Detecting LLM-Generated Text: A Naive Style Similarity Approach

Team: Jiacheng Zhu, Hongkai Ding, Jingyi Guo

1) Abstract

This study investigates the detection of text generated by large language models (LLMs), a vital task as the distinction between human and machine-generated content becomes crucial in maintaining informational integrity and preventing misinformation. We frame this as a binary classification problem, further complicated by scenarios like prompt engineering. To build a comprehensive dataset, three separate sources of human-written essays and their instructions were used to prompt GPT-4 for corresponding LLM texts. Hypothesizing that LLMs, such as GPT, have detectable writing styles, we employed classic author attribution techniques by creating an author representation through mean pooling sentence embeddings and comparing these to test sentence embeddings. Our analysis across three scenarios—identical prompts in training and testing, varied prompts, and prompt-engineered test texts—revealed that our naive approach succeeded in the first two cases but only achieved random guessing performance in the third due to style alterations through prompt engineering. We then enhanced our model by incorporating a specific human representation and fitting PCA to the development set sentence embeddings, then transforming both human and LLM representations before using Euclidean distance for similarity measurement with the test set, which improved the accuracy of the failure case to 74%. The effectiveness of the proposed method is largely dependent on the choice of sentence transformer, with transformers specifically designed to encode stylistic features, such as Wegmann embedding, demonstrating superior performance [1]. However, the approach does not consistently succeed, especially when influenced by prompt engineering in GPT-generated sentence production. The contributions of this study include a detailed analysis of failure cases and subsequent model enhancements to address these identified shortcomings, aiming to inform and enhance future research in LLM text detection.

2) Introduction

2.1 Inputs and Outputs

The primary task of this project is to determine whether a piece of text has been generated by a Large Language Model (LLM). The system is designed to receive free text as its input and to produce a binary output: '0' signifies that the text is not LLM-generated, whereas '1' indicates that it is. This straightforward input-output setup allows for clear and direct application in various contexts where the authenticity of text is questioned.

2.2 Data

The data for this project comprises both human-written and LLM-generated texts, categorized into three distinct domains to facilitate comprehensive training and testing (Data Source Link in Appendix). Domain 1 uses a shared prompt dataset, where training and testing data are based on the same prompt. An example use case is plagiarism detection, in which a teacher assigns an essay to students and both the teacher and the students know the instructions. In this case, the training data and testing data are very similar as they share the same prompt (Figure 1).

Domain 2 utilizes an unshared prompt dataset across training and testing. This setup prepares the model for real-world applications such as verifying the authenticity of articles found online or on social media, where the prompts used to generate the text are unknown. The training data includes essays from the FeedBack Prize 3 collection and corresponding LLM-generated texts, while testing involves texts from the Ivypanda database and their LLM-generated equivalents. This ensures that the prompts do not overlap between the training and testing phases. This domain is designed to test the model's robustness and adaptability to new and unseen prompts (Figure 2).

Domain 3 extends Domain 2 and includes prompt-engineered LLM-generated texts in the test set. This setup simulates advanced scenarios where LLMs are specifically prompted to emulate certain styles or content, thus presenting additional challenges for the detection model. The prompt engineering techniques employed by the team involve few-shot learning and attribute prompting, as detailed in the appendix under the data source link section. Utilizing two student-written essays, the team created a two-shot learning prompt with additional attributes that required GPT to generate texts exhibiting diverse styles such as expository, descriptive, persuasive, etc.; diverse tones such as excited, depressed, etc.; and diverse language styles like academic, formal, informal, etc. The texts generated from these engineered prompts replaced the LLM texts in Domain 2's test set. The structure of the training, development, and testing datasets otherwise remains consistent with Domain 2, providing continuity in experimental conditions while introducing complexity in the test cases (Figure 3).

2.3 Evaluation Metrics and Social Impact

The efficacy of the proposed system is measured using accuracy, recall, precision, and especially the F1 score, due to its balance of precision and recall, making it a comprehensive metric for this application. The model's performance will be benchmarked against baseline models to quantify improvements and establish its utility. The capability to accurately detect LLM-generated texts holds significant societal benefits, particularly in combating misinformation on social media and preventing academic dishonesty across educational levels. While the project poses minimal ethical risks, due to the consensual collection of human-written texts, its potential to address critical issues of truth and authenticity in digital content underscores its importance.

3) Background

The proposed system and hypothesis, inspired by the field of author attribution, reference a 2021 paper titled *Writing Style Author Embedding Evaluation*, which investigates the effectiveness of sentence embeddings in distinguishing writing styles (https://aclanthology.org/2021.eval4nlp-1.9.pdf) [4]. The demonstrated effectiveness of embeddings in author attribution encouraged the team to adapt this approach to the field of LLM-text detection. Our system, commonly used in author attribution, builds an author-style representation and compares it with testing sentences to classify authorship. This commonly used system was adapted to the domain of LLM-text detection, and the system's flowchart is illustrated in figure 4.

