

Group 25 Progress Report: AI Essay Detector

Fei Xie, Ghena Hatoum, Haniye Hamidizadeh
`{xief17,hatoumg,hamidizf}@mcmaster.ca`

1 Introduction

The widespread and accessible nature of Generative AI (GenAI) tools, such as ChatGPT and Gemini, has created a significant shift in the educational landscape. While these tools are valuable for quick information retrieval, their rapid adoption precedes the necessary adaptations in academic practices. The core issue is that unchecked reliance on GenAI for academic work can have long-term developmental consequences for students. It prevents them from engaging in the process of formulating original thoughts and arguments, which is critical for developing their unique intellectual voice. A reliable AI Text Detector, integrated within the education system, would allow educators to foster responsible use of GenAI as an educational aid rather than a replacement for student effort.

Developing an effective detector is challenging for several reasons:

- Low Accuracy of Existing Models: Current models are not accurate enough for real-world academic use.
- Evolving Realism: GenAI systems are constantly improving, generating increasingly human-like text, requiring the detector to be dynamically updated.
- Critical False Positive Rate: The model must achieve a near-zero false positive rate to ensure no student is wrongfully accused of plagiarism while maintaining high accuracy in identifying genuine misuse.

2 Related Work

Detecting AI-generated text is closely connected to research in text classification. An overview of how methods such as TF-IDF have been used in traditional machine-learning models before the development of deep-learning techniques as seen in

the survey “A Survey on Text Classification: From Traditional to Deep Learning” ([Li et al., 2022](#)). In practical applications, AI text detection has been explored through competitions, such as the kaggle AI Text Detection Competition ([Al-Ahmadi](#)). Public notebooks from the competition show examples of baseline systems that use TF-IDF combined with classifiers such as Logistic Regression or SVM. Two examples are the notebooks by Faris Al-Ahmadi ([Al-Ahmadi](#)) and Jasmine Mohamed ([Mohammad](#)), where TF-IDF is used to represent essays numerically before training classification models.

More recent work focuses on neural network approaches. LSTM models are often used for text classification tasks because they can process text as a sequence and keep track of earlier context in the input. An example of this is shown in an LSTM text-classification notebook ([Khotijah](#)). Transformer-based models such as DistilBERT have also been applied to similar classification problems. The KerasNLP starter notebook from the same Kaggle competition shows how DistilBERT can be fine-tuned for detecting AI-generated text ([Audevert](#)).

There is also research specifically focused on identifying patterns unique to AI-generated writing. The “DetectGPT” paper ([Mitchell et al., 2023](#)) shows that AI-generated text often has different patterns in how words are used compared to human writing and discusses ways to detect those differences.

3 Dataset

We will be using the DAIGT V2 Train Dataset from Kaggle ([Klecze](#)). This comprehensive dataset is a collection of other datasets. One of the sub datasets contain argumentative essays written by grade 6 to 12 students. The remaining sub datasets contain AI-generated essays from numerous models as seen in Table 1.

The dataset is prelabelled and additional columns include prompt_name (original persuade prompt), source (original dataset), text (actual text content), and RDizzl3_seven (Classifier built for a previous Kaggle competition, indicating whether the essays were written in response to one of the seven essay prompts for the competition).

Our team has dropped all the features other than text and the label.

3.1 Preprocessing

Using the text feature, we have extracted additional features. These include:

Word Statistics

The features include the total number of words in the text/document, the sum of words that repeat frequently (greater or equal to 5 times), and the sum of words that appear infrequently (less than 5 times). All these features are integers.

Punctuations Statistics

The features include the number and ratios of the following punctuations: [', -, ;, :, ., !, ?, (,), [,], {, }, /, ", ', _]. The model incorporates the percentage of characters that are one of the 17 different punctuation marks shown above. These features are integers. It also includes the total punctuation percentage, which is the sum of these individual percentages. These values are between 0 and 1.

TF-IDF Vectorizer

The model uses the TF-IDF vectorizer provided by ski-kit as another feature set. Term frequency (TF) measures how often a word appears in a specific document as shown in Equation 1. A higher frequency suggests greater importance within that document. Inverse document frequency (IDF) captures the term popularity as an inverse of the overall corpus as shown in Equation 2. Our TF-IDF matrix has a max of 10000 features.

4 Features

Our detection strategy relies entirely on feature engineering and makes no use of pre-trained language models or learned embeddings. The features are explicitly designed to capture distinct characteristics of the text: structural patterns, writing complexity, and vocabulary.

The model input, represented by the matrix X, is a concatenation of three separate, hand-engineered feature sets:

Structural Features (Punctuation Metrics): These

18 features (17 individual punctuation ratios + total percentage) capture the fine-grained writing style. This set is dense and low-dimensional.

Complexity Features (Word Statistics): These 3 features (total word count, high-repeat word count, uncommon word count) attempt to measure stylistic complexity and originality. This set is also dense and low-dimensional.

