# 



School of InfoComm Technology



**Deep Learning Assignment**

Diploma in CSF / FI / IT

Apr 2022 Semester

**ASSIGNMENT 2**

(40% of DL Module)

4th Jul 2022 – 12th Aug 2022

**Submission Deadline:**

**Presentation: 12th Aug 2022 (Week 17),**

**Report: 12th Aug 2022 (Friday), 11:59PM**

|  |  |  |
| --- | --- | --- |
| **Tutorial Group** | **:** | **P01 / P02 / P03 / P04** |
| **Student Name** | **:** | Dominic Lee |
| **Student Number** | **:** | S10222894 |

**Penalty for late submission:**

10% of the marks will be deducted every calendar day after the deadline.

**NO** submission will be accepted after 21st Aug 2022 (Sunday), 11:59PM.

Table of Contents

[1](#_Toc111214267)

[**1.** **Overview (500-1000)** 2](#_Toc111214268)

[**2. Data Loading and Processing (500-1000)** 3](#_Toc111214269)

[7](#_Toc111214270)

[7](#_Toc111214271)

[**3. Develop the Sequence Generator Model(s) (1000-2000)** 7](#_Toc111214272)

[**4. Use the developed Model to Generate Texts (500-1000)** 29](#_Toc111214273)

[30](#_Toc111214274)

[**5. Summary (500-1000)** 32](#_Toc111214275)

# **Overview (500-1000)**

**Objective**

The objective of the problem is to build a recurrent neural network that generates text character by character. The aim is to optimise the model’s validation accuracy as well as ensure that the text it produces is of good quality. The reason why the quality of text is not the main metric to optimise is that it is extremely difficult to compare different text generated and say one is better than the other, especially if they are similar in quality. Hence, validation accuracy will be the main metric used to separate text quality and by definition if a validation accuracy of a model is higher than that of another model, it would mean that the former model is to be preferred.

**Problem**

A problem that occurs is that it is hard to determine and compare whether the text generated is better than another generated text or is even good. As such, the metrics used to judge and evaluate the models includes

1. Validation accuracy
2. My own judgement/intuition
3. Rogue score

The metrics in this list goes from most to least important. Regarding my own judgement, sentences will be analysed and the spelling of the words, the coherency of the sentences generated, and the grammar of the text will be considered. For accuracy, validation accuracy will be used as the main metric as it shows how well the model can generate text on unseen data which is vital. Lastly, rogue score will be used only when the text generated have similar accuracies and are perceived to be similar in quality of text generated. In simple terms, Rogue score calculates the similarity between the produced text (model produced text) and the reference text (actual correct text). The problem with Rogue score is that it does not account for meaning of the words and is rather a measure of how statistically similar two sentences are. It will play a small role in deciding between text quality.

Another problem is there are many hyperparameters to tune and it will be extremely hard to extensively test all the hyperparameters. Hence, it is important that we test a select few hyperparameter combinations and that these hyperparameters are important to the model.

Lastly, as the book is rather long which would indicate long training times added with the constraints from the hyperparameter problem, time is an issue which limits the amount of extensive testing able to be done.

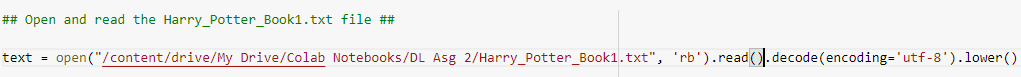
**Approach**

For problem 2, two models will be tuned and tested against each other. Firstly, Conv1D layers will be used for one of the models and this is a non-RNN based model. The other model which will be used is a RNN model. As there are various types of RNN layers, tests between LSTM, Bidirectional LSTM and GRU layers will be done at the start to determine which one works best. By building two models, one RNN and one non-RNN, the pros and cons of the models will be seen.

For both models, they will be trained following the universal workflow of machine learning, where a baseline model is initialised, the hyperparameters are tuned and finally techniques to delay overfitting are implemented.

Doing so allows for systematic tuning and training of the models which would produce a model that reaches an accuracy which comes close to the optimal model’s accuracy.