Our system also involves generating sentence embeddings, following instructions from Hugging Face (huggingface.co/sentence-transformers/all-roberta-large-v1) [5]. While some previous literature, such as the 2020 paper *Ranking Scalar Adjectives with Contextualised Representations* (aclanthology.org/2020.emnlp-main.598.pdf), suggests using the second-to-last or third-to-last hidden layers of BERT for style embeddings, our team opted for the last hidden layer after finding the Wegmann style embedding model to be superior to BERT[6][1][7]. The Wegmann style embedding was introduced by Anna Wegmann in her 2022 paper *Same Author or Just Same Topic? Towards Content-Independent Style Representations* (https://arxiv.org/abs/2204.04907) [1]. In her paper, Wegmann introduced a sentence transformer that can capture an author's style without capturing context [7], an objective highly aligned with this project's hypothesis and allowing the proposed system to capture pure LLM-style representations.

LLM-text data generation was a significant component of this project, and the team used the Data Dreamer library to assist with data generation (https://arxiv.org/abs/2402.10379) [3]. This library enables the team to generate a large amount of GPT-4 text using various prompt engineering methods. One such method, attributed prompts, was referenced in the 2023 research *Large Language Model as Attributed Training Data Generator: A Tale of Diversity and Bias*, which investigates the performance of attributed prompts in data generation (https://arxiv.org/abs/2306.15895) [2]. Among all the research referenced, Wegmann's paper on content-independent style representation is the most relevant to our project.

4) Summary of Our Contributions

The team's first contribution is the analysis of failure cases. This task involves determining the scenarios or subdomains of the test data in which the proposed system will succeed or fail. Many LLM-text detection tools are available; however, they do not always perform successfully, especially in complex scenarios involving prompt engineering. The team investigates three scenarios. First, the LLM-text in the training and testing sets is generated using the same prompts (figure 1). Second, the LLM-text in the training and testing sets is generated using different prompts (figure 2). Third, the LLM-text in training and testing are different, and the LLM-text in testing is prompt-engineered to simulate a student writing style using few-shot learning and to encourage diversity in writing style using attributed prompting (figure 3) [2].

The second contribution involves extending the model to address the failure cases identified in the first contribution. In this part, the system incorporates a human representation, and then fits PCA to the development set, which is then used to transform testing sentences, human, and GPT representations. Subsequently, Euclidean distance is used as a similarity measure to determine a classification threshold for classification.

5) Detailed Description of Contributions

5.1 Methods

In contribution 1, where we perform failure case analysis using the original proposed system, the high-level structure involves the following steps: In training, (1) Obtain k diverse samples of texts generated by a Large Language Model (LLM). (2) Transform each piece of text into a sentence embedding using your chosen model. (3) Average the sentence embeddings to get the GPT-style representation. In inferencing, (1) Compute cosine similarity between the GPT representation and the sentence embedding of the testing text. (2) Determine if the similarity meets or exceeds a specific threshold for binary classification. Additional specific details include the method of obtaining the sentence embedding through mean pooling of the last hidden layer's word embeddings. The team experimented with two different sentence transformers in the proposed system: RoBERTa and Wegmann [2]. Wegmann is particularly powerful as it focuses on generating context-independent style embeddings, which are well-suited for our use case. Figure 4 illustrates the workflow.

In Contribution 2, where we extended the proposed system to improve failure case performance, the extended system involves the following steps: During training, (1) Obtain *k* diverse samples of texts generated by an LLM and *m* diverse samples of texts written by humans. (2) Transform each piece of text into a sentence embedding using your chosen model. (3) Mean pool the sentence embeddings to create the GPT-style representation and Human-Style representation. (4) Use a development set to fit 2 principal components. (5) Transform the GPT-style and Human-Style representations into 2-dimensional space using the fitted principal components. In inferencing, (1) Transform the testing sentence into 2-dimensional space using the fitted principal components. (2) Use Euclidean distance to measure the similarity between the testing sentence and both the GPT-Style and Human-Style representations to obtain two similarity scores. (3) Calculate the absolute value of the difference between the two similarity scores. (4) Apply a threshold to the absolute value of the difference to classify the text. Additional specific details include setting a threshold on the development set. The sentence transformer used in this extension is Wegmann [1]. Figure 5 illustrates the workflow.

