
# EE4483 Artificial Intelligence and Data Mining

## Mini Project (Option 1)

**Sentiment Analysis**

**School of Electrical and Electronic Engineering**

**Academic Year 2021/2022**

**Semester 1**

**Name: Lim Jia Wei (U1920462G)**

**Khor Chin Yi (U1920193A)**

**Shoon Zhen Yong (U1920283L)**

**Feature Format: One Hot Encoding**

One Hot Encoding is chosen as the feature format for our data samples. One Hot Encoding is a common way to preprocess categorical data variables. One Hot Encoding will help us to transform strings into numbers where every word will be written in the form of vectors, constituting only 1 and 0. Every different word will be encoded as a unique one hot vector. This will allow the word to be identified uniquely by its one hot vector and there will be no two words with the same vector representation. Therefore, the reviews will be encoded into an array of one hot encoded vector by using One Hot Encoding. Since our dataset is an array of reviews, this will result in a three-dimensional tensor that can be fed to our neural network. Besides, as our review comes with a different length, we will be able to rescale it easily by using One Hot Encoding.

**Data Preprocessing**

Before encoding the data, data cleaning is performed to remove some valueless information in the dataset. Hyperlinks, newline characters, additional whitespaces and non-alphanumeric characters are removed with the use of the regular expression. Stopwords such as “the”, “a”, “an” are also removed with the use of the Natural Language Toolkit. The cleaned data is then stored as an array to perform one hot encoding. One hot encoding is carried out with a vocabulary size set as 20000. This data is retrieved by getting the unique number of words in the reviews of train data. The total unique word in train data is 13533. Therefore, by setting the vocabulary size as 20000, we provide a tolerance for test data as the test data might contain words that do not appear in train data. The encoded reviews are then padded with 0 if they have a length shorter than 300, while encoded reviews which are longer than 300 are truncated to ensure all reviews have the same length. The encoded and padded reviews are stored back in the dataset.

**Data Splitting**

| Dataset | Count |
|---|---|
| Train | 5921 |
| Validation | 1480 |
| Test | 1851 |

*Table 1 : Data Splitting*

The table above shows the total number of data that we have for training, validation and testing.

**Data Distribution**

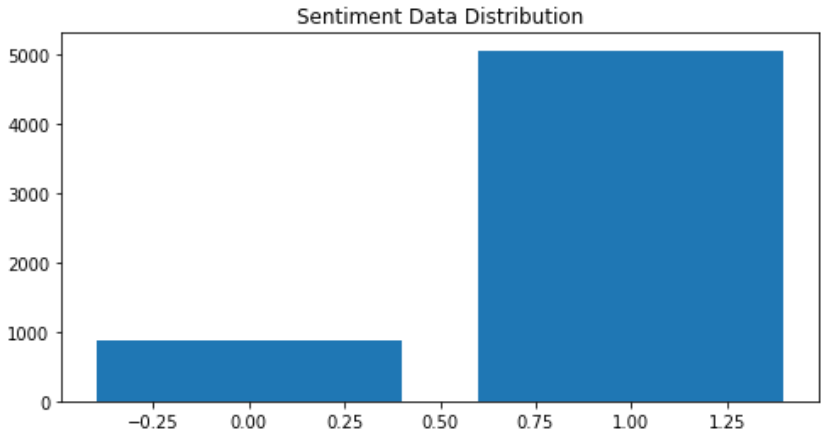


*Figure 1 : Data Distribution of train.json*

Figure 1 shows the distribution for the sentiments of the review in our dataset. From the graph, we can see that most of our dataset has a sentiment value of 1. Uneven distributed datasets will greatly affect the accuracy of our model as there might be too little data for our model to learn the pattern of negative sentiments.

# RNN Model

We had chosen the RNN model to build our classifier.

Recurrent neural networks(RNNs) are the algorithms for sequential data like time series, speech,text, financial data and audio etc. It is the algorithm that could remember its input due to the existence of internal memory. Thus, it is very precise in predicting what's coming next.

## How do Recurrent Neural Networks Works?

To understand how RNN works, we need to know feed-forward neural networks and sequential data.

Sequential data is ordered data in which related things follow each other. Examples are time series data, a series of data points listed in time order.

In the feed-forward neural network, the information only moves in one direction from the input layer, through the hidden layer, and finally to the output layer. Due to its characteristic that only considers the current input, it simply can't remember anything about what happened in the past except its training. But, in RNN, its information cycle is through a loop. When it makes a decision, it will consider the current input and what it learned from the inputs it received previously.

The image below illustrates the difference in information flow between RNN and feed-forward neural networks.

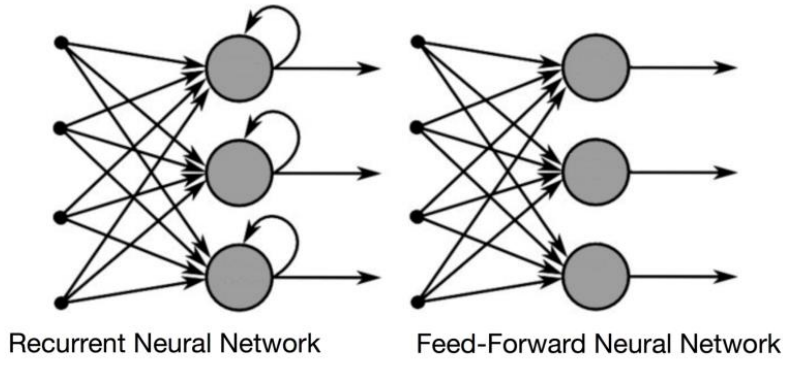


*Figure 2: Diagram of RNN and normal Neural Network*

I will give one example to illustrate the concept of neural network memory. Imagine we have a normal feed-forward neural network and give it the word “hello” as its input, and it processes the word character by character. By the time it reaches the character “l”, it has already forgotten about “h” and “e”. This makes it almost impossible to predict which character will come next. However, RNN is able to remember those characters because of its internal memory. It produces output, copies it, and loops it back into the network.

