AI Content Forensics: Detect, Rewrite, and Defend

1. Background and Motivation

With the rapid rise of large language models (LLMs) such as ChatGPT, distinguishing between human-written and AI-generated text has become an important challenge in education, journalism, and online communication. While AI-generated text can be useful and efficient, it also raises concerns regarding authorship, academic integrity, misinformation, and over-reliance on automated systems. At the same time, purely statistical detectors often fail when text is rewritten or stylistically modified.

The motivation behind this project is to explore AI content forensics by combining classical NLP techniques with modern large language models. Rather than treating AI detection as a black-box decision, this project emphasizes interpretability, interaction, and resilience testing. The system not only detects whether text appears AI-generated or human-written, but also analyzes sentiment and formality, explains its reasoning, and tests whether GPT-based rewriting can successfully fool the detector.

This project demonstrates how classical NLP and neural models can be integrated with LLMs to build an interactive, end-to-end forensic analysis tool.

2. Problem Formulation

The core objective of this project is to design an NLP system capable of:

Detecting textual attributes

AI-generated vs. human-written text

Formal vs. informal writing style

Positive vs. negative sentiment

Explaining classifier decisions

Provide probabilistic outputs and stylistic interpretation

Strategically rewriting text using GPT

Modify text to change its classification outcome

Base rewrites on actual classifier signals

Testing detector robustness

Allow users to attempt adversarial rewrites

Re-evaluate rewritten text to see whether the detector is fooled

This formulation directly corresponds to Option 3: “AI Content Forensics — Detect + Rewrite + Defend.”

3. Dataset Description

Multiple datasets were used during development:

AI vs Human Detection

Public datasets containing human-written text and AI-generated text were used to train a binary classifier using TF-IDF features and Logistic Regression.

Formality Classification

Labeled datasets distinguishing formal and informal writing styles were used, again employing TF-IDF vectorization and Logistic Regression.

Sentiment Analysis

A labeled sentiment dataset was used to train:

A classical logistic regression baseline (later removed)

A neural Multi-Layer Perceptron (MLP) implemented in TensorFlow

All datasets were preprocessed using standard NLP techniques including tokenization, normalization, and vectorization.

4. Classical NLP Pipeline

4.1 Preprocessing

Text preprocessing includes:

Tokenization

Lowercasing

Vocabulary filtering through TF-IDF vectorization

Stopword handling and feature weighting are implicitly managed through the TF-IDF vectorizer.

4.2 Vectorization

TF-IDF (Term Frequency–Inverse Document Frequency) was used for all classical models.

TF-IDF captures lexical importance while reducing the influence of overly common words.

4.3 Classical Models

Two Logistic Regression classifiers were trained:

Formality Classifier

AI vs Human Classifier

Logistic Regression was chosen due to its interpretability, efficiency, and strong performance on sparse text features.

4.4 Neural Baseline

A Neural Sentiment Classifier (MLP) was implemented using TensorFlow/Keras.

This model serves as a neural baseline and improvement over the classical sentiment classifier.

During experimentation, a logistic regression sentiment baseline was evaluated but removed from the final system due to:

Over-reliance on surface-level lexical cues

Less stable probability calibration compared to the neural model

5. Model Evaluation

Each classifier was evaluated using standard metrics:

Precision

Recall

F1-score

The neural sentiment model demonstrated more stable confidence distributions than the classical baseline. The AI vs Human classifier performed well on longer texts but showed sensitivity to stylistic rewriting, which is intentionally explored in this project.

While detailed confusion matrices are not displayed in the CLI, evaluation was conducted during development and informed model selection.

6. OpenAI API Integration

6.1 GPT Usage

The OpenAI API is used for three distinct tasks:

Explanation Generation

GPT explains classifier outcomes using sentiment, formality, and AI-detection probabilities.

Human-Style Rewriting

GPT rewrites text to sound more human and natural.

Adversarial Rewriting

GPT attempts to intentionally fool the AI detector.

Each GPT call is conditioned on actual classifier outputs, ensuring that the LLM is grounded in statistical analysis rather than hallucinated reasoning.

6.2 Prompt Engineering

System prompts are carefully designed to:

Maintain a consistent “AI content forensics expert” role

Reference classifier probabilities

Control tone and rewriting objectives

This satisfies the requirement that LLM behavior is guided by classical NLP outputs.

7. End-to-End Interaction Loop

The system follows a multi-module pipeline:

User inputs text

Classical NLP classifiers analyze the text

Probabilistic summaries are generated

GPT receives classifier outputs as context

GPT generates explanations or rewrites

Rewritten text is re-evaluated by classifiers

This loop allows users to actively test detector robustness and observe how stylistic changes affect predictions.

8. Ethical Considerations

AI content detection raises several ethical concerns:

Bias: Classifiers may associate formality or vocabulary richness with AI authorship.

False Positives: Human-written academic text may be misclassified as AI-generated.

Misuse: Adversarial rewriting tools could be misused to evade detection systems.

This project addresses these issues by emphasizing transparency, explanation, and experimentation rather than enforcement.

9. Limitations and Future Work

Limitations

Detection accuracy decreases for short or heavily edited texts.

Feature-level interpretability (e.g., top TF-IDF tokens) is not displayed in the CLI.

The system does not claim definitive AI detection, only probabilistic analysis.

Future Work

Visualization of embedding spaces

Integration of attention-based models

Web-based UI

Cross-model ensemble detection

Improved feature attribution explanations

10. Conclusion

This project demonstrates a complete AI Content Forensics system that integrates classical NLP, neural models, and modern LLMs. By combining detection, explanation, rewriting, and adversarial testing, the system goes beyond static classification and offers an interactive exploration of AI-generated text. The project highlights both the power and limitations of AI detection, making it a practical and educational contribution to modern NLP studies.

Author

Hastin Ghasadiya

NLP Final Project — Intelligent Language Systems