Emotion Analysis of Pages in a Book to Play Background Music

MSc Research Project
Data Analytics

Joshma Joseph
Student ID: X20206968

School of Computing
National College of Ireland

Supervisor: Hicham Rifai

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Joshma Joseph
Student ID:	X20206968
Programme:	Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Hicham Rifai
Submission Due Date:	15/08/2022
Project Title:	Emotion Analysis of Pages in a Book to Play Background Music
Word Count:	XXX
Page Count:	8

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:	
Date:	11th August 2022

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):	

Emotion Analysis of Pages in a Book to Play Background Music

Joshma Joseph
X20206968

1 Hardware specification:

- The accompanying figure shows the machine's specifications, which were utilised in this study. The system is a windows 10 system with 8 GB installed RAM and 64-bit Operating system. The processor of the machine is 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz.

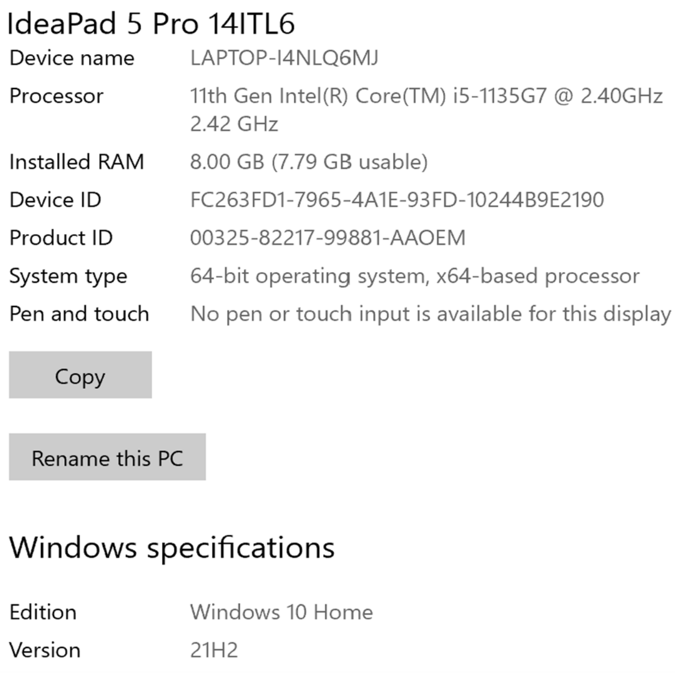

Figure 1: Hardware specification

2 Software Specification:

- For accessing the Jupyter environment in this research, an Anaconda software installation is done on the computer. [1]

[1] `https://docs.anaconda.com/`

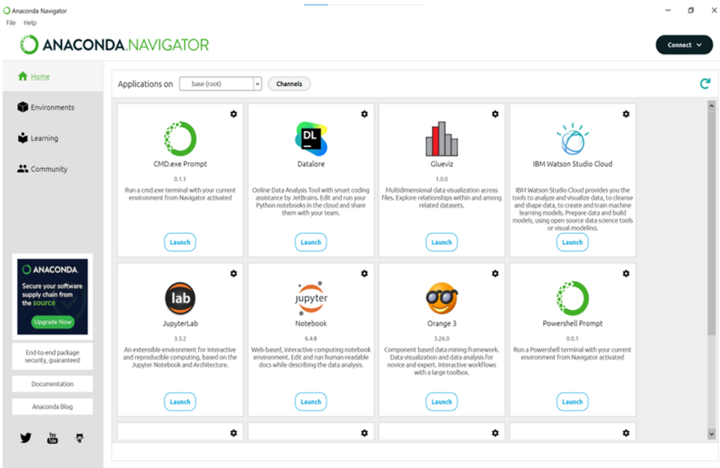

Figure 2: Software specification

- As shown in the below figure 3 the Python 3 option from new can be selected to start a Jupyter notebook. Three distinct notebooks are made in this study for dataset merging, Nave Bayes model, and RNN model.

Figure 3: Jupyter notebook

- For the RNN Model, Google Collaboratory Lab will be used along with TensorFlow for better and faster performance.[2]

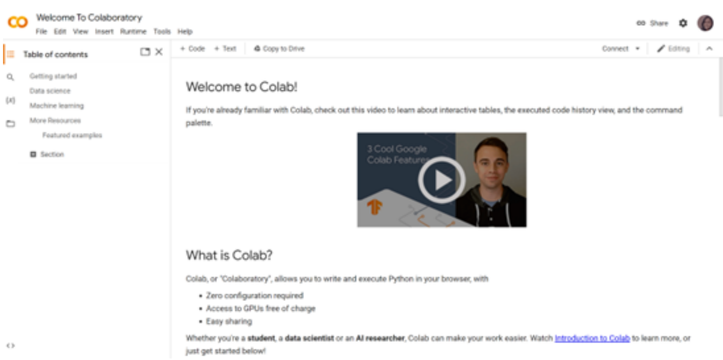

Figure 4: Google Colab

[2]https://colab.research.google.com/

3 Loading required packages:

- Here, the required libraries are loaded to carry out the data preparation for the text analysis.