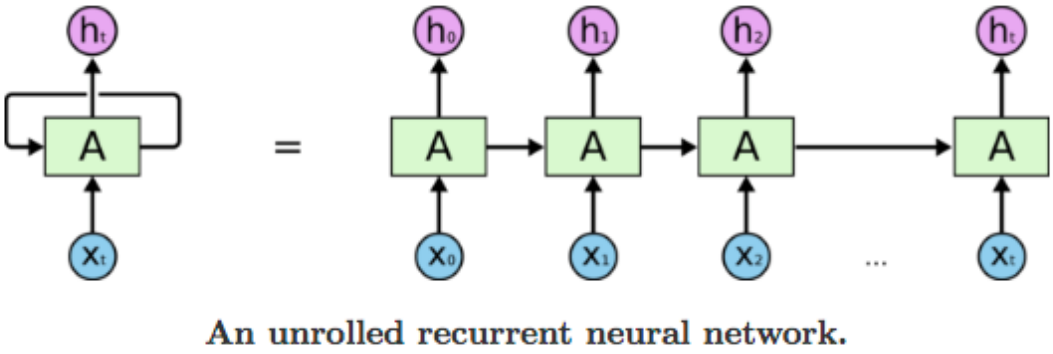


*Figure 3. Diagram of unrolled RNNs*

The figure above represents the unrolled recurrent neural network through a time step for us to imagine how the RNN works across a time series.

Our RNN model consists of

1. Embedding Layer: that converts our word vector (integers) into embedding of specific size.
2. Bidirectional LSTM Layer: We have three layers of bidirectional LSTM layers which each have 32, 8 and 4 neurons respectively.
3. Dropout Layer: The Dropout layer is added after each bidirectional LSTM layer at the rate of 0.5.
4. Sigmoid Activation Layer: that turns all output values in a value between 0 and 1
5. Output: Sigmoid output from the last timestep is considered as the final output of this network.

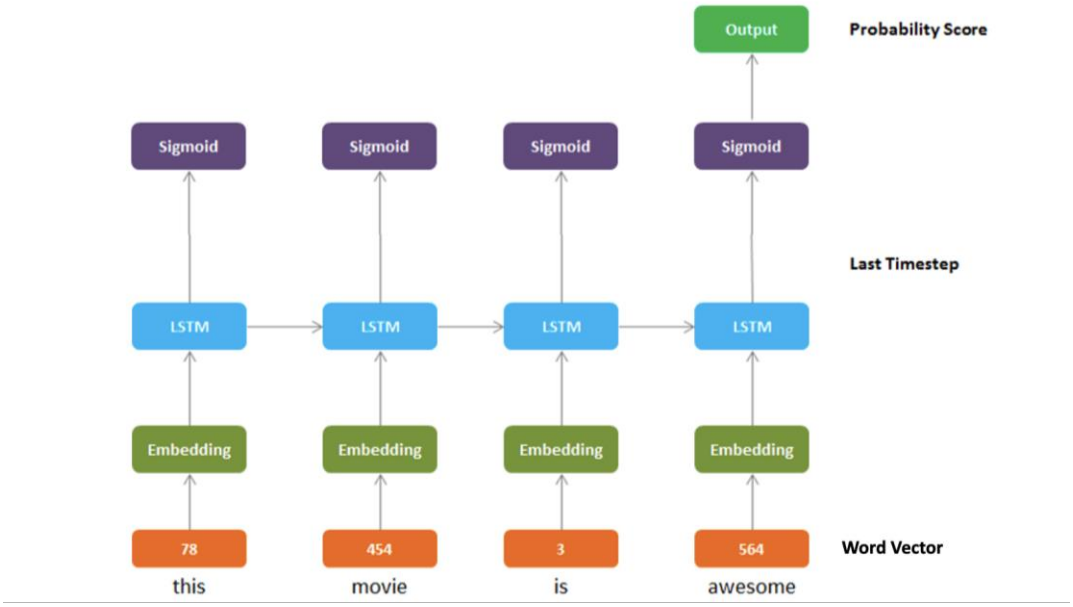


*Figure 4. Structure of our RNNs model*

Training configurations:

1. Embedding Layer: Input Dimension=20000, Output Dimension=32, Input Length=300
2. Output Dimension: 1
3. Number of epochs:12
4. Number of trainable parameters: 1,621,993
5. Batch size: 32
6. Optimizer: Adaptive Moment Estimation (Adam)
7. Initial learning rate: 0.0001
8. Activation Function: Sigmoid
9. Loss function: Binary Cross Entropy
    a. The Binary Cross Entropy compares each of the predicted probabilities to the actual class output which can be either 0 or 1. It then calculates the score that penalizes the probabilities based on the distance from the expected value. That means how close or far from the actual value.

b. Loss for our prediction is calculated as follows:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

*Figure 5. Formula of Binary Cross Entropy Loss Function*

where y is the label positive of the review and its value will be 1.

p(y) is the predicted probability of the point being positive for all N points.

