National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Avis Massey
Student ID: x21199752

School of Computing
National College of Ireland

Supervisor: Aaloka Anant

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Avis Massey
Student ID:	x21199752
Programme:	Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Aaloka Anant
Submission Due Date:	14/12/2023
Project Title:	Configuration Manual
Word Count:	404
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:	14th December 2023

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

Avis Massey
x21199752

1 Introduction

This configuration manual describes the hardware and the software requirements whcih is essential to build the counterfeit review identifier using Neural Network - LSTM & CNN, Classifier - SVM and Transfer learning models BERT, RoBERTa, DistilBERT and ALBERT.

2 Hardware and Software Requirement

To train the LSTM, CNN and SVM models along with the transfer learning models such as BERT, RoBERTa, DistilBERT, and ALBERT, Google Collaboratory cloud is used which is a cloud machine since we have a vast dataset to train. For whcih you can find below the specification of the host device.

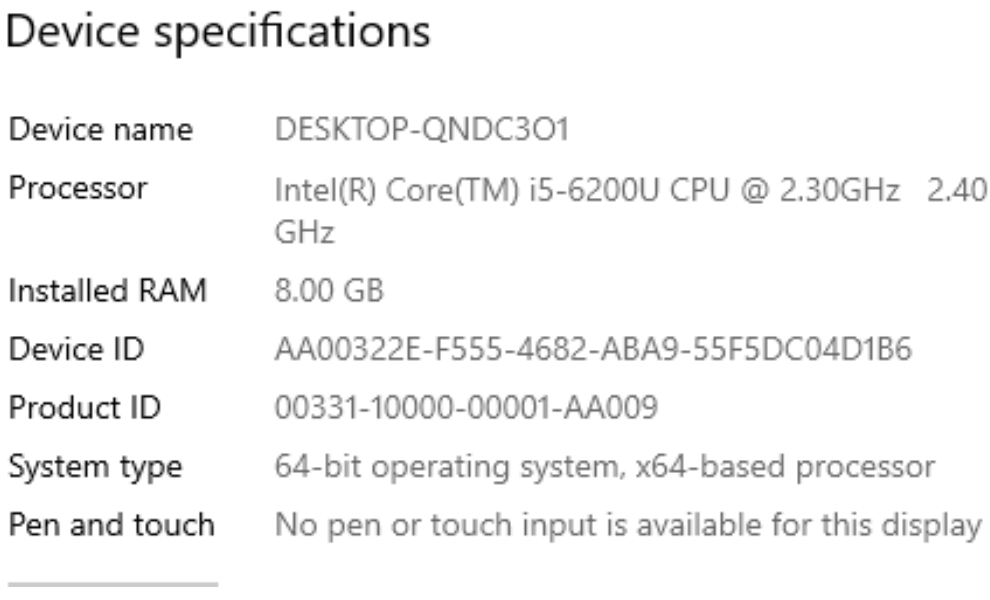

Figure 1: Device Specifications

The following table 1 provides information regarding the programming languages, computaional unit, and libraries employed throughout the project.

Specification	Value
IDE	Google Colab
Computation	GPU
Number of GPU	1
Programming language	Python
Modeling library	SimpleTransformer, HuggingFace Transformer, Sklearn, Pandas, Numpy, Matplotlib, Seaborn, Wandb, Keras, PyTorch, tqdm, SciPy

Table 1: System Specifications for the Project

3 Dataset

The dataset whcih we have implied in the project is in the CSV format. Originally these reviews are from amazon and the fake are AI generated using GPT. There are namely 5 column, Category, Label, Rating, Reviews and Fake_Review_flag. The Reviews and Fake_Review_flag are the most important columns. Salminen et al. (2022)