# **2. Data Loading and Processing (500-1000)**

**Loading data**

The data came in a text file which was encoded in an utf-8 format. Thus, after loading the text file, the file was decoded and all text in it was lowercase.

Text

Description automatically generated with medium confidence

The total number of characters in the text file was printed out to better understand the volume of the data that is given.

**Data pre-processing**

The next step would be to pre-process the data. This includes cleaning the data as well as vectorizing the data. By cleaning the data, lines and characters that have no meaning to the model which only add noise to the model will be removed. These include special characters which are rarely used, or white space lines that have no meaning. Vectorization of the data is needed as the model can only process numeric characters although the book uses word. As such, it manipulates the data to be understood by the model which lets the model train on it.

The first step taken was to convert the text into a list which is split every line by the character “\n”. The aim was to remove the useless lines which included whitespace lines (lines which had no characters) and lines that contained the page numbers.

By converting the text into a list, these lines would be isolated where they can be removed without affecting other text.

Text

Description automatically generated

The whitespace lines were seen as empty strings “” in the list and the lines containing the page numbers all started with the string “page |”. Using these a for loop was used to iterate through the list and remove all lines which contained these patterns.

Next, the characters that were redundant had to be removed, these characters rarely occur in text and even if they do, they do not provide any semantic meaning to the text.

Graphical user interface, text, application

Description automatically generated

The code contains a list of characters that are deemed as redundant characters and the loop in the screenshot iterates through the entire text to replace these characters with a whitespace.





After removing these characters and the page numbers, the total number of characters in the text file decreased by approximately 40000 characters. This is evident of how infrequent these characters occur, which would also indicate they are more likely to confuse the model then to help improve the model’s generalisation and accuracy (Majority of the characters removed come from the removal of page numbers).

Following the removal of unnecessary characters, another for loop will be used to iterate through the text and find a list of distinct characters found in the text. This is important as the length of this list will be the number of neurons present in the text generation model’s output layer.

Text

Description automatically generated

From the screenshot above, the characters that are present in the text are characters from 0-9, a-z as well as punctuation that occurs frequently in sentences or those that carry semantic meaning such as “!” which could show a sense or urgence or excitement.

Graphical user interface, text, application

Description automatically generated

The penultimate step in the data pre-processing for problem 2 is to prepare data into training text and labels (X & y). This was done using the sliding window technique where there are two pointers that basically iterate through the entire text, generating the training and test data.

All the training samples will be set to a of length 60 (maxlen=60), and the number of steps taken (characters moved) per iteration will be 3 (step=3). For each iteration of the loop, the sentences list (training data) will have a string of length 60 appended to it while the next\_chars list (test data) will have the next character (character right after the string) appended to it. That will serve as the output that the model would have to predict. Additionally in this step, all the distinct characters are mapped to a number preparing for it to be fed into the model.

The final step in the data pre-processing is to vectorize the text into numeric tensors so that they can be fed into the model for training.

The code above vectorizes the data by doing the following. It first creates two NumPy zero arrays with the X array having the shape len(sentences), maxlen, len(chars) and the y array having the shape len(sentences), len(chars). Using the parameter dtype=bool, both arrays are filled with Boolean False values. The reason why X is a 3D array is that it will contain all the sentences which is why its first dimension is of len(sentences). The second dimension is of length 60 (maxlen) because each sentence consists of 60 characters and the last dimension is of length 47 (len(chars)) because there are 47 different options for every character, and only one of these 47 values will be True for every character. The same holds through for the y array except that it does not have the dimension (maxlen) because it only contains one character.

Text

Description automatically generated

# Graphical user interface, text Description automatically generated

# Table Description automatically generated

Once vectorization of the data is complete, the data pre-processing step is over, and the data is ready to be fed into the model for the model to train.

# **3. Develop the Sequence Generator Model(s) (1000-2000)**

**RNN Model**

The way that overfitting will be detected during training will be using early callbacks which have a patience of 15 and the monitored metric will be validation loss. The rationale is that if the validation loss of a model has been on an increasing trend for 15 epochs in a row, it is almost certain that overfitting is taking place and training can be stopped prematurely.