5.2 Experiments and Results

In Contribution 1 of the project, which focuses on analytical contributions, the team hypothesizes that LLMs, such as GPT, possess their own unique preconceptions or styles of writing, similar to human authors. This has been underscored by several very recent studies, which suggest that GPT tends to use certain words or phrases, such as 'commendable' and 'innovative', that are not typically used by humans [8][9]. This previous literature motivates us to explore when GPT-generated texts can be detected and when it is nearly impossible to detect them. The baselines used to compare with our proposed system in each data domain include two models. In particular, we have employed the RoBERTa sequence classification model without fine-tuning, as well as the RoBERTa sequence classification model with fine-tuning, to perform binary classification to distinguish between human-written and AI-generated texts. The

BERT-based pre-trained model has been a commonly used baseline for many classification tasks, such as fake news detection, where it has been frequently utilized [10].

In the Contribution 1 experiment, we examined three data domains mentioned in the introduction. In Domain 1, derived from a recent Kaggle competition [11], both LLM-generated and human-generated texts used identical prompts during training and testing, resulting in over 99% accuracy for the baseline, fine-tuned RoBERTa, and our proposed system. The scenario where the prompt domain remained constant across training and testing, confirmed its simplicity through competition leaderboards and our results in Table 1.

In Use Case 2, training and testing texts were generated from distinct, non-overlapping prompts, presenting a significant domain shift between training and testing. This shift reduced the performance of two baseline models to mere random guessing (50% accuracy), while our proposed system powered by RoBERTa embeddings saw a performance drop to 76% accuracy. Replacing RoBERTa embeddings with Wegmann-style embeddings, designed for style-independent content, improved accuracy to 92% [2] (Table 2). As shown in Figure 6, where the distribution of similarity scores was visualized, we can see a clear Gaussian-mixture distribution which allows the team to select a clear threshold. The superior performance of Wegmann embedding comes from its design objective to generate context-free style embeddings, which are very suitable for Use Case 2.

In Use Case 3, we further manipulate the domain by introducing prompt engineering during the testing phase to potentially obscure GPT styles, making it harder to differentiate from human text. The specific prompt engineering techniques used include few-shot learning, which prompts GPT with examples of student writing to simulate human writing style, and attributed prompts, which encourage GPT to generate diverse writing styles and tones [2][3]. As a result, this shift in the test data domain successfully degraded the performance of all systems to random guessing. Figure 7 demonstrates the distribution of similarity scores in the development set, where we can no longer see a two-component Gaussian-mixture model distribution like what we observed in Use Case 2, indicating that the system is unable to separate LLM-generated sentences from human sentences. Therefore, the previously successful Wegmann-style embedding saw its accuracy drop to 55% (Table 3).

The experiments in Use Cases 1 and 2 support our hypothesis that GPT-generated text retains a detectable style, capturable by embeddings traditionally used for human styles, aligning with findings on GPT's linguistic tendencies [8][9]. However, in Use Case 3, we undermine the basis of our hypothesis through prompt engineering, stripping away GPT's predefined style or conception in writing, which results in a significant degradation of performance to random guessing. These results illustrate that when advanced techniques such as prompt engineering are involved, it becomes extremely challenging for a naive system like the proposed one to accurately classify which texts are generated by an LLM and which are not.

The team's Contribution 2 focused on model extension, initiated after identifying the failure case in Use Case 3. The team expanded the proposed system to include several new components. The first addition was a human-style representation. Alongside the GPT style representation, the team decided to compute a human-style representation using the training dataset, which allows us to test which style the testing sentence is closer to. Before this, the team fit 2 principal components on the development dataset and projected both the human representation and GPT style representation into 2 dimensions. During testing, the team also projected the testing sentence into 2 dimensions, and then calculated the Euclidean distance between the testing sentence and both the human and GPT representations, respectively. The team then calculated the absolute difference between the two distances, using a threshold to determine the classification of the testing sentence. The threshold was determined through hyperparameter tuning on the development set. Using this extended system, the team was able to improve the system to 74% accuracy. The team also tested multiple other similarity measures such as the dot product without PCA, Euclidean distance without PCA, and cosine similarity without PCA, but all resulted in mere performance of around 60% accuracy (Table 4). In summary, the team was able to improve the proposed system through extension to 74% accuracy, which is significantly better than random guessing from the original proposed system; however, this performance is still far from being deployable and trustworthy. The reason that PCA improves the extended system might be related to the curse of dimensionality, which suggests that Euclidean distance is a more powerful measure of similarity when dimensionality is reduced.

6) Compute/Other Resources Used

The team used a data generation library called Data Dreamer and computing platform Google Colab, specifically A100 and L4 instances [3].

