CPSC 552 Project Report
Detecting Human-AI Generated Texts: A Multi-Method Classification Approach with Compact Style Embeddings

Shurui Wang(sw2349)
shurui.wang@yale.edu

Alex Wang(ww445)
alex.wang.ww445@yale.edu

Lang Ding(ld698)
lang.ding@yale.edu

Abstract

This project addresses the challenge of distinguishing texts generated by humans, artificial intelligence (AI), or a combination of both. We introduce a high-performance model that leverages style embeddings from pre-trained lightweight models, such as DistilBERT and MobileBERT, to analyze authorship based solely on text style, independent of content. The model uses a triplet loss function to enhance the differentiation of stylistic features. Our datasets include a mixture of human-generated, AI-generated, and mixed-origin texts, allowing us to train and test our model effectively. We evaluated our approach using various classification heads, including KNN, SVMs, Logistic Regressions and Decision Trees, and a deep learning Multilayer Perceptron (MLP). Our findings reveal that while MLP offers high accuracy, traditional machine learning classifiers provide competitive performance with lower computational demand. Our project contributes a scalable and efficient solution for the real-time detection of text authorship, enhancing the integrity of content across various digital platforms. Code is available at $https://github.com/JadenWSR/CPSC552\_Final\_Project$.Video explaining out project can be found at $https://www.youtube.com/watch?v = sOyAG4tG4jc$

1 Introduction and Motivation

The increasing sophistication of AI-generated texts presents significant challenges in distinguishing them from human-generated content, especially in contexts requiring the integrity of authored works, such as academic writing, legal documentation, and online media. The ability to accurately identify the origin of text—whether it be human, AI, or a combination thereof—is essential for preventing plagiarism, securing the integrity of information, and curating high-quality datasets for training large language models. However, many current models that offer high accuracy in detection often require substantial computational resources, are slow to process, and primarily focus on clear-cut cases of authorship, neglecting the more prevalent and complex real-world scenarios of mixed authorship.

In response, our project develops a rapid, high-performance model capable of identifying whether text is generated by humans, by AI, or is a product of both. By redefining the challenge as an extension of the broader authorship verification issue, we aim to discern the source of text without the traditional dependencies on heavy hardware. Utilizing content-independent style embeddings, our model learns to capture unique authorship features from the style of texts alone, irrespective of content. This approach enables us to deploy efficient machine learning classifiers on top of the style embeddings, facilitating faster, more scalable, and practical detection.

2 Background and Related Work

In the evolving field of AI-generated text detection, initial frameworks like DetectGPT[7] have introduced zero-shot detection methods based on probability curvature, which significantly enhanced the ability to discriminate between human and AI-generated content, particularly in fake news. GPTZero[10], meanwhile, uses both zero-shot and supervised techniques to focus on changes in statistical properties to determine AI involvement in text creation. These methods have proven effective but primarily target clear-cut cases of AI authorship, neglecting more nuanced scenarios of mixed authorship where contributions from both humans and AI are interwoven.

Furthermore, Lazebnik and Rosenfeld [6] explored the limitations of these detection strategies, which often fail to recognize subtle shifts in writing style indicative of mixed authorship. Recent advancements, such as those by Wegmann et al. [11], propose using style embeddings to achieve a more refined analysis of authorship that goes beyond content, which addresses these limitations by capturing stylistic nuances independent of the text content.

Despite these technological strides, current detectors such as GPTZero[10] and DetectGPT[7] are computationally intensive and less effective in scenarios involving mixed authorship. Our approach aims to bridge this gap by incorporating a trained style embedding model with various classification techniques—both deep learning (e.g., MLP [4]) and traditional machine learning (e.g., SVMs [1], Decision Trees [12], logistic regression [2], KNN [3]). To enhance the efficiency and scalability of our model, we base our style embeddings on pre-trained lightweight models, specifically DistilBERT [8] and MobileBERT [9]. These models provide a balance between performance and computational resource requirements, which makes them ideal for real-time applications. This integration promises enhanced detection capabilities with greater efficiency and adaptability in distinguishing text origins across different authorship contexts.