The first model that I will be building is the model which will use RNN layers and below are the hyperparameters that will be tested to optimise the model’s performance:

1. LSTM/GRU/Bidirectional LSTM
2. Number of neurons in each RNN layer and Number of RNN layers
3. Number of neurons in each Dense layer and Number of Dense layers
4. Layer Normalisation
5. Learning Rate
6. Dropout
7. Regularizers
8. **LSTM vs GRU vs Bidirectional LSTM**

When testing these 3 different layers against each other, all hyperparameters values were kept the same except for the type of layer used.

The difference between LSTMs and bidirectional LSTMs is that the unidirectional LSTM only preserves information from the past while bidirectional LSTM preserves information from both the past and future, allowing them to understand the context of sentences better.

GRU on the other hand, only has 2 gates compared to LSTM’s 3 gates. This allows it to run faster seeing that it is less complex while giving similar levels of accuracy. It is generally better suited on smaller datasets compared to LSTMs.

**Baseline model**

Text

Description automatically generated

The baseline model architecture consists of two RNN layers followed by two dense layers. It uses the Adam optimizer with a learning rate of 0.003. The batch size used is 256 and the loss function is categorical cross entropy.

**GRU**Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Model overfits ~ Epoch 10 Validation Accuracy ~ 55%

**Bidirectional LSTM (Selected)**

Chart, line chart

Description automatically generatedChart

Description automatically generated

Model overfits ~ Epoch 15 Validation Accuracy ~ 55%

**LSTM** Chart

Description automatically generatedA picture containing histogram

Description automatically generated

Model overfits ~ Epoch 30 Validation Accuracy ~ 54%

The LSTM layer will not be used as it has the lowest accuracy out of the three and takes the longest to converge. Both the bidirectional LSTM and GRU layers hit 55% validation accuracy but as bidirectional had less noise during training it is chosen as the RNN layer to be used.

**Summary**

**Table

Description automatically generated**

1. **Number of neurons in each RNN layer and Number of RNN layers**

This is where the model’s architecture is tuned, the optimal architecture is one where it is as simple as possible yet reaches the highest validation accuracy out of all possible architectures.

**Bidirectional +1 layer**

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Model overfits ~ Epoch 14 Validation Accuracy ~ 55%

**Bidirectional 128 Nodes (Selected)**

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Model overfits ~ Epoch 10 Validation Accuracy ~ 56%

**Bidirectional 256 Nodes**

Chart, line chart

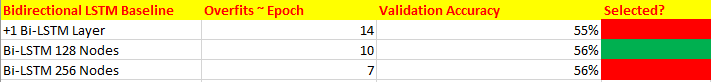
Description automatically generatedChart, line chart

Description automatically generated

Model overfits ~ Epoch 7 Validation Accuracy ~ 56%

Though there is an increase in the model’s complexity through an addition of a layer, there are no improvements however, with 128 and 256 nodes the validation accuracy increases to 56%. As the model using 128 nodes is less complex than 256 nodes, the 128 nodes model is selected.

**Summary**



1. **Number of neurons in each Dense layer and Number of Dense layers**

Similarly, this parameter will help form the model’s architecture and generally, the more complex the model, the better it can generalise, but the sooner overfitting will occur.

**+1 Dense Layer**Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Model overfits ~ Epoch 8 Validation Accuracy ~ 56%

**+2 Dense Layers**Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Model overfits ~ Epoch 9 Validation Accuracy ~ 55%

**128 Nodes (Selected)**

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Model overfits ~ Epoch 8 Validation Accuracy ~ 57%

**256 Nodes**

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Model overfits ~ Epoch 7 Validation Accuracy ~ 57%

Here the increase in the number of dense layers had no effect on the model’s validation accuracy, however by increasing the number of nodes in the dense layer, the validation increased by 1% to 57%. As both 128 and 256 nodes gave similar accuracies, the model with 128 nodes will be selected as it is less complex.

**Summary**

**Graphical user interface, table

Description automatically generated**

1. **Learning Rate**

Learning rate is the hyperparameter that tunes how fast the model learns and converges. It is an essential parameter to tune as it plays a big role as to how high of an accuracy the model can hit. Too high a learning rate causes the model to converge too quickly and too low a learning rate will cause the model to train very slowly and potentially get stuck in a local minimum resulting in lower accuracies.

