CSCE 479/879 Homework 2: Sentiment Analysis with Sequential Models

Izzat Adly, Ryan Bockmon, Quan Nguyen

March 13, 2021

Abstract

We where tasked with applying sequential neural network models to the problem of sentiment analysis on the IMDB dataset. We came up with four different models; two using Long Short Term Memory (LSTM) and two using Gated Recurrent Units (GRU). We used a 3 split k-fold cross validation to access accuracy of the models. In the end all 4 models all did fairly similar (around 0.75 validation accuracy and validation loss around 0.50).

1 Introduction

We where tasked with applying sequential neural network models to the problem of sentiment analysis on the IMDB dataset. The data set consisted of 50,000 reviews labeled as positive or negative. We had to implement at least two sequential architectures for this problem. We came up with four different models; two using LSTM and two using GRU. We also used a 3 split k-fold cross validation to access accuracy of the models. In the end all 4 models all did fairly similar (around 0.75 validation accuracy and validation loss around 0.50)

2 Problem Description

Sentiment analysis is a technique used to determine whether data is positive, negative or neutral. It is is often performed on textual data to help businesses monitor brand and product sentiment in customer feedback, and understand customer needs. Sentiment analysis help companies to under stand users attitudes towards a product based on their feedback.

The data set that we had consisted of 50,000 reviews labeled as positive or negative from the IMDB dataset. We had to implement at least two sequential architectures for this problem. These could be based on LSTMs, GRUs, or transformers. For each architecture we needed to use an attention mechanism and a regularizer, and try at least two settings of hyperparameter values. In the end we had to evaluate four models. We used k-fold cross-validation for this process.

3 Report Approach

For the homework, we decided to study the effectiveness of LSTM and GRU architectures in learning and classifying IMDB reviews. The benchmarks for performance are two: performance in validation accuracy and loss, and training taken for training. We decided to focus on LSTM and GRU because of its major advantages over previous learning model that we have learnt. One advantage of both models is in the ability to remember long term information. Previous models struggle to remember information for a long time. LSTM and GRUs are built with such features as its default behavior. Hence, it is suitable for tackling the problem of classifying movie reviews into positive and negative characteristics. Since IMDB reviews are sentences which are made up of a combination of words, LSTMs and GRUs are able to pick up and remember the patterns between the words and conduct classification on the IMDB review data set. The choosing of LSTM and GRU as main hidden layers is not a coincidence.

LSTM, specifically, has an advantage due to its complex structure. Firstly, there are input gates that control how much new information to keep. The input gates are made up logistic and tanh functions and takes in the previous short term state and the current inputs of the data set. There is also a forget gates where irrelevant or contextual data gets removed from the long term state between each learning and lastly; there is an output gate that controls what kind of information and how much of the information would be released.

GRUs on the other hand are just simpler versions of LSTMs and does less mathematical operations compared to LSTM models. In GRUs, the long term and short term states are combined into one single input vector; there is only one single gate controller that controls both the input gate and forget gate; and there is no output gate and only a single output of a full state vector at every time stamp. Since the GRUs has less gate controllers and keeps track of less states, it is theoretically more efficient and quicker in training compared to LSTMs.

For the homework, we will dive deeper into which architecture impacts the final evaluation of the model. For the first model, we decided on a model with a single LSTM layer. The second differed in that there will be 3 LSTM layers. Model 3 will be one hidden layer of GRUs and model 4 will be 3 GRU layers. The combined units of each model was 200. The second model will have a rhombus shape (50 nodes on the first layer, 100 nodes on the second layer, and 50 nodes on the third layer).The rest of the hyper parameters will be kept the same.The reason we have chosen the rhombus shape is because it will theoretically allow for more changes in the weights of the model. Hence, this allows for the model to take into account the details of the IMDB reviews. For the rest of the report, we will refer to the first model as LSTMmodel1, the second model as LSTMmodel2, the third model as GRUmodel3 and the fourth model as GRUmodel4.

Before conducting any experiments, we hypothesized that the LSTMmodel2 will outperform LSTMmodel1 due to the rhombus shape. This will also hold true between GRUmodel3 and GRUmodel4, in that GRUmodel4 will outperform GRUmodel3 because of the rhombus shape. The reasoning for the above two hypothesis is that a rhombus shape model has more weights to train as 200 units are split up into 3 sections instead of clumping together into 1. Besides that, With more nodes in different layers, it will be more flexible, and hopefully learn better. In terms of training time, we hypothesize that LSTMmodel1 and GRUModel3 will outperform the LSTMmodel2 and GRUModel4 respectively. This is because LSTMmodel1 and GRUModel3 has less nodes to compute because of having only one hidden layer. Thus, LSTMmodel1 and GRUModel3 should train faster than their model counterparts.

