Fast, Interpretable AI-Generated Text Detection Using Style Embeddings

Stanford CS224N Custom Project

Giulia Socolof
Department of Computer Science
Stanford University
`gsocolof@stanford.edu`

Ritika Kacholia
Department of Computer Science
Stanford University
`ritikak@stanford.edu`

Abstract

As the prevalence of AI-generated text grows, it becomes increasingly necessary to detect when text is AI-generated to maintain accountability, avoid plagiarism, and many other reasons. In this project, our goal is to target style embeddings as a core component to (1) further pretrain an existing style embedding model on human and AI-written texts to detect whether the author of a text was a human or an AI, and (2) to explore adversarial settings by determining whether a model can detect and accurately classify mixed human-AI text. We implement three simple classification heads on top of the embedding model: a Multi-Layer Perceptron, K-Means Clustering, and K-Nearest Neighbors. Our findings reveal that we learn a strong representation of human, AI, and mixed text in embedding space and in natural language, outperform baselines on classification, and all with a model that is more understandable and more efficient than current detection methods.

1 Key Information

- CS224N Mentor: Tathagat Verma
- External Collaborators: Abhay Singhal
- Contributions: Giulia and Ritika both worked on running experiments (baselines, mixset, etc) and writing papers together. Giulia implemented the continued pretraining of the style embedding model and Ritika implemented the data processing/testing of the mixset dataset class, getting inputs from each other throughout the whole process. We credit our external collaborator Abhay Singhal (and his original project partner, Daniel Ma) for the initial style embedding idea, implementation of experimental infrastructure, and DetectGPT baseline.
- Our working branch of the GitHub repository can be found here: generally treat commits from either Giulia or Ritika as joint, since we often work off VSCode LiveShare together.

2 Introduction

As the prevalence of AI-generated text grows, it becomes increasingly necessary to detect when a piece of text is AI-generated to maintain trust and accountability, avoid plagiarism, identify high-quality data for further large language model (LLM) training, and generally attribute authorship and provenance correctly. The space of these models is therefore quite large, but many of the models achieving state-of-the-art accuracy (>90 %) require expensive hardware to compute, lack interpretability (especially to the average human who is being accused of plagiarism by one of these models), and/or are slow to evaluate. Additionally, little research has been conducted into detecting the difficult cases where a piece of text is created with some human and some AI involvement, which is the most common use case in the real world.

In this paper, we create a model to detect whether a piece of text was written by a human, by an AI, or a mixture; this model is **high-performing**, **interpretable**, and **fast**. We frame the AI text detection problem as an extension of the more general authorship verification problem, where the goal is to determine which author in a set of known authors wrote a text. As a core component, we target content-independent style embeddings, as introduced in Wegmann et al. (2022). The training procedure compares and contrasts authors of texts while controlling for the content of the text to learn a representation of authorship in embedding space. We can then train simple neural classification heads on top of the embedding model.

3 Related Work

There has been some interest in using style as a metric for assessing language model involvement in text generation, such as in Lazebnik and Rosenfeld (2024), which builds upon a style model from Lazebnik and Rosenfeld (2023) to predict whether AI-generated text is present in scientific literature. It is also critical to distinguish style from content; though they are often correlated (the content in a scientific paper, for example, is most commonly described using formal language), it is possible to discuss the same content in many different styles. As such, we use the procedure in Wegmann et al. (2022) to achieve a robust, content-independent representation of text.

There are many models which attempt to detect AI-generated text, some which require some amount of training and some of which are entirely zero-shot. GPTZero is a prominent example which is commonly used industry and educational settings through their API (Tian and Cui, 2023), though it has sometimes been criticized for lack of interpretability and imperfect classification (Tangermann). A recent training-based model which has proven quite strong is Robust AI Detection via Adversarial Learning, or RADAR (Hu et al., 2023), which pits a paraphraser model and a detector model against each other in an adversarial training scheme. For zero-shot learning, a high-performing example is DetectGPT, which uses a proxy LLM to estimate the curvature of the probability distribution near the text (Mitchell et al., 2023). We employ RADAR and DetectGPT (actually, a faster implementation of DetectGPT known as Fast-DetectGPT (Bao et al., 2024)) as baselines.

