Configuration Manual

MSc Research Project
Data Analytics

Sharon George
Student ID: 21245240

School of Computing
National College of Ireland

Supervisor: Mayank Jain

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Sharon George
Student ID:	21245240
Programme:	Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Mayank Jain
Submission Due Date:	14/12/2023
Project Title:	Configuration Manual
Word Count:	602
Page Count:	6

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Sharon George
Date:	31st January 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.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

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

Configuration Manual

Sharon George
21245240

1 Introduction

This config manual has all the information needed to use the project. It tells you the least and best things you need to do in the thesis. It connects the project and the thesis, explaining the software and hardware you need and showing the important parts with code bits. It gives steps for getting data, running the project and showing the main results. All of this is focused on an AI-based Product Recommendation System that looks at YouTube comments.

2 Hardware Configuration

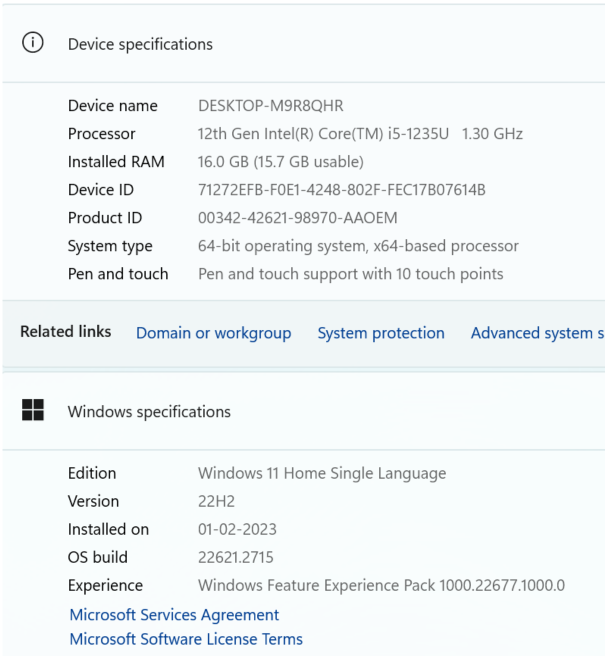

Figure 1: Hardware specifications

Figure 1 above outlines the required software specifications for implementing the project on the local machine.

3 Software Configuration

The following software must be installed fpr the project to run successfully:
1. Anaconda Navigator for Windows

2. Jupyter Notebook
3. Python
4. YouTube

4 Data Selection

The dataset for the project is available from an open data source called Kaggle. The URL is https://www.kaggle.com/datasets/kazanova/sentiment140. It has around 1.6 lakh tweets from a twitter api.

5 Code files of the project

- 1_GPT2_Feature_Extraction.ipynb - The aim of this code is to collection the data, data cleaning, data preprocessing and create a new csv file with the extracted features.The below Figure 2 displays all the libraries used by this file.Foundation (year of the documentation)

```
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
pd.set_option("display.max_columns", None)
import matplotlib.pyplot as plt
%matplotlib inline
from tqdm import tqdm
from lib_file import lib_path
import re
import torch
from transformers import GPT2Model, GPT2Tokenizer
import re
import contractions
from langdetect import detect
import wordcloud
```

Figure 2: Imported Libraries for 1_GPT2_Feature_Extraction.ipynb file

- 2_ModelTraining.ipynb - A CLSTM model is created on the extracted features and saved.Also three common evaluation metrics used for classification tasks are accuracy score, classification report, and confusion matrix.The below Figure 3 displays all the libraries used by this file.

```python
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
pd.set_option("display.max_columns", None)
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.utils import resample
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
from lib_file import lib_path
```

