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

Paper ID: 09

Presenting Author

Mr. L D M S Sai Teja

Co-Author(s): **Dr. Partha Pakray, Mr. Annepaka Yadagiri, and Mr. M Srikar Vardhan**

Outline

- Introduction
- Literature Survey
- Problem Statement and Dataset Description
- System Description
- Experimental Setup, Data Sampling and Model Selection
- Adversarial Detection
- MGT Classification
- Results
- Conclusion

Introduction

- Large Language Models (LLMs) have quickly established themselves as transformative tools in Natural Language Processing (NLP).

- Despite these advancements, ethical concerns have surfaced regarding inherent risks such as the potential for misinformation, hallucinations in generated outputs, and even biases against certain groups.

- Current detection methods are generally classified into three main categories: statistical approaches that use metrics like entropy, perplexity, and log-likelihood; neural classifiers trained on supervised datasets labeled as human or AI-generated; and watermarking techniques that embed subtle patterns into AI-generated text.
- The impact of adversarial attacks on detector performance in complex, real-world conditions is still largely unexamined.

Literature Survey

- **Adversarial Attacks:**

  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).

- 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.

- (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 AIgenerated.

Problem Statement and Dataset Description

- To develop a model capable of detecting and classifying text based on its domain of generation.
- **Dataset**:
  - The RAID dataset contains over 10 million documents spanning 11 LLMs, 11 genres, 4 decoding strategies, and 12 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.

Problem Statement and Dataset Description

- To develop a model capable of detecting and classifying text based on its domain of generation.
- **Dataset**:
  - The RAID dataset contains over 10 million documents spanning 11 LLMs, 11 genres, 4 decoding strategies, and 12 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.
- **Evaluation Metric**:
  - The official evaluation metric is TPR @ FPR=5%.
  - The accuracy of the model on detecting machine-generated text at a fixed false positive rate of 5%.

Dataset Description Cont.,

Dataset statistics:

- 11 LLMs, 11 genres, 4 decoding strategies, and 12 adversarial attacks.
- Adversarial techniques like paraphrasing, homoglyph, perplexity misspelling, synonyms, whitespace, upperlower, number, insert_paragraphs, article_deletion, alternative_ spelling, and zero_width_space.

Dataset Description Cont.,

Dataset statistics:

- 11 LLMs, 11 genres, 4 decoding strategies, and 12 adversarial attacks.
- Adversarial techniques like paraphrasing, homoglyph, perplexity misspelling, synonyms, whitespace, upperlower, number, insert_paragraphs, article_deletion, alternative_ spelling, and zero_width_space.

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.

Dataset Description Cont.,

Dataset statistics:

- 11 LLMs, 11 genres, 4 decoding strategies, and 12 adversarial attacks.
- Adversarial techniques like paraphrasing, homoglyph, perplexity misspelling, synonyms, whitespace, upperlower, number, insert_paragraphs, article_deletion, alternative_ spelling, and zero_width_space.

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.

