National College of Ireland

Detecting AI-Generated Text Using a Hybrid Deep Learning Framework for Authenticity Verification

MSc Research Project
Research Practicum

Ridhi Vipin Sharma
Student ID: 23297093

School of Computing
National College of Ireland

Supervisor: Cristina Hava Muntean

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ridhi Vipin Sharma
Student ID:	23297093
Programme:	Msc in Data Analytics **Year:** 2025-2026
Module:	Research Practicum
Lecturer: **Submission Due Date:**	January 28,2026
Project Title:	Detecting AI-Generated Text Using a Hybrid Deep Learning Framework for Authenticity Verification
Word Count:	960 **Page Count:** 5

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: Ridhi Vipin Sharma

Date: January 27,2026

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):	

Detecting AI-Generated Text Using a Hybrid Deep Learning Framework for Authenticity Verification

Ridhi Vipin Sharma
Student ID: 23297093

1. Introduction

The project works on a framework that is used to compute the distinction between human-written text and text produced by a powerful large language model (LLM) like GPT-3.5, Claude-v1, and LLaMA-30B. With an increasing emulation of human linguistic patterns by the modern LLMs, the ability to reliably identify synthetic text grew as a complex problem, surpassing the capabilities of the older machine learning algorithms, which used shallow surface-level lexical features. To address the limitations, the project deploys an end-to-end analytical pipeline with classical ML baselines, deep learning sequence models, and a hybrid ALBERT+LSTM architecture. The system can use the parameter-efficient contextual embeddings of ALBERT and LSTM strengths in temporal modelling to provide a scalable, accurate, and interpretable text detection solution using AI, achieving a final accuracy of 0.91.

2. System Specifications

The configuration described below reflects the environment used to preprocess the dataset, extract features, train models, generate embeddings, and evaluate performance.

Hardware Requirements

- **CPU:** Multi-core x86_64 processor (≥ 4 cores recommended)
- **GPU:** NVIDIA GPU with CUDA support
- **RAM:** Minimum 8GB (16 GB recommended for transformer embedding generation)
- **Storage:** At least 8 GB free disk space for datasets, embeddings, and model files

Software Requirements

- **Operating System:** Ubuntu 20.04+, Windows 10/11, or Google Colab Linux VM
- **Python Version:** 3.8–3.10
- **Environment:** Conda, venv, or Google Colab GPU Runtime
- **CUDA Toolkit:** CUDA 11.x
- **cuDNN:** Version compatible with installed TensorFlow

3. Required Libraries

The system is based on a family of fundamental Python libraries supporting an effective process of data pre-processing, feature engineering, model development and evaluation. The basic numerical computation tools and structured data manipulation tools are found in NumPy (version 1.26) and Pandas (version 2.1). Matplotlib (version 3.8) and Seaborn (version 0.13) help to create visual analytics by creating plots of exploratory data analysis and visualisations of performance. The NLTK library (version 3.9) is used to perform text pre-processing tasks, such as tokenisation, stopword elimination, and lemmatisation. It uses scikit-learn (version

1.3), which provides TF-IDF vectorisation, classical machine learning models, metric computation and hyperparameter tuning by use of GridSearchCV. The implementation of deep learning elements, such as GRU, LSTM, and the hybrid architecture, is performed with the help of TensorFlow and Keras (version 2.15). The Hugging Face Transformers library (version 4.36) is used to generate contextual embeddings to the ALBERT model. Also, the TQDM (version 4.66) tool is utilized to offer progress bars when performing the embedding generation process to guarantee the visibility and tracking of the activity in case of long-term operations.

The necessary libraries can be installed using the following commands:

a) **pip install numpy==1.26 pandas==2.1 matplotlib==3.8 seaborn==0.13 nltk==3.9 scikit-learn==1.3**
b) **pip install tensorflow==2.15 transformers==4.36 tqdm==4.66**

4. Dataset description

The data that was utilized in this research is the publicly available Human vs LLM Text Corpus on Kaggle that comprises 31,180 entries in five columns and is marked as Human, GPT-3.5, Claude-v1 or LLaMA-generated text. It offers a subset of samples of all types that are selected and diversified based on social media, online conversation, and the work of various language models. The records contain the raw text, source label and metadata of the text prompt ID, text length and word count. The data set is in a wide range of writing styles, sentence structure, and contextual differences, which include short informal postings and long, consistent responses. This language diversity contributes to the fact that the corpus is useful in studying the stylistic and semantic variations between human and AI-generated language and provides a strong basis to train and evaluate detection models.

5. Models Implemented

The modelling approach is an evolution of the conventional lexical-frequency approaches to the deep contextual sequence learning and eventually leads to the hybrid ALBERT-LSTM architecture. First trained on TF-IDF features to set interpretable performance baselines, classical machine learning baselines, such as Logistic Regression, Linear SVM, and Random Forest, are limited by their use of sparse lexical information to understand semantics. To learn temporal structure, Bidirectional GRU and Bidirectional LSTM models were then applied using tokenised sequences, which can learn stylistic and sequential patterns, but not deep contextual representation. The last and the most successful model is a combination of the contextual embeddings of ALBERT and the LSTM network, which combines semantic depth modelling with the time dependency modelling. Such a hybrid structure enables the system to recognize the subtle linguistic differences and makes the distinction between the human and LLM-written text far more effective.