**Lr = 0.001**Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Model overfits ~ Epoch 12 Validation Accuracy ~ 55%

**Lr = 0.005 (Selected)**Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Model overfits ~ Epoch 7 Validation Accuracy ~ 57%

The original learning rate was 0.003. Using a lower learning rate, the validation accuracy hit was lower at 55%, and the model took longer to converge only overfitting at epoch 12. However, using a higher learning rate of 57%, the model hit roughly the same validation accuracy and converged faster thus, that will be the model chosen.

**Summary**

****

1. **Layer Normalisation**

Layer normalisation aims to provide a uniform scale for the weights and biases in the layer it is applied to. Normally when there is numerical data that varies in a huge range, the model is prone to be biased to the data that are in the larger ranges. Layer normalisation ensures that the data are in an appropriate range which would allow the model to hit better accuracies.

Layer normalisation is similar to batch normalisation. The difference is that it works better than batch normalisation in RNNs but worse in CNNs, and it is not dependent on batch sizes since it normalises through layers and not batches.

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Model overfits ~ Epoch 11 Validation Accuracy ~ 56%

Overall, the model with layer normalisation hit a lower accuracy than the current selected model of 57% and will not be implemented.

1. **Dropout**

Dropout is an overfitting technique that is used to combat overfitting. There is a possibility the model might be able to train to higher validation accuracies when overfitting is delayed. Dropout works by randomly dropping neurons during training in each iteration which prevents the model from relying on certain neurons too much.

Dropout will be implemented in the recurrent and dense layers in the model. The higher the dropout rate, the stronger the effect it will have on preventing overfitting.

**Dropout 0.1 (Selected)**

Chart

Description automatically generatedChart

Description automatically generated

Model overfits ~ Epoch Nil Validation Accuracy ~ 58%

**Dropout 0.2**Chart

Description automatically generatedA picture containing chart

Description automatically generated

Model overfits ~ Nil Validation Accuracy ~ 58%

**Dropout 0.3**Chart

Description automatically generated with medium confidenceChart

Description automatically generated

Model overfits ~ Epoch Nil Validation Accuracy ~ 57%

From the results of the models, it is observed that dropout delays overfitting substantially. For dropout 0.1 and 0.2 the validation accuracy increased till 58% from 57%, the validation loss graph shows no sign of overfitting for 0.2 and slight signs of overfitting for 0.1, the validation loss however have stagnated indicating that the accuracies are very unlikely to improve any further. Also, for a dropout value of 0.3, the accuracy hit was 57% and the model also showed no signs of overfitting other than the validation loss stagnating from around epoch 50 onwards. A dropout value of 0.1 will be used.

It is also important to note that for these models with dropout, the training accuracy tends to be very low compared to the validation accuracy. Reason being that dropout is only applied during training, and not during validation.

**Summary**

**Table

Description automatically generated**

1. **Regularizers**

The other technique to combat overfitting is using regularizers. Regularizers prevent overfitting by penalising the neurons with a value called alpha. This is done to prevent the weight of neurons from exploding and to allow the model to generalise better, achieving higher accuracies.

As dropout is already implemented, adding too much regularisation will lengthen the training process by too much. Hence, only recurrent regularizers (Bidirectional layer) were tested with.

A picture containing chart

Description automatically generatedChart

Description automatically generated**L2 = 0.01**

Model overfits ~ Epoch 20 Validation Accuracy ~ 54%

**L2 = 0.007**A picture containing chart

Description automatically generatedChart

Description automatically generated

Model overfits ~ Epoch 20 Validation Accuracy ~ 54%

The regularizers did not help the model as the validation accuracy for both values decreased to 54%. Both validation loss graphs showed slight signs of overfitting at around epoch 20, though it is not obvious. However, it is safe to say that the accuracy would not increase further.

**Summary**

****

**Final Model**Chart

Description automatically generated

**Analysis of text generated**

**Text

Description automatically generatedFinal Text**

From the text generated, the temperature of 0.5 produces text that are the most coherent accurate in spelling. Almost all the words are actual English words, the sentences are somewhat coherent and there are some forms of punctuation. The issues with it would be the lack of meaning in the sentences that are generated. The text also introduces some names from the Harry potter book such as Harry which indicates that the text generated will always be in a Harry Potter context.