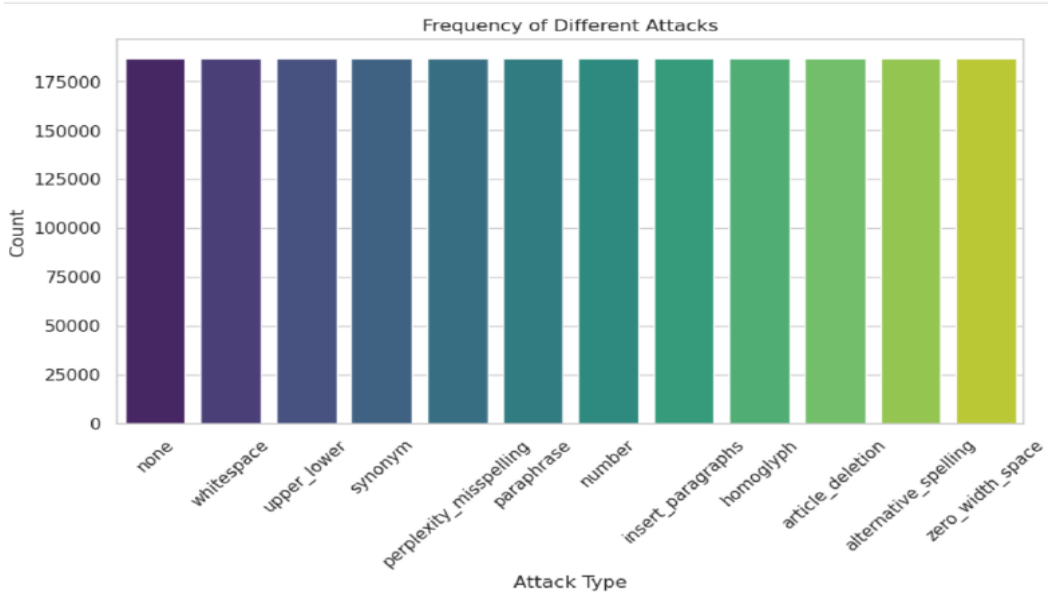


Fig 1: Frequency of Different attack types

System Description

Our methodology outlines an approach for detecting Machine- Generated Text (MGT), particularly focusing on both non-adversarial and adversarial scenarios in cross-domain text. This involves two main steps: **1) Adversarial Detection, 2) MGT Classification.**

- The analysis demonstrates that all attacks target plain text, with the dataset balanced across different attack types.
- The preprocessing pipeline is designed to clean and standardize attacked and non-attacked text.
- The preprocessing steps address these by applying transformations such as replacing homoglyphs with predefined mappings, normalizing alternative spellings (e.g., British vs. American English), converting numbers to words, removing extra or zero-width spaces, converting text to lowercase, and stripping punctuation.

System Description

Our methodology outlines an approach for detecting Machine- Generated Text (MGT), particularly focusing on both non-adversarial and adversarial scenarios in cross-domain text. This involves two main steps: **1) Adversarial Detection, 2) MGT Classification.**

- The analysis demonstrates that all attacks target plain text, with the dataset balanced across different attack types.
- The preprocessing pipeline is designed to clean and standardize attacked and non-attacked text.
- The preprocessing steps address these by applying transformations such as replacing homoglyphs with predefined mappings, normalizing alternative spellings (e.g., British vs. American English), converting numbers to words, removing extra or zero-width spaces, converting text to lowercase, and stripping punctuation.

```
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."
```

Fig 2: Text Preprocessing addressing Adversarial Attacks

Experimental Setup, Data Sampling and Model Selection

- The experiment was conducted in a Jupyter Notebook on a ma-chine with an Intel® Xeon® W-2155 CPU @ 3.30GHz, 20 cores, and an NVIDIA Quadro P2000 GPU, along with 64 GB of RAM. Python was used with libraries like Numpy, Pandas, SKlearn, and TensorFlow. To reduce computational load, 40% of the adversarial data was sampled based on unique adv source id, ensuring a manageable yet representative dataset. The sampled data was reset for indexing and prepared for the next pipeline step.

Experimental Setup, Data Sampling and Model Selection

- The experiment was conducted in a Jupyter Notebook on a ma-chine with an Intel® Xeon® W-2155 CPU @ 3.30GHz, 20 cores, and an NVIDIA Quadro P2000 GPU, along with 64 GB of RAM. Python was used with libraries like Numpy, Pandas, SKlearn, and TensorFlow. To reduce computational load, 40% of the adversarial data was sampled based on unique adv source id, ensuring a manageable yet representative dataset. The sampled data was reset for indexing and prepared for the next pipeline step.
- With a large dataset of over 10 million samples, we needed a model that balances performance and efficiency. DistilBERT, a faster and smaller variant of BERT, offers significant speed improvements while maintaining much of BERT's performance. Its reduced size ensures faster training and inference, making it ideal for handling large datasets without excessive computational costs.

Adversarial Detection

- For this, we used 4 methods namely: **Cosine Similarity, Cosine Similarity and Edit Distance, Word Overlap ratio, and Homoglyph Substitution Count.**

Adversarial Detection

- For this, we used 4 methods namely: **Cosine Similarity, Cosine Similarity and Edit Distance, Word Overlap ratio, and Homoglyph Substitution Count.**

❖ **Cosine Similarity** in NLP is a measure of similarity between two vectors by calculating the cosine of the angle between them, reflecting how similar the texts are in terms of their word vector representations.

Adversarial Detection

- For this, we used 4 methods namely: **Cosine Similarity, Cosine Similarity and Edit Distance, Word Overlap ratio, and Homoglyph Substitution Count.**

❖ **Cosine Similarity** in NLP is a measure of similarity between two vectors by calculating the cosine of the angle between them, reflecting how similar the texts are in terms of their word vector representations.