6. Code snippets and plots:

1) Descriptive analysis

```
df.clean_word_count.describe()
```

	clean_word_count
count	23359.000000
mean	276.298771
std	357.439088
min	9.000000
25%	93.000000
50%	222.000000
75%	341.000000
max	15710.000000

2) Source distribution

```
df.source.value_counts()
```

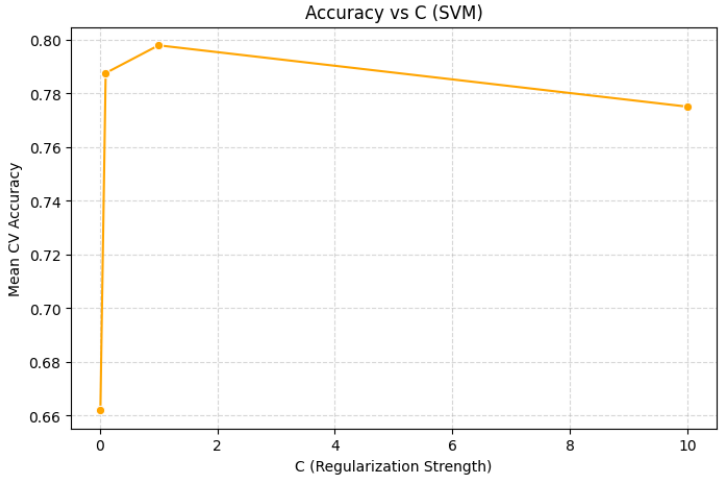
source	count
Human	7902
GPT-3.5	7391
LLaMA-30B	5209
Claude-v1	2857

3) Splitting data

```
from sklearn.model_selection import train_test_split
# Define features (X) and target (y)
X = X_tfidf
y = df['source']

# Split 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,
    stratify=y
)
print("Training set shape:", X_train.shape)
print("Testing set shape:", X_test.shape)
```

```
Training set shape: (18687, 5000)
Testing set shape: (4672, 5000)
```

4) SVM hyperparameter plot



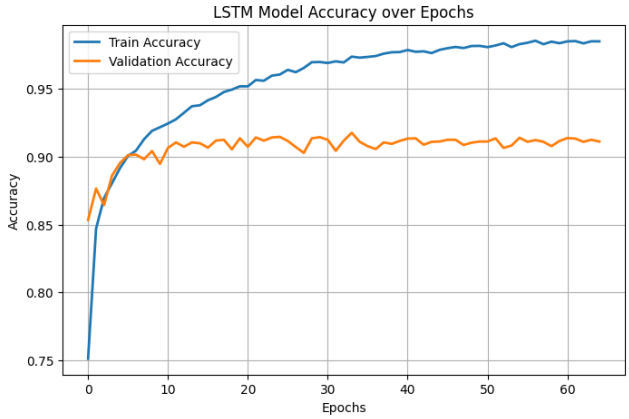
5) Albert embeddings

```
from transformers import AutoTokenizer, TFAutoModel
model_name = "albert-base-v2"
tokenizer = AutoTokenizer.from_pretrained(model_name)
albert_model = TFAutoModel.from_pretrained(model_name, from_pt=True)
from tqdm import tqdm

def get_albert_embeddings(texts, batch_size=32):
    embeddings = []
    if not isinstance(texts, list):
        texts = list(texts)

    for i in tqdm(range(0, len(texts), batch_size), desc="Embedding batches"):
        batch_texts = texts[i:i+batch_size]
        inputs = tokenizer(
            batch_texts,
            padding=True,
            truncation=True,
            max_length=256,
            return_tensors='tf'
        )
        outputs = albert_model(inputs['input_ids'], attention_mask=inputs['attention_mask'])
        cls_emb = outputs.last_hidden_state[:,0,:]
        embeddings.append(cls_emb.numpy())
```

6) Lstm on albert embeddings

```
model = Sequential([
    Input(shape=(X_train_emb.shape[1], X_train_emb.shape[2])),
    LSTM(128, return_sequences=False),
    Dropout(0.3),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(num_classes, activation='softmax')
])

# Compile
model.compile(
    loss=tf.keras.losses.SparseCategoricalCrossentropy(),
    optimizer=tf.keras.optimizers.Adam(learning_rate=3e-4),
    metrics=['accuracy']
)
```

7) Accuracy plot of ALBERT + LSTM



8) Comparison of evaluation metrics

Table 1. Performance Comparison of Machine Learning, Deep Learning, and Hybrid Models

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	0.81	0.83	0.81	0.82
Logistic Regression	0.80	0.82	0.81	0.81
SVM (Linear)	0.81	0.82	0.82	0.82
GRU	0.82	0.83	0.82	0.82
LSTM	0.82	0.82	0.82	0.81
ALBERT+LSTM	0.91	0.91	0.91	0.91

7. Conclusion:

The workflow and system configuration outlined in the present paper will offer a robust and reproducible system setup regarding the use of natural language text to verify its authenticity. The project provides a solid baseline using structured pre-processing, TF-IDF modelling and sequential deep networks. With the inclusion of ALBERT embeddings and LSTM, a massive boost in performance is observed, reaching 0.91 accuracy with balanced precision, recall and F1. This validates the hybrid model as the most practical and computationally efficient architecture to use to differentiate between human text and LLM-generated text. Modular design allows multilingual datasets or alternative transformer encoders or use in real-time content screening applications.

References:

1. https://huggingface.co/docs/transformers/en/index
2. https://www.nltk.org/
3. https://www.tensorflow.org/
4. https://scikit-learn.org/stable/
5. https://numpy.org/
6. https://pandas.pydata.org/