1-p(y) is the predicted probability of the point being positive for all N points.

```
#create a sequential model
model = Sequential()
model.add(Embedding(input_dim=20000, output_dim=32, input_length=300))
# Add 2 bidirectional LSTMs
model.add(Bidirectional(LSTM(32, return_sequences=True)))
model.add(Dropout(0.5))
model.add(Bidirectional(LSTM(8, return_sequences=True)))
model.add(Dropout(0.5))
model.add(Bidirectional(LSTM(4)))
model.add(Dropout(0.5))
# Add a classifier
model.add(Dense(1, activation="sigmoid"))
# model = keras.Model(inputs, outputs)
model.summary()
```

*Figure 6. RNN implementation in Python code*

## 1. Embedding Layer

Embedding layer is one of the layers available in Keras API. This is mainly used in Natural Language Processing related applications such as language modeling.

In dealing with this sentiment analysis problem, we train our own embeddings using Keras embedding layer. This embedding layer will help us to convert each input word into a fixed-length vector of defined size. The resultant vector is a dense one and helps us to represent words in a better way along with reduced dimension.

Three parameters had been set to our embedding layer:

1. Input_dim: 20000, which is the vocabulary size defined in the data processing stage
2. out_dim: 32, we use a 32-dimensional tensor to represent the 20k words in the database

3. Input_layer: 300, which is the length of word vector padded by pad_sequence in the data processing stage.

## 2. LSTM Model

We used bidirectional LSTM in our RNN model. To better understand the LSTM model, we first talk about LSTM.

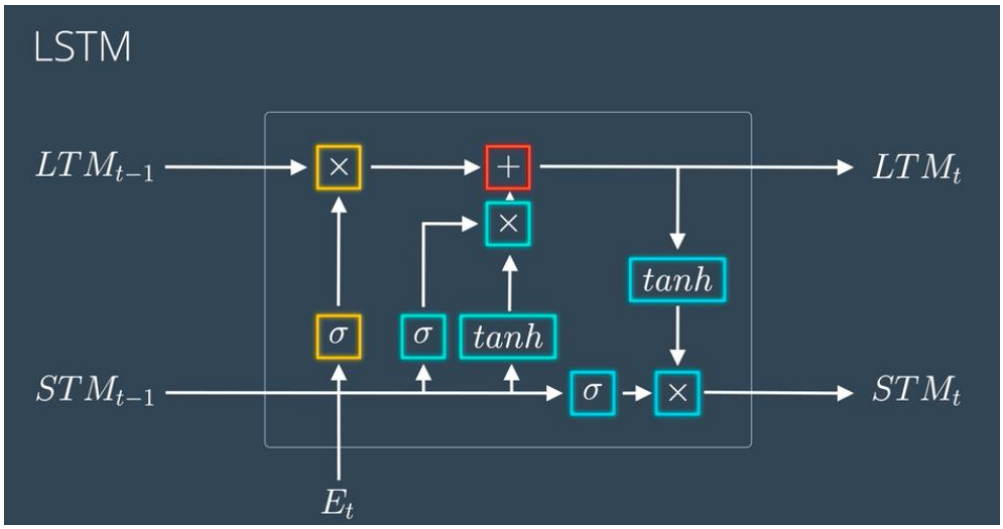


*Figure 7 Structure of LSTM*

### What is LSTM?

Due to stochastic gradients' failure, RNNs are unable to detect long-term dependencies in lengthy sequences. Thus, LSTM was proposed to address this issue. LSTM networks are RNN extensions designed to learn sequential (temporal) data and their long-term connections more precisely than standard RNNs. They are commonly used in deep learning applications like natural language processing.

In Bidirectional LSTM, the input flows in both directions, and it's capable of utilizing information from both sides. It's also a powerful tool for modeling the sequential dependencies between words and phrases in both directions of the sequence.

Bidirectional LSTM adds one more LSTM layer, which reverses the direction of information flow. It means that the input sequence flows backward in the additional LSTM layer. Then we combine the outputs from both LSTM layers in several ways, such as average, sum, multiplication, or

concatenation. Thus, it can produce a more meaningful output, combining LSTM layers from both directions.

For example,

“Apple” is a fruit but can also be a famous smartphone company.

Let us consider these two sentences:

a) Apple is a ***good*** company.
b) Apple is ***good*** to eat.

Thus, using bidirectional LSTM in these two sentences will result in different outputs for every word in the different sentences.