❖ **Levenshtein distance**, or edit distance, in NLP is the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one string into another.

Adversarial Detection

- For this, we used 4 methods namely: **Cosine Similarity, Cosine Similarity and Edit Distance, Word Overlap ratio, and Homoglyph Substitution Count.**

❖ **Cosine Similarity** in NLP is a measure of similarity between two vectors by calculating the cosine of the angle between them, reflecting how similar the texts are in terms of their word vector representations.

❖ **Levenshtein distance**, or edit distance, in NLP is the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one string into another.

❖ **Word overlap ratio** in NLP is a measure of similarity between two texts based on the proportion of common words shared between them relative to the total number of unique words in both texts.

Adversarial Detection

- For this, we used 4 methods namely: **Cosine Similarity, Cosine Similarity and Edit Distance, Word Overlap ratio, and Homoglyph Substitution Count.**

❖ **Cosine Similarity** in NLP is a measure of similarity between two vectors by calculating the cosine of the angle between them, reflecting how similar the texts are in terms of their word vector representations.

❖ **Levenshtein distance**, or edit distance, in NLP is the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one string into another.

❖ **Word overlap ratio** in NLP is a measure of similarity between two texts based on the proportion of common words shared between them relative to the total number of unique words in both texts.

❖ **Homoglyph substitution count** in NLP refers to the number of character substitutions where visually similar characters (homoglyphs) are swapped between two text strings, potentially affecting text similarity or detection.

Adversarial Detection Cont.,

Mathematical Formulae:

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

$$\text{Levenshtein Distance (LD)}(s_1, s_2) = \min \begin{Bmatrix} \text{Insert}, \\ \text{Delete}, \\ \text{Substitute} \end{Bmatrix} \tag{2}$$

$$S_{new\_hybrid} = \alpha \cdot \text{Cosine Similarity}(A, B) + (1 - \alpha) \cdot \left(1 - \frac{\text{Levenshtein Distance}(A, B)}{\max(\text{len}(A), \text{len}(B))}\right) \tag{3}$$

$$\text{Word Overlap Ratio} = \frac{|W_1 \cap W_2|}{|W_1 \cup W_2|} \tag{4}$$

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: **CosineSimilarity(x, x′) ≈ 1, 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: **CosineSimilarity(x, x′)≉ 1, when x≉ x′.**

Do these really matter, as combined with the embeddings of the pretrained Transformer?

Adversarial Attack	Cosine Similarity	Edit Distance	Word Overlap Ratio	Homoglyph Substitution Count
Article Deletion	✅	✅	✅	❌
Homoglyph Substitution	✅	✅	❌	✅
Number Swap	✅	✅	❌	❌
Paraphrase	✅	❌	✅	❌
Synonym Swap	✅	❌	✅	❌
Misspelling	✅	✅	❌	❌
Whitespace Addition	✅	✅	✅	❌
Upper-Lower Case Swap	✅	✅	❌	❌
Zero-Width Space	❌	✅	❌	❌
Insert Paragraphs	✅	✅	✅	❌
Alternative Spelling	✅	✅	❌	❌

MGT Classification

Model Setup: A fine-tuned DistilBERT model classifies human- vs machine-generated text using text embeddings combined with adversarial detection factors.

Training Details: Trained with batch size 16, for 3 epochs, using AdamW optimizer and CrossEntropyLoss.

Performance Metrics: Evaluated using accuracy, precision, recall, and F1 score.

Adversarial Robustness: The model is fine-tuned to remain robust against adversarial modifications.

Real-World Applicability: This approach ensures accurate classification in scenarios involving adversarially modified text.

MGT Classificatior

Model Setup: A fine-tuned DistilBERT model classifies human- vs machine-generated text using text embeddings combined with adversarial detection factors.

Training Details: Trained with batch size 16, for 3 epochs, using AdamW optimizer and CrossEntropyLoss.

Performance Metrics: Evaluated using accuracy, precision, recall, and F1 score.

Adversarial Robustness: The model is fine-tuned to remain robust against adversarial modifications.

Real-World Applicability: This approach ensures accurate classification in scenarios involving adversarially modified text.

