Statistical and Representation-Based Analysis of Human vs AI-Generated Text

Project Group:
Subhadip Baidya (221092)
Souptik Majumder (221079)
Anant Srivastava (220133)

Contents

1 Abstract

With the rapid advancement of large language models (LLMs), distinguishing between human-written and AI-generated text has become increasingly challenging. This project investigates whether a combination of **statistical entropy analysis**, **KL divergence–based layer selection**, and **deep learning classifiers** can effectively separate the two. We compute entropy metrics across human and AI text, extract layer-wise features using Qwen models, compute KL-based informativeness scores, and train binary classifiers (MLP, Random Forest, SVM). Our results show that statistical signals exist, and SVM achieves the strongest performance.

2 Overview

This project performs a comprehensive **statistical** and **representation-based** comparison between human-written text and AI-generated text.

Our goals:

- Analyse entropy characteristics of human vs AI text.
- Identify the most informative transformer layer using KL divergence.
- Extract embeddings from Qwen2-based models.
- Train binary classifiers on the extracted features.

We test whether combinations of these approaches can reliably distinguish between human and AI-generated content.

3 Problem Statement

The explosion of machine-generated content (text, images, videos) has amplified misinformation and synthetic media. One major challenge is:

How do we reliably detect whether a text was written by a human or an AI model?

This project explores statistical signatures and deep model representations that could help solve this problem.

4 Steps to Achieve the Goal

Step 1: Statistical Analysis

Compute entropy-based properties of text (token entropy, normalized entropy, sequence entropy) and visualize distributions.

Step 2: Deep Learning Approach

Use Qwen-1.5B/GTE-Qwen models to extract hidden layer features and compute KL divergence between layers to identify the most informative layer.

Final Goal: Train classifiers (MLP, RF, SVM) using selected layer embeddings to classify text as human or AI.

5 Data Sources

5.1 Human-Generated Text

Human-written samples were obtained from the Kaggle dataset[4]:

5.2 AI-Generated Text

Prompts were taken from the HuggingFace *WritingPrompts* dataset[5]:
Following Dataset contained texts from the model response.

- **deepseek-r1:latest**
- **qwen2.5:7b**
- **llama3.2**

6 Statistical Analysis (Entropy-Based)

We computed:

1. **Token entropy**
2. **Normalized entropy**
3. **Sequence-level entropy**

These provide measures of randomness and predictability in text sequences [3].

6.1 Visualizations

6.1.1 Human-written Text Entropy Distribution

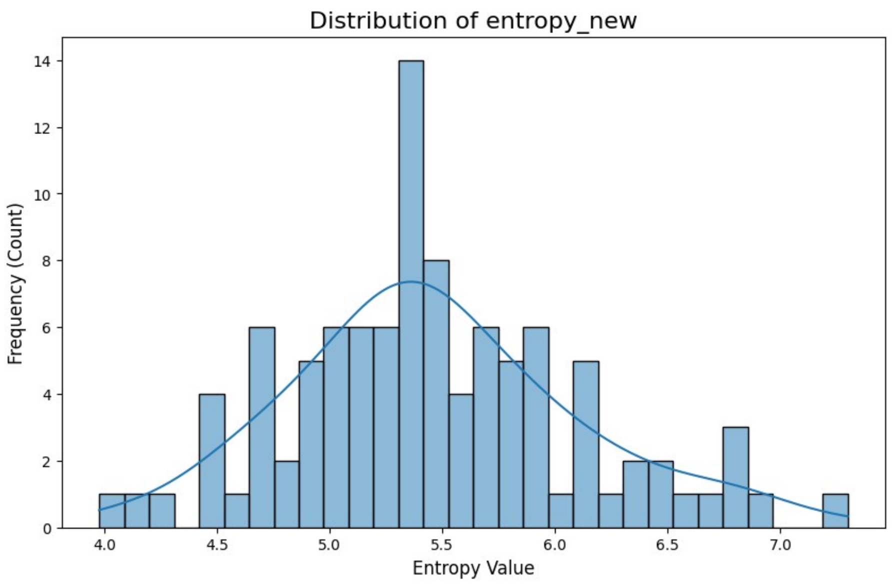

Figure 1: Entropy distribution for human-written text.