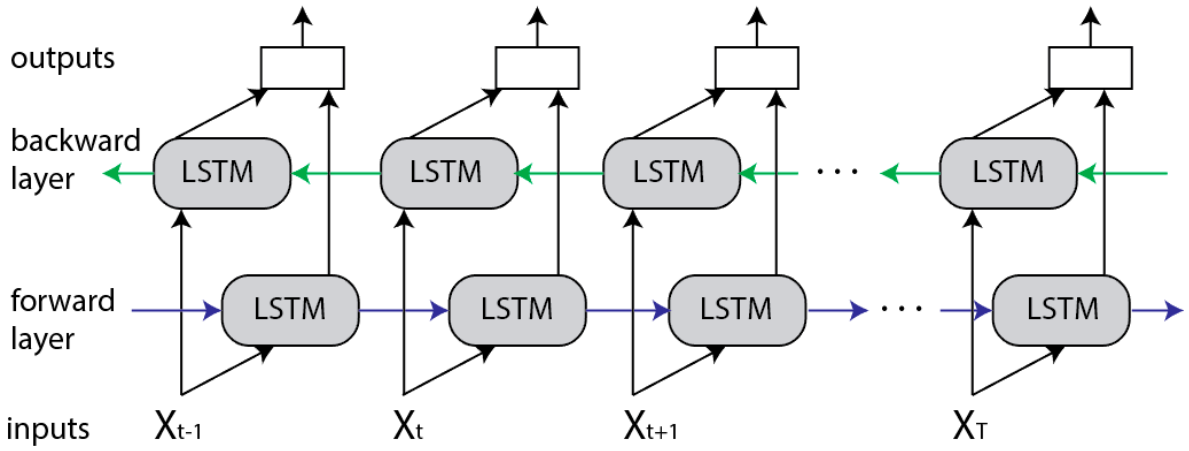


*Figure 8 Unrolled Bidirectional LSTM*

### 3. Dropout Layer

To avoid overfitting our RNN model, we added a dropout layer after each bidirectional LSTM layer. The Dropout layer will randomly set input units to 0 with a frequency rate at each step during training time, which helps to prevent overfitting. Our RNN model building had a rate of 0.5 for our Dropout layer.

### 4. Sigmoid Activation Layer

In this binary classification problem that predicts the project review, the sigmoid function is used to predict the probability of a binary variable.

The sigmoid function is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

*Figure 9 Formula of sigmoid function*

where

X is the input to the sigmoid function

E is the Euler number

**Training Strategy:**

To ensure the producibility of our result, we do not use a random split of our training dataset when training our data. We had fixed that the first 80% of the dataset is the training data, whereas the remaining 20% is testing data. This ensures that we always use the same training set when we do the training and reduces the risk of getting different results in the training process.

Besides, we also tabulate the loss vs. epochs function and accuracy vs. epochs function to evaluate the performance of our RNN model. This is to prevent our RNN model from overfitting and underfitting problems by observing whether the accuracy of validation results had improved for certain epochs. When the overfitting problem is observed, we can perform several methods like performing an early stopping, reducing the number of neurons and layers to reduce the complexity of the network .This ensures we can find the optimum epoch when training our machine-learning model.

## Model Parameters

### a) Learning Rate

#### Learning rate = 0.001

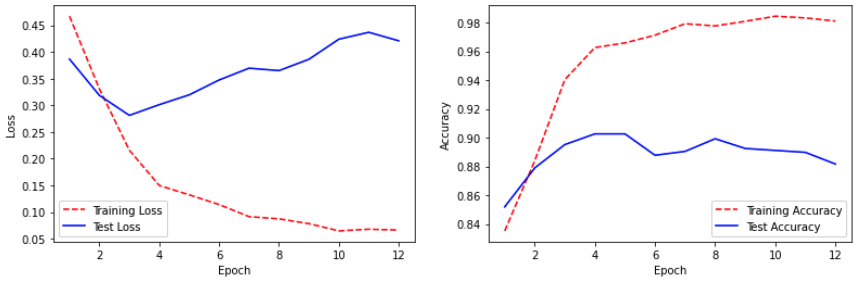


*Figure 9 : Loss vs Epochs and Accuracy vs Epochs (LR=0.001)*

#### Learning rate = 0.0001

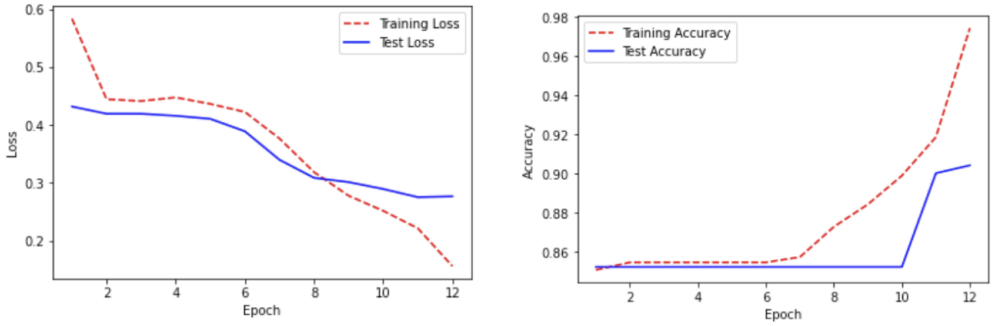


*Figure 10 : Loss vs Epochs and Accuracy vs Epochs (LR=0.0001)*

From the observations above, larger learning rate result in overfitting of the model as the validation loss is increasing after 3 epochs. Thus, we choose to set our learning rate to a much lower rate at 0.0001 to prevent our model from overfitting.

### b) Batch_Size

The batch size should be a power of 2 to take full advantage of the GPU's processing. After several rounds of testing, we found that 32 is the optimum batch size as 64 batch size is too big and the GPU is not able to train appropriately and to lead to poor generalisation, whereas 16 batch size is too small as the weighting based on small batch will be more noisy result and will prevent the descent from fully converging to an optimum.

