CNLP-NITS-PP at GenAI Detection Task 3: Cross-Domain Machine-Generated Text Detection Using DistilBERT Techniques

L D M S Sai Teja, Annepaka Yadagiri, M Srikar Vardhan and Partha Pakray
Department of Computer Science & Engineering
National Institute of Technology Silchar, Assam, India, 788010
{lekkalad_ug, annepaka22_rs, mangadoddis_ug, partha}@cse.nits.ac.in

Abstract

This paper presents a Cross-domain Machine-Generated Text Detection model developed for the COLING 2025 Workshop on Detecting AI-generated Content (DAIGenC). As large language models evolve, detecting machine-generated text becomes increasingly challenging, particularly in contexts like misinformation and academic integrity. While current detectors perform well on unseen data, they remain vulnerable to adversarial strategies, including paraphrasing, homoglyphs, misspellings, synonyms, whitespace manipulations, etc. We introduce a framework to address these adversarial tactics designed to bypass detection systems by adversarial training. Our team DistilBERT-NITS detector placed 7th in the Non-Adversarial Attacks category, and Adversarial-submission-3 achieved 17th in the Adversarial Attacks category.

1 Introduction

Large Language Models (*LLMs*) (Touvron et al., 2023; Anil et al., 2023) have quickly established themselves as transformative tools in Natural Language Processing (*NLP*). These models gain substantial internal knowledge by undergoing extensive pre-training on massive datasets in a self-supervised manner, enabling them to excel in various tasks, from answering factual queries and generating coherent text to handling intricate reasoning processes. This versatility has brought substantial advancements across various NLP application areas.

Despite these advancements, ethical concerns have surfaced regarding inherent risks (McKenna et al., 2023; Bian et al., 2023; Ferrara, 2023), such as the potential for misinformation, hallucinations in generated outputs, and even biases against certain groups. Growing awareness of these issues has spurred research into detecting AI-generated text. However, AI-text detectors may carry similar vulnerabilities as neural network models (Szegedy, 2013), inspiring related studies (Sadasivan et al., 2023; Krishna et al., 2024) that explore paraphrasing attacks designed to deceive detector predictions. It is contended that examining potential adversarial attacks on text detectors is crucial, as weaknesses in AI detection systems can be identified before deployment in practical settings, such as academic plagiarism detection, thereby supporting the development of effective countermeasures.

Current detection methods are generally classified into three main categories: statistical approaches (Mitchell et al., 2023) that use metrics like entropy, perplexity, and log-likelihood; neural classifiers (Guo et al., 2023) trained on supervised datasets labeled as human or AI-generated; and watermarking techniques (Kirchenbauer et al., 2023) that embed subtle patterns into AI-generated text. However, research on adversarial perturbations specifically targeting AI-text detectors still needs to be completed. For example, (Sadasivan et al., 2023) investigated paraphrasing to alter Machine-Generated Text (MGT) in adversarial attacks, while (Shi et al., 2024) employed LLMs to create adversarial word candidates through a search-based approach. Although these studies have shown that AI detectors can be vulnerable to adversarial modifications, the impact of such attacks on detector performance in complex, real-world conditions is still largely unexamined.

2 Related Work

Most research on adversarial attacks has focused on image detection (Kong et al., 2021; Akhtar et al., 2021; Xu et al., 2020), as text data presents unique challenges due to its discrete structure, making it harder to create imperceptible modifications compared to image data, where subtle pixel changes can go largely unnoticed (Peng et al., 2023). Some general text classification adversarial attacks, such

Proceedings of the 1st Workshop on GenAI Content Detection (GenAIDetect), pages 334–339

as those by (Damodaran, 2021; Gao et al., 2018), have demonstrated this. Recently, studies have turned toward adversarial attacks on neural text detectors: (Xu et al., 2020) found that introducing minor spelling errors and homoglyph replacements can significantly lower detection rates for GPT-2-generated text. Similarly, (Liang et al., 2023a) showed that character-level perturbations also affect RoBERTa-based detectors (Liang et al., 2023b) further revealed that existing detectors are vulnerable to simple rephrasing and may even mistakenly label texts written by non-native speakers as AI-generated.

Due to the susceptibility of current methods to adversarial attacks, several researchers have proposed approaches to enhance their robustness, including work by (Liang et al., 2023b; Shi et al., 2024). Although watermarking techniques have also been explored for identifying AI-generated text, they are generally considered vulnerable to adversarial tactics, particularly those based on mutation and paraphrasing (Sadasivan et al., 2023; Kirchenbauer et al., 2023).