4 Approach

Our approach has two key components. Firstly, we continue pretraining the embedding model with human and AI text. Following the approach in Wegmann et al. (2022), we also use a contrastive learning task with a triplet loss and a conversation-based content control, where we consider two utterances to be in the same conversation if they are responding to the same prompt. Each triplet includes three utterances. Utterances (A, B) are written by the same author but different content, and two others (A1, C) are discussing the same topic but have different style (here, same author is used as a proxy for similar style). The model is then trained using a triplet loss, pushing two utterances with similar style closer together and pulling apart utterances with a different style. This aims to de-emphasize the importance of content defining an author's style. The loss function can be found in equation 1, where α is the margin (Wikipedia, 2024).

$$\mathcal{L}\left(A,B,C\right) = max\left(\left\|f\left(A\right) - f\left(B\right)\right\|_2 - \left\|f\left(A\right) - f\left(C\right)\right\|_2 + \alpha, 0\right) \tag{1}$$

Secondly, by using simple neural classification heads–such as a multilayer perceptron (MLP), k-nearest neighbors (KNN), and k-means clustering–on top of the fine-tuned style embeddings to create a model with few parameters, we maintain speed and computational efficiency. The embedding model which we continued pretraining can be frozen, and the classification heads alone are used to classify on other datasets. Thus, our method is easily extendable to further datasets with minimal compute required, and as we see later, the simple classification heads are easy to understand.

5 Experiments

We describe our datasets, experiments, and details such as training time and hyperparameters.

5.1 Data

We test our implementation on four data sets. Three contain only pure human-written text (HWT) and pure machine-generated text (MGT). These are the LLM Detect AI Generated Text (DAIGT) dataset from a recent Kaggle competition (Kłeczek, 2023), the Human-ChatGPT Comparison Corpus (HC3) (Guo et al.), and Word Embedding Over Linguistic Features for Fake News Detection (WELFake) (Verma et al., 2021). The fourth is MixSet, which contains several subsets of HWT, MGT, and mixed human-AI text (Gao et al., 2024). We use a 80-20 train/test split, with the validation set being drawn from 5% of the train set. Further details of the datasets can be found in the appendix.

5.2 Experimental details

5.2.1 Continued pretraining of style embedding model

We continue pretraining of the style embedding model in (Wegmann et al., 2022) using only the texts found in the HC3 dataset. We initialize the model weights from the HuggingFace model card. We first convert the HC3 dataset to a ConvoKit Corpus (Chang et al., 2020). We then use tools in the style embeddings repository to generate 210,000 triplets in the train set, and 45,000 triplets each in the validation and test sets (the same number of triplets as in (Wegmann et al., 2022)).

We train the model for one epoch, with the same hyperparameters as in (Wegmann et al., 2022): a batch size of 8, with 10% of the training data used for warm-up steps, the Adam optimizer with a learning rate of 0.00002 and a weight decay constant of 0.01, and a triplet loss with a 0.5 margin. The resulting style embedding of each text is a 768-dimensional vector. Once the contrastive triplets were generated, the model took approximately 4 hours to train on an NVIDIA-P100 GPU.

5.2.2 Classification heads

We consider three basic classification heads: k-means clustering, k-nearest neighbors (KNN), and a multilayer perceptron (MLP). k-means and KNN are implemented with SciKit-Learn and are wrapped with SciKit-Learn's CalibratedClassifierCV, which performs Platt scaling with cross-validation on the train set to calibrate the probabilities (Pedregosa et al., 2011). We determine the human cluster in k-means by a majority vote on a held-out subset of the train set after training. KNN uses the 10 nearest neighbors for classification. The MLP has two hidden layers of size 100, uses ReLU activations, a softmax on the output, and L2 regularization with a strength of 0.01. It is optimized with Adam, with the default 0.001 learning rate, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. To account for class imbalance, particularly in MixSet, we optimize a cross-entropy loss weighted by class support.

We use 2-class classification on all datasets with all of the above classification heads. On the MixSet, we consider all pure MGT and all mixed texts to be labeled as AI-generated in one experiment. We additionally test 3-class classification on MixSet, where pure HWT and MGT are contrasted with mixed texts, with the MLP and KNN only.

