National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Ashlyn Nivita Mahendran Joseph Solomon
Student ID: x22163549

School of Computing
National College of Ireland

Supervisor: Shubham Subhnil

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Ashlyn Nivita Mahendran Joseph Solomon
Student ID:	x22163549
Programme:	Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Shubham Subhnil
Submission Due Date:	14/12/2023
Project Title:	Configuration Manual
Word Count:	758
Page Count:	7

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:	Ashlyn Nivita Mahendran Joseph Solomon
Date:	12th 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

Ashlyn Nivita Mahendran Joseph Solomon
x22163549

1 Overview

This configuration manual contains the step-by-step process that was involved in implementing the project "Deciphering Sarcasm in Textual Data: A Comparative Study of Machine Learning and Deep Learning Methods and a Nuanced Dive into Topic Modeling".

2 Environment Setup

Below are the detailed hardware and software specifications necessary to create a suitable environment for the development and analysis work involved in this research project.

2.1 Hardware Requirements

The hardware configuration of the system on which this research project is implemented is detailed below.

- Operating System - Windows 11 Home Single Language, Version: 22H2
- Processor - 12th Gen Intel(R) Core(TM) i5-1235U 1.30 GHz
- Storage – 476 GB
- RAM - 16.0 GB

2.2 Software Requirements

Softwares required for the build and execution are as follows.

- Anaconda Version 23.1.0
- Jupyter Notebook Version 6.4.12
- Python Version 3.9.13
- MS Excel

3 Implementation

3.1 Data Source

The News Headlines dataset used in this project is taken from Kaggle website (refer Figure 1. It contains 26,710 records and 3 columns. The file is in JSON Format (Misra and Arora (2023)).

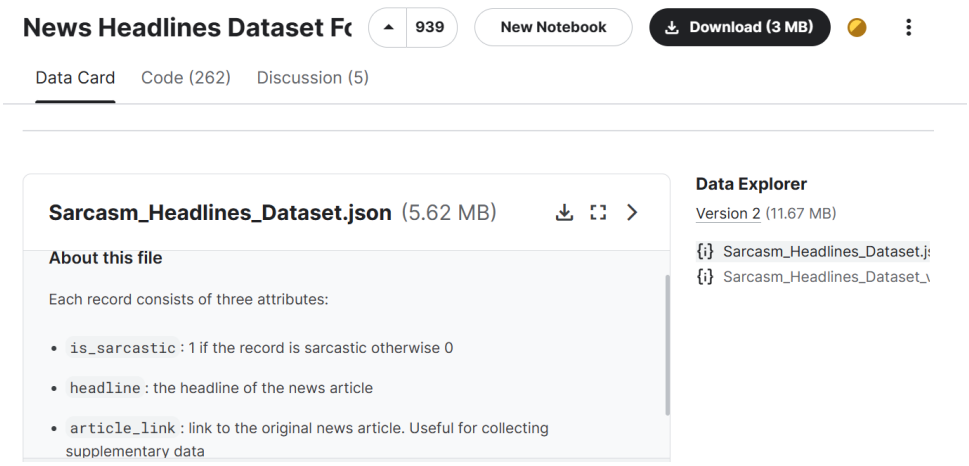

Figure 1: News Headlines Dataset from Kaggle

In addition, the project utilizes a text file containing the 100-dimensional GloVe embeddings, which are essential for training the deep learning model. This file is sourced from Kaggle and will be instrumental in the model's word representation capabilities (refer Figure 2).

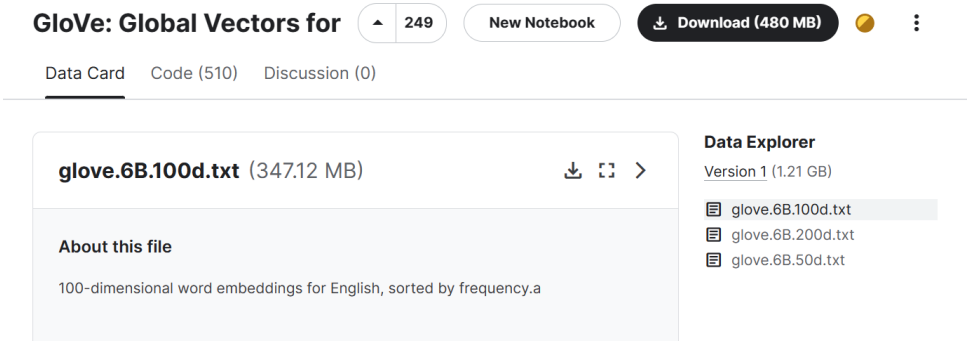

Figure 2: GloVe Embeddings File from Kaggle

3.2 Python Libraries Used

The python libraries used for implementing this project are shown in the Figure 3. The versions of the most important ones are listed below.