3 Methodology

3.1 Dataset Description

As shown in Table 1, the RAID dataset contains over 10 million generated samples across diverse models, content domains, decoding strategies, and adversarial attacks. Models include ChatGPT, GPT-4, GPT-3, Llama 2, Cohere, MPT-30B, and Mistral 7B, covering content from Reddit, IMDb, Wikipedia, and news articles. Decoding strategies such as Greedy, Sampling, Greedy+Repetition Penalty, and Sampling+Repetition Penalty are used alongside adversarial techniques like paraphrasing, homoglyph, perplexity misspelling, synonyms, whitespace, upperlower, number, insert_paragraphs, article_deletion, alternative_spelling, and zero_width_space. This dataset supports research on model performance, generation diversity, and robustness against adversarial attacks.

Task	Label	Train	Dev
Non-Adversarial	Human (0)	13,371	4,855
	Machine (1)	454,614	165,070
Adversarial	Human (0)	160,452	58,260
	Machine (1)	5,455,368	1,980,840

Table 1: Statistics of Train and Development Data for Non-Adversarial and Adversarial Tasks.

3.2 System Description

This paper presents our approach to Task 3 in the COLING Workshop on MGT Detection, which emphasizes cross-domain robustness in AI-generated content detection (Dugan et al., 2025). The primary objective of this task is to classify whether a given text is machine-generated or human-authored, even when the content spans multiple domains. We participated in both Subtask A *(Non-Adversarial Cross-Domain MGT Detection*) and Subtask B (*Adversarial Cross-Domain MGT Detection*), which involve handling text from eight diverse domains, produced by eleven generative models and four decoding strategies. We first classify whether the text has been adversarially attacked to detect adversarial attacks in text. If an attack is detected, the text undergoes preprocessing to mitigate the attack, after which the preprocessed text proceeds to our model for further MGT detection. Our approach to finetuning the DistilBERT model uses hyperparameters to extract semantic features.

3.2.1 Experimental Setup and Data Sampling

The experiment was conducted in a Jupyter Notebook on a machine powered by an ***Intel® Xeon® W-2155 CPU @ 3.30GHz with 20 cores and an NVIDIA Quadro P2000 GPU*** for handling LLM tasks. The system was also equipped with 64 GB of RAM. Python served as the programming language, utilizing the libraries `Numpy`, `Pandas`, `SKlearn`, and `TensorFlow`.

To reduce the computational load, only 40% of the adversarial data is sampled based on the unique `adv_source_id`. This is done by selecting a random sample of rows corresponding to 40% of the unique IDs in the training set. This sampled data is then prepared for further processing, ensuring the dataset remains manageable while representing a substantial portion of the original data. The sampled data is reset for indexing and is ready for the pipeline's next steps.

3.2.2 Preprocessing:

As evidenced by the analysis, all attacks target plain text, and the dataset maintains a balance, with an equal number of rows for each attack type, as reflected in the supporting figure. Next, the code implements a preprocessing pipeline to clean and standardize text (*Every text attacked and non-attacked*). The attacks, such *as paraphrasing, homoglyph, perplexity misspelling, synonyms, whitespace, upperlower, number, insert_paragraphs, article_deletion,*

alternative_spelling, and zero_width_space can confuse NLP systems. The preprocessing steps address these issues by applying several transformations. Homoglyphs, which are visually similar but distinct characters (*e.g., numbers or symbols resembling letters*), are replaced using a predefined mapping. Additionally, alternative spellings (*like British versus American English*) are normalized, ensuring consistency in spelling. Numbers are converted to their word equivalents, and extra spaces or zero-width spaces are removed. The text is also converted to lowercase, and punctuation is stripped for uniformity. By implementing these techniques, the pipeline cleans up adversarially manipulated text, making it more suitable for analysis while maintaining its original meaning. This process ensures that NLP models can better understand and process the input text without being misled by adversarial perturbations.

```
input_text = "Th!s ls @n ex@mple txt w1th
    h0m0glyphs, zerowidth\u200bspaces, and
    incorr3ct spelling."

After preprocessing:

output_text = "this is an example text with
    homoglyphs, zerowidthspaces, and incorrect
    spelling."
```

3.2.3 Adversarial Detection:

After preprocessing, we obtained both the Raw Text and the Preprocessed Text for each input. We employed several factors like the combination of **Cosine Similarity** and **Edit Distance**, **Word Overlap ratio**, and **Homoglyph Substitution Count** to analyze surface-level changes (*e.g., homoglyph substitutions, misspellings*) on text embeddings generated by the `distilbert-base-uncased` model.