6.1.2 LLaMA 3.2 Text Entropy Distribution

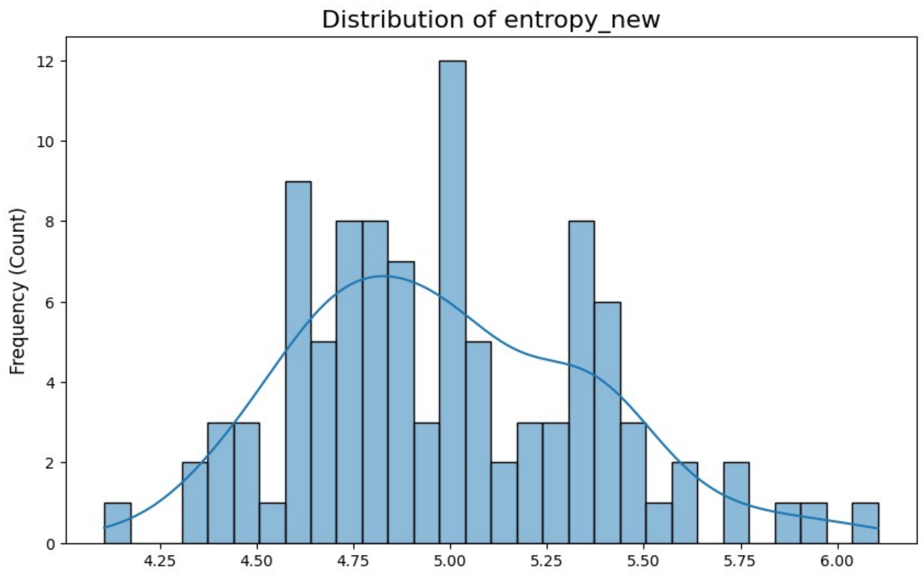

Figure 2: Entropy distribution for LLaMA 3.2 generated text.

6.1.3 Qwen2.5 7B Text Entropy Distribution

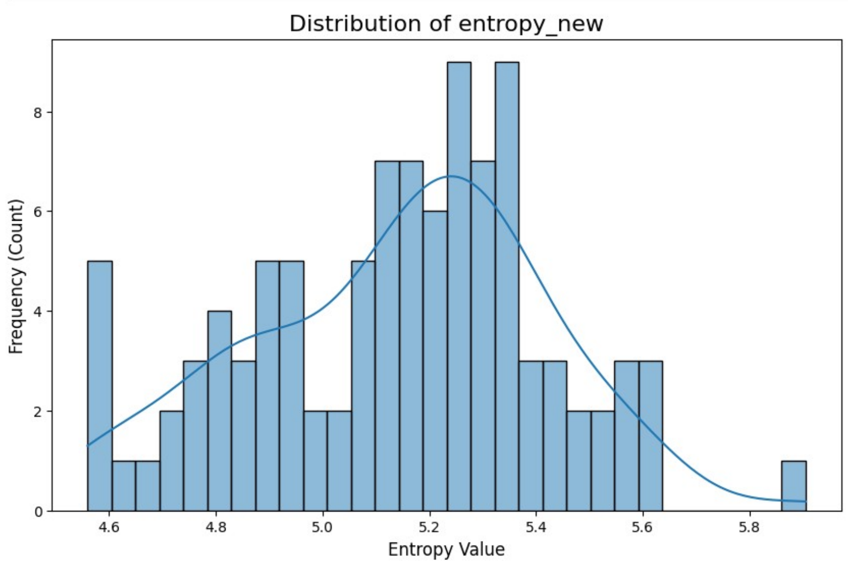

Figure 3: Entropy distribution for Qwen2.5 7B generated text.