Generating embeddings for all texts in e.g. HC3, which has 85,000 examples, takes approximately 4 hours on CPU (about 6 texts/second) or 20 minutes on GPU. Once the embeddings are cached, training on the classification heads is a matter of 2-3 minutes on CPU, and inference is completed extremely fast–in 0.05s for the HC3 test set, or over 500,000 texts/second. Roughly, if the embeddings were computed at inference time, inference could be completed at 6 texts/second on CPU.

5.2.3 Baselines

We compare performance with two strong, state-of-the-art baselines: Fast-DetectGPT (Bao et al., 2024), which is zero-shot, and RADAR (Hu et al., 2023), which must be trained. For DetectGPT, GPT-Neo 2.7B is used as the proxy model (Black et al., 2022). For RADAR, we finetune it for one epoch on each of HC3, DAIGT, and WELFake individually, for fair comparison with our one epoch training procedure on the style embedding model with HC3. It was trained for 3 epochs on MixSet and a selection of pure HWT and MGT, since MixSet is much smaller. We use a 85-15 train/test split and the same hyperparameters as in (Gao et al., 2024).

Finetuning RADAR on an NVIDIA-P100 took 3.5 hours. Inference is completed at approximately 15 texts/second on GPU, and would likely be significantly slower on CPU.

5.2.4 Out-of-domain testing

We compare our model's ability to generalize out-of-domain with the RADAR baseline (this is nonsensical for DetectGPT, since it is zero-shot). We train each model on HC3 and test it on DAIGT and MixSet 2-class.

5.3 Evaluation method

5.3.1 Quantitative metrics

To assess our model's performance, we use two quantitative metrics. One is the F1 score, which balances precision (true positives) and recall (number of correctly predicted positives) and is more robust than raw accuracy. The other is the ROC-AUC, or simply AUC (Area Under the Curve of a Receiver Operating Characteristic), which computes the area under the curve traced out by the accuracy at different thresholds values; it can be interpreted as the probability of correct classification. The metrics are both implemented with SciKit-Learn, weighted by class imbalance, for 2-class and multiclass.

5.3.2 Qualitative metrics

We consider two qualitative metrics as well. Firstly, to determine how well the style embedding model was able to distinguish the texts in embedding space, we plot the embeddings, using t-SNE to perform a nonlinear dimensionality reduction of the embeddings from 768-dimensional space to 2-dimensional space. Secondly, to examine the content-independent style more directly, we use the results of the KNN classifier to retrieve similar texts to a human or AI-written texts.

6 Results

We present the results of the binary classification in table 1, as compared to the baselines[1]. We bold the highest-performing detector on each metric, where we consider two detectors to be tied if the score is within ± 0.001 precision. Most of the our detectors tie or exceed the RADAR baseline, and even the KNN classifier, which just uses the Euclidean distance of the raw embeddings, is a significant improvement over DetectGPT.

Table 1: Binary Classification

Dataset	Subset	Eval	Our Model			Baseline	
			MLP	K-Means	KNN	RADAR	DetectGPT
DAIGT	all	AUC	**0.9984**	0.7786	0.9932	**0.9997**	0.8772
		F1 Score	0.9943	0.7868	0.9814	**0.9971**	
HC3	all	AUC	**0.9998**	0.9924	**0.9995**	**0.9999**	0.9711
		F1 Score	**0.9987**	0.9924	**0.9987**	**0.9998**	
WELFake	all	AUC	0.9952	0.4997	0.9945	**1.000**	
		F1 Score	0.9632	0.4933	0.9701	**0.9987**	
MixSet	2 class	AUC	0.9792	0.7564	0.9442	**0.9942**	0.582
		F1 Score	**0.9563**	0.6853	0.9535	0.876	0.635

In table 2, we examine the results of the three-class classification, as compared to the three-class classifications carried out by the authors of (Gao et al., 2024) on DetectGPT and two model-based detectors, ChatGPT Detector (Guo et al.) and Distillbert (Sanh et al., 2020) (also trained on the MixSet and some pure HWT/MGT text). The 3-class F1 score using our method vastly outperforms the baselines, but is slightly worse than the two-class classification.

[1]Our GPU was too small to actually run DetectGPT, but our external collaborator had previously computed the AUC on DAIGT and HC3, so we report that. The authors of MixSet had run DetectGPT, RADAR, and other baselines as well, so we use their results. The MixSet authors also implement the F1 score and AUC using sklearn with the same function parameters.