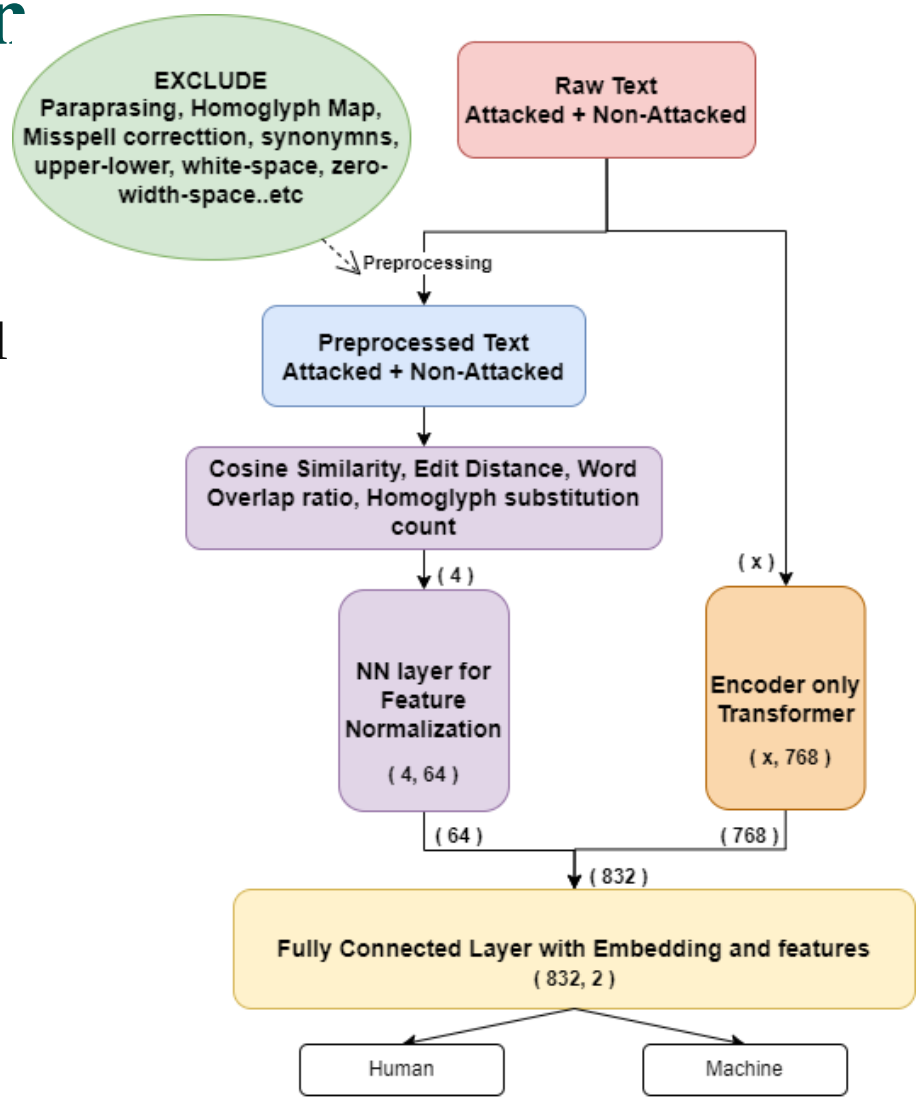
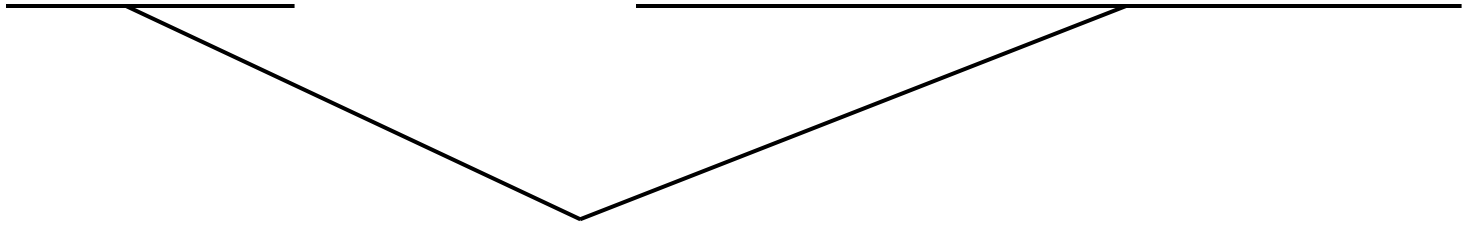