7 Why Transformer Models Can Detect AI-Generated Text

The entropy distributions of human-written and AI-generated text exhibit fundamentally different trends, which helps explain why transformer-based methods such as Text Fluoroscopy are effective for the detection task.

7.1 Entropy Behaviour of Human vs. AI Text

AI-generated text shows a broad and irregular entropy distribution. The histogram displays noticeable fluctuations, multiple small peaks, and a long-tailed spread. This indicates that humans naturally vary their lexical choices, sentence structure, and token predictability. As a result, the entropy signal is more chaotic and heterogeneous.

In contrast, **Human-generated text** exhibits a much narrower, smoother, and more symmetric entropy distribution. The histogram forms a shape that is closer to a unimodal, Gaussian-like curve with significantly reduced variance. This is because large language models generate text through probability optimization, maintaining stable token-level uncertainty during generation. Sampling strategies (temperature, top-k, top-p) further constrain the entropy into a compact range.

7.2 How These Trends Relate to Transformer Internal Representations

Transformers process text through multiple layers, where:

- lower layers capture surface-level linguistic features,
- higher layers capture semantic meaning,
- **middle layers encode the compositional process that combines syntax and semantics**.

The key observation is that this compositional process differs significantly between humans and AI models. AI text, with its irregular entropy patterns, induces more varied and unstable internal activations across transformer layers. Human-generated text, with smoother and more uniform entropy, produces much more regular internal activation patterns.

7.3 Why KL Divergence Identifies the Most Informative Layer

Text Fluoroscopy compares the probability distributions of each layer to those of the first and last layers using KL divergence. Because Human text produces consistently smooth internal distributions while AI text induces greater variation, the middle layer with the highest divergence naturally highlights this difference. This layer effectively captures the *intrinsic writing signature* of human versus machine-generated text.

7.4 Overall Intuition

In summary, transformer-based detectors succeed because:

- AI generate text with high variability and entropy irregularity.
- Human models generate text with stable, low-variance entropy behaviour.
- These differences create distinct patterns inside the middle transformer layers.
- KL divergence is able to isolate the layer where these differences are most pronounced.

Thus, the contrasting shapes of the entropy distributions directly explain why transformers are able to distinguish AI-generated text using internal representations rather than superficial linguistic cues.

8 KL Divergence Layer Scoring (Text Fluoroscopy)

To find the most informative transformer layer M, we compute KL divergence[1] between:

- First layer distribution q_0
- Last layer distribution q_N
- Middle layer distribution q_j

$$\text{Score}(j) = D_{\text{KL}}(q_j \parallel q_0) + D_{\text{KL}}(q_j \parallel q_N)$$

The layer with highest score is selected:

$$M = \arg\max_j \text{Score}(j)$$

Then we extract the hidden state vector from layer M:

$$h_{t-1}^{(M)}$$

This encoded vector becomes the input to the binary classifier.

9 Text Fluoroscopy: Theory and Motivations

Text Fluoroscopy [2] is a recent method designed to detect LLM-generated text by using **intrinsic middle-layer features** rather than semantic (last-layer) or linguistic (n-gram) features. The motivation comes from the observation that transformer layers behave differently:

- The **first layers** capture low-level linguistic information.
- The **last layer** captures high-level semantic meaning.
- The **middle layers** encode the intermediate composition process where lexical patterns are transformed into coherent semantic structures.

This transition region between syntax and semantics is where human and AI text differ the most.

9.1 First vs Last vs Middle Layers

Previous approaches rely either on:

- **Semantic features** (final layer) — tend to overfit domains.
- **Linguistic features** — fragile to paraphrasing.

Text Fluoroscopy argues that **middle-layer intrinsic features** are more robust and generalizable.