7) Conclusions

The outcomes of this project highlight three main conclusions. First, a context-independent sentence embedding such as the Wegmann embedding can significantly improve an LLM-detection system, especially when there is no prompt engineering involved in the testing sentences [1]. Second, across all three data domains/use cases tested, if no prompt engineering is involved in the LLM-generated text, a naive system such as the one proposed is sufficient to detect LLM-generated text. This aligns with results from some Kaggle competitions and recent research [11][8][9]. Such results can partially support our hypothesis that LLMs like GPT have predefined conceptions and distinct writing styles that can be captured; however, after experiments, we must add a condition to this hypothesis that it holds only when no prompt engineering is involved. Third, the team's extended system, which utilizes an additional human-style representation and PCA to project all sentences and representations into 2 dimensions, helped improve the proposed system beyond random guessing in the failure case where prompt engineering was involved, highlighting a potential path for future research. Moreover, future studies could benefit from expanding this research to include more diverse data since the team's human data was solely comprised of essays from 9th-12th grade high school students. Scaling up this project to make the LLM-detection system more useful and generalizable could potentially have a large impact, spanning from plagiarism detection to social media misinformation detection, where platforms like X (formerly Twitter) could mark tweets generated by LLMs and warn users about potential risks associated with these tweets.

8) Reference

[1] Wegmann, S., Fu, X., & Volodina, E. (2022). Same author or just same topic? Towards content-independent style representations. In *Proceedings of the 7th Workshop on Representation Learning for NLP (RepL4NLP 2022)*. https://aclanthology.org/2022.repl4nlp-1.26

[2] Yu, Y., Zhuang, Y., Zhang, J., Meng, Y., Ratner, A., Krishna, R., Shen, J., & Zhang, C. (2023). Large language model as attributed training data generator: A tale of diversity and bias. *arXiv preprint arXiv:2306.15895*. https://arxiv.org/abs/2306.15895

[3] Patel, A., Raffel, C., & Callison-Burch, C. (2024). DataDreamer: A tool for synthetic data generation and reproducible LLM workflows. *arXiv preprint arXiv:2402.10379*. https://arxiv.org/abs/2402.10379

[4] Terreau, E., Gourru, A., & Velcin, J. (2021). Writing style author embedding evaluation. In *Proceedings of the 2021 Workshop on Evaluation and Comparison of NLP Systems (EVAL4NLP)*. https://api.semanticscholar.org/CorpusID:241583359

[5] Hugging Face. (n.d.). Sentence transformers: All-RoBERTa-large-v1 usage example. https://huggingface.co/sentence-transformers/all-roberta-large-v1

[6] Garí Soler, A., & Apidianaki, M. (2020). BERT knows Punta Cana is not just beautiful, it's gorgeous: Ranking scalar adjectives with contextualised representations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 7371-7385). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.598

[7] Wegmann, A. (n.d.). Style embedding. Hugging Face. https://huggingface.co/AnnaWegmann/Style-Embedding

[8] Gray, A. (2024). ChatGPT "contamination": Estimating the prevalence of LLMs in the scholarly literature. *arXiv preprint arXiv:2403.16887*. https://arxiv.org/abs/2403.16887

[9] Liang, W., Izzo, Z., Zhang, Y., Lepp, H., Cao, H., Zhao, X., Chen, L., Ye, H., Liu, S., Huang, Z., McFarland, D. A., & Zou, J. Y. (2024). Monitoring AI-modified content at scale: A case study on the impact of ChatGPT on AI conference peer reviews. *arXiv preprint arXiv:2403.07183*. https://arxiv.org/abs/2403.07183

[10] Palani, B., Elango, S., & Viswanathan, K. V. (2022). CB-Fake: A multimodal deep learning framework for automatic fake news detection using capsule neural network and BERT. *Multimedia Tools and Applications, 81*(15), 5587–5620. https://doi.org/10.1007/s11042-021-11782-3

[11] Kaggle. (n.d.). LLM-Detect AI-generated text. https://www.kaggle.com/competitions/llm-detect-ai-generated-text/overview

9) Tables

Table 1: Data Domain 1 - Shared Prompts across Train and Test				
	Accuracy	Precision	Recall	F1 Score
Baseline 1: Roberta No Finetuning	0.957299	0.921277	**1.0**	0.959025
Baseline 2: Roberta W/ Finetuning	**0.999423**	**0.998847**	**1.0**	**0.999423**
Proposed: GPT-Rep-Roberta	0.998269	0.996548	**1.0**	0.998271

Table 2: Data Domain 2 - Unoverlap Prompts across Train and Test				
	Accuracy	Precision	Recall	F1 Score
Baseline 1: Roberta No Finetuning	0.509688	0.504891	**1.000000**	0.671000
Baseline 2: Roberta W/ Finetuning	0.532024	0.516542	1.000000	0.681210
Proposed: GPT-Rep-Roberta	0.764263	0.773690	0.747040	0.760131
Proposed: GPT-Rep-Wegmann	**0.925996**	**0.897140**	0.962325	**0.928590**

Table 3: Data Domain 3 - Domain 2 + Prompt Engineered LLM text in Test				
	Accuracy	Precision	Recall	F1 Score
Baseline: Roberta No Finetuning	0.509150	0.504622	**0.998924**	**0.670520**
Baseline: Roberta W/ Finetuning	0.510495	0.505544	0.956943	0.661581
Proposed: GPT-Rep-Roberta	0.317815	0.363011	0.482777	0.414414
Proposed: GPT-Rep-Wegmann	**0.554629**	**0.547541**	0.629171	0.585525