## c) Number of Epochs

### At 15 Epochs,

| At 15th Epoch | Loss | Accuracy |
|---|---|---|
| Train | 0.1484 | 0.9762 |
| Validation | 0.2872 | 0.9047 |

*Table 2: Loss and Accuracy of Train and Validation dataset at 15 Epochs*

By looking at the validation loss and validation accuracy we observed that the validation loss did not improve after 10 epochs. Thus, this means that the model might be overfitted. We had tried to reduce the number of epochs to 10 but the validation accuracy is too low for prediction. Thus, we had reduced it to 12 epochs and below are results of this epoch.

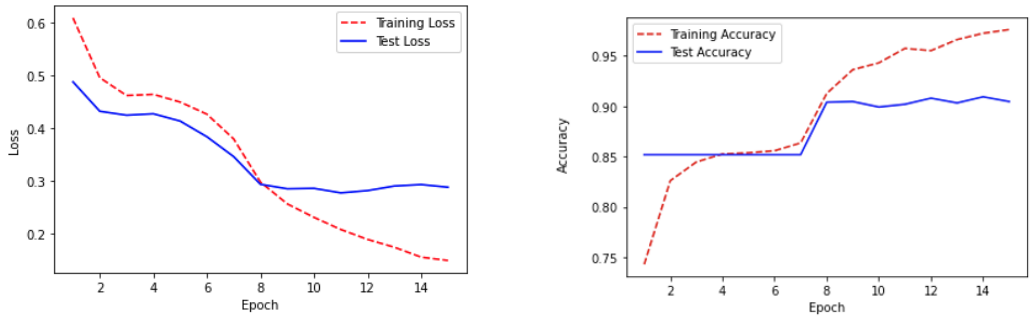


*Figure 11: Loss and Accuracy of Train and Validation dataset across epochs*

### At 12 Epochs,

| At 12th Epoch | Loss | Accuracy |
|---|---|---|
| Train | 0.1559 | 0.9743 |
| Validation | 0.2765 | 0.9040 |

*Table 2: Loss and Accuracy of Train and Validation dataset at 12 epochs*

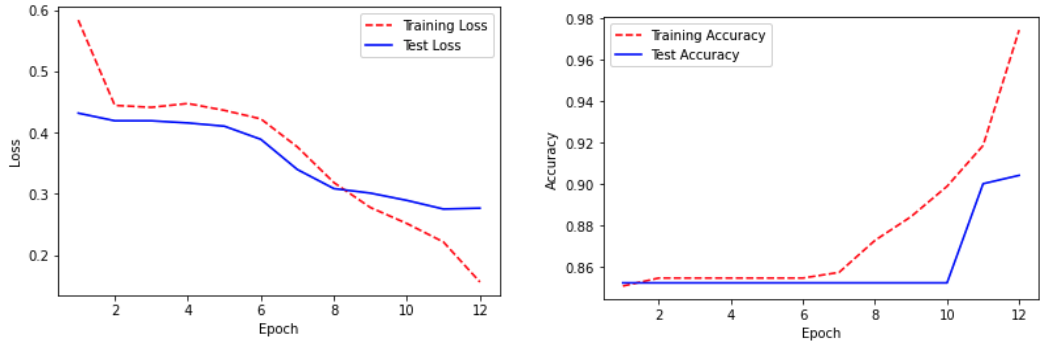


*Figure 12: Loss and Accuracy of Train and Validation dataset across epochs*

At epoch 12, the validation loss has become steady in the Loss vs. Ecoph graph. Therefore we choose our epoch at 12.

**How to run the code**

1. Add train.json and test.json dataset into the content folder in the folder that the jupyter notebook located at.
2. To train our model:

   Run all the cells ( go to the kernel and choose Restart & Run All ) option in the jupyter notebook code that has been submitted.

3. When running the code, the jupyter notebook will help us to
   1. Do data processing includes data splitting, data cleaning and word encoding.
   2. Build the RNN model with the model parameter set by us mentioned above
   3. Train the model using train.json by using the constructed RNN model
   4. Tabulate the loss vs epoch graph and accurate vs epoch graph to evaluate the performance of the model.
   5. Construct the heatmap to visualize the accuracy through the comparison of predicted value and actual value.
   6. Use the trained RNN model to the result in test.json.
   7. Create and save the predicted result into submission.csv and save it in the content folder

**Correctly and Incorrectly classified samples in Test set**

There are 4 different scenarios that our model will produce in the test set.

1. Classify the **positive** sentiments **correctly**

| Reviews | Sentiments |
|---|---|
| everything about the transaction (price, delivery time, quality of item) was ***great***. I wouldn't hesitate to purchase something again from this seller | 1 |
| I purchased last years model of the AT202 (not as pretty, but the same functionally) and took it on my trip to Peru for 4 weeks. The bag is huge and it is well separated so you can divide your things exceptionally well. It can be wheeled, carried (with top handles) or hefted on your back. I put the bag heavily loaded on my back and carried it for about 3 blocks in Cusco. While it wasn't the most ***comfortable*** carry - it allowed me to move ***faster*** and more ***easily***.<br>The one thing I was concerned with the older bag were the zippers. At one point the teeth were not meshing on either the top or bottom sections and I was worried they were ruined. But I simply ran both the zippers to the same end (using small force) and the teeth seemed to be fixed. I really love this bag and will use it often for all my travels, so the potential zipper problem is the only thing I can mention. I highly recommend the bag | 1 |
| These are truly wrinkle free and longer than the average womans botton down, which I ***love***!! Overall, these are ***fabulous*** shirts and you can't beat the price | 1 |

Our model classifies the positive sentiment correctly as there are a lot of positive sentiment words ( great, comfortable, love and fabulous ) in the sentences.

