Configuration Manual

MSc Research Project
Data Analytics

Dev Sharma
Student ID: X23139676

School of Computing
National College of Ireland

Supervisor: Abdul Qayum

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Dev Sharma
Student ID:	X23139676
Programme:	Data Analytics
Year:	2024
Module:	MSc Research Project
Supervisor:	Abdul Qayum
Submission Due Date:	12/08/2024
Project Title:	Configuration Manual
Word Count:	1361
Page Count:	5

Signature:	Dev Sharma
Date:	12th August 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Dev Sharma
X23139676

1 Introduction

This configuration manual is designed in such a way that it provides a comprehensive and detailed instructions to set up, run and reproduce the research titled as "LSTM Hyperparameter Optimization". The main purpose of this manual is to make sure that every person be it a researcher, students or practitioners can reproduce this study easily without any hassle and difficulties and validate the results that are presented in the thesis titled as above. With the help of the following steps in the manual, users will be able to recreate the execution environment. Preprocess the data, can implement the model and conduct the experiments in a systematic order.

The manual serves the following main purposes:

1. Reproducibility: This ensures that the experiments conducted by the researchers can replicated, which is important for validation purpose. Reproducibility is one the primary fundamental aspect of any scientific research that increases the credibility and the reliability of the results.

2. Ease of Use: This manual provides a clear and step by step instruction that are required to set up the complex working environment in an easy way. Ease of use becomes important because any person that has less experience will be able to set up and execute the program without any issues.

3. Troubleshooting Errors: This manual will also help researchers to solve any kind of errors that may occur during or before the execution of the program. This will ensure efficiency and reduce significant delays that may occur during the process.

4. Educational Resource: Students who are in process of learning and have no prior experience will be able to read this manual and execute the programs on their own without any hassle. This will help in their learning process and ability to tackle problems.

This manual covers the following section:

- System Requirement: Minimum hardware and software requirement to replicate the model working.

- Environment Setup: Gives a detailed instructions about how to set up the programming environment and includes the installations of the dependencies.

- Data Preparation: Data preprocessing steps discussed.

- Model Implementation: Architecture of the LSTM model. Snippets of the code provided for implementation purpose.
- Training the LSTM model: This sub section will explain the configurations which are suitable for model implementation.
- Evaluation and Testing: Methods used for the evaluation purpose.
- Hyperparameter Tuning: Optimization process to improve the model performance.

2 System Requirements

To successfully execute the programs and validate the obtained results, it becomes very crucial to correctly setup the appropriate environment and install the necessary software dependencies. In this section a comprehensive overview of the both the hardware and software requirements and the dependencies that must be installed in order to successfully will be discussed.

Hardware Requirements are as follows:

1. **CPU**: An Intel Core i5 (for windows user) or MacBook (with Apple silicon M1 or later or i7 Intel chip (for older variants)) is recommended to efficiently handle the computational task.
2. **RAM**: Minimum requirement is 8GB to ensure uninterrupted processing.
3. **Disk Space**: At least 10 GB free disk space is required to store the files that includes the dataset, the main code and the intermediate files that are created when the execution starts.
4. **GPU**: GPU is optional but can be used to increase the processing speed. An NVIDIA GPU along with CUDA can drastically reduce the training time of the model and increase the performance.

Software Requirements are as follows:

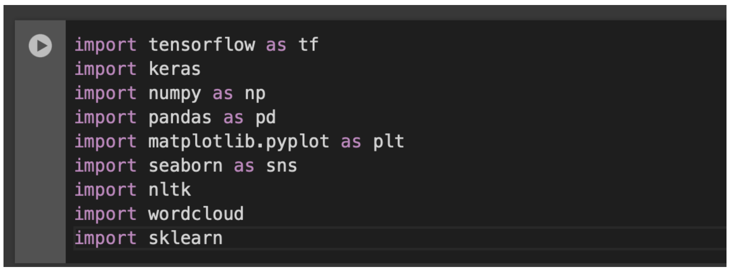
1. **OS (Operating System)**: Systems operation on Ubuntu 18.04 or later versions, or Windows 10 or later, or MacOS BigSur or later are compatible.
2. **Python**: Python version 3.7 or later is required to run the model. Python programming language is used for the research because of the wide variety of available libraries which reduces the need to write complex programs.