For this report, We decided to not compare between LSTM and GRU models directly, as we feel it would not be a fair comparison. The previous apples to apples comparison allows us to split the rest of the report into 2 categories, comparisons between LSTM models and comparisons between GRU models.

4 Experimental Setup

4.1 General Setup

- Regarding the input, the default train dataset of IMDB was split into a train dataset and a validation dataset. Train dataset accounts for 90% of the default data while the validation dataset accounts for 10%. During splitting, the dataset was shuffled.
- Both were allowed to run for 50 epochs. Early stopping was implemented by monitoring loss values. The model would stop if the loss values had not improved for 5 epochs
- Both LSTM and GRU layer in both model setup were put into a Bidirectional layer due to the usage of the attention function for the a.
- Loss function was "binary crossentropy"
- Optimizer was Adam
- Metric was accuracy
- 3 k-folds was use for cross-validation. The number of folds was chosen do to time constraints.

4.2 LSTM model Setup

- General structures:

1. Embedding layer
2. Dropout layer with rate = 0.4
3. LSTM layers (varied between models)
   - LSTMmodel1: a single LSTM layer with 200 units
   - LSTMmodel2: 3 LSTM layers: first one has 50 units, second one has 100 units, and third one has 50 units.
4. Dropout layer with rate = 0.3
5. Attention layer
6. Dropout layer with rate = 0.5
7. Dense layer with 512 features
8. Leaky ReLU layer. Similar to ReLU but the negative values were multiplied by an alpha = 0.2 instead of setting it to 0. The decision behind this was to avoid the overscaling of using both ReLU and Dropout.
9. Output layer with 1 feature and activation was "sigmoid"

4.3 GRU model Setup

- General structures:

1. Embedding layer
2. Dropout layer with rate = 0.4
3. LSTM layers (varied between models)
   - GRUmodel3: a single GRU layer with 200 units
   - GRUmodel4: 3 GRU layers: first one has 50 units, second one has 100 units, and third one has 50 units.
4. Dropout layer with rate = 0.3
5. Attention layer
6. Dropout layer with rate = 0.5
7. Dense layer with 512 features
8. Leaky ReLU layer. Similar to ReLU but the negative values were multiplied by an alpha = 0.2 instead of setting it to 0. The decision behind this was to avoid the overscaling of using both ReLU and Dropout.
9. Output layer with 1 feature and activation was "sigmoid"

5 Experimental Results

5.1 LSTM model results

Tables 1, 2, and 3 shows the summary of results for each of the first approach using LSTM. Tables 1 and 2 specifically show the k-fold validation spit into 3 folds while Table 3 shows the results of running one fold split with 90% training data and 10% testing data. LSTMModel1 had an average accuracy of 0.76 and loss of 0.50 across all folds. LSTMModel2 had a slightly higher accuracy on the 90/10 run than each of the folds (0.7634 vs 0.7583) and slightly lower loss on the 90/10 run that each of the folds (0.4891 vs 0.50). This could be because splitting the data into three folds resulted in a smaller training and larger testing set (a 66/33 train/test split). Figures 1 and 6 shows the graph of each models accuracy over each epoch. Figures 3 and Figures 4 shows the confusion matrix of both LSTM models.

Table 1: Results of 3 k-folds for LSTMmodel1.

Model 1:	Validation Loss	Validation Accuracy
Fold 1	0.5207	0.7584
Fold 2	0.4844	0.7721
Fold 3	0.4988	0.7568

Table 2: Results of 3 k-folds for LSTMmodel2.

Model 1:	Validation Loss	Validation Accuracy
Fold 1	0.5146	0.7583
Fold 2	0.4902	0.7583
Fold 3	0.5114	0.7563

Table 3: Overall results for the LSTM models.