2. Classify the **positive** sentiments **incorrectly**

| Reviews | Sentiments |
|---|---|
| My son ***lves*** the light up cars shoes and they fit ***himm comfortably*** | 0 |
| These are a very ***good*** quality set of socks and the web site I used to ship them was on time and ***fantistic*** | 0 |

The model wrongly classify the positive sentiments sentence when there is a grammatical error in the sentences such as

a) lves → loves
b) himm → him
c) fantistic → fantastic

although the sentences consist of positive sentiment words such as comfortably and good.

3. Classify the **negative** sentiments **correctly**

| Reviews | Sentiments |
|---|---|
| I give the order processing an ***A+*** due to receiving the product within three days.<br><br>The bag, on the other had, leaves a bit to be desired. While the material used is soft, the inexpensive zippers are **difficult to opergate** with one hand | 0 |
| the product is ***fine*** however it you you way too **damn long** to ship!!!!!! | 0 |
| Whoever shot the photos for this bag did a ***great*** job - the bag was not as nice as it looked in the photos. It also **did not fit** the oversized laptop that it seemed it would. The wheels didn't seem like they would take much abuse, and the organizer pockets were cheap plastic. It earns points for color selection and for the handle. All in all, I sent it back despite the fact that the eBag policy is to only give a partial refund. | 0 |

The model classifies the negative sentiments correctly although the beginning of the sentences starts with positive sentiments words such as A+, fine and great.

4. Classify the **negative** sentiments **incorrectly**

| Reviews | Sentiments |
|---|---|
| ***Decent*** lightweight jacket for anytime, not sure if it is just my jacekt, but one of my sleeves seems to be sewed slightly skewed so I have to twist it around to get my arm to go in. Could send it back, but not worth the trouble | 1 |
| Sadly, this ***well designed*** item doesn't work for me. It comes detached from its clip while I'm exercising and plunges to the floor dragging my headphones off | 1 |
| i own this bag in a blue color,it seemed ***nice***.but after 4 months the strapp is | 1 |

| | |
|---|---|
| starting to tear off the bag.now im shopping for a replacment | |

The model wrongly classifies words like decent, well and nice as positive sentiments.

## Strength and Weaknesses of the Model

### a) Strength of Model

1. Able to predict correct sentiments although the user provides different sentiments at the beginning of the reviews
2. The use of bidirectional LSTM models could differentiate the same word that carry different meanings by analyzing the sentences and provide a more accurate result on it.

### b) Weakness of Model

1. Unable to provide correct classification when there is a grammatical error.
2. Unable to provide correct classification when there aren't correct sentiment words in the sentence.

## Difference Choices of Feature Format

### a) One Hot Encoder

One-hot encoding converts the categorical data into numeric data by splitting the column into multiple columns. The numbers are replaced by 1s and 0s, depending on which column has what value.

| Set of all the words in the corpus | | R1: Great Restaurant and great service ! | R2: They can do better to provide better service | R3: Only two thumbs up, worst service ever |
|---|---|---|---|---|
| great | | 1 | 0 | 0 |
| restaurant | | 1 | 0 | 0 |
| and | | 1 | 0 | 0 |
| service | | 1 | 1 | 0 |
| they | | 0 | 1 | 0 |
| can | | 0 | 1 | 0 |
| do | | 0 | 1 | 0 |
| better | | 0 | 1 | 0 |
| to | | 0 | 1 | 0 |
| provide | | 0 | 1 | 0 |
| only | | 0 | 0 | 1 |
| Two | | 0 | 0 | 1 |
| thumbs | | 0 | 0 | 1 |
| up | | 0 | 0 | 1 |
| worst | | 0 | 0 | 1 |
| ever | | 0 | 0 | 1 |

*Figure 13 : One Hot Encoding*

Our model uses One Hot Encoder as our Feature Format. The resource consumption for running One Hot Encoder is 1.16GB for RAM and 22.55GB for Disk.


*Figure 14 : Resource Consumption*

The accuracy of the model using this feature format is 90.87%.

**b) Count Vectors**

This algorithm is very similar to the One-Hot Encoding, but it has the advantage of identifying the frequency/counts of the words in the documents they appear.

| Set of all the words in the corpus | R1: Great Restaurant and great service ! | R2: They can do better to provide better service | R3: Only two thumbs up, worst service ever |
|---|---|---|---|
| great | 2 | 0 | 0 |
| restaurant | 1 | 0 | 0 |
| and | 1 | 0 | 0 |
| service | 1 | 1 | 0 |
| they | 0 | 1 | 0 |
| can | 0 | 1 | 0 |
| do | 0 | 1 | 0 |
| better | 0 | 2 | 0 |
| to | 0 | 1 | 0 |
| provide | 0 | 1 | 0 |
| only | 0 | 0 | 1 |
| Two | 0 | 0 | 1 |
| thumbs | 0 | 0 | 1 |
| up | 0 | 0 | 1 |
| worst | 0 | 0 | 1 |
| ever | 0 | 0 | 1 |

*Figure 15 : Count Vectors*

As the model is quite similar to One Hot Encoding, the resource consumption of the model should be roughly the same. The accuracy of the count vectors will be lower as the frequency of words does not have a direct relationship with the sentiment of the message.

**c) Term Frequency-inverse document frequency (tf-idf)**

The tf–idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general.

This feature format gives more weight to words which appear in fewer documents and less weight to words which appear in many documents.