3 Environment Setup

For setting up the environment several python libraries are needed to be installed for the smooth flow of model development. Firstly, TensorFlow and Keras should be installed as these libraries provide the necessary tools that are required for implementation of the LSTM model. Next is numpy and pandas which are the fundamental libraries that are required for any machine learning project. For the visualization purpose such as graphs and plots matplotlib and seaborn are used. Wordcloud library is used for generating words clouds from the textual data. Lastly NLTK which is a fundamental library in any natural language processing project is installed which provides tools that will be helpful when working with the human language data. Broadly these are the major libraries that need to pre-installed before the development process to have a smooth workflow.

Google Colab is an open platform offered from Google which lets its user run python codes in their browser which any need for expensive hardware setup. It consists of many runtime can be configured according to the requirement and the type of applications it has been used for.

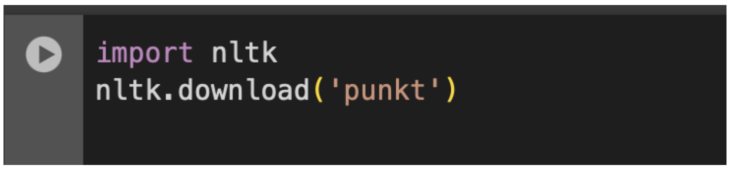
To verify that the libraries are installed correctly the code snippet shown in Fig. 1 should be executed.



```python
import tensorflow as tf
import keras
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
import wordcloud
import sklearn
```

Figure 1: Required Libraries for Development

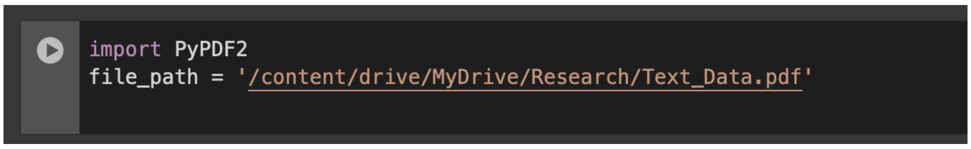
To install the NLTK data run the following code as shown in Fig.2



```python
import nltk
nltk.download('punkt')
```

Figure 2: Installing NLTK Data

By following all these steps the working environment is perfectly configured and ready for the research process. Another important step is adding the correct path of the dataset in the system. The following Fig. 3 shows where the file path needs to be changed according to where the user has the text file.



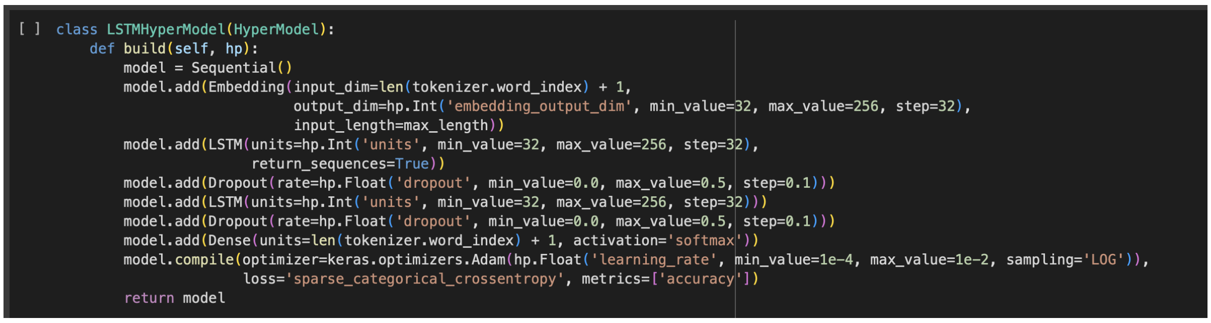
```python
import PyPDF2
file_path = '/content/drive/MyDrive/Research/Text_Data.pdf'
```

Figure 3: Path of file need to be changed according to users file location

4 LSTM Modelling and Training

This section will provide the instructions about how to implement the LSTM model. A complete overview of development starting from defining the model architecture, configurations of the training process will done.