Table 4: Extended System Performance on Data Domain 3				
	Accuracy	Precision	Recall	F1 Score
Method 1: Cosine Similarity	0.656620	0.618005	0.820237	0.704903
Method 2: Euclidean Distance	0.636168	0.611947	0.744349	0.671685
Method 3: Dot Product	0.671690	0.646062	0.759419	0.698169
Method 4: PCA -> Eculidean Distance	**0.746771**	**0.703868**	**0.851991**	**0.770879**

10) Figures

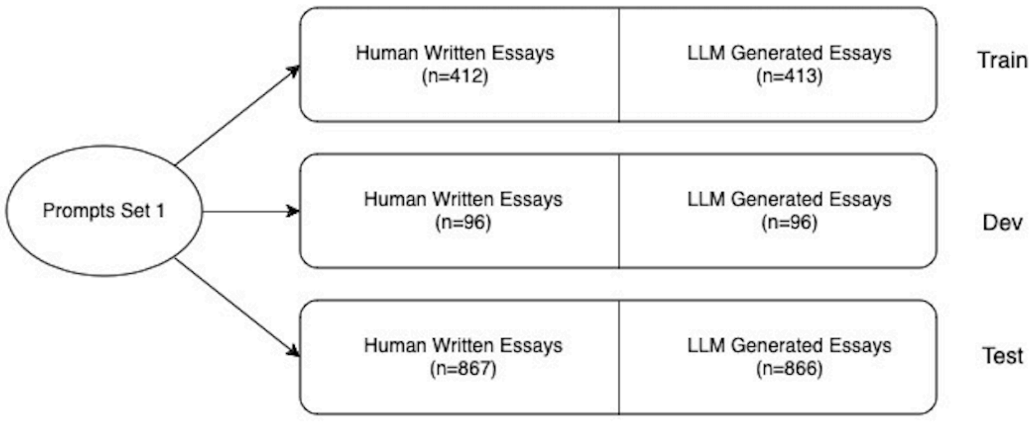
Prompts Set 1
Human Written Essays (n=412)
LLM Generated Essays (n=413)
Train
Human Written Essays (n=96)
LLM Generated Essays (n=96)
Dev
Human Written Essays (n=867)
LLM Generated Essays (n=866)
Test

Figure 1: Use Case 1 Data Setup

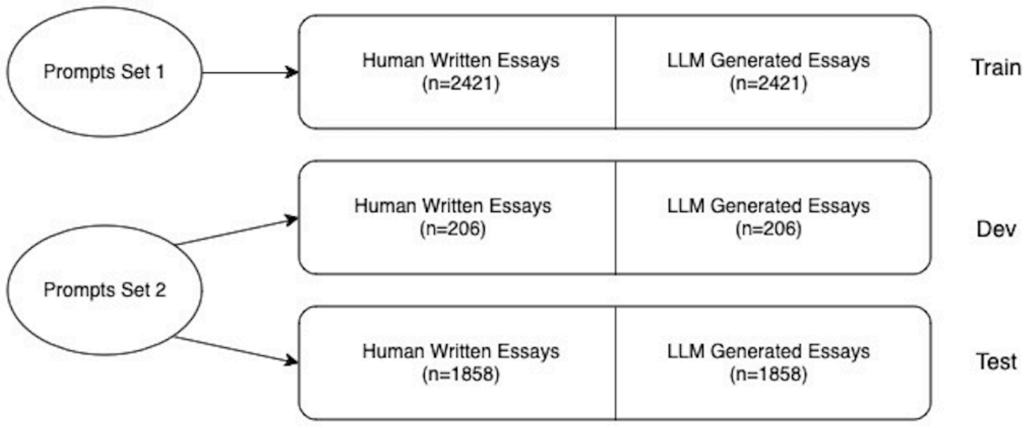
Prompts Set 1
Human Written Essays (n=2421)
LLM Generated Essays (n=2421)
Train
Prompts Set 2
Human Written Essays (n=206)
LLM Generated Essays (n=206)
Dev
Human Written Essays (n=1858)
LLM Generated Essays (n=1858)
Test