Load packages

```
In [1]: import pandas as pd
        import numpy as np
```

Load data visualization packages

```
In [2]: import matplotlib.pyplot as plt
        import seaborn as sns
```

Text cleaning packages

```
In [3]: #!pip install neattext
        import neattext.functions as nfs
```

Figure 5: Basic Libraries loading

- Similarly, even for the specific models used for the analysis, the required libraries and packages need to be loaded as shown in the below screenshots.

```
33]: from sklearn.naive_bayes import MultinomialNB

     #Vectorizer
     from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

     #Metrics
     from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, plot_confusion_matrix
```

Figure 6: Libraries for Naive Bayes

- We will use the Keras Library for the RNN model.[3]

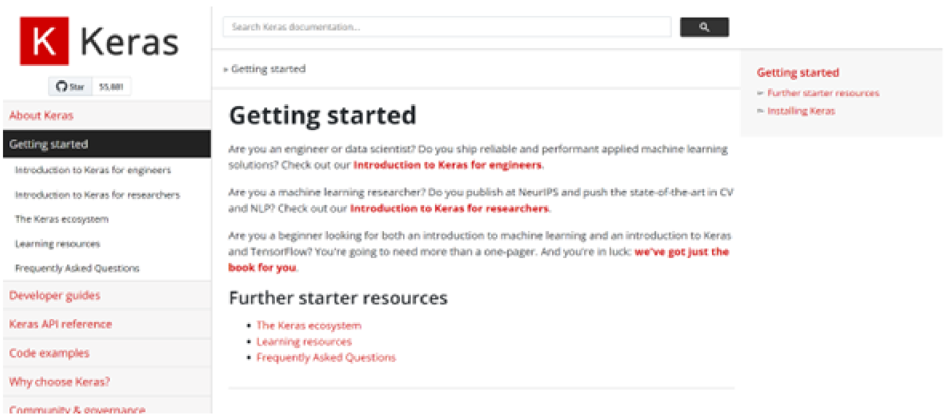

Figure 7: Keras Documentation

- The below snippet shows the required packages for RNN model

[3] `https://keras.io/`

```
import re
import nltk
import numpy as np
import pandas as pd

from nltk.stem import PorterStemmer
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

import tensorflow as tf
import keras.backend as K
from tensorflow import keras
from tensorflow.keras.preprocessing.text import text_to_word_sequence
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.text import one_hot
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras import Sequential
from keras.layers import Dense, SimpleRNN, Embedding, Flatten, Dropout
```

Figure 8: Libraries for RNN model

4 Data Preparation:

- Initially, the data needs to be cleaned and unwanted columns need to be removed as shown in the snippet below.

```
In [30]: frndsemo18 = frndsemo18.drop(frndsemo18.columns[[0]], axis=1)
         frndsemo18 = frndsemo18.drop(frndsemo18.columns[[0]], axis=1)
         frndsemo18 = frndsemo18.drop(frndsemo18.columns[[2]], axis=1)
         frndsemo18.rename(columns = {'utterance':'Text', 'emotion': 'Emotion'}, inplace = True)
         frndsemo18.tail()
```

Figure 9: Dropping unwanted columns

- Since we are using a combination of two data-sets we need to merge the data. For this purpose, we created a database in MongoDB as seen in the figure below.

```
In [34]: from pymongo import MongoClient as mc

In [35]: client = mc()
         print(client)

         MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True)

In [36]: client = mc('localhost', 27017)

In [37]: db = client.frdata
         print(db)

         Database(MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True), 'frdata')

In [38]: friends19 = db.friends19
```

Figure 10: MongoDB for data storing

- The next step is the pre-processing of data which falls under Natural language processing, in this step we need to tokenize the data and remove stop-words and punctuations. Along with this process, we will perform stemming and lemmatization.

```
In [15]: df['Clean_text'] = df['Text'].apply(nfx.remove_stopwords)

In [16]: df['Clean_text'] = df['Clean_text'].apply(nfx.remove_userhandles)

In [17]: df['Clean_text'] = df['Clean_text'].apply(nfx.remove_punctuations)

In [18]: df[['Text', 'Clean_text']]
```

Figure 11: NLP pre-processing

- In the next phase we will extract the most common keywords identified for each emotion.

• Extract most common words per class of emotion

```
In [19]: from collections import Counter

In [20]: def extract_keywords(text, num=50):
             tokens = [tok for tok in text.split()]
             most_common_tokens = Counter(tokens).most_common(num)
             return dict(most_common_tokens)
```

Figure 12: Keyword Extraction

- For the next phase, we will perform the Transformation of the data. In this step, we will perform the transformation of data based on the model being used. For Naïve Bayes, we will use countvectorizer and for the RNN model we will use one-hot encoding along with padding.

```
In [37]: #Vectorizer
         cv = CountVectorizer()
         X= cv.fit_transform(Xfeatures)

In [38]: #Get features by name
         cv.get_feature_names_out()
```

Figure 13: Data transformation for Naive Bayes

```
      corpus.append(text)

    # one-hot encode each sentence; converts each sentence to a vector
    one_hot_word = [one_hot(input_text=sentence, n=vocab_size) for sentence in corpus]
    # makes all vectors the same length
    pad = pad_sequences(sequences=one_hot_word,maxlen=max_len,padding='pre')

    return pad
```

Figure 14: Data transformation for RNN

5 Model Building:

- Once the pre-processing and data transformation is completed we move on to the model-building process. We will need to split the data into train and test data to train and evaluate our models.

```
In [39]: #Split dataset
         X_train, X_test, y_train, y_test = train_test_split(X, ylabels, test_size =0.3, random_state = 42)
```

Figure 15: Train and test data split

- Now that we have data to train, we will first train the Multinomial Naïve Bayes model.

