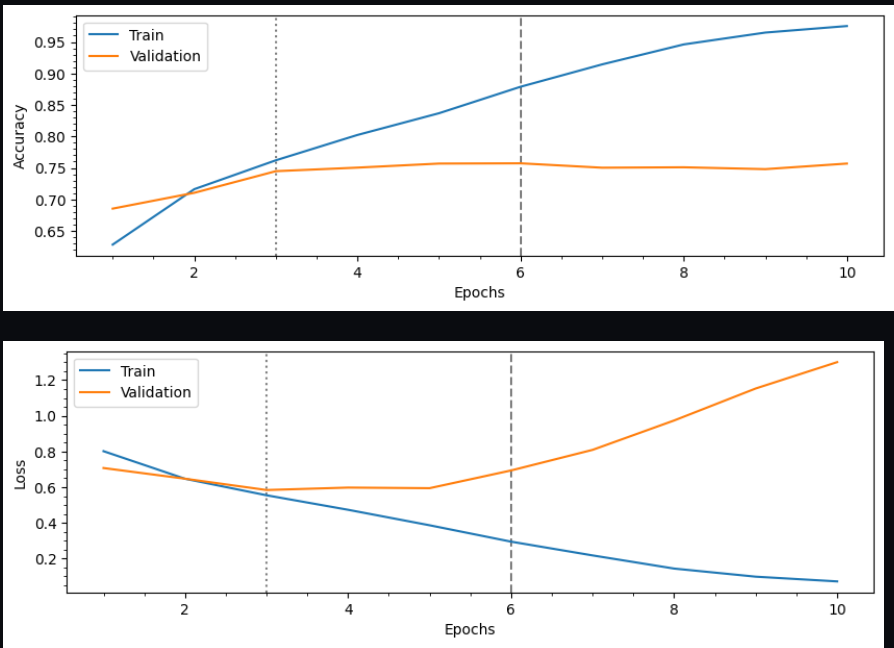
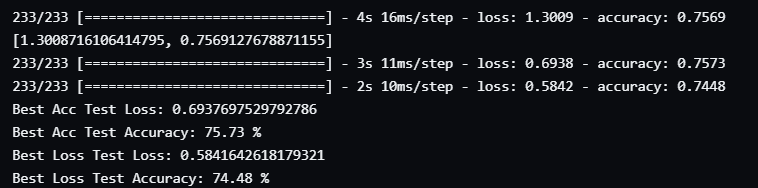
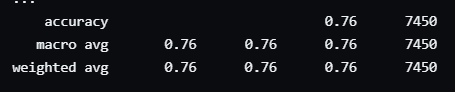
* Run at 50 epochs
* Included Early Stopping Callback Function monitoring for max validation accuracy with a patience of 5
* Batch size of 120

# Model evaluation and classification report

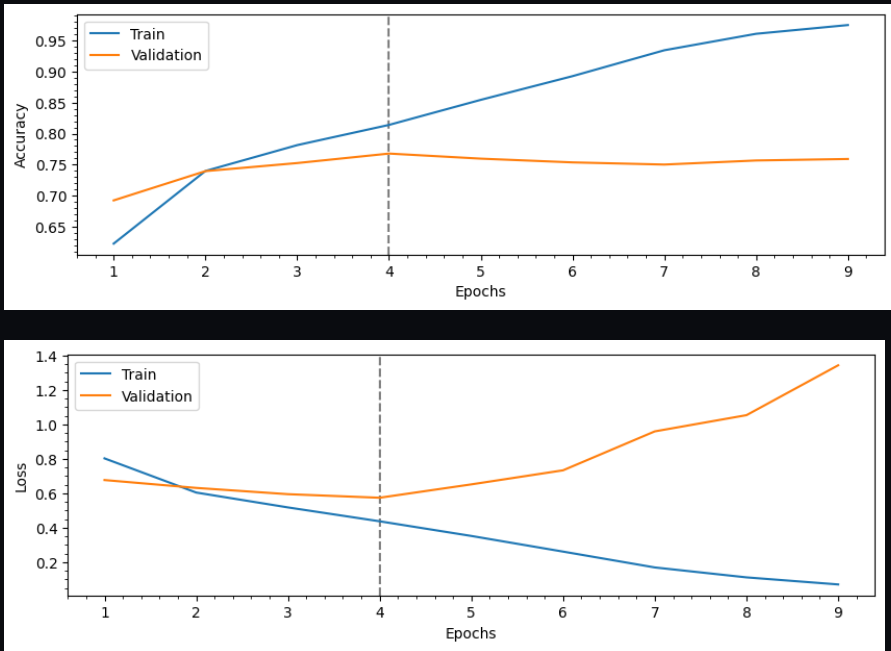
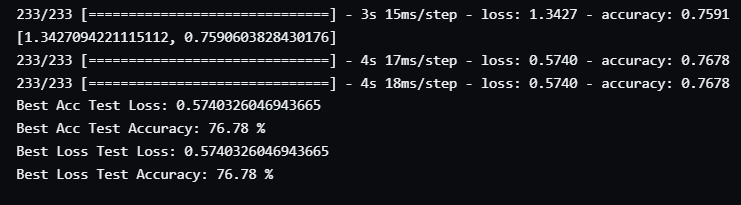
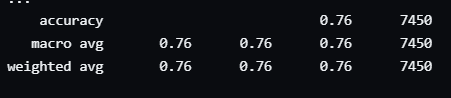
The **Simple-RNN model** performed the worst among the 3 models with a test accuracy of 66.479%. I also generated a classification report for precision, accuracy and f1 score results.