Model:	Model 1	Model 2
Validation accuracy	0.7634	0.7698
Validation Loss	0.509	0.4891
Sign of over fitting	Yes	Yes
Error 95% confidence interval	(0.231, 0.255)	(0.225, 0.249)
Time taken per epoch	100s per epoch	121s per epoch
Overall time for training	59 minutes	79 minutes

5.2 GRU model results

Tables 4, 5, and 6 shows the summary of results for each of the second approaches models using GRU. Tables 4 and 5 specifically show the k-fold validation spit

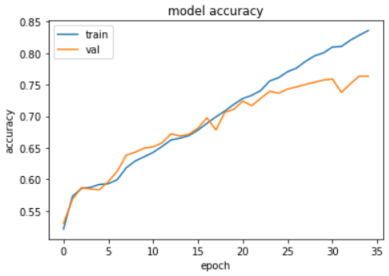

Figure 1: LSTMmodel1's Train and Validation accuracy and after each epoch

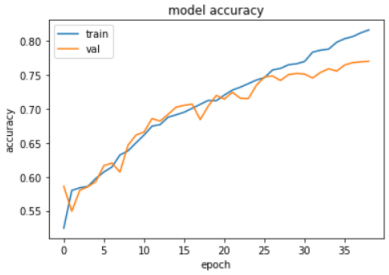

Figure 2: LSTMmodel2's Train and Validation accuracy and after each epoch

into 3 folds while Table 6 shows the results of running one fold split with 90% training data and 10% testing data. GRUModel3 had a slightly higher accuracy on the 90/10 run than each of the 3 folds (0.776 vs 0.76) and slightly lower loss on the 90/10 run that each of the 3 folds (0.485 vs 0.51). Again this could be because splitting the data into three folds resulted in a smaller training and larger testing set (a 66/33 train/test split). GRUModel4 had an average accuracy of 0.77 and loss of 0.51 across all folds. Figures **??** and 7 shows the graph of each models accuracy and loss over each epoch. Figures **??** and Figures 8 shows the confusion matrix of both GRU models.

Table 4: Results of 3 k-folds for GRUmodel3.

Model 1:	Validation Loss	Validation Accuracy
Fold 1	0.5323	0.7684
Fold 2	0.4901	0.7616
Fold 3	0.5336	0.7523

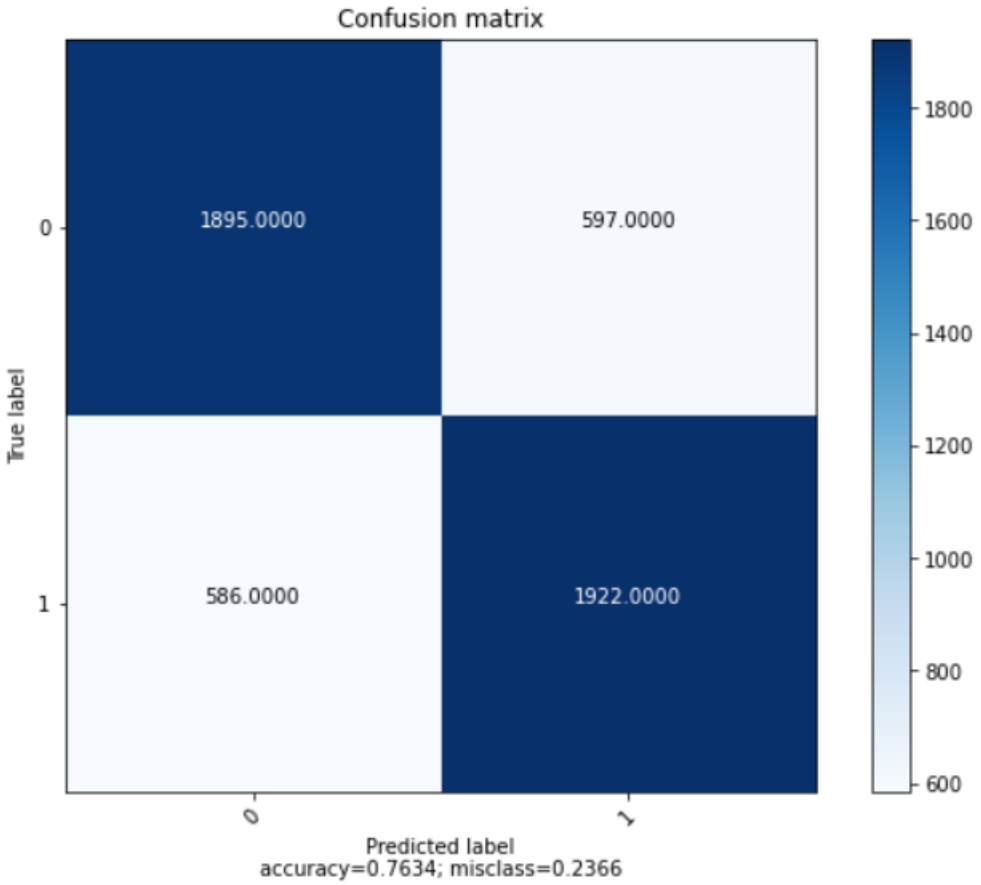

Figure 3: LSTMmodel1's confusion matrix

Table 5: Results of 3 k-folds for GRUmodel4.