Category	Rating	Label	Reviews
Home_and_Kitchen_5	5	CG	Love this! Well made, sturdy, and very comfortable. I love it!Very pretty
Home_and_Kitchen_5	5	CG	love it, a great upgrade from the original. I've had mine for a couple of years
Home_and_Kitchen_5	5	CG	This pillow saved my back. I love the look and feel of this pillow.
Home_and_Kitchen_5	1	CG	Missing information on how to use it, but it is a great product for the price! I
Home_and_Kitchen_5	5	CG	Very nice set. Good quality. We have had the set for two months now and have not been
Home_and_Kitchen_5	3	CG	I WANTED DIFFERENT FLAVORS BUT THEY ARE NOT.
Home_and_Kitchen_5	5	CG	They are the perfect touch for me and the only thing I wish they had a little more space.
Home_and_Kitchen_5	3	CG	These done fit well and look great. I love the smoothness of the edges and the extra
Home_and_Kitchen_5	5	CG	Great big numbers & easy to read, the only thing I didn't like is the size of the
Home_and_Kitchen_5	5	CG	My son loves this comforter and it is very well made. We also have a baby
Home_and_Kitchen_5	5	CG	As advertised. 5th one I've had. The only problem is that it's not really a

Figure 2: Dataset overview

4 Implementation of the project

In the Google Colab we chage the runtime to make use of the GPU provided by the google whcih is the T4 GPU showing exceptional computation capabilities.

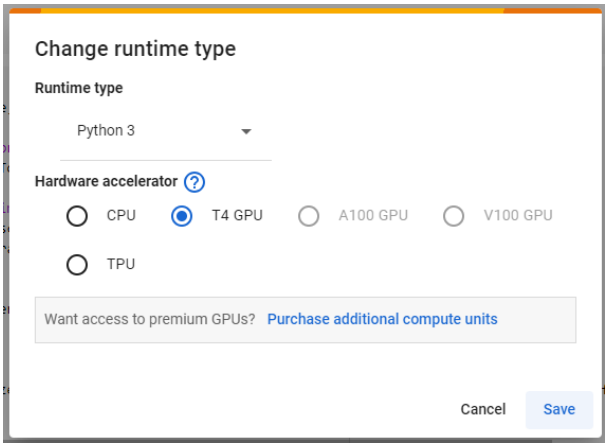

Figure 3: change run time to GPU

4.1 Installation of Libraries

The 4 shows installtion of required libraries such as numpy pandas sentencepiece scipy torch transformers.

```
[1]  !pip install numpy pandas sentencepiece scipy torch transformers

     Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packag
     Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packa
```

Figure 4: Dataset overview

Once installed the imported libraries are shown below in 5

```
[ ]
from keras.layers import Conv1D, Dense, Embedding, GlobalMaxPooling1D, LSTM
from keras.models import Sequential
from keras.preprocessing.sequence import pad_sequences
from keras.preprocessing.text import Tokenizer

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

from torch.utils.data import DataLoader, RandomSampler, SequentialSampler, TensorDataset

from tqdm import tqdm

from transformers import AlbertTokenizer, AlbertForSequenceClassification, BertTokenizer,

from scipy.stats import zscore
```

Figure 5: Import Libraries

4.2 Importing the dataset

We import the data into the dataframe df using the pandas library.

```
# # Load the dataset
df = pd.read_csv("./fake_reviews_dataset.csv")


data = df.sample(frac=0.5)

# Extracting Reviews and Labels
X = data["Reviews"]
y = data["Fake_review_flag"]

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Figure 6: Store the dataset in dataframe.

4.3 Tokenize for LSTM and CNN

We tokenize the dataset for LSTM and CNN models.

```
[ ]  # Tokenizer for LSTM and CNN
     max_features = 10000  # Number of words to consider as features
     tokenizer = Tokenizer(num_words=max_features)
     tokenizer.fit_on_texts(X_train)
     X_train_seq = tokenizer.texts_to_sequences(X_train)
     X_test_seq = tokenizer.texts_to_sequences(X_test)

     # Padding sequences for LSTM and CNN
     maxlen = 100  # Cuts off reviews after 100 words
     X_train_pad = pad_sequences(X_train_seq, maxlen=maxlen)
     X_test_pad = pad_sequences(X_test_seq, maxlen=maxlen)

     # TF-IDF for SVM
     vectorizer = TfidfVectorizer(max_features=max_features)
     X_train_tfidf = vectorizer.fit_transform(X_train)
     X_test_tfidf = vectorizer.transform(X_test)
```

Figure 7: Tokenization for the models

4.4 Building the LSTM model and CNN model

The LSTM and CNN models are built and executed, for whcih the outputs are given below.

