National College of Ireland

EFFICIENT CYBERBULLYINGDETECTION ON SOCIALMEDIA USING LIGHTWEIGHT TRANSFORMER AND BILSTM HYBRID ARCHITECTURE

MSc Research Project
Msc. Data Analytics

Anusha Koritala
Student ID: X23329076

School of Computing
National College of Ireland

Supervisor : Yalemisew Abgaz

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Anusha Koritala
Student ID:	x23329076
Programme	MSc. Data Analytics **Year:** 2024- 2025
Module:	MSc. Research Project
Supervisor:	Yalemisew Abgaz
Submission Due Date:	15-09-2025
Project Title:	Efficient Cyberbullying Detection on Social Media Using Lightweight Transformer and BiLSTM Hybrid Architecture
Word Count:	**643** **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: Koritala Anusha
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):	

EFFICIENT CYBERBULLYING DETECTION ON SOCIALMEDIA USING LIGHTWEIGHT TRANSFORMER ANDBILSTM HYBRID ARCHITECTURE

Anusha Koritala
X23329076

1. Environment Setup

- Install Python 3.8 or above
- Install required libraries: transformers, torch, scikit-learn, lightgbm, pandas, numpy, matplotlib
- Ensure GPU availability for faster transformer training (optional but recommended)
- Set random seeds for reproducibility

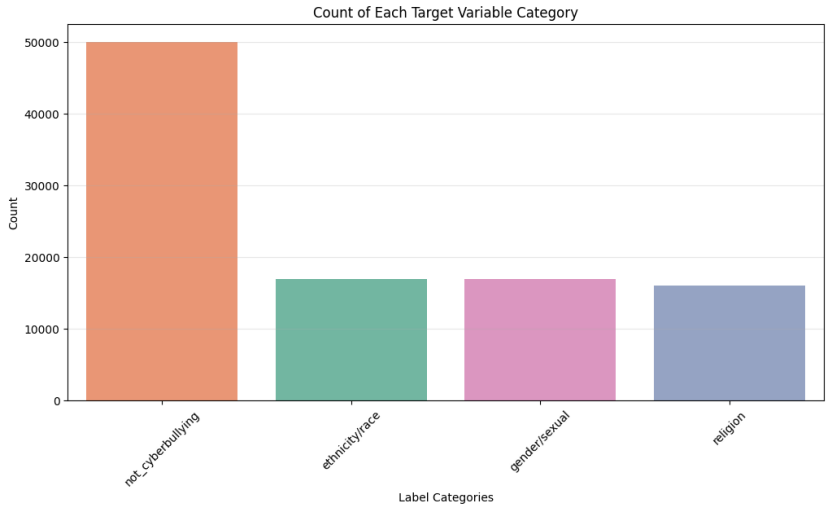
2. Dataset Preparation

The dataset used in this study is the **Cyberbullying Classification Dataset** from Kaggle.It consists of **47,000+ labeled social media posts**, classified into the following categories: **ethnicity/race, religion, gender/sex, and not_cyberbullying**. This dataset provides a diverse set of real-world examples that reflect the complexity of online conversations.

The dataset is publicly available at the following link:

https://www.kaggle.com/datasets/andrewmvd/cyberbullying-classification

- Use labeled social media dataset with categories:
    - ethnicity/race
    - religion
    - gender/sexual
    - not_cyberbullying
- Store in CSV/JSON format with two columns: text and label
- Remove duplicates and irrelevant records



3. Data Preprocessing

- Lowercase all text
- Remove:
    - Special characters and punctuation
    - Usernames/mentions (@username)
    - URLs
    - Stopwords (optional for transformers)
- Handle class imbalance using:
    - Class weights in loss function
    - Oversampling (SMOTE) or undersampling
- Tokenization:
    - Use DistilBERT tokenizer with max sequence length (e.g., 128–256 tokens)
    - Apply padding and truncation

4. Baseline Model Configuration

- **Random Forest Classifier (RFC)**
- **Light GBM**
- **KNN**

```python
# === Imports ===
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (accuracy_score, classification_report,
    confusion_matrix, roc_curve, auc)
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

# === Encode Labels ===
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# === Train-Test Split ===
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)

# === Train Random Forest with Varying Estimators ===
estimators = [10, 25, 50, 75, 100, 150, 200]
accuracies = []

for n in estimators:
    clf = RandomForestClassifier(n_estimators=n, max_depth=10, random_state=42)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    accuracies.append(acc)
```

5. Deep Learning Model Configuration (BiLSTM)

```python
import numpy as np
import random
import tensorflow as tf
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout, Bidirectional
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
import os

# Set seeds for reproducibility
seed_value = 42
os.environ['PYTHONHASHSEED'] = str(seed_value)
np.random.seed(seed_value)
random.seed(seed_value)
tf.random.set_seed(seed_value)

# Tokenize and pad
tokenizer = Tokenizer(num_words=10000, oov_token="<OOV>")
tokenizer.fit_on_texts(df['cleaned_text'])
X = tokenizer.texts_to_sequences(df['cleaned_text'])
X = pad_sequences(X, padding='post', maxlen=100)

# Encode labels
y = LabelEncoder().fit_transform(df['label'])

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=seed_value)

# Build improved model
model = Sequential()
model.add(Embedding(input_dim=10000, output_dim=64))
```