Figure 3: Imported Libraries for 2_ModelTraining.ipynb file

- 3_Inference.ipynb - The inference code begins with the user providing a YouTube link. Once the user provides a YouTube link, the system proceeds to collect all comments associated with the video. Following this, the collected data undergoes cleaning and preparation before being fed into the model, which ultimately produces the result.The below Figure 4 displays all the libraries used by this file.

```python
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
pd.set_option("display.max_columns", None)
import numpy as np
from tensorflow.keras.models import load_model
import re
from bs4 import BeautifulSoup
from tqdm import tqdm
import torch
from transformers import GPT2Model, GPT2Tokenizer
from transformers import pipeline
from urllib.parse import urlparse
from lib_file import lib_path
import googleapiclient
from langdetect import detect
from IPython import display
```

Figure 4: Imported Libraries for 3_Inference.ipynb file

6 Code snippets

6.1 1_GPT2_Feature_Extraction file

The below code snippet in Figure 5 depicts the text_cleaning function which is used for preprocessing textual data.

```python
def remove_emoticons(text):
    emoticons_to_remove = [":)", ":(", ":D", ";D", ":*", ":'(", ":/", "O:)", ":P", ":O", "&)", "^_^", ">:O", ":3", ">:(", "8
    for emoticon in emoticons_to_remove:
        text = text.replace(emoticon, '')
    return text
```

```python
def text_cleaning(text):
    text = remove_emoticons(text)
    emoji_pattern = re.compile("["
                               u"\U0001F600-\U0001F64F"  # Emojis
                               u"\U0001F300-\U0001F5FF"  # Symbols & pictographs
                               u"\U0001F680-\U0001F6FF"  # Transport & map symbols
                               u"\U0001F700-\U0001F77F"  # Alphabetic presentation forms
                               u"\U0001F780-\U0001F7FF"  # Geometric shapes
                               u"\U0001F800-\U0001F8FF"  # Miscellaneous symbols"
                               u"\U0001F900-\U0001F9FF"  # Supplemental symbols & pictographs
                               u"\U0001FA00-\U0001FA6F"  # Extended-A
                               u"\U0001FA70-\U0001FAFF"  # Extended-B
                               u"\U0001F004-\U0001F0CF"  # Mahjong tiles
                               u"\U0001F170-\U0001F251"  # Enclosed characters
                               u"\U00020000-\U0002F73F"  # Chinese, Japanese, and Korean characters
                               u"\U000E0000-\U000E007F"  # Tags
                               "]+", flags=re.UNICODE)
    text = re.sub(emoji_pattern, '', text)
    text = re.sub(r"http\S+|www\S+|https\S+", '', text, flags=re.MULTILINE)
    text = re.sub(r'(@\w+|#\w+)', '', text)
    text = contractions.fix(text)
    text = re.sub(r"^[^a-zA-Z0-9]+", '', text)
    try:
        lang = detect(text)
        if lang == 'en':
            return text.lower()
        else:
            return 'empty'
    except:
        return 'empty'
```

```python
sentences = []
for text in tqdm(df['text'].values):
    sentences.append(text_cleaning(text))
```

```
100%|██████████| 40000/40000 [07:05<00:00, 93.95it/s]
```

Figure 5: Text cleaning function

This code shown in Figure 6 essentially processes a list of cleaned text data using a pre-trained GPT-2 model, extracting features from the hidden states of the model for each text.

```python
# Load a pre-trained GPT-2 model
gpt2_model = GPT2Model.from_pretrained('gpt2')
tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
```

Figure 6: Applying GPT-2 to the cleaned text

The code snippet below in Figure 7 is the result after applyint the GPT-2 Model

	column_1	column_2	column_3	column_4	column_5	column_6	column_7	column_8	column_9	column_10	column_11	column_12	column_13	colu
0	0.195140	-0.011653	-0.742695	0.362702	0.032608	-0.239953	0.929984	-0.001241	-0.180009	0.067178	0.127039	-0.118616	-0.270507	0.
1	0.241021	-0.084276	-0.397343	0.001570	-0.031986	-0.352723	3.009794	-0.288340	0.055039	-0.030605	0.260705	-0.297391	-0.267551	-0.(
2	-0.004207	-0.226263	-0.467169	0.172692	0.268597	-0.652958	2.138371	0.163865	0.007916	0.087473	0.012659	-0.093617	-0.198733	-0.(
3	0.253612	-0.147663	0.152705	-0.101110	0.050246	-0.108569	6.792430	0.071317	-0.057158	-0.450624	-0.211453	0.035497	-0.177794	0.(
4	0.065767	-0.042519	-0.466257	-0.115363	-0.015952	-0.158690	1.311224	-0.226336	-0.327245	-0.185646	-0.114596	-0.080248	-0.227208	0.(

```python
new_df.to_csv("extracted_features.csv", index=False)
```