The vector size generated by tf-idf is roughly the same as the number of unique vocabulary in the dataset which is around 10000 dimensions. Hence, the resource consumption of the tf-idf will be much more higher than One Hot Encoding. Furthermore, the accuracy will be similar to our model as each word is still captured in a standalone manner, thus the context in which it occurs is not captured.

**d) Co-occurrence Vector**

This algorithm is based on the principle that similar words will occur together and will also have similar context.

For example, given a context window, we will count how many times the neighbouring word appears with the word of interest and place it into a matrix as shown in Figure 16.

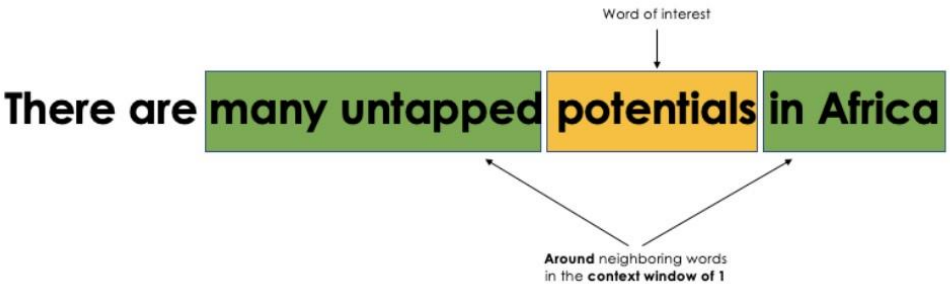


*Figure 16 : Relationship between words of interest and neigbouring words*

Neighboring words

| Words of interest | there | are | many | untapped | potentials | in | africa |
|---|---|---|---|---|---|---|---|
| there | | 1 | 1 | 0 | 0 | 0 | 0 |
| are | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| many | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| untapped | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| potentials | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| in | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| africa | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Figure 17 : Data Matrix of Words of Interest and Neighbouring Words*

The co-occurrence captures the semantic relationship between the words of the same sentence. As we will need to create a huge matrix, it will require a lot of resource assumption although it provides higher accuracy to the sentiment analysis.

With lower resource consumption and higher accuracy, hence we use one-hot encoder for data pre-processing instead of the other selections

## Sentiment Classification on Hotel Reviews

### a) Only Raw Text

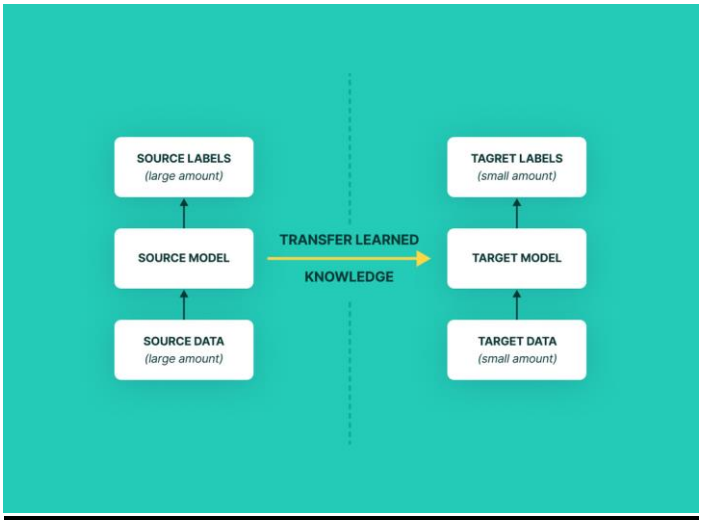


*Figure 18 : Diagram of Transfer Learning*

Since we have already built a model to perform sentiment analysis on product reviews, we would like to perform transfer learning to our current model. In transfer learning, knowledge from the trained model can be leveraged and applied to newer and related tasks. Since both tasks are related to sentiment classification and reviews, we could also use the same model to perform sentiment classification on hotel reviews. The learning in sentiment analysis of product reviews will be utilized and the knowledge will be generalized for hotel reviews.

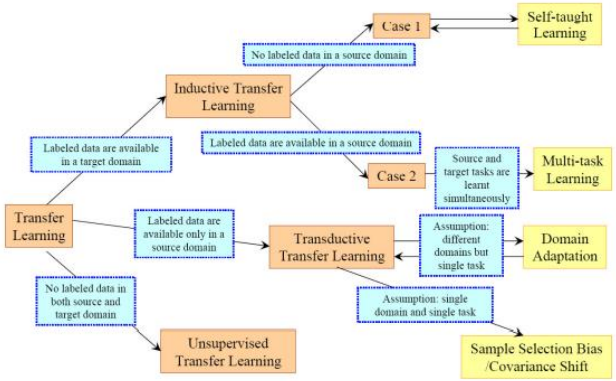


*Figure 19 : Different Transfer Learning Strategies and Techniques*

There are different transfer learning strategies and techniques, which can be applied based on the domain, task at hand and the availability of data. It includes inductive transfer learning, transductive transfer learning and unsupervised transfer learning. In our case, product reviews are labeled with sentiment value, while the hotel reviews do not come with rating scores. Besides, there are some similarities between the two tasks as both are to label the reviews into an integer value. Therefore, transductive transfer learning is the suitable strategy that we are looking into. Moreover, the domain in both cases is the same as both reviews come in raw text and can be converted to the same feature space by using One Hot Encoding. Hence, covariance shift should be performed in the transductive transfer learning.