Table 2: MixSet 3 Class Classification {Human, AI, Mix}

Eval	Our Model		Baseline		
	MLP	KNN	DetectGPT	ChatGPT Detector	Distillbert
F1 Score	0.8710	**0.8776**	0.255	0.304	0.261

In Figure 1, it demonstrates the binary classification of each subset of the MixSet dataset. The AUC and F1 scores are computed by testing an MLP classification head on our further pre-trained embedding model. Apart from the two "MGT Adapt" subsets, our F1 scores for all other subsets are higher then all baselines presented in the Gao et al. (2024) paper.

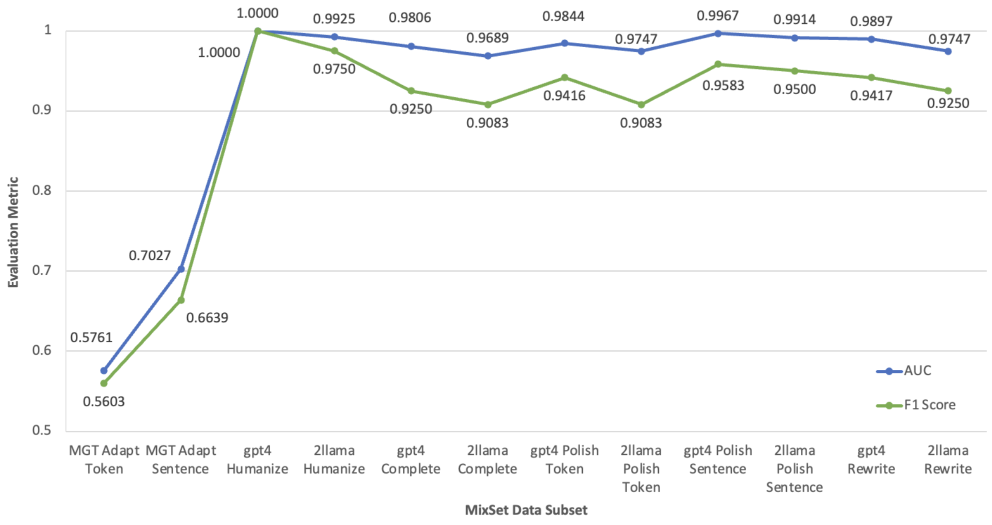


Figure 1: AUC and F1 Scores for All Subsets of MixSet

The out-of-domain test AUCs and F1 scores in table 3 are also significantly better than or equal to the baseline.

Table 3: Out-Of-Domain Test Results

Train Dataset	Test Dataset	Eval	Our Model			Baseline
			MLP	K-Means	KNN	RADAR
HC3	DAIGT	AUC	**0.8656**	0.5438	0.8252	0.8028
		F1 Score	0.7320	0.3119	**0.7503**	0.3262
HC3	MixSet 2C	AUC	0.4875	**0.5323**	0.4922	**0.5330**
		F1 Score	0.4169	**0.5811**	0.4251	**0.5474**

In Figures 2 and 3, we visualize the 2-D reduced embeddings of the MixSet, in both a 2-class and 3-class setting, and of HC3. We represent the AI generated text embeddings with red, human with blue, and mixed with purple in the 3-class case. Though there is some overlap, the AI-generated text strongly clusters to the left of the plot, whereas there is a fairly strong human cluster to the right. The mixed embeddings sit between the two in Figure 3.