Vocabulary Features (TF-IDF): This is the high-dimensional representation of the text. It converts the actual words and two-word phrases (unigrams and bigrams) into 10,000 numerical weights, with higher weights indicating a word is both frequent in a document and rare across the entire dataset. This set is sparse (mostly zeros), making it memory-efficient.

The combined matrix X (with 10,021 total features) is fed directly into the machine learning pipeline, where the classifier learns the relationship between these feature values and the binary label (Human/AI).

5 Implementation

The process of building the model is divided into three logical phases: Feature Preparation, Data Combination, and Training Pipeline & Optimization.

As described earlier, the initial step is to calculate three distinct types of features from the text data: Word Statistics, Punctuation Metrics, and the TF-IDF Vectorizer. This preparation phase focuses solely on generating these numerical values.

After feature generation, the data is prepared for the classifier.

The calculated features are combined into a single, comprehensive input matrix. Because the TF-IDF matrix is typically sparse (mostly zeros), a specialized function checks if any feature group is sparse. If so, it converts all other feature groups (like Punctuation Ratios) to a sparse format before stacking them together. This ensures memory efficiency when dealing with the large number of TF-IDF features.

The full dataset is then split, reserving 11.1% of the data as an independent test set.

The final step is to build and train the classification model:

1. Pipeline Construction: A sequential pipeline is created to automate the workflow.
2. Data Scaling: The first step in the pipeline is

to standardize the features by scaling their values. We skip centering the data (ignoring the mean) before scaling. This is done to maintain the efficiency of the sparse TF-IDF matrix, which is a common practice in natural language processing when using sparse vectorizers.

3. Model Training: A Logistic Regression classifier is trained on the scaled training data. The model is configured to run for a maximum of 200 iterations and utilizes all available CPU cores for faster parallel processing. The model fits its parameters to the training set by minimizing the underlying loss function.

The model trained using only Punctuation Metrics serves as our Official Baseline. Any subsequent model must outperform this 83.00% accurate baseline to be considered a valuable improvement.

From these results we can see that the TF-IDF is the key predictor. We see that model using TF-IDF and Word Counts and the model using TF-IDF only have similar performance and that TF-IDF model achieves a near-perfect performance. This shows that it's overwhelmingly the most critical element. We can also see that adding more features leads to negligible changes.

6 Results and Evaluation

To evaluate our model, we split the data into training, validation, and test sets using an 80/10/10 split. The validation set allows us to monitor how well the model is learning during training, and the test set provides a final measure of performance on data the model has not seen before. For evaluation, we use both Accuracy and F1 Score. Accuracy indicates how often the model predicts the correct label overall, while the F1 Score reflects how well the model balances detecting AI-generated essays without incorrectly flagging human-written ones.

We began by establishing baselines using different subsets of features. The first baseline used only punctuation-based features (for example, the relative frequency of commas, periods, and other punctuation marks). This model achieved approximately 0.83 accuracy and around 0.77 F1 on the test set. This suggests that while punctuation usage differs to some extent between human and AI-generated essays, punctuation alone is not sufficient for reliable classification.

Next, we trained a model using only TF-IDF with N-grams, which represents each essay based

on how strongly certain words and short phrases appear relative to the rest of the dataset. This model performed very well, achieving approximately 0.997 accuracy and about 0.997 F1 on the test set. We then combined TF-IDF with punctuation-based features and word-count-related features. The results were effectively the same as using TF-IDF alone, indicating that most of the meaningful signal for distinguishing between human and AI-written text is captured through the phrasing and word usage patterns identified by TF-IDF.

The model performance metrics are summarized in Table 2, where the full set of results is available. These results suggest that differences in expression style and repeated phrasing patterns play a significant role in distinguishing AI-generated writing. Human authors tend to show greater variation in sentence structure and vocabulary, while AI models may reuse similar transitions or stylistic patterns. Since TF-IDF highlights which words and phrases are most characteristic within each essay, it was able to capture these distinctions effectively, which explains the strong performance of the TF-IDF-based model.

7 Feedback and Plans

We have received relatively good feedback from the TA with our current implementation of using logistic regression. Even with our accuracy already being quite high, in the high 90s, we discussed with the TA to implement our model as a neural network instead. We will update our model to a Long Short Term Memory network. This will allow us to utilize our initial text feature, that we weren't able to use beforehand with a simple logistic regression approach. If our team have time, we will also investigate using a transformer, such as DistilBERT (Sanh et al., 2020), for text classification. Based on the suggestion from the TA, we have updated our model training process by splitting up our dataset into training, validation, and test sets, as opposed to only having a training and test set. Our next step is to add k-fold cross validation to our three sets, this will allow us to detect any overfitting of our model and finetune the hyperparameters. Other potential next steps is to increase our dataset's sample with more AI generated essays from new models, such as the GPT5 family from ChatGPT and model 4 family from Claude. This will allow us to judge our models accuracy, as cur-

rent AI generated samples were from older models, allowing us to determine if certain characteristics that were present in older models, such as the excessive use of certain punctuations were over-contributing to our model predictions.