Cosine similarity: A widely used metric for its simplicity, interpretability, and computational efficiency for capturing semantic meaning. Its values range from -1 (*completely dissimilar*) to 1 (*identical*), providing an intuitive similarity measure. This will focus solely on the directional alignment of embeddings. Furthermore, its low computational complexity ensures scalability, making it ideal for processing large datasets efficiently.

$$\text{Cosine Similarity: } \cos(\theta) = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\|\|\vec{B}\|} \quad (1)$$

Here, $\vec{A}$ and $\vec{B}$ represent the embedding vectors, $\vec{A} \cdot \vec{B}$ denotes their dot product, and $\|\vec{A}\|$ and $\|\vec{B}\|$ represent their magnitudes (*L2 norms*). The resulting similarity score ranges between -1 and 1. Where: **1** indicates perfect similarity (*identical direction*), **0** indicates orthogonality (*no similarity*), and **-1** indicates complete opposition (*opposite direction*).

Edit Distance: Specifically, Levenshtein Distance calculates the minimum number of edits required to transform one string into another. This detects small, surface-level changes.

$$\text{Levenshtein Distance}(s_1, s_2) = \min \left\{ \begin{matrix} \text{Insert}, \\ \text{Delete}, \\ \text{Substitute} \end{matrix} \right\}$$

To combine these two measures, we can apply a hybrid approach that leverages the strengths of both metrics. The combined similarity score, S_{new}, can be represented as:

$$S_{new} = \alpha \cdot \text{CS} + (1 - \alpha) \cdot \beta \quad (2)$$

$$CS = \text{Cosine Similarity}(A, B) \quad (3)$$

$$\beta = (1 - \frac{\text{Levenshtein Distance}(A, B)}{\max(\text{len}(A), \text{len}(B))}) \quad (4)$$

Where: - A and B are the two texts being compared. - α is a weight parameter that controls the contribution of each metric. - $\text{len}(A)$ and $\text{len}(B)$ are the lengths of the two texts. - β normalizes the Levenshtein distance to a range between 0 and 1.

Word Overlap Ratio: is a metric used to quantify the similarity between two text sequences by comparing the number of common words to the total number of unique words across both sequences.

Let W_1 and W_2 represent the sets of words in two text sequences.

$$\text{Word Overlap Ratio} = \frac{|W_1 \cap W_2|}{|W_1 \cup W_2|} \quad (5)$$

Let x represent a text that is not attacked. Upon preprocessing, x remains unchanged, denoted as x'. Computing the cosine similarity between x and x', we obtain a value of 1, as x and x' are identical:

$$\text{CosineSimilarity}(x, x') = 1, \quad \text{when } x = x'.$$

Conversely, if x is an attacked text, preprocessing yields a modified version x'. The cosine similarity between x and x' will deviate from 1, reflecting the difference introduced by the attack:

$$\text{CosineSimilarity}(x, x') \neq 1, \quad \text{when } x \neq x'.$$

This approach effectively captures adversarial manipulations, enabling robust detection based on the interplay between cosine similarity and edit distance metrics.

3.2.4 Classification Model Architecture:

After calculating the adversarial detecting factors, the text embeddings and the factors combined undergo classification using a fine-tuned **`DistilBERT`** model as shown in Figure 1 for distinguishing human-generated and machine-generated text. The detailed architecture of the DistilBERT model is depicted in Figure 2. The Figure 2 illustrates the internal workings of the model, particularly highlighting the implementation process obtained from the code `model.to(device)`. The model is trained with a batch size of 16 for 3 epochs using the `AdamW` optimizer and `CrossEntropyLoss`, all the hyperparameters are shown in the Table 2. During training, the model's performance is evaluated on key metrics like accuracy, precision, recall, and F1 score, ensuring robustness against adversarial attacks. By fine-tuning the model with this approach, it can better classify text accurately in real-world scenarios, even when it contains adversarial modifications.