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



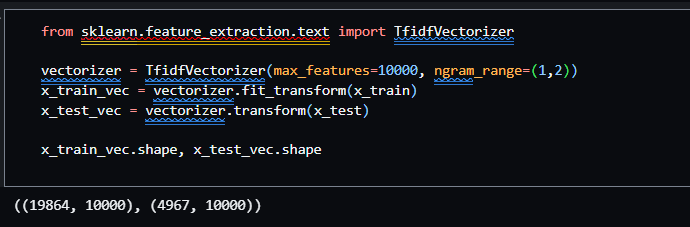
The GRU model performed the best of all the 3 models with a test accuracy of 76.78%.



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

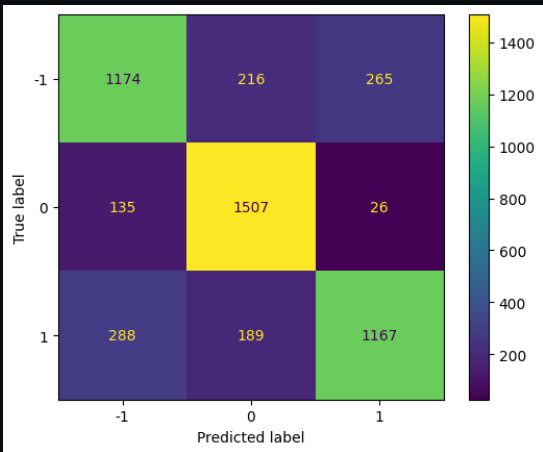
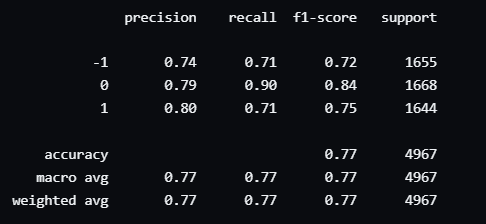
Since I only compared the model accuracy and efficiency of three deep learning models from the same family – RNN, I decided to use some traditional machine learning models to compare based on the same dataset used for the deep learning models.

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



## Random Forest Classifier

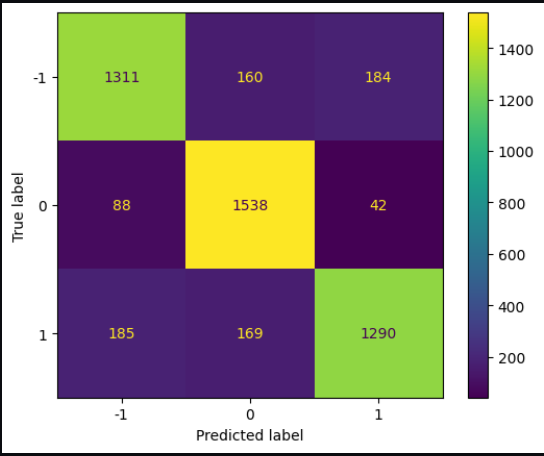
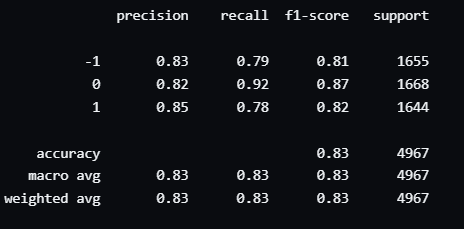
For the Random Forest Classifier Model, I did a simple model.fit, model.score and model.predict to find out how it did. Below are the classification report results and the confusion matrix respectively:



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



# Hyperparameter Tuning

With the aim of reducing overfitting of Gradient Boosting Regressor by increasing regularization, I've decided to perform Grid Search to search for the optimal hyperparameters to reduce high variance while retaining the low bias characteristics of our model.

