National College of Ireland

Configuration Manual

MSc Research Project
Practicum

Sampath Reddy Kalwa
X23337702

School of Computing
National College of Ireland

Supervisor: Lavish Thomas

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……. …Sampath Reddy Kalwa……

Student ID: ……X23337702……..……

Programme: ……MSC in AI…… **Year:** ……2025…..

Module: ……Practicum 2……..……

Lecturer: ……Lavish Thomas……..……

Submission Due Date: ……15/09/2025……..……

Project Title: ……Classification of hate speech using machine learning and natural language processing ……..……

Word Count: ……1141……

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: ……Sampath kalwa……

Date: ……15/09/2025……

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

Sampath Reddy Kalwa
X23337702

1 Introduction

This configuration manual includes any information regarding hardware and software specifications employed in the process of this research. The sections below demonstrate measures that ought to be taken to make an environment to run this research study. It also has various application that needs configuration and usage.

2 System Specifications

The system specification has all the resources utilized so as to accomplish the research study. Figure 1 demonstrates the local system where this project is run during research purpose. Figure 2 provides useful information regarding Jupyter Nootbook. All the hardware accelerators have undergone research use and must be cautious because of the limit of use.

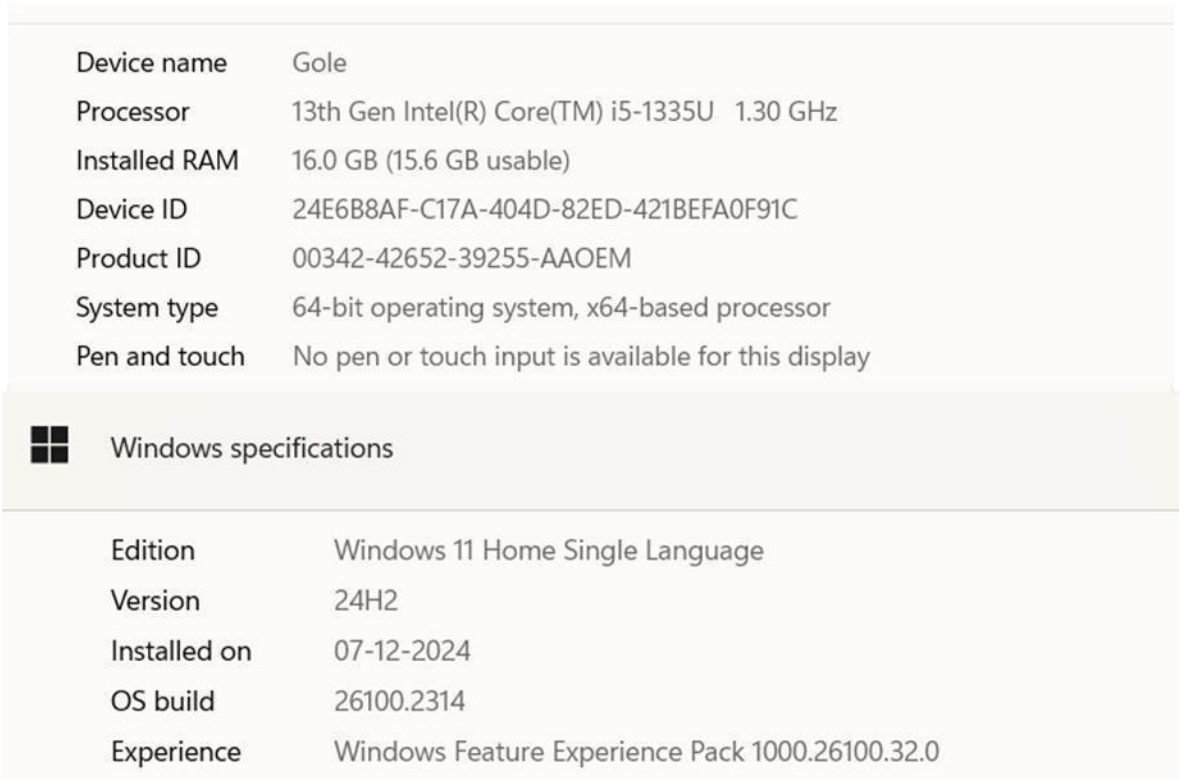

Figure 1: System Specifications

Figure 2: Jupyter Notebook

Software Used:

- Microsoft excel: Used for custom dataset description.
- Jupyter Notebook: Used for all processing and as code runtime environment

3 Dataset specification

Data have been sourced from the "Hate Speech and Offensive Language dataset" available on Kaggle (Samoshyn, 2020). This dataset comprises 24,783 manually annotated English tweets labelled as hate speech, offensive language, or neither. The dependent variable is the categorical label for hate content detection. Independent variables include token counts, TF-IDF vectors, embedding dimensions, tweet length, and punctuation frequency, derived via tokenisation, stop-word removal, lemmatisation, and vectorisation. This dataset offers sufficient volume, quality, and balanced class distribution to train machine learning models, ensuring reproducibility and guiding strategies to curb hate speech.

	count	hate_spee	offensive_	neither	class	tweet
0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't complain about cleaning up your house. & as a man you should always take the trash out...
1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn bad for cuffin dat hoe in the 1st place!!
2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby4life: You ever fuck a bitch and she start to cry? You be confused as shit
3	3	0	2	1	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she look like a tranny
4	6	0	6	0	1	!!!!!!!!!!!! RT @ShenikaRoberts: The shit you hear about me might be true or it might be faker than the bitch who told it to ya 
5	3	1	2	0	1	!!!!!!!!!!!!!!!!!"@T_Madison_x: The shit just blows me..claim you so faithful and down for somebody but still fucking with hoes! 😂😂😂"
6	3	0	3	0	1	!!!!!!"@__BrighterDays: I can not just sit up and HATE on another bitch .. I got too much shit going on!"
7	3	0	3	0	1	!!!!“@selfiequeenbri: cause I'm tired of you big bitches coming for us skinny girls!!”
8	3	0	3	0	1	" & you might not get ya bitch back & thats that "
9	3	1	2	0	1	"
10	3	0	3	0	1	" Keeks is a bitch she curves everyone " lol I walked into a conversation like this. Smh
11	3	0	3	0	1	" Murda Gang bitch its Gang Land "
12	3	0	2	1	1	" So hoes that smoke are losers ? " yea ... go on IG
13	3	0	3	0	1	" bad bitches is the only thing that i like "
14	3	1	2	0	1	" bitch get up off me "
15	3	0	3	0	1	" bitch nigga miss me with it "
16	3	0	3	0	1	" bitch plz whatever "
17	3	1	2	0	1	" bitch who do you love "
18	3	0	3	0	1	" bitches get cut off everyday B "
19	3	0	3	0	1	" black bottle & a bad bitch "
20	3	0	3	0	1	" broke bitch cant tell me nothing "
21	3	0	3	0	1	" cancel that bitch like Nino "
22	3	0	3	0	1	" cant you see these hoes wont change "
23	3	0	3	0	1	" fuck no that bitch dont even suck dick " 😂😂😂 the Kermit videos bout to fuck IG up
24	3	0	3	0	1	" got ya bitch tip toeing on my hardwood floors " 😂 http://t.co/cOU2WQ5L4q
25	3	0	2	1	1	" her pussy lips like Heaven doors " 😌