```
[ ]  # Building the LSTM model
     lstm_model = Sequential()
     lstm_model.add(Embedding(max_features, 128, input_length=maxlen))
     lstm_model.add(LSTM(64, dropout=0.2, recurrent_dropout=0.2))
     lstm_model.add(Dense(1, activation="sigmoid"))

     # Compile the model
     lstm_model.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy"])

     # Train the model
     lstm_model.fit(X_train_pad, y_train, batch_size=batch_size, epochs=epochs, validation_data=(X_test_pad, y_test))

     WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic
     Epoch 1/3
     4489/4489 [==============================] - 2479s 550ms/step - loss: 0.3162 - accuracy: 0.8970 - val_loss: 0.3092
     Epoch 2/3
     4489/4489 [==============================] - 2415s 538ms/step - loss: 0.2942 - accuracy: 0.8975 - val_loss: 0.3071
     Epoch 3/3
     4489/4489 [==============================] - 2471s 551ms/step - loss: 0.2764 - accuracy: 0.8992 - val_loss: 0.3194
     <keras.src.callbacks.History at 0x7c8e88660eb0>
```

Figure 8: Building the LSTM Model

4.5 Predicting the LSTM and CNN models

The CNN models outperfomr the LSTM model after evaluation.

4.6 SVM building and Evaluation

SVM is built and executed, the evaluation states that it can be an alternative.

```
[ ] # Building the CNN model
    cnn_model = Sequential()
    cnn_model.add(Embedding(max_features, 128, input_length=maxlen))
    cnn_model.add(Conv1D(64, 5, activation="relu"))
    cnn_model.add(GlobalMaxPooling1D())
    cnn_model.add(Dense(1, activation="sigmoid"))

    # Compile the model
    cnn_model.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy"])

    # Train the model
    cnn_model.fit(X_train_pad, y_train, batch_size=batch_size, epochs=epochs, validation_data=(X_test_pad, y_test))

    Epoch 1/3
    4489/4489 [==============================] - 70s 14ms/step - loss: 0.3115 - accuracy: 0.8959 - val_loss: 0.3089 - val_accuracy: 0.8971
    Epoch 2/3
    4489/4489 [==============================] - 35s 8ms/step - loss: 0.2809 - accuracy: 0.8977 - val_loss: 0.3172 - val_accuracy: 0.8936
    Epoch 3/3
    4489/4489 [==============================] - 31s 7ms/step - loss: 0.2293 - accuracy: 0.9100 - val_loss: 0.3566 - val_accuracy: 0.8878
    <keras.src.callbacks.History at 0x7c8e8705ae30>
```

Figure 9: Building the CNN model

```
[ ] # Predicting with LSTM model
    y_pred_lstm = lstm_model.predict(X_test_pad)
    y_pred_lstm_classes = (y_pred_lstm > 0.5).astype("int32")

    # Predicting with CNN model
    y_pred_cnn = cnn_model.predict(X_test_pad)
    y_pred_cnn_classes = (y_pred_cnn > 0.5).astype("int32")

    # Evaluation for LSTM model
    print("LSTM Model Evaluation")
    print(classification_report(y_test, y_pred_lstm_classes))

    # Evaluation for CNN model
    print("CNN Model Evaluation")
    print(classification_report(y_test, y_pred_cnn_classes))

    253/253 [==============================] - 8s 31ms/step
    253/253 [==============================] - 1s 2ms/step
    LSTM Model Evaluation
```

Figure 10: Predicting the models

5 Transfer Models

The transformer model hyperparameter are shown in 15 which is used to fine tune the model.

6 Results

The BERT function model is executed for which the reference is given below.

The evaluation of the model is given in 16

Similarly, we build all the other transformer models.

References

Salminen, J., Kandpal, C., Kamel, A., Jung, S. G. and Jansen, B. (2022). Creating and detecting fake reviews of online products, *Journal of Retailing and Consumer Services* **64**.