**Conv1D Model**

Below, a model will be trained using Conv1D layers which is not a RNN layer. This is used to compare the performance between RNN and non-RNN models.

The hyperparameters that will be tuned goes in the order of:

1. Number of Conv1D/Maxpooling layers and number of filters in each Conv1D layer
2. Number of Dense layers and neurons in each Dense layer
3. Learning Rate
4. Kernel Size
5. Batch Normalisation
6. Dropout

**Baseline model**

Graphical user interface, text, application

Description automatically generated

This is the baseline model for my Conv1D model. It consists of two Conv1D, Maxpool1D and Dense layers. The number of neurons in all layers is 64, the optimizer is Adam with a learning rate of 0.003, activation function for all layers except the output layer (Softmax) is ReLU. Kernel size and padding are 3 and “same” for Conv1D layers respectively.

Chart, scatter chart

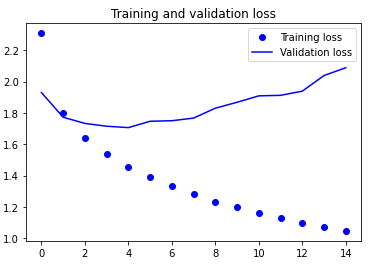
Description automatically generatedChart, scatter chart

Description automatically generated

Model overfits ~ Epoch 3 Validation accuracy ~ 47%

1. **Number of Conv1D/Maxpooling layers and number of filters in each Conv1D layer**

**+1 Conv1D/Maxpool layer (Selected)**Chart, scatter chart

Description automatically generated

Model overfits ~ Epoch 4 Validation accuracy ~ 50%

**+2 Conv1D/Maxpool layer**

Chart, scatter chart

Description automatically generatedChart, line chart, scatter chart

Description automatically generated

Model overfits ~ Epoch 5 Validation accuracy ~ 50%

When a Conv1D layer is added, by default a Maxpooling layer will also be added. By adding a Conv1D layer, the validation increased from 47% to 50%. This accuracy remains when two Conv1D layers are added. Hence, the model with an additional Conv1D layer is selected.

**Summary**



**Conv1D 128 Filters (Selected)**

Chart, line chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generated

Model overfits ~ Epoch 3 Validation accuracy ~ 51%

**Conv1D 256 Filters** Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generated

Model overfits ~ Epoch 3 Validation accuracy ~ 50%

When 128 filters are used, there is an increase in validation accuracy to 51%, hence the model will use 128 filters in the Conv1D layers.

**Summary**

**Table

Description automatically generated**

1. **Number of Dense layers and neurons in each Dense layer**

**+1 Dense layer**

Chart, line chart, scatter chart

Description automatically generated Chart, scatter chart

Description automatically generated

Model overfits ~ Epoch 5 Validation accuracy ~ 51%

Chart, line chart

Description automatically generated**128 Nodes**Chart, scatter chart

Description automatically generated

Model overfits ~ Epoch 3 Validation accuracy ~ 51%

The increase in complexity of the model by increasing the number of nodes in the hidden dense layer shows no improvement and the current selected model will continue to be tested with.

1. **Learning Rate**

**Learning rate = 0.005**

Chart, line chart

Description automatically generatedChart, scatter chart

Description automatically generated

Model overfits ~ Epoch 6 Validation accuracy ~ 50%

Chart, line chart, scatter chart

Description automatically generated**Learning rate = 0.001**Chart, scatter chart

Description automatically generated

Model overfits ~ Epoch 5 Validation accuracy ~ 50%

Likewise, the learning rate being higher or lower has no improvements to the model’s performance but rather decreases its validation accuracy to 50%.

1. **Kernel Size**

The kernel size of a conv1D layer refers to the length of the sliding window used in the convolution operation. What it means for a conv1D layer to have a kernel size of 3 and filters=128 is that there will be 128 filters with each filter having a length of 3.