Figure 3: Dataset

4 Project Development

```python
# 1. Imports
import pandas as pd
import numpy as np
import re
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import SMOTE
from sklearn.utils import class_weight
from sklearn.utils.class_weight import compute_class_weight
import nltk
from nltk.corpus import stopwords
from wordcloud import WordCloud
from nltk.stem import WordNetLemmatizer
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# For deep learning
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SimpleRNN, LSTM, Dense, Dropout, Bidirectional
from tensorflow.keras.utils import to_categorical

# 2. Download required NLTK data
# nltk.download('stopwords')
# nltk.download('wordnet')
# nltk.download('vader_lexicon')

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

Figure 4: Import Libraries

These are progressive entry points of the code preconditioning of an effective machine learning workflow and an interpretable language conversion pipeline. The importation of libraries is highly crucial because it enables the importation of libraries that are mandatory in data manipulation, machine learning, as well as deep learning. Analysis and manipulation of data is performed using pandas and NumPy, and visualisation is performed using Matplotlib. In libraries like Scikit-learn, there are functions which can be used to build and evaluate the model and perform activities like dividing the dataset, generating a classification report, and adding up confusion matrices. NLTK can also be used during pre-processing (tokenisation, removal of stopwords, lemmatisation), which is essential in the preparation of the data in the text towards machine learning.

```python
# 3. Load the dataset
df = pd.read_csv('labeled_data.csv', encoding='latin-1')
```

Figure 5: Data Load

The second important thing is to load the data set with the help of Pandas. This will enable the reading of the dataset in the system, which is in CSV format. The specifications of the Latin-1 encoding allow it to process special characters that can be present in text data that represents social media content.

```python
# Keep only the relevant columns and drop rows with missing values
df = df[['tweet', 'class']].dropna()
```

```python
# Map numeric classes to descriptive labels
class_mapping = {0: 'hate_speech', 1: 'offensive_language', 2: 'neither'}
df['label'] = df['class'].map(class_mapping)
```

Figure 6: Drop rows and Numeric classes

The figures demonstrate significant steps of data pre-processing in a pipeline of a machine learning model to detect hate speech. The first figure illustrates how the value of the dataset can be cleaned by choosing the necessary columns (in the given case, the 'tweet' and 'class' columns) and dropping the row consisting of missing values with the help of the dropna() function. This step will also make the data complete and fit for analysis. The latter segment assigns the descriptive labels to the numeric ('hate_speech', 'offensive_language', 'neither') class labels based on a dictionary of classes (Kovács, Alonso and Saini, 2021). This is necessary because it enhances the sentences of the dataset to understandable data by taking numbers instead of words in their labels.

```
# 5. Text Preprocessing
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()
```

```
def clean_text(text):
    """
    - Lowercase
    - Remove URLs
    - Remove mentions (@user)
    - Remove non-alphabetic characters
    - Tokenize, remove stopwords, and lemmatize
    """
    text = text.lower()
    text = re.sub(r"http\S+|www\S+|https\S+", '', text)      # remove URLs
    text = re.sub(r'@\w+', '', text)                         # remove @mentions
    text = re.sub(r'[^a-z\s]', '', text)                     # keep only letters and spaces
    tokens = text.split()
    tokens = [lemmatizer.lemmatize(word) for word in tokens if word not in stop_words]
    return ' '.join(tokens)
```

Figure 7: Text Preprocessing

Texture pre-processing is another text pre-processing that is essential to natural language processing (NLP). NLTK is used to remove common stop words of the English language, and a WordNet Lemmatiser is initialised. The text clean_text () does the following work, it changes the text to lowercase, removes the URLs, and mentions (@user), non-alphabetical characters, and then tokenises the text.

```
# Apply cleaning to the sampled dataframe
df_sample['cleaned'] = df_sample['tweet'].apply(clean_text)
```

```
# 6. Sentiment Analysis using VADER
sia = SentimentIntensityAnalyzer()
df_sample['sentiment_score'] = df_sample['cleaned'].apply(lambda x: sia.polari
```

```
# Categorize sentiment into negative, neutral, positive
df_sample['sentiment_label'] = pd.cut(
    df_sample['sentiment_score'],
    bins=[-1.0, -0.05, 0.05, 1.0],
    labels=['negative', 'neutral', 'positive']
)
```

Figure 8: Sentiment Analysis

Sentiment scores of each of the cleaned tweets are then calculated using the polarity_scores () method of Vader to generate a compound score between -1 and + 1. These scores are binned as negative (<= -0.05), neutral (-0.05 to 0.05), and positive (> 0.05). A distribution of the outcome, 7,257 negative, 2,319 neutral, and 2,816 positive tweets, gives a picture of the emotional background of the data.