9.2 Distribution Comparison Across Layers

For each transformer layer j, the vocabulary head $\phi(\cdot)$ is applied to obtain the next-token probability distribution:

$$q_j(x_t \mid x_{<t}) = \text{softmax}\big(\phi(h_t^{(j)})\big).$$

To measure how different the middle-layer distribution is from the first and last layers, KL divergence is computed:

$$KL_j = D_{\text{KL}}(q_j \parallel q_0) + D_{\text{KL}}(q_j \parallel q_N).$$

The most informative layer is selected as:

$$M = \arg\max_j KL_j.$$

This intrinsic layer representation $h_{t-1}^{(M)}$ becomes the final feature vector.

10 Our Implementation of Text Fluoroscopy

We implemented the full Text Fluoroscopy pipeline exactly as described in the paper. The following steps summarize our method.

10.1 Step 1: Extract Hidden States for All Layers

Using `Qwen/Qwen2-1.5B-Instruct`, we extract hidden states from all transformer layers:

$$H_j = \{h_0^{(j)}, \dots, h_{t-1}^{(j)}\}.$$

The vocabulary head is applied to each hidden state:

```
logits_j = model.lm_head(outputs.hidden_states[j])
```

We compute:

- q_0: distribution from the first layer,
- q_N: distribution from the final layer,
- q_j: distributions from all middle layers.

10.2 Step 2: KL Divergence Computation

For each middle layer:

```
kl_first = KL(q_j || q_0)
kl_last  = KL(q_j || q_N)
kl_score = kl_first + kl_last
```

This yields a KL score per layer per text sample.

10.3 Step 3: Select the Most Informative Layer

The intrinsic layer index is:

$$M = \arg\max_{j} KL_j.$$

Using the **gte-Qwen2-1.5B** embedding model, we extract concatenated embeddings:

$$E = [h^{(0)}, h^{(1)}, \ldots, h^{(N)}].$$

If each layer has dimension d, the selected intrinsic embedding is:

$$\mathbf{x}_{\text{final}} = E[M \cdot d : (M+1) \cdot d].$$

10.4 Step 5: Train Binary Classifiers

We trained three detectors:

- Multi-Layer Perceptron (MLP),
- Random Forest,
- SVM (best overall performance).

All models were trained on the KL-selected intrinsic embeddings.

11 Embedding Extraction Using GTE-Qwen2-1.5B

For each text, we extract hidden states from all layers:

$$E = [h^{(0)}, h^{(1)}, \ldots, h^{(N)}]$$

We use a custom pooling function `last_token_pool` to extract representations only from the last valid token.

Then we concatenate all layers:

$$\mathbf{X} = h^{(0)} \| h^{(1)} \| \ldots \| h^{(N)}$$

These embeddings are saved as `.pt` files.

11.1 KL-based Layer Selection

Given per-layer dimension d and total layers L:

$$\mathbf{X} \in \mathbb{R}^{dL}$$

The embedding for the best KL layer M is:

$$\mathbf{x}_{\text{final}} = \mathbf{X}[Md : (M+1)d]$$

12 Binary Classifier Architecture

After extracting the intrinsic middle-layer embeddings using the KL-divergence based selection method, we trained three classical machine learning models. Each model captures different aspects of the representation space and allows a diverse comparison of learning behavior.

12.1 Multi-Layer Perceptron (MLP)

The MLP serves as a non-linear neural classifier that learns a mapping from the intrinsic feature vector to the binary label (human vs. AI-generated text). It is composed of fully connected layers with ReLU activations, enabling the network to model complex non-linear relationships in the embedding space.

Architecture

- Input dimension: determined by the selected transformer layer.
- Hidden layers: two layers of sizes 1024 and 512.
- Activation function: ReLU.
- Regularization: Dropout with probability 0.2.
- Output layer: Fully connected layer with 2 logits.