The kernel is used to extract features from an image, using a larger kernel the feature we extract will be larger and less specific compared to small kernel sizes. What this means is that the larger kernel sizes might miss out on specific details, but it would remove more noise from the text.

Chart, line chart, scatter chart

Description automatically generated**Kernel Size = 5**Chart, scatter chart

Description automatically generated

Model overfits ~ Epoch 4 Validation accuracy ~ 52%

**Kernel Size = 7**

Chart, scatter chart

Description automatically generatedChart, line chart, scatter chart

Description automatically generated

Model overfits ~ Epoch 3 Validation accuracy ~ 53%

The kernel size of the Conv1D layers had a significant impact on the model’s performance. Using a size of 5 increased the validation accuracy to 52% and 7 increased the validation accuracy to 53%.

**Summary**

**A picture containing table

Description automatically generated**

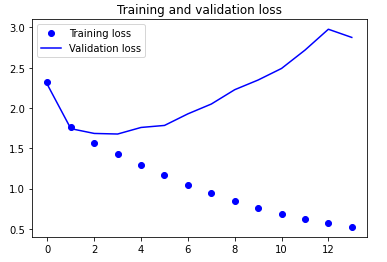
1. **Batch Normalisation**

Batch normalisation normalises to inputs within the layer, it is similar to layer normalisation except that it normalises each layer’s inputs in mini batches. It uses the mean and variance of the current mini batch that it is applied to normalise the mini batch. It tends to work very well on convolutional models.

In theory batch normalisation allows a network to train faster, provides some form of regularisation and can give better overall results for the neural network.

Batch normalisation is applied to all Conv1D layers.

Chart, scatter chart

Description automatically generated

Model overfits ~ Epoch 3 Validation accuracy ~ 51%

There is a drop in validation accuracy and as such, we will not implement batch normalisation.

1. **Dropout**

Dropout is more effective on problems where there is a limited amount of training data, and the model is likely to overfit on the training data. As many of the models overfit early into training and there is not an abundance of training data, dropout will be the only regularisation used to combat overfitting.

A picture containing graphical user interface

Description automatically generated**Dropout 0.3**Chart

Description automatically generated

Model overfits ~ Epoch Nil Validation accuracy ~ 54%

**Dropout 0.2 (Selected)**

Chart

Description automatically generatedA picture containing histogram

Description automatically generated

Model overfits ~ Epoch Nil Validation accuracy ~ 55%

**Dropout 0.1** Chart

Description automatically generatedChart

Description automatically generated

Model overfits ~ Epoch 10 Validation accuracy ~ 54%

The model using a dropout of 0.2 hits the highest validation accuracy of 55% and will be the final model. The other dropout values come close at 54%. Note that the models using dropout values of 0.2 and 0.3 do not overfit, rather their validation loss has stagnated, and it is highly likely that their accuracies will not increase further.

**Summary**

**A picture containing chart

Description automatically generated**

**Final Model**

**Graphical user interface

Description automatically generated**

**Analyse generated text**

**Final Model**

**Text

Description automatically generated**

The code above is the generated text from the final model and the validation accuracy of the model was 55%. From the temperatures listed, a temperature of 0.5 seems to give the most desirable text output. The text generated are all proper English words except for the word “yefless”. The text shows slight signs of the sentences making sense and lastly, it is not repetitive. Also, a Rogue score of 0.62 means that 0.62% of n-grams in the text generated are also present in the actual sentence.

**Overall**

The best model generated would be the RNN model as it manged to hit higher a higher accuracy. Though it is important to note that there are no significant differences in the quality of the text generated as both are around the same quality.

Even though rogue score for the Conv1D model is higher, it has been seen to be an imprecise metric seeing that it only accounts for the statistical properties of the text.

# **4. Use the developed Model to Generate Texts (500-1000)**

The next step will be to develop the chosen model to generate texts on real-life text inputs that are not from the Harry Potter book. The generated text will then be analysed to see if the quality of text is good.

The steps taken to apply the model on a real-life text input will be illustrated followed by the analysis of the generated text.

**Saving the best model**

**Text

Description automatically generated**

Saving the RNN model was the first step taken, the model will be loaded in the steps below to be used to make predictions (generate text).