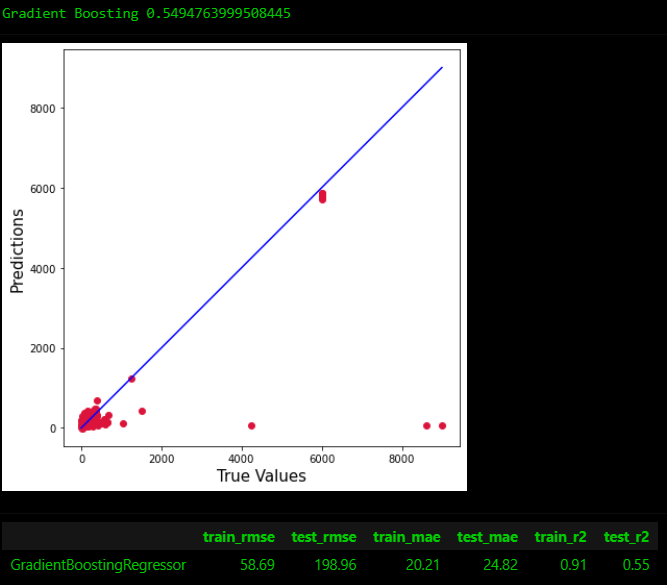
Due to time and computational resources constraint, I have only managed to perform grid search some models, although I wanted to try for SVR as well. The following is the set of hyperparameters that I have used to build my final Machine Learning pipeline.

#### n\_estimators:100

For the hyper-parameter tuning, I created my own function to run a Grid Search CV on all the models I chose including Lasso and Ridge (which are not the main models so not mentioned). The best params for K Neighbors Regressor was ‘n\_neighbors:3’, Random Forest Regressor was ‘n\_estimators: 100’ and Gradient Boosting Regressor was ‘n\_estimators: 100’.

# Model Evaluation

After performing the hyper-parameter tuning, the two best models that gave the best r2 test scores were Random Forest Regressor and Gradient Boosting Regressor of 0.54 and 0.55. The worst performing model was the Linear Regression Model with a r2 test score of 0.32 (rounded to 2 d.p.).



*Fig 6 Gradient Boosting Regressor Model Evaluation*

Since I am using Gradient Boosting Regressor as my final model, we can visualize the model importance to investigate which features helps us better in modelling to investigate the relationship between the features and target variables.’

Fig 8 shows the Model Attributes like Hotel Property Type and Strict Cancellation Policy with Grace Period is significant in affecting the model’s decision which is consistent with the findings by (Hati, S.R.H. et al., 2021). Moreover, Geographic Attributes like Longitude and Latitude are indispensable in the study of Airbnb Price Estimation.



*Fig 7 Feature Importance*

# Conclusion

Overall, I have managed to build a machine learning model that can help us in estimating Airbnb prices with R2 test score of 0.55. Besides, based on EDA and Model Interpretation, I have identified several important attributes like Hotel Property Type and Strict Cancellation Policy and Geographical Attributes like latitude and longitude and CBD which are dominant towards the impact of Airbnb prices. The model can be improved with more computation resources for hyperparameter tuning as well as other Neural Network or Random Search CV that can better model non-linear relationships between the features and price.

REFERENCES

[1] [Bettendorf, B., 2019. Berlin Airbnb Data. *Kaggle*. Available at: https://www.kaggle.com/datasets/brittabettendorf/berlin-airbnb-data?select=listings\_summary.csv [Accessed June 9, 2022].](https://www.kaggle.com/datasets/brittabettendorf/berlin-airbnb-data?select=listings_summary.csv)

[2] [Anon, Get the Data. *Inside Airbnb*. Available at: http://insideairbnb.com/get-the-data/ [Accessed June 9, 2022].](Anon,%20Get%20the%20Data.%20Inside%20Airbnb.%20Available%20at:%20http://insideairbnb.com/get-the-data/%20%5bAccessed%20June%209,%202022%5d.)

[3] [Anon, 2022. Airbnb. *Wikipedia*. Available at: https://en.wikipedia.org/wiki/Airbnb [Accessed June 9, 2022].](https://en.wikipedia.org/wiki/Airbnb)

[4] [Anon, Sign in. *RPubs*. Available at: https://rpubs.com/jeryl\_goh/airbnb\_SG [Accessed June 9, 2022].](https://rpubs.com/jeryl_goh/airbnb_SG)

[5] [Hati, S.R.H. et al., 2021. A decade of systematic literature review on airbnb: The sharing economy from a multiple stakeholder perspective. *Heliyon*. Available at: https://www.sciencedirect.com/science/article/pii/S2405844021023252 [Accessed June 9, 2022].](https://www.sciencedirect.com/science/article/pii/S2405844021023252)