The MLP is trained using cross-entropy loss:

$$\mathcal{L} = -\sum_{i=1}^{N} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}).$$

12.2 Random Forest Classifier

The Random Forest classifier is an ensemble-based method that constructs a large number of decision trees and aggregates their predictions. It is known for its robustness, ability to capture non-linear boundaries, and resistance to overfitting when sufficiently randomized.

Model Description

- Number of trees: 6000
- Maximum tree depth: 400
- Class weighting: balanced (to address label imbalance)
- Parallelization: 2 processing threads

Each tree is trained on a bootstrap sample of the training data, and feature splits are randomized at each node. The final prediction is obtained through majority voting. This ensemble approach stabilizes decision boundaries and prevents the model from memorizing noise present in individual samples.

12.3 Support Vector Machine (SVM) with RBF Kernel

The SVM is a margin-based classifier that attempts to find the optimal separating hyperplane between the two classes in a high-dimensional space. We use the Radial Basis Function (RBF) kernel to allow non-linear separation.

Model Description

- Kernel: RBF (Gaussian)
- Penalty parameter: $C = 20$
- Kernel scale: $\gamma = \text{scale}$
- Class weighting: balanced
- Probability output: enabled via `probability=True`

The RBF kernel projects the intrinsic embeddings into an infinite-dimensional feature space, allowing the SVM to capture complex geometric boundaries between human and AI-generated text. The margin maximization principle of SVMs makes the classifier less susceptible to overfitting compared to models that optimize prediction error directly.

12.4 Evaluation Metrics

Each classifier was evaluated using:

- Accuracy
- Precision
- Recall
- F1 Score
- AUROC
- Confusion Matrix

These metrics collectively measure the classifier's discriminative capability, robustness to class imbalance, and generalization performance across in-domain (validation) and out-of-domain (test) datasets.

13 Results

Table 1: Performance Metrics Across Models and Epochs

Model	**Epoc/estimators**	**Accuracy**	**Recall**	**Precision**	**F1 Score**	**AUROC**
MLP	10	0.8255	0.7651	0.8702	0.8142	**0.9109**
MLP	50	0.8555	0.8004	0.8889	0.8451	**0.9216**
Random Forest	600	0.7550	0.8926	0.7000	0.7846	**0.8876**
Random Forest	6000	0.8065	0.8926	0.7000	0.8308	**0.9390**
SVM (RBF)	10	0.8691	0.8121	0.9167	0.8612	**0.9589**

Table 2: Performance Metrics with Confusion Matrices for All Models

Model	**Dataset**	**Accuracy**	**Precision**	**Recall**	**F1**	**AUROC**	**Confusion Matrix**
MLP	Test: gpt4-pub-gpt3	0.5503	0.6271	0.2483	0.3557	**0.5935**	$\begin{bmatrix} 127 & 22 \\ 112 & 37 \end{bmatrix}$
	Test: gpt4-writing-gpt3	0.7700	0.8403	0.6667	0.7435	**0.8770**	$\begin{bmatrix} 118 & 32 \\ 35 & 115 \end{bmatrix}$
Random Forest	Test: gpt4-pub-gpt3	0.4731	0.4737	0.4833	0.4784	**0.4870**	$\begin{bmatrix} 87 & 62 \\ 87 & 65 \end{bmatrix}$
	Test: gpt4-writing-gpt3	0.7466	0.7434	0.7533	0.7483	**0.8611**	$\begin{bmatrix} 111 & 39 \\ 37 & 113 \end{bmatrix}$
SVM	Test: gpt4-pub-gpt3	0.5234	0.5479	0.2684	0.3604	**0.4952**	$\begin{bmatrix} 116 & 33 \\ 109 & 40 \end{bmatrix}$
	Test: gpt4-writing-gpt3	0.8300	0.9159	0.7267	0.8104	**0.8940**	$\begin{bmatrix} 143 & 7 \\ 48 & 102 \end{bmatrix}$

14 Conclusion and Inference

The performance comparison across MLP, Random Forest, and SVM reveals several important insights into how different classifiers exploit the intrinsic representations extracted through KL-based layer selection.