**Loading the model**

**Text

Description automatically generated**

**Processing user input**

# **Text Description automatically generated**

The code above takes a user input and whatever characters that are not in the list chars will be removed. Furthermore, the text is lowercased and because the model only accepts inputs of length 60, if the input if shorter than 60 characters, it will be pre-padded with whitespaces, and if the input is longer than 60 characters, it will be post-truncated till it is of length 60.

**Generating text**

**Text

Description automatically generated**

Above is the code that is used to generate the text. The outermost for loop iterates through the various temperature ranges that needs to be tested with, the next for loop is responsible for generating the number of characters the model predicts, and the innermost nested for loop is responsible for vectorizing the user input. This is achieved by first creating a 3D array called sampled, which has the dimensions, 1 x 60 x 47. After, the model makes a prediction using the predict and sample function which generates the model’s next predicted character. This generated character is appended to the variable generated\_text, which was assigned to the user input, before being sliced to omit the first character hence maintaining a length of 60 characters.

**Analysis of model predictions**

As a temperature of 0.5 was the best performing temperature for both models, it will be used as the temperature to generate the text.

There were 3 test cases used, the first test cased tested when the input was longer than 60 characters long, the seconds test case tested special characters with a sentence that does not make sense and the last test cases used a short incomplete sentence.

**Test case 1**

Text

Description automatically generated

**Test case 2**

Text

Description automatically generated

**Test case 3**Text

Description automatically generated

From the above three test cases, the text generated do not relate to the context of the inputs and generally start to have no contextual meaning a few words in. The use of the connector “and” is widely seen in all three cases and word that closely relate to Harry Potter are generated. These words include “Hogwarts”, “broomstick”, “Hagrid”, “Gryffindor”. Sentences produced had grammar mistakes and had little meaning behind them. Finally, the use of punctuation was quite appropriate seeing that it was well spaced throughout the text and the positions they were used in made sense.

Overall, the model could not make insightful predictions that related to the context of the inputs, it also could not generate sentences with meaning though the structure of the sentences were present. However, the model could generate text that were actual English words, use punctuation rather well, and use terms that are only in the context of Harry Potter.

# **5. Summary (500-1000)**

**Summary**

For problem 2, the task was to build a model that could generate text using a Harry Potter book. The initial step taken was to pre-process the data so that it could be fed into the model. This was done by removing useless characters and lines in the data and vectorizing the data using One Hot Encoding into a numeric form so that the model can train on it.

A RNN model and a non-RNN model were built to perform text generation on the dataset. Both model’s hyperparameter were tuned in the following order:

* + - 1. Model’s architecture
      2. Model’s parameters
      3. Parameters that deal with overfitting

During training, the validation accuracy was the metric that was optimized.

Thereafter, the model’s accuracy and text generation using a sequence in the dataset were compared and analysed to determine the best model which in this case was the RNN model. Both models had similar quality of text generated but the RNN had a higher validation accuracy which is why it was chosen.

With the selected model, custom inputs were made to observe how the model would fare at generating text on these inputs and the output of the model was analysed to assess the model’s limitations.

**Further Improvements**

The objective was to improve and optimize the validation accuracy of the model which would by in most cases allow the model to generate better quality text.

As seen in the above analysis, the model tends to generate text in the context of the data that it is trained on. Thus, depending on the general context of inputs that the model will receive e.g., Fictional books. The model can be trained more on those type of text. In this case, another Harry Potter book could be used to train as it would provide more data with the same writing style and context.

Furthermore, a larger variety of models and hyperparameters could be used to tune the model. For example, at the start of the RNN model test, both GRU and Bidirectional LSTM hit validation accuracies of 55%, but only the Bidirectional LSTM model was chosen. Certain hyperparameters like batch size were also not tuned and left out of the training phase. This lack of testing could be attribute to length and time constraints of the problem.

Lastly, pertaining to the length of input that the model can accept, due to technical incapabilities the model is only able to accept inputs of length 60 and inputs will be pre-padded with whitespaces and post-truncated to a length of 60. This padding can potentially confuse the mode and the truncation takes away some meaning of the input.