3 Dataset

To address the challenge of text source detection, we utilize two distinct datasets. The first, known as the Augmented Data for LLM from Kaggle[5], comprises 433,564 texts, labeled as either AI-generated or human-generated, drawing from the most advanced AI models like GPT and Claude. This dataset is already partitioned into an 80/20 training and testing split. The second dataset, MixSet[13] from GitHub, includes a more diverse collection of texts—3,600 entries distributed across 12 JSON files, featuring human-written, AI-generated, and mixed human-AI texts. Due to the limited size of MixSet, we enhance it by integrating samples from the Augmented Data for LLM, creating a balanced dataset with three categories: pure human, pure AI, and mixed texts. This balanced dataset comprises 7,500 entries for training and 1,500 for testing, with a further 5% of the training set reserved as a validation set.

4 Method

4.1 Model Framework and Mathematical Setting

Our method can be divided into two fundamental components as detailed in Algorithm 1. After preparing the dataset, we utilize pre-trained lightweight base models, DistilBERT and MobileBERT, to initialize our Style Embedding Model. This model adapts the technique outlined by Wegmann et al.[11] and is trained using a Triplet Loss function, which aims to optimize the embeddings for textual data within the input dataset. The triplet loss function is defined as $L(A, B, C) = \max(\|f(A) - f(B)\|_2 - \|f(A) - f(C)\|_2 + \alpha, 0)$, where α is a margin that enhances stylistic differentiation. The entire training and embedding generation process takes approximately one hour, after which we can swiftly apply various classification heads—MLP, KNN, SVM, Decision Tree, and Logistic Regression—to the style embedding results, obtaining results within seconds. This efficient workflow allows for the reuse of the style embedding model across different tasks by merely switching the classification heads, thereby enhancing flexibility and conserving computational resources.

However, during the integration of these models into our workflow, we encountered technical challenges that necessitated further adaptations. Specifically, when transitioning the base model from MobileBERT to DistilBERT, we faced the issue of the absence of a ‘pooler_output‘ in DistilBERT. In

MobileBERT, this is used to obtain a fixed-size sentence-level embedding from the [CLS] token. This output in MobileBERT is processed through an additional dense layer and a non-linear activation (tanh). To adapt our Style Embedding Model for DistilBERT, we introduced a manual replication of this process. Specifically, we added a linear layer followed by a tanh activation function to the [CLS] token's representation from DistilBERT's last hidden state. This modification allowed us to maintain a consistent output format and comparable performance in our triplet loss-based tasks, despite the architectural differences between the two models.

Algorithm 1 Training and Classification Using Triplet Style Embedding Model and Multiple Classification Heads

```
 1: Input: Accept input data in a pandas DataFrame with columns Text and Label.
 2: Load a pre-trained tokenizer appropriate for DistilBERT or MobileBERT.

 3: procedure EMBEDDING MODEL TRAINING
 4:     Initialize TripletTextDataset:
 5:     Tokenize each text entry up to a maximum length.
 6:     Map each text label to an integer.
 7:     Organize indices of texts by their labels for sampling.
 8:     Triplet Sampling: For each text (anchor), randomly select another text (positive) with the
        same label and one text (negative) with a different label.
 9:     Define Model: Initialize DistilBERT or MobileBERT as basemodel.
10:     Compute Loss and Backpropagate: Calculate triplet loss and backpropagate errors to
        optimize embeddings. Use Adam optimizer with learning rate of $1e - 4$ and weight decay of
        $0.01$. Define triplet loss as $L(A, B, C) = \max(\|f(A) - f(B)\|_2 - \|f(A) - f(C)\|_2 + \alpha, 0)$,
        where $\alpha = 0.5$.
11: end procedure

12: procedure APPLYING CLASSIFICATION HEADS
13:     Extract Embeddings: Pass all dataset texts through the trained model to obtain style
        embeddings.
14:     Train Classifiers: Train classifiers such as MLP, KNN, SVM, Decision Tree, and Logistic
        Regression using embeddings.
15:     Classify: Use trained classifiers to predict new text labels based on embeddings.
16: end procedure
```

4.2 Custom TripletTextDataset Dataset

To support our style embedding model, detailed in the previous section, we developed the TripletTextDataset class, which was specifically designed to improve the model's proficiency in distinguishing texts originated by humans, AI, or a combination of both, primarily through stylistic differences. This class organizes the data into triplets, where each triplet set comprises an anchor sample (A1) from any category, a positive sample (A2) from the same category but differing in content, and a negative sample (B) from a different category. This setting is designed to maximize the learning of stylistic nuances by contrasting similar and dissimilar text samples. Formally, a triplet in our dataset is defined as $(x^{(a)}, x^{(p)}, x^{(n)})$ where $x^{(a)}$ and $x^{(p)}$ belong to the same class but differ in content, and $x^{(n)}$ is from a different class. The inclusion of all text features, including stopwords, ensures that full stylistic integrity is maintained for accurate style-based classification.