6. Hybrid Model Configuration (DistilBERT + BiLSTM)

```python
# Train-test split
X_train_ids, X_test_ids, y_train, y_test = train_test_split(
    input_ids, labels, test_size=0.2, random_state=42
)
X_train_mask, X_test_mask, _, _ = train_test_split(
    attention_mask, labels, test_size=0.2, random_state=42
)

# === Custom DistilBERT Layer===
class DistilBertEncoder(Layer):
    def __init__(self, model_name, **kwargs):
        super(DistilBertEncoder, self).__init__(**kwargs)
        self.bert = TFAutoModel.from_pretrained(model_name)

        # Freeze all layers except last 2
        for layer in self.bert.distilbert.transformer.layer[:-2]:
            layer.trainable = False
        for layer in self.bert.distilbert.transformer.layer[-2:]:
            layer.trainable = True

    def call(self, inputs):
        input_ids, attention_mask = inputs
        outputs = self.bert(input_ids=input_ids, attention_mask=attention_mask)
        return outputs.last_hidden_state  # (batch, seq_len, hidden_size)

# === Build the Model ===
input_ids_layer = Input(shape=(128,), dtype=tf.int32, name='input_ids')
attention_mask_layer = Input(shape=(128,), dtype=tf.int32, name='attention_mask')

bert_output = DistilBertEncoder(model_name)([input_ids_layer, attention_mask_layer])
x = GaussianNoise(0.2)(bert_output)
x = Bidirectional(LSTM(32))(x)
x = Dropout(0.5)(x)
output = Dense(1, activation='sigmoid')(x)
```

7. Training Process

- Split dataset into train/validation/test (e.g., 70/15/15)
- Apply stratification to maintain class distribution
- Train baseline ML models for comparison
- Train BiLSTM model separately
- Train hybrid DistilBERT + BiLSTM model
- Monitor metrics: Accuracy, Precision, Recall, F1-score

8. Evaluation & Visualization

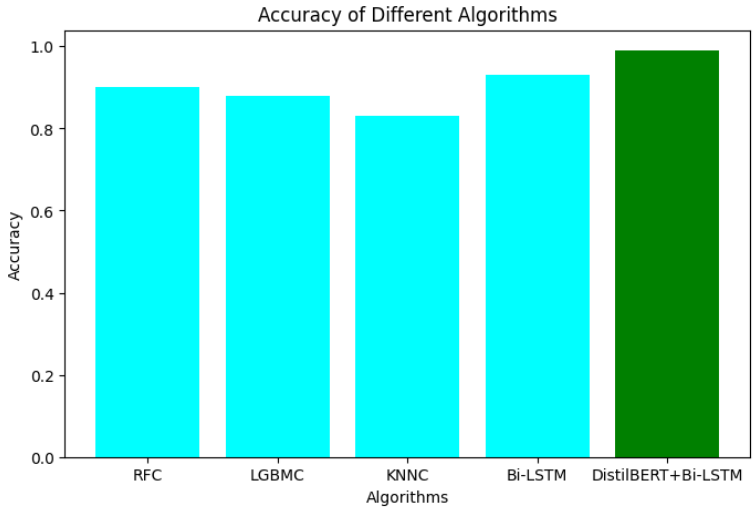
```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      9916
           1       0.99      1.00      0.99     10082

    accuracy                           0.99     19998
   macro avg       0.99      0.99      0.99     19998
weighted avg       0.99      0.99      0.99     19998

Confusion Matrix:
 [[ 9807   109]
 [   35 10047]]
```

- Use classification report for detailed metrics per class
- Generate confusion matrix to analyze misclassifications
- Compare models:
    - RFC, LGBMC, KNNC, BiLSTM, DistilBERT+BiLSTM



References

Python – General-purpose programming – https://docs.python.org/3/

NumPy – Numerical computing & arrays – https://numpy.org/doc/

Pandas – Data manipulation & EDA – https://pandas.pydata.org/docs/

Scikit-learn – ML models & metrics – https://scikit-learn.org/stable/documentation.html

PyTorch – Deep learning (BiLSTM& hybrid) – https://pytorch.org/docs/stable/index.html

TensorFlow/Keras – Model prototyping – https://www.tensorflow.org/api_docs/python/tf/keras

Hugging Face Transformers – DistilBERT embeddings – https://huggingface.co/docs/transformers/index

LightGBM / XGBoost – Gradient boosting – https://lightgbm.readthedocs.io/en/latest/ | https://xgboost.readthedocs.io/en/stable/

Matplotlib / Seaborn – Visualizations – https://matplotlib.org/stable/users/index.html | https://seaborn.pydata.org/

Plotly – Interactive visualizations – https://plotly.com/python/