Figure 2: Use Case 2 Data Setup

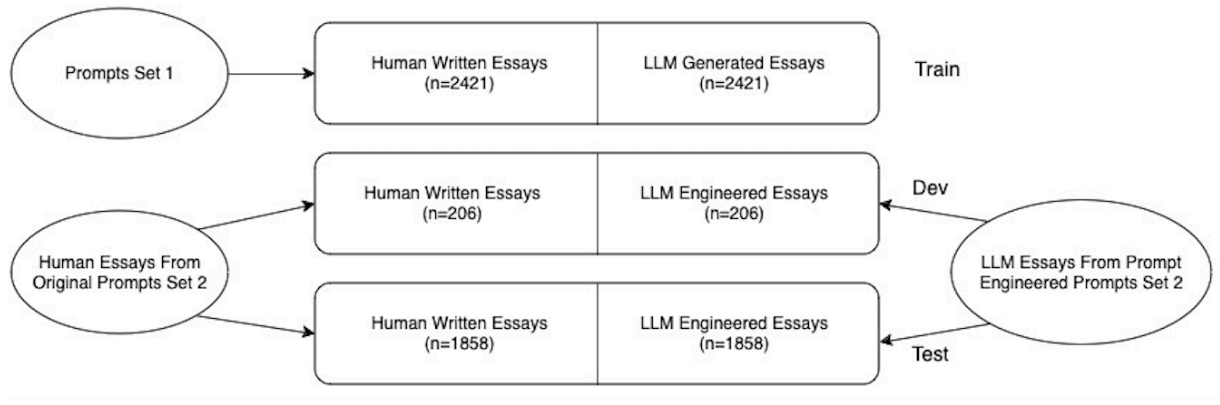
Prompts Set 1
Human Written Essays (n=2421)
LLM Generated Essays (n=2421)
Train
Human Essays From Original Prompts Set 2
Human Written Essays (n=206)
LLM Engineered Essays (n=206)
Dev
Human Written Essays (n=1858)
LLM Engineered Essays (n=1858)
Test
LLM Essays From Prompt Engineered Prompts Set 2