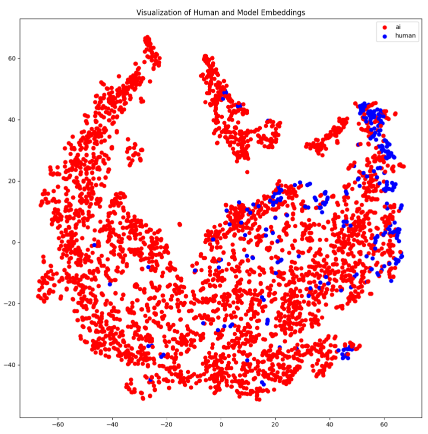


Figure 2: MixSet 2-Class Embeddings

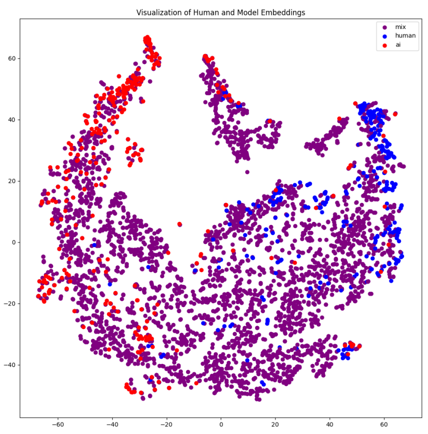
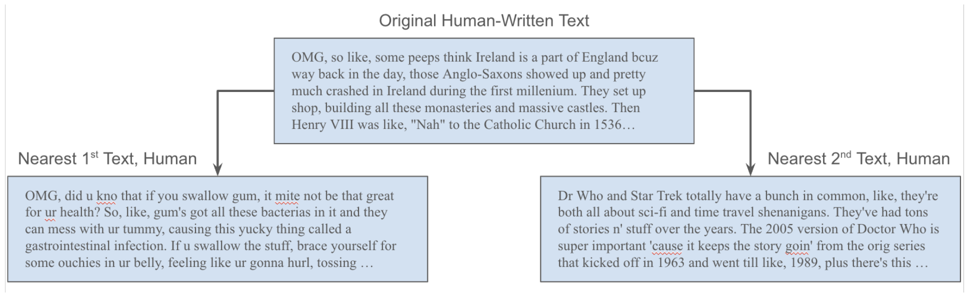


Figure 3: MixSet 3-Class Embeddings


Original Human-Written Text

OMG, so like, some peeps think Ireland is a part of England bcuz way back in the day, those Anglo-Saxons showed up and pretty much crashed in Ireland during the first millenium. They set up shop, building all these monasteries and massive castles. Then Henry VIII was like, "Nah" to the Catholic Church in 1536…

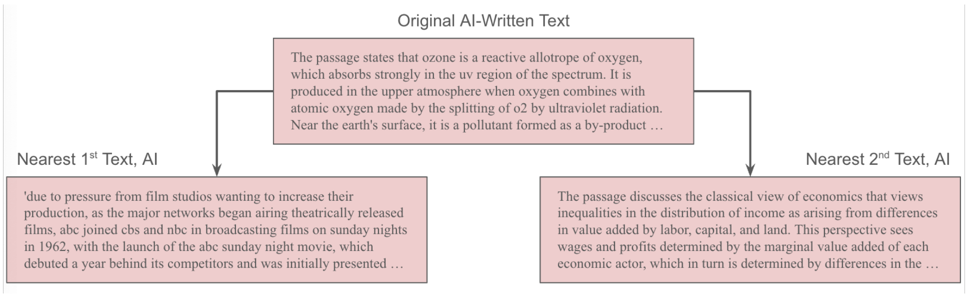
Nearest 1st Text, Human

OMG, did u kno that if you swallow gum, it mite not be that great for ur health? So, like, gum's got all these bacterias in it and they can mess with ur tummy, causing this yucky thing called a gastrointestinal infection. If u swallow the stuff, brace yourself for some ouchies in ur belly, feeling like ur gonna hurl, tossing …

Nearest 2nd Text, Human

Dr Who and Star Trek totally have a bunch in common, like, they're both all about sci-fi and time travel shenanigans. They've had tons of stories n' stuff over the years. The 2005 version of Doctor Who is super important 'cause it keeps the story goin' from the orig series that kicked off in 1963 and went till like, 1989, plus there's this …

Figure 4: Two Nearest Neighbors for a Human-Written Text


Original AI-Written Text

The passage states that ozone is a reactive allotrope of oxygen, which absorbs strongly in the uv region of the spectrum. It is produced in the upper atmosphere when oxygen combines with atomic oxygen made by the splitting of o2 by ultraviolet radiation. Near the earth's surface, it is a pollutant formed as a by-product …