8 Team Contributions

Fei: Setup and initial implementation of the model, feature engineering for feature sets Word Statistics and Punctuation Statistics excluding TFIDF. Wrote the Dataset and Feedback and Plans sections for the progress report, and ported over the Introduction and Related work sections from team's word document.

Haniye: Improved the feature processing by generating the TF-IDF representation with N-grams and saving the vectorizer to a pickle file so it can be reused without retraining. Also optimized the punctuation feature extraction and updated the data split to include separate train, validation, and test sets. Wrote the Related Work and Results/Evaluation sections.

Ghena: Added the punctuation function to the model, wrote up the implementation, preprocessing and feature sections.

References

- Faris Al-Ahmadi. Ai text detection competition. <https://www.kaggle.com/code/farisalahmdi/ai-text-detection-competition>. Accessed: 2025-08-03.
- Alexia Audevert. Kerasnlp starter notebook llm - detect ai generate. <https://www.kaggle.com/code/alexia/kerasnlp-starter-notebook-llm-detect-ai-generate>. Accessed: 2025-08-03.
- Nicholas Broad. 2023a. Daigt data - llama 70b + gpt4 + falcon180b dataset. <https://www.kaggle.com/datasets/nbroad/daigt-data-llama-70b-and-falcon180b>. Accessed: 2025-11-08.
- Nicholas Broad. 2023b. Persuade corpus 2.0. <https://www.kaggle.com/datasets/nbroad/persuade-corpus-2/>. Accessed: 2025-11-08.
- Darragh Hanley. 2023. 1000 essays from anthropic claudie. <https://www.kaggle.com/datasets/darraghdog/hello-claude-1000-essays-from-anthropic>. Accessed: 2025-11-08.
- Siti Khotijah. Using lstm for nlp: Text classification. <https://www.kaggle.com/code/khotijahs1/using-lstm-for-nlp-text-classification>. Accessed: 2025-08-03.
- Darek Kleczek. Daigt v2 train dataset. <https://www.kaggle.com/datasets/thedrcat/daigt-v2-train-dataset/data>. Accessed: 2025-08-03.
- Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S. Yu, and Lifang He. 2022. **A survey on text classification: From traditional to deep learning.** *ACM Trans. Intell. Syst. Technol.*, 13(2).
- Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. **Detectgpt: Zero-shot machine-generated text detection using probability curvature.** *Preprint*, arXiv:2301.11305.
- Jasmine Mohammad. Detecting ai-text(svm, nn) + full ml guide. <https://www.kaggle.com/code/jasminemohamed2545/detecting-ai-text-svm-nn-full-ml-guide>. Accessed: 2025-08-03.
- Radek Osmulski. 2023. Llm generated essays. <https://www.kaggle.com/datasets/radek1/llm-generated-essays>. Accessed: 2025-11-08.
- Alejo Paullier. 2023. Daigt external dataset. <https://www.kaggle.com/datasets/alejopaullier/daigt-external-dataset>. Accessed: 2025-11-08.
- Muhammad Rizqi. 2023. Llm-generated essays using palm. <https://www.kaggle.com/datasets/kingki19/llm-generated-essay-using-palm-from-google-gen-ai>. Accessed: 2025-11-08.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. **Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.** *Preprint*, arXiv:1910.01108.

Tables and figures

See Table 1

Equations

$$TF(w, d) = \frac{\text{count}(w, d)}{\text{total words in } d} \quad (1)$$

$$IDF(w, C) = \log \left(\frac{|C|}{|\{d \in C : w \in d\}|} \right) \quad (2)$$

Dataset Name	Description
Persuade Corpus 2.0 (Broad, 2023b)	Provides argumentative essays produced by 6 to 12 grade students. It was created by The Learning Agency and Vanderbilt University, originally pulled from the following GitHub repository.
ChatGPT (Paullier, 2023)	Contains 2.5k student written texts sourced from the FeedBack Prize 3 Competition, and 2.5k AI-generated texts using ChatGPT. The compiled dataset includes only AI-generated texts and prompts.
Llama 70b + GPT-4 (Broad, 2023a)	Contains 9k essays generated by Llama 70b and Falcon 180b. Prompts come from the Persuade Corpus and GPT4, using a total of 35 prompts for generation.
LLM Generated Essays (Osmulski, 2023)	Contains 700 essays generated by LLMs 500 from GPT3.5Turbo and 200 from GPT4.
Claude Essays (Hanley, 2023)	Contains 1000 essays generated by Claude-Instant-1 using 15 prompts from the Persuade Corpus. Prompts were sourced from the competition discussion.
PaLM Essays (Rizqi, 2023)	Contains 1384 essays generated by PaLM. Prompts were sourced from a Kaggle competition notebook.

Table 1: Summary of Datasets Used in the DAIGT V2 Train Dataset

Table 2: Performance Accross Different Feature Sets

Features Used	Total Features	Test Accuracy	Test F1 Score
Punction (Baseline)	18	0.83	0.7758
TF-IDF	10 000	0.9973	0.9966
Word Counts + TF-IDF	10 003	0.9973	0.9966
All features	10 021	0.9971	0.9963