Figure 7: Extracted features

6.2 2_ModelTraining file

The code snippet below is the model architecture to build the CLSTM Model

Algorithm : Convolutional Long Short-Term Memory

```
]:  from tensorflow.keras import Sequential
    from tensorflow.keras.layers import Conv1D,MaxPool1D,Flatten,Dense,Dropout,LSTM,BatchNormalization
```

```
]:  clstm_model = Sequential()
    clstm_model.add(Conv1D(filters=32, kernel_size=3, activation='relu', padding='same', input_shape=(x_
    clstm_model.add(Conv1D(filters=32, kernel_size=3, activation='relu', padding='same'))
    clstm_model.add(MaxPool1D())
    clstm_model.add(BatchNormalization())
    clstm_model.add(Conv1D(filters=64, kernel_size=3, activation='relu', padding='same'))
    clstm_model.add(Conv1D(filters=64, kernel_size=3, activation='relu', padding='same'))
    clstm_model.add(MaxPool1D())
    clstm_model.add(BatchNormalization())
    clstm_model.add(Conv1D(filters=128, kernel_size=3, activation='relu', padding='same'))
    clstm_model.add(Conv1D(filters=128, kernel_size=3, activation='relu', padding='same'))
    clstm_model.add(MaxPool1D())
    clstm_model.add(BatchNormalization())
    clstm_model.add(Conv1D(filters=256, kernel_size=3, activation='relu', padding='same'))
    clstm_model.add(Conv1D(filters=256, kernel_size=3, activation='relu', padding='same'))
    clstm_model.add(MaxPool1D())
    clstm_model.add(BatchNormalization())
    clstm_model.add(Conv1D(filters=512, kernel_size=3, activation='relu', padding='same'))
    clstm_model.add(Conv1D(filters=512, kernel_size=3, activation='relu', padding='same'))
    clstm_model.add(MaxPool1D())
    clstm_model.add(BatchNormalization())
    clstm_model.add(Conv1D(filters=1024, kernel_size=3, activation='relu', padding='same'))
    clstm_model.add(Conv1D(filters=1024, kernel_size=3, activation='relu', padding='same'))
    clstm_model.add(MaxPool1D())
    clstm_model.add(BatchNormalization())
    clstm_model.add(Conv1D(filters=2048, kernel_size=3, activation='relu', padding='same'))
    clstm_model.add(Conv1D(filters=2048, kernel_size=3, activation='relu', padding='same'))
    clstm_model.add(MaxPool1D())
    clstm_model.add(BatchNormalization())
    clstm_model.add(LSTM(units=256, return_sequences=True))
    clstm_model.add(LSTM(units=256, return_sequences=True))
    clstm_model.add(Flatten())
    clstm_model.add(Dense(512, activation='relu'))
    clstm_model.add(Dropout(0.4))
    clstm_model.add(Dense(2, activation='sigmoid'))
    clstm_model.compile(loss='binary_crossentropy', optimizer="adam", metrics=['accuracy'])
```

Figure 8: Model architecture - CLSTM

The Figure 9 below shows the Accuracy Score and Classification report of the model.