Figure 3: Use Case 3 Data Setup

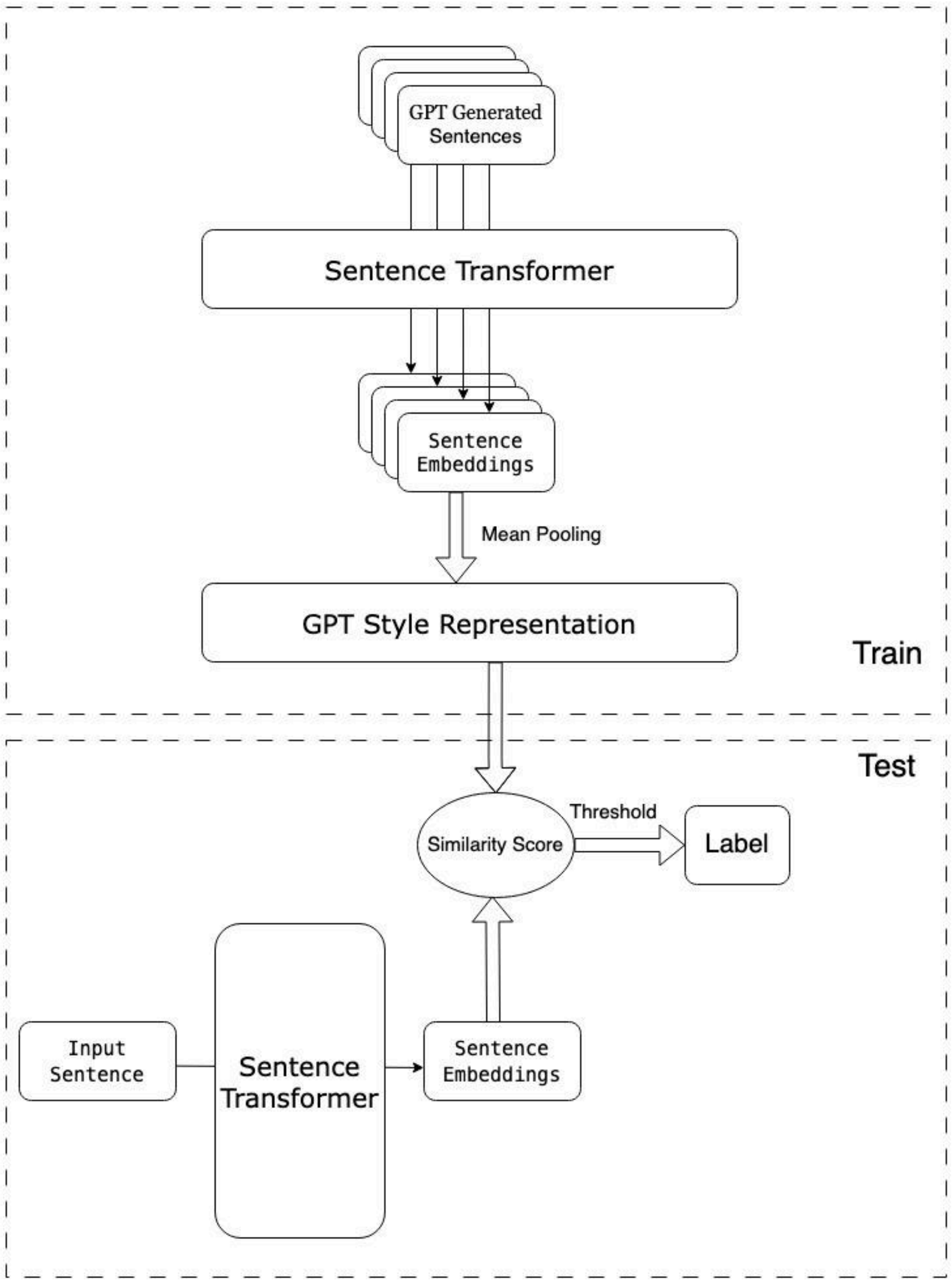

Figure 4: Proposed System in Fail Case Analysis

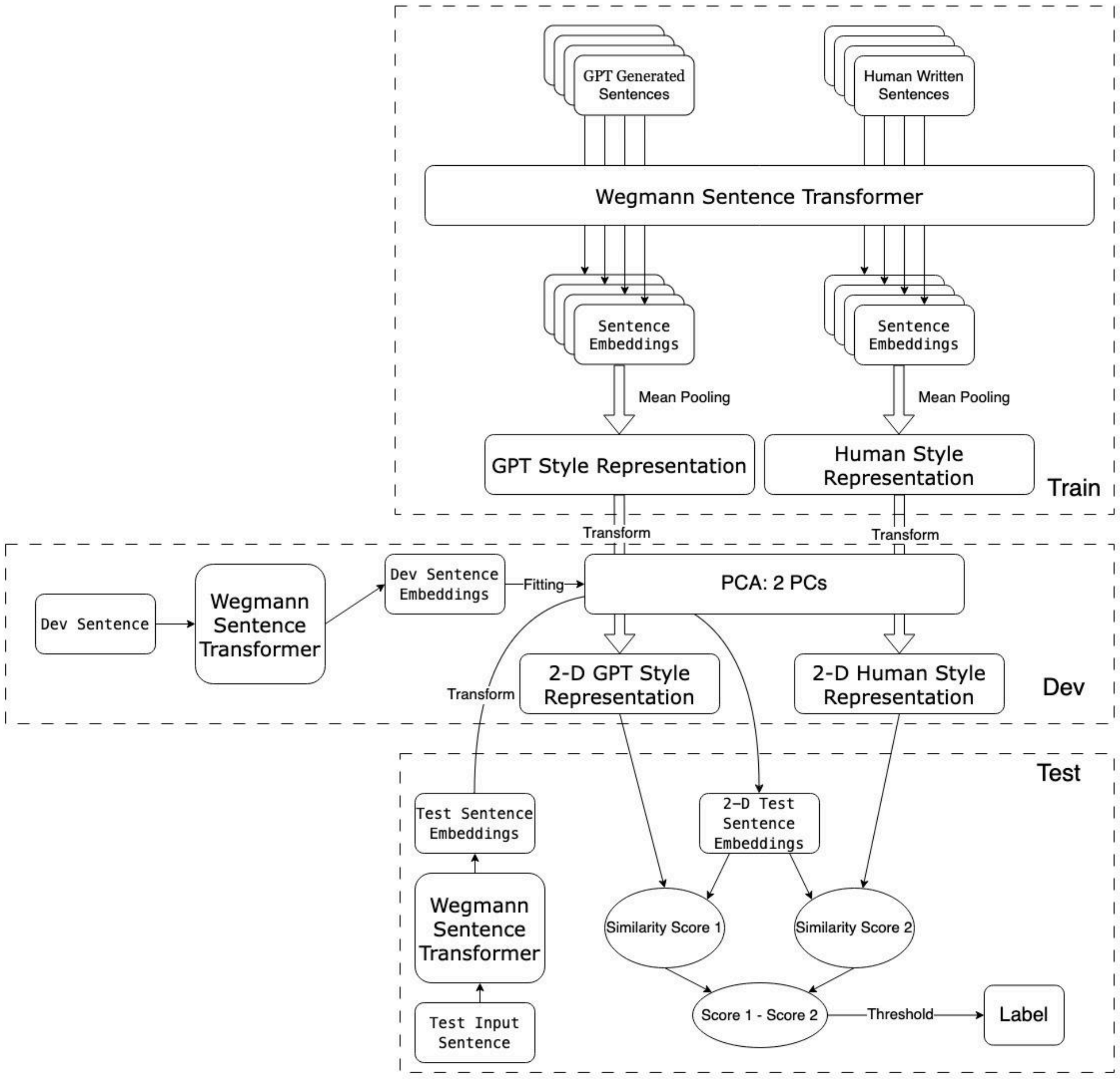

Figure 5: Extended System to improve performance on Fail case

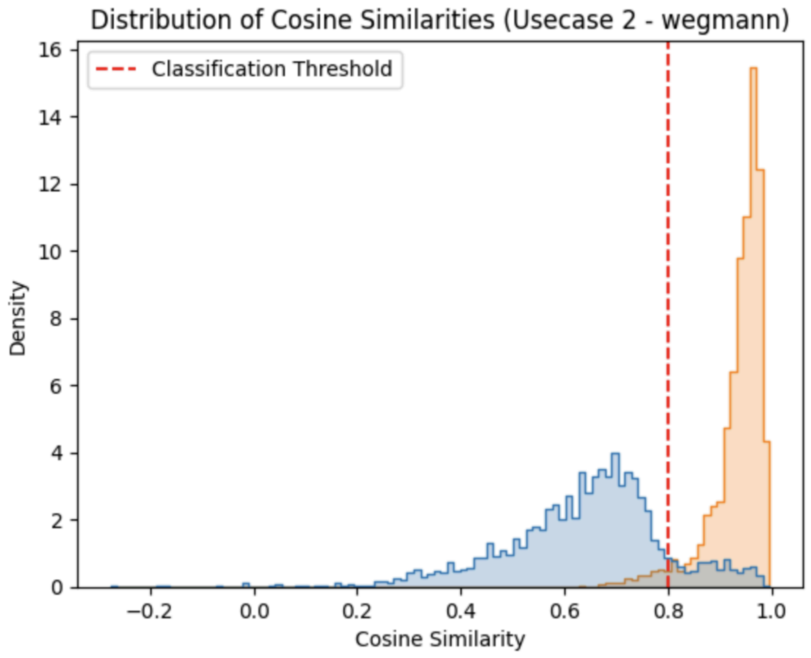

Figure 6: Distribution of Cosine Similarity in Use Case 2 + Wegmann Embedding

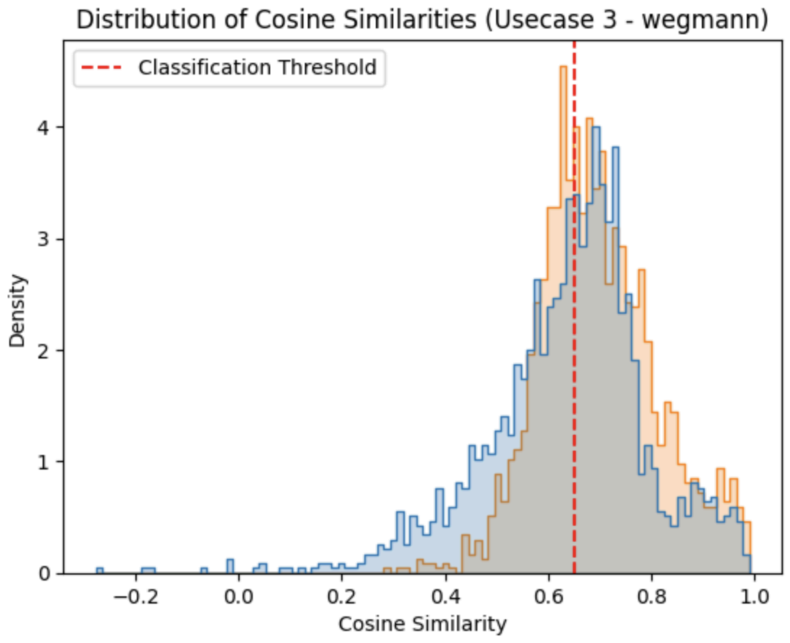

Figure 7: Distribution of Cosine Similarity in Use Case 3 + Wegmann Embedding

11) Data Source Link

- Domain 1
  - Human-written text: Domain 1 Human Data - High School Student Essay
  - LLM-generated text: Domain 1 LLM Data Generated by Team using GPT4
- Domain 2
  - Human-text train: FeedBack Prize 3 - High School Student Essay
  - LLM-text train: LLM text generated from Human-written text 1's prompt
  - Human-text test: ivypanda - Sample Essay for High School Students
  - LLM-text test: LLM text generated from Human-written text 2's prompt
- Domain 3
  - LLM-text-prompt-engineered: Prompt Engineered LLM-text test from Domain 2

Engineered Prompt and attributes, the library used to assist data generation is Data Dreamer:

```
style = ['Expository', 'Descriptive', 'Persuasive', 'Narrative',
'Creative', 'Argumentative']
tone = ['excited', 'depressed', 'sarcastic', 'frightened', 'hopeful']
language = ['academic', 'formal', 'informal', 'Colloquial ']

'''
Here are some sample pieces of text written by high school students:
   (1) {example1}
   (2) {example2}

Now, write an essay simulates a high school student's writing
using {style} writing styles, {tone} tone, {language} language, around
length of {length} words
with respect to following instruction:
'''
```