- numpy version: 1.23.5
- pandas version: 2.1.3

- matplotlib version: 3.5.2
- seaborn version: 0.11.2
- nltk version: 3.7
- sklearn version: 1.0.2
- gensim version: 4.2.0
- keras version: 2.12.0
- tensorflow version: 2.12.0

```
In [1]: # Basic Data Manipulation and Regular Expressions.
import pandas as pd
import re
import numpy as np
import string

# Plotly and Matplotlib for Data Visualization.
import chart_studio.plotly as py
import plotly.graph_objs as go
from plotly.offline import init_notebook_mode, iplot
from plotly import tools
from matplotlib import pyplot as plt
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import STOPWORDS

# Natural Language Processing and Text Analysis.
import nltk
nltk.download('stopwords')
import itertools
from nltk.corpus import stopwords
from collections import Counter
from collections import defaultdict
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer
from gensim import corpora
from gensim.models.ldamodel import LdaModel
from gensim.models.coherencemodel import CoherenceModel

# TensorFlow and Keras for Deep Learning.
import tensorflow as tf
import random as python_random
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from keras.layers import LSTM, Activation, Dense, Dropout, Input, Embedding
from keras.layers import Input, Embedding, LSTM, Dense, Dropout, Bidirectional, GRU
from keras.optimizers import RMSprop
from keras.models import Model
from keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.regularizers import l1_l2
from tensorflow.keras.layers import Conv1D, MaxPooling1D, GlobalMaxPooling1D
from tensorflow.keras.optimizers import Adam

# Scikit-Learn for Machine Learning and Model Evaluation.
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
```

Figure 3: Imported Python Libraries

3.3 Data Loading

The dataset "Sarcasm_Headlines_Dataset.json" is loaded into Jupyter Notebook and converted to a Pandas Dataframe. The code snippet for this process and a few sample records are depicted in the below Figure 4.

In [2]:
```
# Load the dataset and display the first five records.
sar_acc = pd.read_json('Sarcasm_Headlines_Dataset.json',lines=True)
sar_acc['source'] = sar_acc['article_link'].apply(lambda x: re.findall(r'\w+', x)[2])
sar_acc.head()
```

Out[2]:

	article_link	headline	is_sarcastic	source
0	https://www.huffingtonpost.com/entry/versace-b...	former versace store clerk sues over secret 'b...	0	huffingtonpost
1	https://www.huffingtonpost.com/entry/roseanne-...	the 'roseanne' revival catches up to our thorn...	0	huffingtonpost
2	https://local.theonion.com/mom-starting-to-fea...	mom starting to fear son's web series closest ...	1	theonion
3	https://politics.theonion.com/boehner-just-wan...	boehner just wants wife to listen, not come up...	1	theonion
4	https://www.huffingtonpost.com/entry/jk-rowlin...	j.k. rowling wishes snape happy birthday in th...	0	huffingtonpost

Figure 4: News Headlines Dataset Loading

3.4 Data Pre-processing and Feature Extraction

Once the dataset is loaded, pre-processing techniques such as tokenization, lemmatization, stemming and stopword removal is performed. Feature extractions such as Vectorization, N-grams and Additional text features such as number of words, unique words, characters, stopwords, and punctuations are extracted to gain more insights into the data.

3.5 Modeling and Evaluation

Four models namely, LDA for topic modeling, Deep Learning Hybrid Model (Nayak and Bolla (2022)), Random Forest and Decision Tree for Sarcasm Detection are built using the cleaned data.

The code snippet for building the LDA model is given in the below Figure 5. Once the model is built, it is evaluated using the coherence score metric.

```
# Building the LDA model.
lda = LdaModel(corpus=doc_term_matrix, id2word=dictionary, num_topics=7, random_state=100,
               update_every=1, chunksize=100, passes=50, alpha='auto', per_word_topics=True)

num_words = 7
topics = lda.print_topics(num_topics=7, num_words=num_words)

for idx, topic in topics:
    print('Topic: {} \nWords: {}'.format(idx, topic))
```

Figure 5: Code Snippet for LDA Model Building

For Sarcasm detection, the dependent and independent variables are first selected where the “headline” column is the independent variable and “is_sarcastic” column is the dependent variable. The data is then split into train and test data where 20% of data is allocated for testing.

Tokenization and generation of GloVe Embeddings is done, post which the hybrid deep learning model is defined (refer Figure 6). The model is then trained with 10 and 20 epochs.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inputs (InputLayer)         [(None, 150)]             0

 embedding (Embedding)       (None, 150, 100)          100000

 conv1d (Conv1D)             (None, 150, 64)           19264

 max_pooling1d (MaxPooling1D  (None, 75, 64)           0
 )

 bidirectional (Bidirectiona  (None, 75, 128)          66048
 l)

 gru (GRU)                   (None, 32)                15552

 dropout (Dropout)           (None, 32)                0

 FC1 (Dense)                 (None, 128)               4224

 dropout_1 (Dropout)         (None, 128)               0

 FC2 (Dense)                 (None, 64)                8256

 out_layer (Dense)           (None, 1)                 65

=================================================================
Total params: 213,409
Trainable params: 213,409
Non-trainable params: 0
```

Figure 6: Hybrid Deep Learning Model Architecture

The model that was trained with 10 epochs resulted in a better performance. Thus, the test data predictions were carried out using this model.