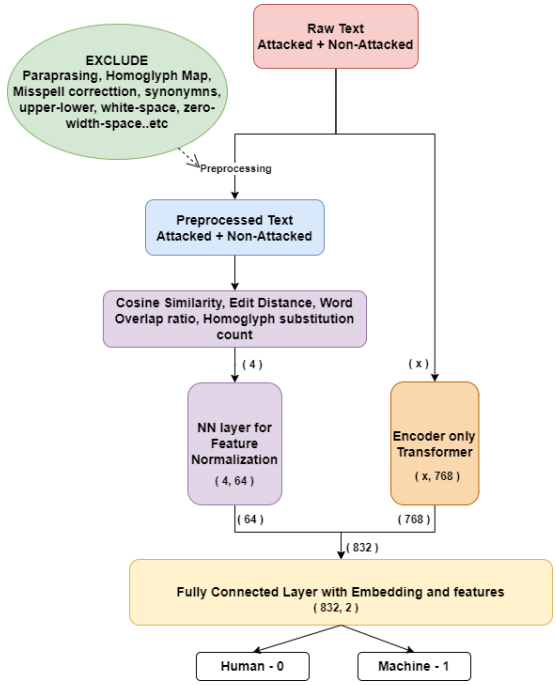

Figure 1: Architecture workflow

3.3 Results Analysis

Due to computational constraints and limited space as per the requirements, we sampled only 40% of the dataset. The table 4 presents the training epochs for the non-adversarial data, the table 5 presents the training epochs for the adversarial data, where the evaluation metrics Accuracy, Precision, Recall, F1 Score, and Loss were used. Additionally, Table 3 shows the test results in final leaderboard performance for both adversarial and non-adversarial data. Although this model may not yet be equipped to handle more advanced semantic and synthetic adversarial attacks, we will consider these and strive to improve our work in the future by incorporating new techniques.

Parameter	Value
Max Seq Leng	128
Batch Size	16, 32
Learning Rate	2e-5,5e-4
Epochs	3, 5
Patience	2
Minimum Delta	0.001
Loss	CrossEntropyLoss
Optimizer	AdamW

Table 2: Model Hyperparameters

4 Conclusion

In this study, we developed a robust framework using a fine-tuned DistilBERT-NITS model to detect MGT across diverse domains, focusing on adversarial scenarios. Our approach ranked 7th in non-adversarial detection and 17th in adversarial detection at the COLING Workshop, involves preprocessing text to mitigate detected adversarial manipulations, enhancing detection accuracy. These findings support the potential of lightweight models to handle adversarial and cross-domain MGT detection effectively. Future work will be focused on refining this method to improve robustness and adaptability against evolving adversarial tactics.

References

Naveed Akhtar, Ajmal Mian, Navid Kardan, and Mubarak Shah. 2021. Advances in adversarial attacks and defenses in computer vision: A survey. *IEEE Access*, 9:155161–155196.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Ning Bian, Peilin Liu, Xianpei Han, Hongyu Lin, Yaojie Lu, Ben He, and Le Sun. 2023. A drop of ink may make a million think: The spread of false information in large language models. *arXiv preprint arXiv:2305.04812*.

Detector	Aggregate	chatgpt	gpt4	gpt3	gpt2	mistral
DistilBERT-NITS (Non-Adv)	0.905	0.989	0.967	0.835	0.880	0.846
Adversarial-submission-3	0.467	0.553	0.533	0.306	0.375	0.543

mistral-chat	cohere	cohere-chat	llama-chat	mpt	mpt-chat
0.976	0.639	0.835	0.987	0.884	0.985
0.514	0.244	0.341	0.591	0.445	0.516

Table 3: Performance Metrics for chatgpt, gpt, mistral, cohere, llama, and mpt Models

Epoch	Dataset	Loss	Acc	F1
1	Train	0.059	0.978	0.975
	Dev	0.132	0.971	0.957
2	Train	0.029	0.988	0.988
	Dev	0.173	0.969	0.956
3	Train	0.013	0.995	0.995
	Dev	0.232	0.956	0.950

Table 4: Training and Development Metrics Across Epochs on Non-Adversarial Data

Epoch	Dataset	Loss	Acc	F1
1	Train	0.080	0.977	0.972
	Dev	0.262	0.919	0.931
2	Train	0.037	0.989	0.988
	Dev	0.261	0.946	0.944
3	Train	0.027	0.992	0.992
	Dev	0.360	0.926	0.934

Table 5: Training and Development Metrics Across Epochs on Adversarial Data

Prithiviraj Damodaran. 2021. Parrot: Paraphrase generation for nlu. *Parrot: Paraphrase generation for nlu*.