Model 1:	Validation Loss	Validation Accuracy
Fold 1	0.5128	0.7623
Fold 2	0.5105	0.7750
Fold 3	0.5112	0.7522

6 Discussion

6.1 LSTM model discussions

For the LSTM model, splitting the units from 1 layer into 3 layers did not impact the final result. Both the validation accuracy and the validation loss were relatively similar for both LSTMmodel1 and LSTMmodel2. Both showed

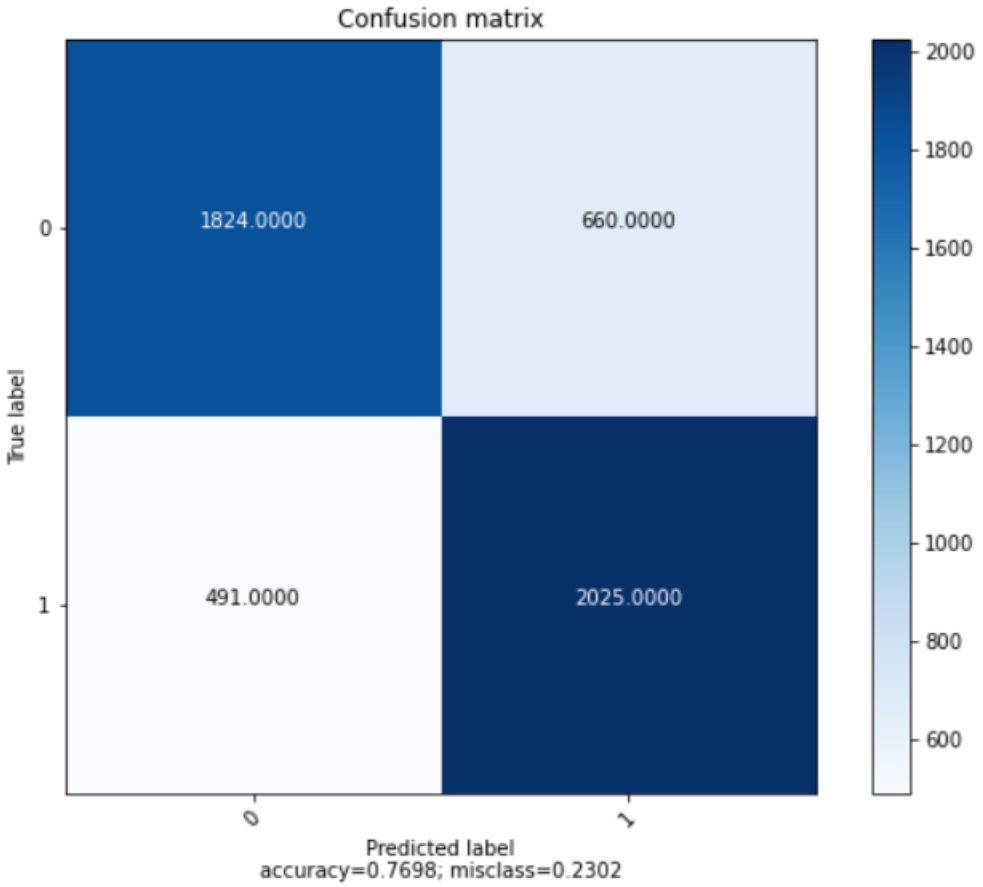

Figure 4: LSTMmodel2's confusion matrix

Table 6: Overall results of the second approach.