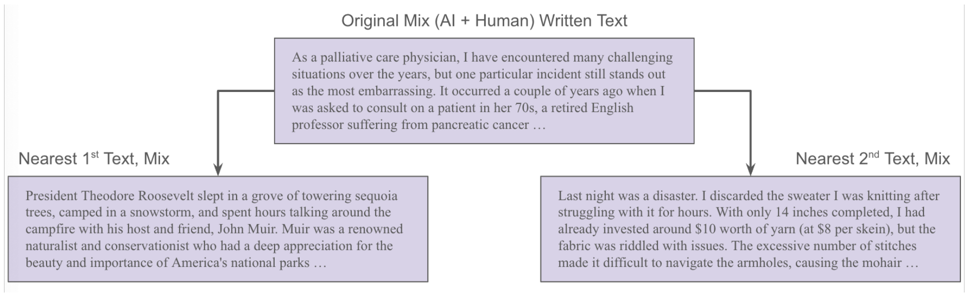
Nearest 1st Text, AI

'due to pressure from film studios wanting to increase their production, as the major networks began airing theatrically released films, abc joined cbs and nbc in broadcasting films on sunday nights in 1962, with the launch of the abc sunday night movie, which debuted a year behind its competitors and was initially presented …

Nearest 2nd Text, AI

The passage discusses the classical view of economics that views inequalities in the distribution of income as arising from differences in value added by labor, capital, and land. This perspective sees wages and profits determined by the marginal value added of each economic actor, which in turn is determined by differences in the …

Figure 5: Two Nearest Neighbors for an AI-Written Text


Original Mix (AI + Human) Written Text

As a palliative care physician, I have encountered many challenging situations over the years, but one particular incident still stands out as the most embarrassing. It occurred a couple of years ago when I was asked to consult on a patient in her 70s, a retired English professor suffering from pancreatic cancer …

Nearest 1st Text, Mix

President Theodore Roosevelt slept in a grove of towering sequoia trees, camped in a snowstorm, and spent hours talking around the campfire with his host and friend, John Muir. Muir was a renowned naturalist and conservationist who had a deep appreciation for the beauty and importance of America's national parks …

Nearest 2nd Text, Mix

Last night was a disaster. I discarded the sweater I was knitting after struggling with it for hours. With only 14 inches completed, I had already invested around $10 worth of yarn (at $8 per skein), but the fabric was riddled with issues. The excessive number of stitches made it difficult to navigate the armholes, causing the mohair …

Figure 6: Two Nearest Neighbors for a mixed Human and AI Written Text

7 Analysis

7.1 Classification Performance

The MLP and KNN classification heads exhibit extremely strong performance on all the pure datasets (HC3, DAIGT, WELFake) and the mixed data set, achieving AUCs and F1 scores greater than 0.95 and often above 0.99, roughly indicating that a text has a at least a 95% chance of correct classification (K-means did not perform as well, but it is fairly weak and very initialization-dependent as a method). It is clear that RADAR is an extremely strong baseline, achieving AUCs and F1 scores very near to 1 in every case but the MixSet. DetectGPT also performs well for being zero-shot, especially on HC3, but it is still worse than either RADAR or our method.

Our frozen embedding model, coupled with the classification heads, recognizes the mixed human-AI style far better than any of the baselines. Furthermore, when we consider three-class classification, our method displays the additional capability of distinguishing mixed text from pure HWT and pure MGT, whereas the baselines struggle severely. Recognizing mixed human-AI involvement as separate from AI is often desirable–perhaps in a scenario where a human writer may want to employ an AI to help clean up grammar, or where a human uses an AI for brainstorming and then reformulates the text as their own, but it would still be impermissible to use entirely AI-generated texts or ideas.

Looking at Figure 1, we can track the performance of our model across different usages of AI for the binary classification to understand which types of mixed text are easiest to detect AI involvement. We see that the strongest performance comes in categories like "Humanize" and "Polish Sentence", where "Humanize" refers to taking an AI-generated text and simply trying to insert perturbations that sometimes are present in human style, like misspellings or abbreviations, while preserving the underlying AI-generated ideas. "Polish Sentence" refers to an AI rewriting human sentences to make them sound more clear, which may not always preserve human ideas as intended. In contrast, on data sets which have comparatively little AI involvement, such as "Polish Token", where the AI purely tries to correct spelling or grammar, the performance is lower, consistent with the fact that spelling and grammar are not always indicative of style, depending on the context in human writing.

Interestingly, we have the most difficulty by far with "MGT Adapt", where a human author is instructed to revise tokens or sentences which they perceive to be inconsistent with humanistic writing to improve fluidity. These texts are most likely misclassified because though the base text is AI-generated, the expert humans have done a good job modifying the text to "sound human." Our model performs incredibly well when trying to classify and detect human-written text which has been modified by an LLM. However, future work should further explore improving classification of machine-written texts which is then modified by humans. That form of mix human-AI writing is more difficult to detect, but also has many relevant use cases.