Liam Dugan, Andrew Zhu, Firoj Alam, Preslav Nakov, Marianna Apidianaki, and Callison-Burch Chris. 2025. Genai content detection task 3: Cross-domain machine generated text detection challenge. In *Proceedings of the 1st Workshop on GenAI Content Detection (GenAIDetect)*, Abu Dhabi, UAE. International Conference on Computational Linguistics.

Emilio Ferrara. 2023. Should chatgpt be biased? challenges and risks of bias in large language models. *arXiv preprint arXiv:2304.03738*.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE.

Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. 2023. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *arXiv preprint arXiv:2301.07597*.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *International Conference on Machine Learning*, pages 17061–17084. PMLR.

Zixiao Kong, Jingfeng Xue, Yong Wang, Lu Huang, Zequn Niu, and Feng Li. 2021. A survey on adversarial attack in the age of artificial intelligence. *Wireless Communications and Mobile Computing*, 2021(1):4907754.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2024. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *Advances in Neural Information Processing Systems*, 36.

Gongbo Liang, Jesus Guerrero, and Izzat Alsmadi. 2023a. Mutation-based adversarial attacks on neural text detectors. *arXiv preprint arXiv:2302.05794*.

Weixin Liang, Mert Yuksekgonul, Yining Mao, Eric Wu, and James Zou. 2023b. Gpt detectors are biased against non-native english writers. *Patterns*, 4(7).

Nick McKenna, Tianyi Li, Liang Cheng, Mohammad Javad Hosseini, Mark Johnson, and Mark Steedman. 2023. Sources of hallucination by large language models on inference tasks. *arXiv preprint arXiv:2305.14552*.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. In *International Conference on Machine Learning*, pages 24950–24962. PMLR.

Hao Peng, Zhe Wang, Dandan Zhao, Yiming Wu, Jianming Han, Shixin Guo, Shouling Ji, and Ming Zhong. 2023. Efficient text-based evolution algorithm to hard-label adversarial attacks on text. *Journal of King Saud University-Computer and Information Sciences*, 35(5):101539.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. 2023. Can ai-generated text be reliably detected? *arXiv preprint arXiv:2303.11156*.

Zhouxing Shi, Yihan Wang, Fan Yin, Xiangning Chen, Kai-Wei Chang, and Cho-Jui Hsieh. 2024. Red teaming language model detectors with language models. *Transactions of the Association for Computational Linguistics*, 12:174–189.

C Szegedy. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K Jain. 2020. Adversarial attacks and defenses in images, graphs and text: A review. *International journal of automation and computing*, 17:151–178.

A Appendix

```
model.to(device)

CustomModel(
  (bert): DistilBertForSequenceClassification(
    (distilbert): DistilBertModel(
      (embeddings): Embeddings(
        (word_embeddings): Embedding(30522, 768, padding_idx=0)
        (position_embeddings): Embedding(512, 768)
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (transformer): Transformer(
        (layer): ModuleList(
          (0-5): 6 x TransformerBlock(
            (attention): MultiHeadSelfAttention(
              (dropout): Dropout(p=0.1, inplace=False)
              (q_lin): Linear(in_features=768, out_features=768, bias=True)
              (k_lin): Linear(in_features=768, out_features=768, bias=True)
              (v_lin): Linear(in_features=768, out_features=768, bias=True)
              (out_lin): Linear(in_features=768, out_features=768, bias=True)
            )
            (sa_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (ffn): FFN(
              (dropout): Dropout(p=0.1, inplace=False)
              (lin1): Linear(in_features=768, out_features=3072, bias=True)
              (lin2): Linear(in_features=3072, out_features=768, bias=True)
              (activation): GELUActivation()
            )
            (output_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          )
        )
      )
    )
    (pre_classifier): Linear(in_features=768, out_features=768, bias=True)
    (classifier): Linear(in_features=768, out_features=2, bias=True)
    (dropout): Dropout(p=0.2, inplace=False)
  )
  (feature_fc): Linear(in_features=4, out_features=64, bias=True)
  (relu): ReLU()
  (dropout): Dropout(p=0.3, inplace=False)
  (classifier): Linear(in_features=832, out_features=2, bias=True)
  (sigmoid): Sigmoid()
)
```

Figure 2: DistilBERT Model Architecture