5 Results

Lang Ding focused on merging datasets, designing the Triplet Dataset, fine-tuning DistilBERT model, and running all experiments with DistilBERT model. Shurui Wang focused on creating the style embedding model, setting up the machine learning structures, fine-tuning MobileBERT model, and running all experiments with MobileBERT model. Alex Wang focused on proposing the theoretical framework, picking and running baseline models, and conducting conclusion analysis.

5.1 Style Embedding Model

After transferring our data to the custom triplet dataset class, we trained the style embedding model using two base models for ten epochs, following the same hyperparameters as described in Wegmann et al.[11]: a batch size of 8, with 10% of the training data allocated for warm-up steps, the Adam optimizer with a learning rate of 0.00002 and a weight decay of 0.01, and triplet loss with a 0.5 margin. The resulting style embeddings for each text are 768-dimensional vectors. Once the contrastive triplets were generated, the model required less than one hour to train on a GPU. Additionally, it took each model about 30 minutes to generate the training, validation, and testing embeddings for the classification heads in the subsequent section.

Table 1: Performance Comparison of Style Embedding Models with Different Base

Base Model	Total Params	Model Size	Test Accuracy	Test Loss	Val Loss	Epochs
MobileBERT	24.6M	98.328MB	**0.9640**	34.442	10.967	10
DistilBERT	67.0M	267.814MB	0.9567	**0.0548**	**0.052**	4

To compare MobileBERT and DistilBERT as base models for our style embedding models, we used several evaluation metrics during model training, validation, and testing. MobileBERT exhibited higher test accuracy (0.9640 vs. 0.9567) but performed worse in terms of test loss (34.442 vs. 0.0548) and validation loss (10.967 vs. 0.052) compared to DistilBERT. DistilBERT also demonstrated faster convergence, requiring only 4 epochs to achieve optimal validation loss with early stopping, while MobileBERT needed 10 epochs (the maximum setting in our experiment). Additionally, the training time for DistilBERT was significantly shorter, about 25 minutes on a GPU due to early stopping, whereas MobileBERT took approximately 1 hour for its training cycle. Despite MobileBERT's smaller model size (98.328 MB vs. 267.814 MB for DistilBERT), DistilBERT's superior loss metrics and training efficiency make it a more effective choice in resource- and time-constrained environments. However, when a smaller model size is required, MobileBERT remains a feasible option, offering slightly inferior performance but an acceptable training duration.

5.2 Human-AI Generated Texts Detection With Multiple Classification Heads

Table 2: Comparison of two Sets of Detection Models with Different Classification Heads

Method	**MobileBERT**		**DistilBERT**	
	Validation Accuracy (%)	Test Accuracy (%)	Validation Accuracy (%)	Test Accuracy (%)
Baseline		**33.33**		**33.33**
Deep Learning Approaches				
MLP	90.40	79.00	98.67	**95.67**
Traditional Machine Learning Methods				
KNN	97.60	93.40	98.40	96.13
SVM	–	94.20	–	96.13
Decision Tree	98.04	94.47	97.33	95.33
Logistic Regression	98.13	94.53	97.33	96.07

**Methods not reporting validation accuracy may have incorporated the validation data during model training or parameter optimization.*

In our experimental setup, machine learning (ML) classification heads, such as SVM and Decision Tree, were fine-tuned using a grid search approach with 5-fold cross-validation to optimize their hyperparameters, ensuring robustness and reliability. For the deep learning (DL) classification head, we implemented a Multilayer Perceptron (MLP) model using the PyTorch framework. This model consists of an input layer that matches the size of the input style embeddings, a hidden layer with 100 neurons, and an output layer designed for three classes. We employed a ReLU activation function, an Adam optimizer with a learning rate of 0.001, and a cross-entropy loss function to optimize classification accuracy. The same set of training, validation, and testing data was used across all experiments.