7.2 Embedding and Text Interpretability

The embeddings plot shows a relatively strong division between the classes. Figure 3 is especially striking. Not only are the human-AI mixed text embeddings visually distinguishable from pure HWT or MGT, but *the mixed texts fall in between pure HWT and MGT geometrically*. If the picture is this clear in 2 dimensions, there must be something fairly close to two separating hyperplanes in 768 dimensions between the AI and mixed texts, and between the mixed texts and human texts.

We can interpret our method as well through the transparency offered by the KNN classifier. The human examples in Figure 4 demonstrate a highly colloquial style common in Internet forums, although the content is different–the text to be classified discusses history, whereas the next-nearest one is about biology and the second-nearest is about science fiction. The AI-written examples in Figure 5 are consistent with the authoritative, almost textbook-style writing that often emerges when humans ask an AI to help them understand a topic, but again, the subject of each text is unique (earth science, film history, and economics). Finally, the mixed examples in Figure 6 show both elements of the authoritative AI writing style common in question-answering and a certain colloquial, almost emotional human touch which is especially apparent in the third text. Similar to before, the nearest texts maintained similar style but diverged in content. Plotting the embeddings and qualitatively analyzing examples demonstrates the effectiveness of our further pretrained style embeddings model.

7.3 Efficiency and Adaptation Across Domains

Referring back to the details of our experiments, generating all of the style embeddings for HC3 on CPU takes about 4 hours. From there, the classification head is trained in a few minutes and inference is extremely fast. Again, RADAR took 3.5 hours to train and another 20 minutes to evaluate on GPU. So, it takes the same amount of time to train and evaluate our approach end-to-end on **CPU** as it took the RADAR baseline on **GPU**. Furthermore, the embeddings only need to be cached by the embedding model once; then, one can experiment with classification heads very quickly.

Even if a user of our model does not have enough data to train or continue to train one of the classification heads, our out-of-domain test results again outperform the RADAR baseline, especially for non-mixed data. The out-of-domain performance is slightly worse than DetectGPT, but given that we could not even run DetectGPT for lack of compute, a typical user with limited compute would be far better off with our approach. We also notice the difficulty of generalizing from training on pure human or AI text only to testing on mixed data.

8 Conclusions and future work

Our contribution can be summarized in three main points.

Simple MLP and KNN classification heads are comparable to or outperform strong baselines across domains. Our approach learns an improved representation of AI style by continuing pretraining on the embedding model. The additional use of a classification head, even a simple one like an MLP, performs on par with or outmatches even the most powerful baselines, both within the same domain as the training data and out-of-domain.

Content-independent style is preserved geometrically and linguistically. We confirmed that the human style is recognizable quite distinctly from an AI author by analyzing the outputs of KNN. Plotting the reduced embeddings allowed us to see that the authorship is preserved in embedding space, with discernible classification boundaries and *with the mixed texts sitting between the human and AI text, clearly representing a linear mixture of human and AI style*.

Our approach is faster, easier to extend, and offers greater interpretability than baselines. Anyone can easily train a classifier on top of the embeddings with limited compute and achieve strong results, even with a comparatively small dataset such as MixSet. In settings where AI text detection is desirable and where a custom, relevant data set might exist–e.g. a teacher who has her students' past essays or a news agency with an archive of articles–the embedding model which we continued pretraining can be used out-of-the-box in conjunction with a classification head with minimal work (even hyperparameter tuning is not really necessary). Again, the end-to-end training time on CPU with our method is as fast as the baselines are on GPU, with dramatic speed improvements once embeddings are cached.

The strong performance of KNN yields not only interpretability, as the author's style can be seen by comparing neighboring texts, but also recourse and accountability in settings related to plagiarism. Conventional detectors in industry simply label a text as human or AI with some confidence or probability. However, with our method, if a student is accused of plagiarism by an AI text detector, one only has to look at the most similar texts to understand why a certain classification was made, and from there it is easier to determine if the classification is borderline or erroneous.