```
253/253 [==============================] - 8s 31ms/step
253/253 [==============================] - 1s 2ms/step
LSTM Model Evaluation
              precision    recall  f1-score   support

           0       0.95      0.94      0.94      4118
           1       0.94      0.95      0.94      3969

    accuracy                           0.94      8087
   macro avg       0.94      0.94      0.94      8087
weighted avg       0.94      0.94      0.94      8087

CNN Model Evaluation
              precision    recall  f1-score   support

           0       0.94      0.93      0.93      4118
           1       0.93      0.94      0.93      3969

    accuracy                           0.93      8087
   macro avg       0.93      0.93      0.93      8087
weighted avg       0.93      0.93      0.93      8087
```

Figure 11: LSTM and CNN evaluation

```
[ ]  # Building and training the SVM model
     svm_model = SVC(kernel="linear")
     svm_model.fit(X_train_tfidf, y_train)

     # Predicting and evaluating on the test set
     y_pred_svm = svm_model.predict(X_test_tfidf)
     accuracy_svm = accuracy_score(y_test, y_pred_svm)
     print(f"SVM Model Accuracy: {accuracy_svm}")
```

Figure 12: SVM model building

```
[ ]  # Evaluation for SVM model
     print("SVM Model Evaluation")
     print(classification_report(y_test, y_pred_svm))
```

```
SVM Model Evaluation
              precision    recall  f1-score   support

           0       0.91      0.90      0.91      4118
           1       0.90      0.91      0.90      3969

    accuracy                           0.91      8087
   macro avg       0.91      0.91      0.91      8087
weighted avg       0.91      0.91      0.91      8087
```

Figure 13: SVM Model Evaluation

```
[ ]  # Hyperparameters
     bert_batch_size = 32
     roberta_batch_size = 32
     distilbert_batch_size = 32
     albert_batch_size = 32

     bert_epochs = 5
     roberta_epochs = 5
     distilbert_epochs = 5
     albert_epochs = 5

     bert_max_len = 64
     roberta_max_len = 64
     distilbert_max_len = 64
     albert_max_len = 64
```

Figure 14: Hyperparameters for the transfer models

```
[ ]  # Function for BERT Tokenization
     def bert_encode(data, tokenizer, max_len):
         input_ids = []
         attention_masks = []

         for i in tqdm(range(len(data))):
             encoded = tokenizer.encode_plus(
                 data.iloc[i],
                 add_special_tokens=True,
                 max_length=max_len,
                 pad_to_max_length=True,
                 return_attention_mask=True,
                 return_tensors="pt",
             )
             input_ids.append(encoded["input_ids"])
             attention_masks.append(encoded["attention_mask"])

         input_ids = torch.cat(input_ids, dim=0)
         attention_masks = torch.cat(attention_masks, dim=0)

         return input_ids, attention_masks
```

Figure 15: BERT Model Function

```
print(classification_report(bert_flat_true_labels, bert_flat_prediction

  0%|          | 0/32345 [00:00<?, ?it/s]Truncation was not explicitly
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_
  warnings.warn(
100%|██████████| 32345/32345 [00:58<00:00, 554.19it/s]
100%|██████████| 8087/8087 [00:14<00:00, 565.63it/s]
Some weights of BertForSequenceClassification were not initialized from
You should probably TRAIN this model on a down-stream task to be able t
/usr/local/lib/python3.10/dist-packages/transformers/optimization.py:41
  warnings.warn(
              precision    recall  f1-score   support

           0       0.99      0.95      0.97      4118
           1       0.95      0.99      0.97      3969

    accuracy                           0.97      8087
   macro avg       0.97      0.97      0.97      8087
weighted avg       0.97      0.97      0.97      8087
```

Figure 16: BERT Model Evaluation