```
_____________________________________________________________
Epoch 1/10
193/193 [==============================] - 47s 209ms/step - loss: 0.7077 - accuracy: 0.5574 - val_loss: 0.6958 - val_accuracy:
0.5718
Epoch 2/10
193/193 [==============================] - 34s 176ms/step - loss: 0.6527 - accuracy: 0.6172 - val_loss: 0.5399 - val_accuracy:
0.7543
Epoch 3/10
193/193 [==============================] - 34s 175ms/step - loss: 0.5143 - accuracy: 0.7703 - val_loss: 0.4617 - val_accuracy:
0.7927
Epoch 4/10
193/193 [==============================] - 35s 182ms/step - loss: 0.4609 - accuracy: 0.7997 - val_loss: 0.4387 - val_accuracy:
0.8067
Epoch 5/10
193/193 [==============================] - 34s 175ms/step - loss: 0.4346 - accuracy: 0.8162 - val_loss: 0.4257 - val_accuracy:
0.8156
Epoch 6/10
193/193 [==============================] - 32s 167ms/step - loss: 0.4155 - accuracy: 0.8277 - val_loss: 0.4235 - val_accuracy:
0.8142
Epoch 7/10
193/193 [==============================] - 34s 177ms/step - loss: 0.3978 - accuracy: 0.8365 - val_loss: 0.4118 - val_accuracy:
0.8175
Epoch 8/10
193/193 [==============================] - 34s 175ms/step - loss: 0.3857 - accuracy: 0.8423 - val_loss: 0.4132 - val_accuracy:
0.8231
Epoch 9/10
193/193 [==============================] - 35s 179ms/step - loss: 0.3707 - accuracy: 0.8525 - val_loss: 0.4348 - val_accuracy:
0.8142
Epoch 10/10
193/193 [==============================] - 34s 175ms/step - loss: 0.3592 - accuracy: 0.8585 - val_loss: 0.4063 - val_accuracy:
0.8198
```

Figure 7: Hybrid Deep Learning Model Training with 10 Epochs

```
# Evaluating the model.
loss, accuracy = enhanced_model.evaluate(test_sequences_matrix, Y_test)
print("Test Accuracy:", accuracy)
print("Loss:", loss)
```

```
167/167 [==============================] - 5s 31ms/step - loss: 0.4067 - accuracy: 0.8220
Test Accuracy: 0.8219767808914185
Loss: 0.4067038595676422
```

Figure 8: Hybrid Deep Learning Model Testing

Next, traditional machine learning models such as Random Forest and Decision Tree were implemented, and predictions were made on the test data (Nayel et al. (2021)). The code snippets for these are given below.

```
# Updated Random Forest classifier with the best parameters.
rf_classifier = RandomForestClassifier(
    n_estimators=200,        # Number of trees in the forest.
    criterion='gini',        # Measure the quality of a split.
    min_samples_split=2,     # Minimum number of samples required to split a node.
    min_samples_leaf=1,      # Minimum number of samples required to be at a leaf node.
    random_state=42          # Random state for reproducibility.
)

# Train the classifier.
rf_classifier.fit(X_train, Y_train.ravel())
```

```
# Make predictions.
Y_pred = rf_classifier.predict(X_test)
```

Figure 9: Code Snippets for Random Forest Model Training and Testing

```
# Train the Decision Tree classifier.
dt_classifier = DecisionTreeClassifier(
    max_depth=10,            # Maximum depth of the tree.
    min_samples_split=5,     # Minimum number of samples required to split an internal node.
    min_samples_leaf=4,      # Minimum number of samples required to be at a leaf node.
    random_state=42          # Random state for reproducibility.
)
dt_classifier.fit(X_train_vect, Y_train)
```

```
# Make predictions.
Y_pred = dt_classifier.predict(X_test_vect)
```

Figure 10: Code Snippets for Decision Tree Model Training and Testing

These models were evaluated using Accuracy, Precision, Recall, F1-Score and a Confusion Matrix (Vujović (2021)).

References

Misra, R. and Arora, P. (2023). Sarcasm detection using news headlines dataset, *AI Open* **4**: 13–18.

Nayak, D. and Bolla, B. (2022). Efficient deep learning methods for sarcasm detection of news headlines, *Machine Learning and Autonomous Systems: Proceedings of ICMLAS 2021*, Springer Nature Singapore, pp. 371–382.

Nayel, H., Amer, E., Allam, A. and Abdallah, H. (2021). Machine learning-based model for sentiment and sarcasm detection, *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, pp. 386–389.

Vujović, (2021). Classification model evaluation metrics, *International Journal of Advanced Computer Science and Applications* **12**(6): 599–606.