```
# 8. Encode target labels for classification
le = LabelEncoder()
df_sample['encoded_label'] = le.fit_transform(df_sample['label'])
```

```
X_text = df_sample['cleaned']
y = df_sample['encoded_label']
```

```
# 9. Train/Test split (80% train, 20% test), stratify to preserve label distri
X_train_text, X_test_text, y_train, y_test = train_test_split(
    X_text, y, test_size=0.2, random_state=42, stratify=y
)
```

```
# 10. TF-IDF Vectorization (only on training text)
tfidf = TfidfVectorizer(max_features=5000)
X_train_tfidf = tfidf.fit_transform(X_train_text)
X_test_tfidf  = tfidf.transform(X_test_text)
```

Figure 9: Encode, split and Vectorisation

The cleaned tweets in are labelled and encoded by numeric targets using LabelEncoder with hate_speech, offensive language, and neither being converted into numbers to be identified with. An optimised split is accomplished on the dataset (stratify=y), which separates the dataset into 80% of the training dataset and 20% of the testing dataset to preserve the original proportion of classes. TF-IDF vectorisation is then performed on the training text (over a maximum of 5,000 features) and the same conversion subsequently performed against the test data, converting text into weighted numerical feature vectors. Lastly, the SMOTE analysis is applied to the training TF-IDF data to artificially over-sample minority classes, maintaining class balances and consequently enhancing the classifier to be able to learn on under-represented instances of hate speech.

```
===== Random Forest Results =====

Classification Report:
                    precision    recall  f1-score   support

       hate_speech       0.41      0.38      0.39       149
           neither       0.80      0.87      0.84       418
offensive_language       0.94      0.92      0.93      1912

          accuracy                           0.88      2479
         macro avg       0.72      0.72      0.72      2479
      weighted avg       0.88      0.88      0.88      2479

Confusion Matrix:
```

```
===== RNN Results =====

Classification Report:
                    precision    recall  f1-score   support

       hate_speech       0.16      0.15      0.15       149
           neither       0.48      0.43      0.46       418
offensive_language       0.85      0.87      0.86      1912

          accuracy                           0.75      2479
         macro avg       0.50      0.48      0.49      2479
      weighted avg       0.74      0.75      0.75      2479
```

```
                    precision    recall  f1-score   support

       hate_speech       0.26      0.50      0.34       149
           neither       0.80      0.72      0.76       418
offensive_language       0.92      0.87      0.89      1912

          accuracy                           0.82      2479
         macro avg       0.66      0.70      0.66      2479
      weighted avg       0.86      0.82      0.84      2479

===== BiLSTM Results =====

Classification Report:
                    precision    recall  f1-score   support

       hate_speech       0.26      0.50      0.34       149
           neither       0.80      0.72      0.76       418
offensive_language       0.92      0.87      0.89      1912

          accuracy                           0.82      2479
         macro avg       0.66      0.70      0.66      2479
      weighted avg       0.86      0.82      0.84      2479
```

Figure 10: Models Report

```
===== Model Comparison =====
               accuracy  precision    recall  f1-score
Random Forest  0.883017   0.882558  0.883017  0.882446
RNN            0.753126   0.744413  0.753126  0.748430
BiLSTM         0.821299   0.856893  0.821299  0.835915
```

Figure 11: Model comparision

References

Samoshyn, A. (2020). *Hate Speech and Offensive Language Dataset*. [online] kaggle.com. Available at: https://www.kaggle.com/datasets/mrmorj/hate-speech-and-offensive-language-dataset [Accessed 26 Jun. 2025].

Kovács, G., Alonso, P. and Saini, R. (2021). Challenges of Hate Speech Detection in Social Media. *SN Computer Science*, [online] 2(2), pp.1–15. doi:https://doi.org/10.1007/s42979-021-00457-3.