Accuracy Score

```
clstm_model_accuracy=accuracy_score(y_true=y_true,y_pred=pred)
print("ConvolutionalLongShortTermMemory model's Validation accuracy is {

ConvolutionalLongShortTermMemory model's Validation accuracy is 89.05%
```

Classification Report

```
print(classification_report(y_true=y_true,y_pred=pred, target_names=clas
```

	precision	recall	f1-score	support
negative	0.84	0.97	0.90	2000
positive	0.96	0.81	0.88	2000
accuracy			0.89	4000
macro avg	0.90	0.89	0.89	4000
weighted avg	0.90	0.89	0.89	4000

Figure 9: Accuracy Score and Classification report

6.3 3 Inference.ipynb file

The Figure 10 below depicts the code snippet to enter the youTube link and verify it

```
video_url = input("Paste youtube's appropriate link...\n")

Paste youtube's appropriate link...
https://youtu.be/t-URmQbR5FY?si=YVG98lQErypub8v7

video_id = get_video_id_from_url(video_url)
if video_id is not None:
    print(f"The video ID, {video_id}, has been detected.")
else:
    print("The video ID could not be retrieved from the URL. Please try a different URL.")

The video ID, t-URmQbR5FY, has been detected.
```

Figure 10: Entering link and verifying youtube link

The below Figure 11 show show to pick the youTube link that would be added to the above code snippet in Figure 10

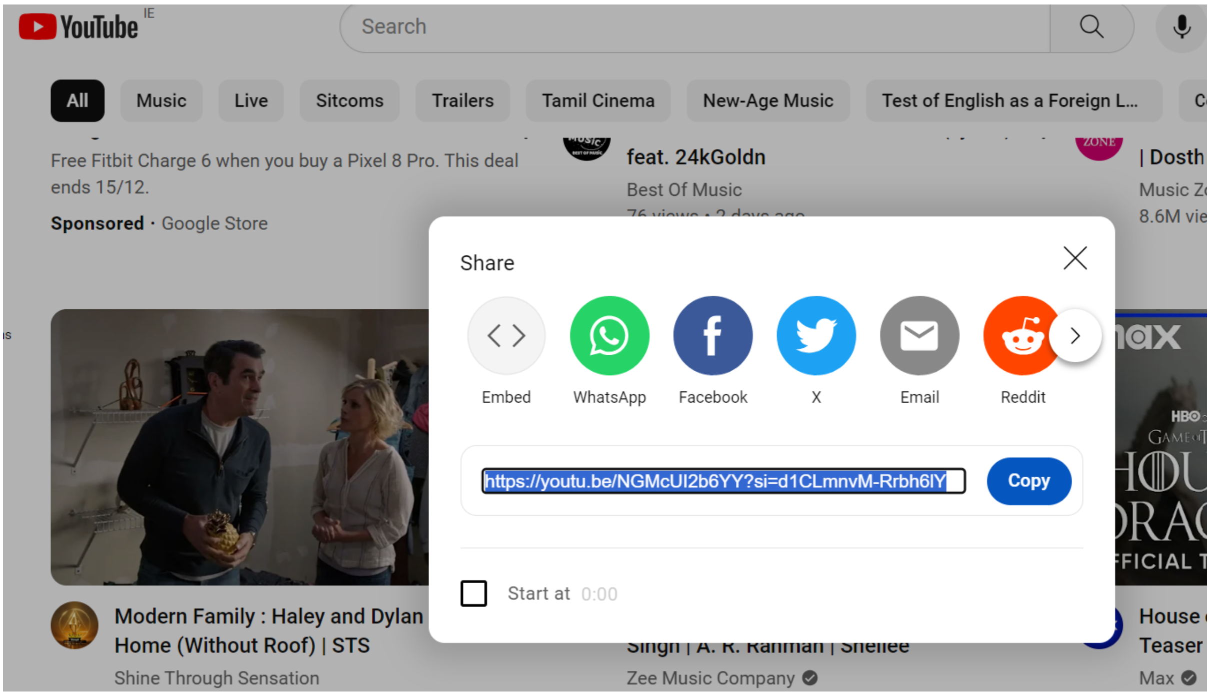

Figure 11: Pick a link from YouTube

References

Foundation, P. S. (year of the documentation). Python documentation. Accessed on: 31st January 2024.
URL: *https://docs.python.org/3/library/index.html*