In transductive transfer learning, product reviews will act as training data and hotel reviews will be the testing data. Both data will be observed beforehand. Learning will be performed from the product reviews and the rating scores of hotel reviews will be predicted. The prediction will be based on the patterns and additional information in the data during the learning process. Instead of building a generic model, transductive learning builds a model that fits both training and testing data points it has already observed. This approach predicts the rating scores of hotel reviews using the knowledge learned from the product reviews.

## Modification

Several actions can be taken in order to improve the performance in sentiment classification on hotel reviews. Firstly, the starting layers could be frozen from the pre-trained model to avoid the loss of all the learning that has already taken place. This will also avoid the additional work of making the model to learn the basic features again.

Besides, additional layers such as LSTM layers and dropout layers could be added on top of the base model. This might help in improving the performance of the new task.

On the other hand, the pre-trained model's final output of sentiment analysis of product reviews and sentiment classification of hotel reviews are different. Sentiment analysis of product reviews only requires the model to predict '0' for negative sentiments and '1' for positive sentiments, while sentiment classification of hotel reviews requires the model to predict the rating scores which are in the range of 1 to 5 (integer value). Therefore, a new output layer needs to be placed in the training.

Lastly, fine-tuning will be performed to improve the performance of the model. Fine-tuning involves unfreezing some parts of the base model and training the entire model at a very

low learning rate. The low learning rate will increase the performance of the model on the sentiment classification of hotel reviews while preventing overfitting.

### b) Raw Text and Rating Scores (Inaccurate)

In this case, both product reviews and hotel reviews come with their sentiment value. Since both sentiment classification is in the same domain, inductive transfer learning can be used to perform sentiment classification on hotel reviews. In inductive transfer learning, the knowledge from the model of product reviews will be used and applied to the model of hotel reviews to improve the prediction. Since the model trained for product reviews already has expertise in the features of the domain, it is a better starting point than if we were to train the model of hotel reviews from scratch.

Moreover, multi-task learning can be performed as both the label of product reviews and hotel reviews are available. There are two most commonly used ways to perform multi-task learning in a neural network, which is hard parameter sharing and soft parameter sharing. Hard parameter sharing is generally applied by sharing the hidden layers between all tasks, while keeping several task-specific output layers. On the other hand, in soft parameter sharing, each task has its own model with its own parameters. The distance between the parameters of the model will then be regularized to encourage the parameters to be similar.

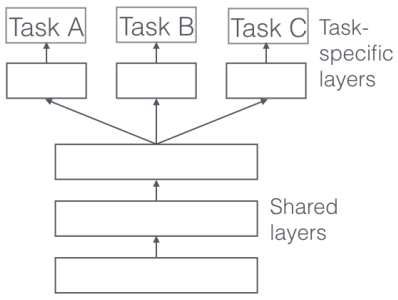


*Figure 20 : Diagram of Hard Parameter Sharing*

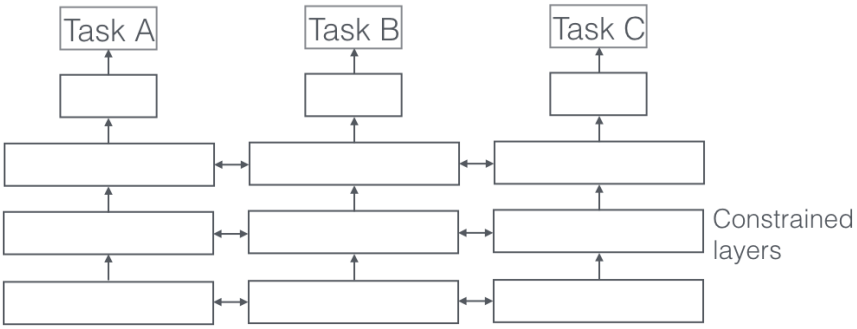


*Figure 21 : Diagram of Soft Parameter Sharing*

In our case, hard parameter sharing will be chosen as the techniques used for inductive transfer learning as it acts as regularization and reduces the risk of overfitting. This model is also hope to learns a representation that will generalize well for all tasks

Three approaches that we can improve our algorithm to perform well in this new problem:

**a) Adding a noise layer over the base model**

As we do not want our base model to learn noise, this noise layer will learn the transition between clean labels and bad labels. The model with the noise layer is hoped to overfit during training. During testing, this layer will then be removed.

**b) Create a small dataset with clean labels**

We can filter the dataset which is not labeled by noise and train the model on this dataset. The model will also be trained on the whole dataset to compare the feature vector from the clean dataset to that of the whole dataset. The similarity between the feature vectors will be used to decide if it is a correct label.

**c) Use a more robust loss function**

Binary cross-entropy loss tends to overconfidence especially when dealing with noisy samples. Therefore, we might consider using a different loss function such as Mean Absolute Error (MAE) Loss in this case. MAE loss is calculated as the average of the absolute difference between the actual and predicted values and it is more robust to outliers.

## Contributions by each team member

| Question | Contributed By |
|---|---|
| **a.** | Lim Jia Wei |
| **b.** | Khor Chin Yi |
| **c.** | Khor Chin Yi |
| **d.** | Khor Chin Yi, Lim Jia Wei, Shoon Zhen Yong |
| **e** | Shoon Zhen Yong |
| **f.** | Shoon Zhen Yong |
| **g** | Lim Jia Wei |
| **h.** | Lim Jia Wei |