Model Performance on Validation Data

Across the validation split, the SVM with RBF kernel achieves the strongest overall performance, obtaining the highest Accuracy (0.8691), Precision (0.9167), and F1 score (0.8612), along with the best AUROC (0.9589). This indicates that the margin-based decision boundary of SVMs is particularly effective in separating human versus AI-generated text when high-dimensional, non-linear intrinsic representations are used.

The MLP shows competitive performance, especially at 50 epochs, achieving an AUROC of 0.9216. This suggests that the neural classifier is capable of learning non-linear patterns, but is slightly more sensitive to optimization noise and requires careful tuning to generalize well.

Random Forest performs reasonably on validation data but lags behind the other two models, even with a large ensemble size (6000 trees). This implies that tree-based methods struggle to fully capture the continuous geometry of the hidden-state embedding space compared to kernel-based or neural approaches.

Inference from Test-Set Performance

A key observation is the drop in performance on the "gpt4-pub-gpt3" test set for all models, with particularly reduced Recall and AUROC. This dataset is distributionally different from the validation set, indicating a domain shift. Here, SVM still maintains the most stable performance among the three models, demonstrating better robustness under domain variation.

On the "gpt4-writing-gpt3" test set, both the MLP and SVM recover strong performance, with the SVM again achieving the highest AUROC (0.8940) and F1 score (0.8104). This further supports the observation that SVMs generalize well when the test distribution partially aligns with the intrinsic structure learned during training.

In contrast, Random Forest exhibits the largest performance fluctuations between test sets, reflecting its limited ability to generalize across different text-generation conditions.

Overall Inference

From the combined results, several conclusions emerge:

- **SVM is the most reliable and robust classifier**, achieving the highest AUROC on both validation and test sets, and demonstrating strong generalization under distribution shift.
- **MLP performs well when trained sufficiently**, but is slightly more sensitive to training dynamics and less stable under domain changes.
- **Random Forest, while effective on in-domain samples**, shows clear limitations when dealing with high-dimensional continuous embeddings and performs noticeably worse on out-of-domain data.
- **The KL-based layer selection significantly boosts model separability**, as evidenced by all classifiers achieving AUROC values above 0.85 on aligned test sets, confirming that intrinsic middle-layer representations contain strong discriminative signals.
- **Performance drops on the gpt4-pub-gpt3 dataset confirm the importance of distributional robustness**, highlighting the need for future detectors to explicitly address domain shift.

Overall, the results validate that combining entropy analysis, intrinsic middle-layer extraction, and classical classifiers—particularly SVM—forms an effective framework for detecting AI-generated text. The strong AUROC values and stable cross-dataset performance indicate that transformer internal representations contain inherent stylistic and structural cues that clearly differentiate human and machine writing, even without relying on surface-level linguistic features.

References

[1] Xiao Yu1,2, Kejiang Chen1,2*, Qi Yang3 , Weiming Zhang1 , Nenghai Yu1 *Text Fluoroscopy: Detecting LLM-Generated Text through Intrinsic Features.* 2024. Available at: https://github.com/Fish-and-Sheep/Text-Fluoroscopy.

[2] Xiao Yu1,2, Kejiang Chen1,2*, Qi Yang3 , Weiming Zhang1 , Nenghai Yu1 https://aclanthology.org/2024.emnlp-main.885.pdf

[3] GLTR: Statistical Detection and Visualization of Generated Text Sebastian Gehrmann,Hendrik Strobelt,Alexander M. Rush https://aclanthology.org/P19-3019.pdf

[4] dataset available at: https://www.kaggle.com/datasets/shanegerami/ai-vs-human-text

[5] dataset available at: https://huggingface.co/datasets/euclaise/writingprompts