Model:	GRUmodel3	GRUmodel4
Validation accuracy	0.776	0.7704
Validation Loss	0.485	0.5048
Sign of over fitting	Yes	Yes
Error 95% confidence interval	(0.213, 0.236)	(0.218, 0.241)
Time taken per epoch	23s per epoch	32s per epoch
Overall time for training	15 minutes	17 minutes

signs of overfitting after 30^{th} epoch. Those results contradicted our hypothesis that the second model would outperform the first model. We came together to discuss and proposed some possible explanations:

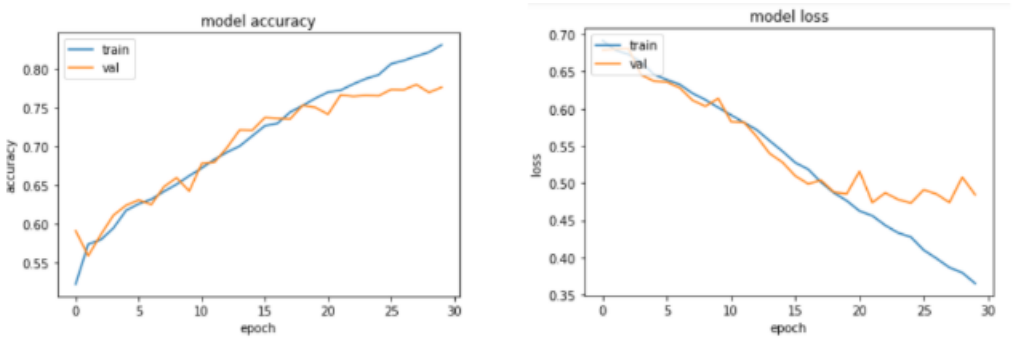

Figure 5: GRUmodel3's accuracy (left) and loss (right) results after each epoch

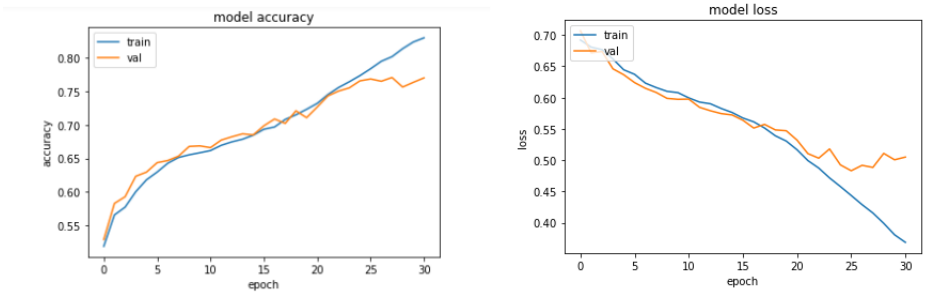

Figure 6: GRUmodel4's accuracy (left) and loss (right) results after each epoch

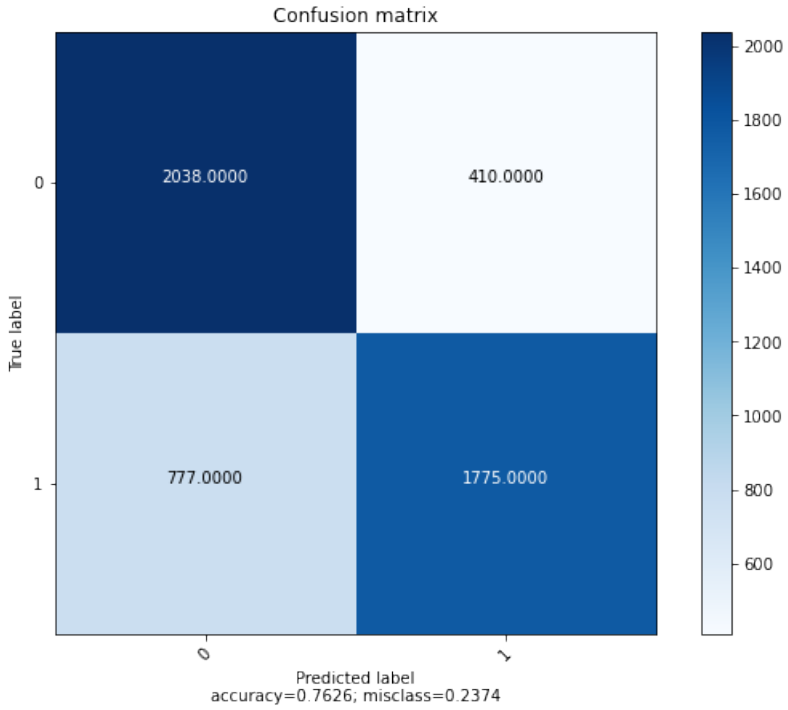

Figure 7: GRUmodel3's confusion matrix

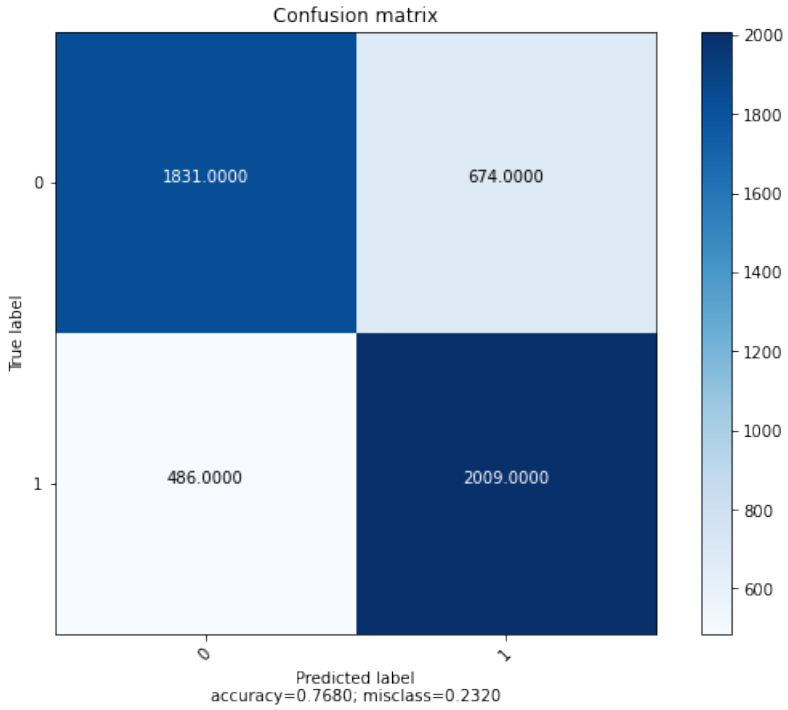

Figure 8: GRUmodel4's confusion matrix

1. The dropout layer rate might be too high that it inhibited the flexibility of the second model.

2. The dropout layer rate might be too low that it did not allow for effective training of nodes in the model

3. The current architecture with only 200 units may be too little units for a spread-out architecture like LSTMmodel2 to outperform the first one.

Moreover, the LSTMmodel1 model also shows time-advantage over the second model. Specifically, the LSTMmodel2 took 33% longer to finish learning compared to the first one. The time difference could be contributed to the difference in designs. The time advantage was significant as both models arrived at the same validation accuracy.

Lastly, based on the heat maps of the confusion matrix, its clear that LSTMmodel1 had more accurate predictions compared to LSTMmodel2 on the validation set. LSTMmodel 2 had more trouble in classifying both positive and negative IMDB reviews.

6.2 GRU model discussions