Fig 2: Architecture workflow

Raw Text	Label (Which model)
Both Attacked and Non-Attacked Text	

Raw Text	Label (Which model)
Both Attacked and Non-Attacked Text	

Raw Text	Preprocessed Text	CS	ED	WOR	HSC	LABEL
Both Attacked and Non-Attacked Text						



These will go into the model

Results

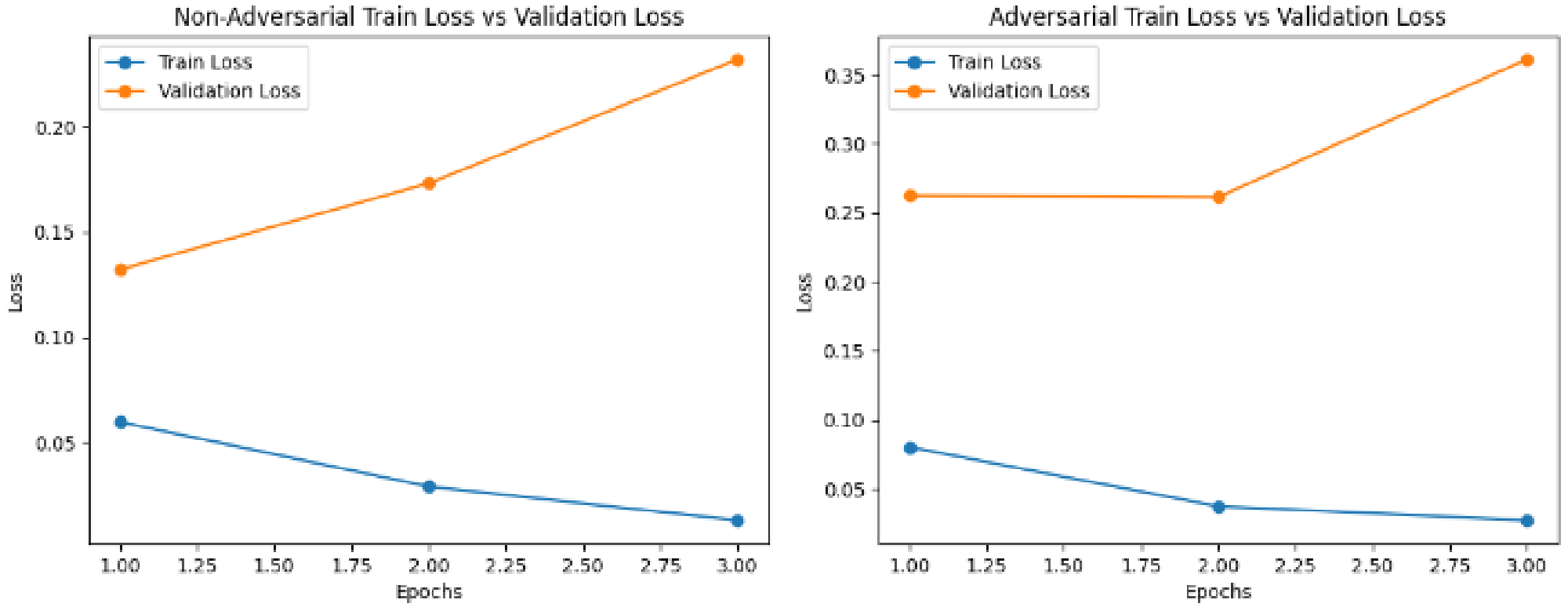


Fig 3: Non-Adversarial data and Adversarial Data Train loss vs Validation loss

Conclusion

- **Framework Development:** A robust framework was developed using a fine-tuned DistilBERT-NITS model to detect machine-generated text (MGT) across diverse domains, focusing on adversarial scenarios.
- **Adversarial Mitigation:** The approach includes preprocessing text to mitigate adversarial manipulations, enhancing detection accuracy.
- **Model Potential:** Demonstrates the capability of lightweight models in handling adversarial and cross-domain MGT detection effectively.
- **Future Focus:** Plans to refine the method to improve robustness and adaptability against evolving adversarial tactics.
- **Ranking Achievement:** Ranked 7th in non-adversarial detection and 17th in adversarial detection at the COLING Workshop.

Thank You.

From
Sai Teja