Given this consistent experimental setup, our comparative analysis of detection models using MobileBERT and DistilBERT as base models revealed varying performance between DL and ML classification heads. For MobileBERT, ML methods outperformed the MLP approach, achieving higher test accuracies (around 94%-95% for all ML methods) compared to the MLP's 79.00%. This superiority of ML methods extends to computational efficiency, as they require less computational power and train significantly faster than their DL counterparts. In contrast, when employing DistilBERT as the base model, the MLP approach demonstrated slightly better performance, achieving the highest test accuracy of 95.67% in fewer than 20 epochs. Interestingly, the training time for MLP was comparable to that of ML methods in this setting. It takes about 30 seconds with a GPU, which is even faster than performing a grid search for some ML methods. This suggests that while ML methods generally require fewer computational resources and offer competitive performance, the choice between DL and ML classification heads may depend on the specific base model used. Furthermore, ML can be a computationally inexpensive yet preferable choice for this task.

6 Conclusion & Discussion

In our project, we developed a novel framework for detecting the origin of texts—whether they are generated by humans, artificial intelligence, or a combination of both—using compact style embeddings derived from lightweight models like DistilBERT and MobileBERT. The evaluation of our model shows that while deep learning classifiers such as Multilayer Perceptrons (MLPs) achieve high accuracy, traditional machine learning classifiers like SVMs and Decision Trees also provide competitive performance with substantially lower computational demands. This balance between performance and efficiency highlights our model's potential in real-time applications where rapid processing is crucial.

Looking ahead, there are several directions for future research that could potentially enhance the robustness and applicability of our model. First, incorporating additional hidden layers or exploring alternative neural network architectures could help in capturing more complex stylistic nuances, potentially improving the model's accuracy beyond the current 95%. Testing our model across a broader range of datasets, especially those that feature subtle distinctions between human and AI-generated texts, could further validate its effectiveness across different contexts.

Additionally, implementing this model in real-time settings, such as monitoring systems for social media platforms or academic review systems, could provide practical insights into its performance and utility in dynamic environments. Collaborating with experts in linguistics and cognitive science might also uncover deeper linguistic features that distinguish between human and AI texts, which could enhance the sensitivity and specificity of our detection methods. These steps would not only aim to refine the accuracy of our model but also broaden its implications, ensuring that it can effectively contribute to safeguarding the integrity of digital content across various platforms.

References

[1] Cortes, C. and Vapnik, V. [1995], 'Support-vector networks', *Machine learning* **20**(3), 273–297.

[2] Cox, D. R. [1958], 'The regression analysis of binary sequences', *Journal of the Royal Statistical Society: Series B (Methodological)* **20**(2), 215–232.

[3] Fix, E. and Hodges, J. L. [1951], Discriminatory analysis. nonparametric discrimination: Consistency properties, Report, USAF School of Aviation Medicine, Randolph Field, Texas. Available as PDF.

[4] Haykin, S. [1994], *Neural networks: a comprehensive foundation*, Prentice Hall PTR.

[5] Herrera, J. [2024], 'Augmented data for llm - detect ai generated text'. Kaggle dataset.
URL: *https://www.kaggle.com/datasets/jdragonxherrera/augmented-data-for-llm-detect-ai-generated-text*

[6] Lazebnik, T. and Rosenfeld, A. [2024], 'Detecting llm-assisted writing in scientific communication: Are we there yet?'.

[7] Mitchell, E., Lee, Y., Khazatsky, A., Manning, C. D. and Finn, C. [2023], 'Detectgpt: Zero-shot machine-generated text detection using probability curvature'.

[8] Sanh, V., Debut, L., Chaumond, J. and Wolf, T. [2020], 'Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter'.

[9] Sun, Z., Yu, H., Song, X., Liu, R., Yang, Y. and Zhou, D. [2020], 'Mobilebert: a compact task-agnostic bert for resource-limited devices'.

[10] Tian, E. and Cui, A. [2023], 'Gptzero: Towards detection of ai-generated text using zero-shot and supervised methods'.
URL: *https://gptzero.me*

[11] Wegmann, A., Schraagen, M. and Nguyen, D. [2022], Same author or just same topic? towards content-independent style representations, *in* 'Proceedings of the 7th Workshop on Representation Learning for NLP', Association for Computational Linguistics, Dublin, Ireland, pp. 249–268.
URL: *https://aclanthology.org/2022.repl4nlp-1.26*

[12] Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Philip, S. Y. et al. [2008], 'Top 10 algorithms in data mining', *Knowledge and information systems* **14**(1), 1–37.

[13] Zhang, Q., Gao, C., Chen, D., Huang, Y., Huang, Y., Sun, Z., Zhang, S., Li, W., Fu, Z., Wan, Y. and Sun, L. [2024], 'Llm-as-a-coauthor: Can mixed human-written and machine-generated text be detected?'.