For the GRU models, the models acted as expected. GRUmodel3 performed faster than GRUmodel4, by around 2 minutes. But in terms of performance, there was no significant difference between the two and both models performed around the same in terms of both validation accuracy and validation loss. This goes against our hypothesis that the extra hidden layers in GRUmodel4 allowed for extra variability in the weight changes. Thus, the extra hidden layers had no effect on the overall structure.

In terms of model overfitting, both models showed forms of overfitting. GRUmodel3 began overfitting at around epoch 25 and GRUmodel4 began overfitting at around the 20th epoch. Possible causes for overfitting in both models:

1. Not having deep enough layers in the model to account for different patterns in the data set.

2. Not having a high enough dropout rates for the model so the whole model is trained properly. Since one model only has one layer and the other has 3 layers, it is difficult to implement proper dropout layers for training.

Moreover, as hypothesized, GRUmodel3 trained much faster compared to GRUmodel4. The reason for this is similar to what happened between the LSTM models; the difference in the designs of each model led to a faster performance for GRUmodel3 compared to GRUmodel4.

Lastly, based on the heat maps of the confusion matrix, it shows that GRUmodel3 had a main problem with with classifying negative IMDB reviews, with making a mistake at least around 28% of the time. For GRUmodel4, it had more

problems with classifying positive reviews than the negative ones. GRUmodel4 made around 25% misclassifications for positive IMDB reviews.

7 Conclusions

In the end each of models in both of the architectures all had very similar results. All achieved an accuracy of around 75% and loss around .50. Each model had signs of over-fitting but with implementing early stopping stopped the models of over-fitting too much. For both type of architectures, if we have more time, we would try it with more architectures like inverted-triangle shape or square shape. We would also lower the dropout rate a bit to see if it affects the final accuracy. Furthermore, we did not clean the output for unexpected characters. Those characters may impact how the models perform by introducing noise. In the end