Define the model as shown in the Fig.4

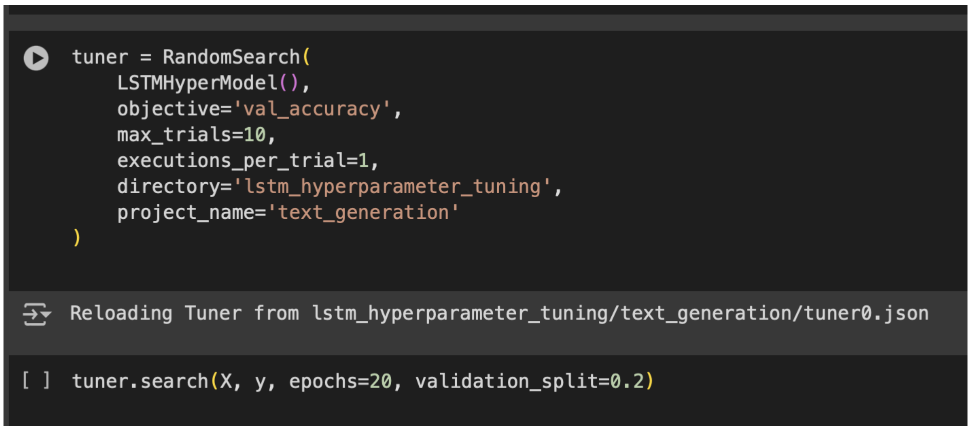


```
[ ] class LSTMHyperModel(HyperModel):
        def build(self, hp):
            model = Sequential()
            model.add(Embedding(input_dim=len(tokenizer.word_index) + 1,
                                output_dim=hp.Int('embedding_output_dim', min_value=32, max_value=256, step=32),
                                input_length=max_length))
            model.add(LSTM(units=hp.Int('units', min_value=32, max_value=256, step=32),
                           return_sequences=True))
            model.add(Dropout(rate=hp.Float('dropout', min_value=0.0, max_value=0.5, step=0.1)))
            model.add(LSTM(units=hp.Int('units', min_value=32, max_value=256, step=32)))
            model.add(Dropout(rate=hp.Float('dropout', min_value=0.0, max_value=0.5, step=0.1)))
            model.add(Dense(units=len(tokenizer.word_index) + 1, activation='softmax'))
            model.compile(optimizer=keras.optimizers.Adam(hp.Float('learning_rate', min_value=1e-4, max_value=1e-2, sampling='LOG')),
                          loss='sparse_categorical_crossentropy', metrics=['accuracy'])
            return model
```

Figure 4: Defining the LSTM Model

Any changes required as per requirement can be done here. This includes adding more layers so as to increase the performance of the model. The different layers include the dense layer, LSTM layer and the dropout layer.

The next step is the using the Keras Tuner's Random Search to find the optimal value of hyperparameter. The following Fig.5 shows the configurations for the tuning process. The number of epochs can be changed as per choice. But it is important to note increasing the epochs lead to increased processing time.



```
tuner = RandomSearch(
    LSTMHyperModel(),
    objective='val_accuracy',
    max_trials=10,
    executions_per_trial=1,
    directory='lstm_hyperparameter_tuning',
    project_name='text_generation'
)
```

```
Reloading Tuner from lstm_hyperparameter_tuning/text_generation/tuner0.json
```

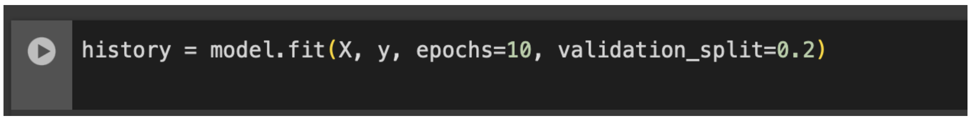
```
[ ] tuner.search(X, y, epochs=20, validation_split=0.2)
```

Figure 5: Keras Random Search Tuning

Here,

- Epochs: 20 (It is the number of time the algorithm will work)
- Validation Split: 0.2 (From original dataset 80% for training purpose and 20% for validation purpose)

Fitting the model on the dataset with the optimized parameter as show in Fig.6.

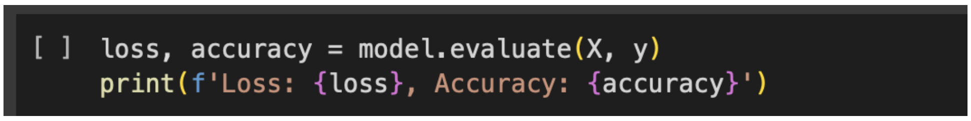


```
history = model.fit(X, y, epochs=10, validation_split=0.2)
```

Figure 6: Model Fitting

5 Evaluation Process

For evaluating the performance of the model. The following code snippet in Fig. 7 is executed it calculates and prints the accuracy and the loss values respectively.



```
[ ]  loss, accuracy = model.evaluate(X, y)
     print(f'Loss: {loss}, Accuracy: {accuracy}')
```

Figure 7: Evaluation Process