Build Model

```
In [40]: nv_model = MultinomialNB()
         nv_model.fit(X_train, y_train)

Out[40]: MultinomialNB()
```

Figure 16: Naive Bayes Model

- Next is the RNN model, as shown in the snippet there are different layers in this model the embedded layer, the RNN layer, the dense layer with sigmoid activation and the dense layer with softmax activation which will provide us with the output in one of the classes of the emotion. Lastly, the model is compiled with Adam optimizer.

```
[ ] def build_model():
      model = Sequential()
      model.add(Embedding(input_dim=vocab_size, input_length=max_len, output_dim=150))
      model.add(Dropout(0.2))
      model.add(SimpleRNN(128))
      model.add(Dropout(0.2))
      model.add(Dense(64,activation='sigmoid'))
      model.add(Dropout(0.2))
      model.add(Dense(8, activation="softmax"))

      model.compile(optimizer='Adam', loss=tf.keras.losses.CategoricalCrossentropy(), metrics=['accuracy',
                                                                                       tf.keras.metrics.Precision(),
                                                                                       tf.keras.metrics.Recall()])

      return model
```

Figure 17: RNN Model

6 Evaluation:

- To Evaluate the model, we will use the metrics such as recall, precision, f1-score and accuracy.

```
In [49]: #Classification
         print(classification_report(y_test, y_pred_for_nv))

                   precision    recall  f1-score   support
```

Figure 18: Evaluation metrics

7 Model Deployment:

- For the analysis, we will use a sample book, separate it according to pages-

```
In [50]: import PyPDF2

In [85]: file = open("C:/Users/rajes/Desktop/College submissions/Research/When-Siggy-FKB.pdf", "rb")
         reader = PyPDF2.PdfFileReader(file, strict=False)
         a = int(reader.numPages)
```

Figure 19: Reading the PDF documents

- and pre-process it according to the input requirements of the models. Once the model has been applied to the pages the accurate emotion will be identified.

```
In [47]: def predict_emotion(sample_text, model):
             myvect = cv.transform(sample_text).toarray()
             prediction = model.predict(myvect)
             pred_proba = model.predict_proba(myvect)
             pred_percent_for_all = dict(zip(model.classes_,pred_proba[0]))
             print("Prediction: {}, Prediction Score: {}".format(prediction[0], np.max(pred_proba)))
             return [pred_percent_for_all, prediction[0]]
```

Figure 20: Naive Bayes processing of book

```
for i in range(21,40):
    page = reader.getPage(i)
    txt = page.extractText()
    #print(txt)
    txt = text_to_word_sequence(txt)
    stemmer = PorterStemmer()
    txt = [stemmer.stem(word) for word in txt if word not in stopwords]
    txt = " ".join(txt)
    corpus = []
    corpus.append(txt)
    one_hot_word = [one_hot(input_text=sentence, n=vocab_size) for sentence in corpus]
    pad_sample = pad_sequences(sequences=one_hot_word,maxlen=max_len,padding='pre')
    predictions = model.predict(pad_sample)
    emotions =  {1:'anger', 2:'sadness', 3:'surprise', 4:'joy', 5:'fear', 6:'neutral', 7:'disgust', 8:'shame'}
    print(emotions[np.argmax(predictions[0])+1])
    print(predictions)
```

Figure 21: RNN preprocessing of book

- Once the emotions have been identified the accurate emotion of the music from the list of manually curated music libraries will be played. For playing the music the playsound library will be utilized.

```python
from playsound import playsound
import os
```

```python
    if p[1] == 'joy':
        playsound('Joy/001.mp3')
        print(p[1])
    elif p[1] == 'fear':
        playsound('Fear/091.mp3')
        print(p[1])
    elif p[1] == 'anger':
        playsound('Anger/121.mp3')
        print(p[1])
    elif p[1] == 'disgust':
        playsound('Disgust/331.mp3')
        print(p[1])
    elif p[1] == 'shame':
        playsound('Shame/031.mp3')
        print(p[1])
    elif p[1] == 'sadness':
        playsound('Sadness/031.mp3')
        print(p[1])
    elif p[1] == 'surprise':
        playsound('Surprise/151.mp3')
        print(p[1])
    elif p[1] == 'neutral':
        playsound('Neutral/046.mp3')
        print(p[1])
    else:
        print("Unknown emotion")
```

Figure 22: Playing Background Music