8.1 Limitations and Future Work

Limitation in the Scale of the MixSet Dataset. Despite the wide coverage in types of texts, the overall scale is relatively small. This could limit the comprehensiveness of model training and evaluation. A larger dataset could be used to further pretrain our model.

Lower accuracy detecting machine-generated texts that are modified by humans. Further pretraining and/or fine-tuning for specific use cases could improve the classification performance of mix texts where humans modify AI ideas and content.

Exploring adversarial settings further. Implementing additional forms of mixed human-AI text generation, testing adversarial prompts where LLMs are directed to write in a specific style/persona, and further out of domain tests potentially for multiclass classification.

References

Guangsheng Bao, Yanbin Zhao, Zhiyang Teng, Linyi Yang, and Yue Zhang. 2024. Fast-DetectGPT: Efficient Zero-Shot Detection of Machine-Generated Text via Conditional Probability Curvature. ArXiv:2310.05130 [cs].

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. GPT-NeoX-20B: An Open-Source Autoregressive Language Model. ArXiv:2204.06745 [cs].

Jonathan P. Chang, Caleb Chiam, Liye Fu, Andrew Z. Wang, Justine Zhang, and Cristian Danescu-Niculescu-Mizil. 2020. ConvoKit: A Toolkit for the Analysis of Conversations. ArXiv:2005.04246 [cs].

Chujie Gao, Dongping Chen, Qihui Zhang, Yue Huang, Yao Wan, and Lichao Sun. 2024. LLM-as-a-Coauthor: The Challenges of Detecting LLM-Human Mixcase. ArXiv:2401.05952 [cs] version: 1.

Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *arXiv preprint arxiv:2301.07597*.

Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho. 2023. RADAR: Robust AI-Text Detection via Adversarial Learning. ArXiv:2307.03838 [cs].

Darek Kłeczek. 2023. Daigt external train dataset.

Teddy Lazebnik and Ariel Rosenfeld. 2023. A Computational Model For Individual Scholars' Writing Style Dynamics. ArXiv:2305.04900 [cs].

Teddy Lazebnik and Ariel Rosenfeld. 2024. Detecting LLM-Assisted Writing in Scientific Communication: Are We There Yet? ArXiv:2401.16807 [cs].

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature. ArXiv:2301.11305 [cs].

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. ArXiv:1910.01108 [cs].

Victor Tangermann. There's a Problem With That App That Detects GPT-Written Text: It's Not Very Accurate.

Edward Tian and Alexander Cui. 2023. Gptzero: Towards detection of ai-generated text using zero-shot and supervised methods.

Pawan Kumar Verma, Prateek Agrawal, Ivone Amorim, and Radu Prodan. 2021. WELFake: Word Embedding Over Linguistic Features for Fake News Detection. *IEEE Transactions on Computational Social Systems*, 8(4):881–893. Conference Name: IEEE Transactions on Computational Social Systems.

Anna Wegmann, Marijn Schraagen, and Dong Nguyen. 2022. Same Author or Just Same Topic? Towards Content-Independent Style Representations. ArXiv:2204.04907 [cs].

Wikipedia. 2024. Triplet loss wikipedia article.

A Appendix

Table 4: Dataset summary

Dataset	**Subset**	**Description**
DAIGT	human	Student essays from Kaggle, persuasive essays from Persuade Corpus
	ai	Essays written by various AI models (ChatGPT, GPT-4, LLama, Falcon, Mistral, Claude)
HC3	reddit_eli5	Humans and AIs respond to questions in the style of the popular subreddit "Explain Like I'm 5"
	open_qa	Humans and AIs respond to open-domain questions
	wiki_csai	Humans and AIs respond to prompts collected from crawling Wikipedia
	medicine	Humans and AIs respond to medicinal dialog
	finance	Humans and AIs respond to finance questions
WELFake	all	A collection of real, fake, and satirical news articles
MixSet	complete	An LLM completes a text given 1/3 of a human-written text
	polish	An AI cleans up a human-written text, either at the token (grammatical) or sentence-level
	rewrite	An LLM reads a human-written text and rewrites it
	humanize	A human (sometimes simulated with an AI) takes a machine-generated text and introduces perturbations commonly found in human texts, such as spelling/grammar errors or abbreviations
	adapt	An expert human takes a machine-generated text and modifies it to be fluent and natural similar to human linguistic habits
	pure	Fully human-written or machine-generated