A Hybrid Approach for Detecting AI-Generated Text in English

Yanyang Gong, Yunfan Zhou
Department of Informatics, University of Zurich
{yanyang.gong, yunfan.zhou}@uzh.ch

Abstract

Detecting AI-generated text is becoming harder as generators improve and writing styles vary across domains. To address the COLING 2025 GenAI Content Detection Task 1 (English), we propose a hybrid detector that combines a train-free statistical signal with a train-based classifier, allowing the two to complement each other. Experiments show that the hybrid model consistently outperforms both individual baselines on validation and test datasets. We further analyze the model's performance across different domains and sources to assess its robustness.

1 Introduction

Rapid advances in large language models (LLMs) have made AI-generated text widespread, raising concerns about academic integrity and content attribution. At the same time, they blur the boundary between human- and machine-written text, making detection increasingly important and challenging.

This project therefore addresses the GenAI Content Detection Task 1 from COLING 2025 (Wang et al., 2025). The task consists of two subtasks: Monolingual subtask A and Multilingual subtask B. In this work, we focus on subtask A, which requires binary classification of English text as either AI-generated or human-written.

To address this task, we review two representative approaches for AI-generated text detection: train-based discriminative models (e.g., DeBERTa classifiers (Gritsai et al., 2025)) and train-free statistical methods (e.g., perplexity-based approaches (Zhu et al., 2025)). Train-based models have been the mainstream approach in this task. However, prior studies point out that these models often overfit to in-distribution features, resulting in poor generalization to out-of-distribution texts (Zhu et al., 2025). On the other hand, train-free methods are also vulnerable: their performance can be undermined when AI-generated text closely mimics human writing styles (Wang et al., 2025) or when applied across different domains.

Motivated by the limitations, our project aims to investigate: *Can we combine complementary signals from train-free and train-based methods to improve robustness and overall detection performance?* To answer this question, we propose a hybrid model that systematically combines these two approaches.

The main contribution of this paper can be summarized as follows: (1) We propose a hybrid classifier framework that combines train-based and train-free methods, leveraging their complementary strengths. (2) We show that the hybrid model consistently outperforms each individual method on both the validation and test sets, including the top-ranked train-based model in the COLING 2025 Task 1 and a train-free baseline. (3) We further examine classifier performance across domains, providing additional insights and analyzing model robustness accordingly.

2 Related work

Existing approaches for detecting machine-generated text can be broadly divided into train-based and train-free methods.

Train-based methods build classifiers using neural networks or fine-tune pre-trained models, such as RoBERTa (Solaiman et al., 2019), Llama (Doan and Inui, 2025), or DeBERTa (Gritsai et al., 2025). However, these methods often suffer from overfitting and poor generalization to out-of-distribution texts.

Train-free methods rely on statistical features without requiring model training. Representative approaches include DetectGPT (Mitchell et al., 2023), which leverages probability curvature, and DNA-DetectLLM (Zhu et al., 2025), which calculates cumulative repair effort to transform input into an "ideal" AI-generated sequence. A key advantage is their low deployment cost, though performance can vary substantially across domains and generation models.

Some methods incorporate statistical features in addition to classifiers. For instance, a recent study extracts linguistic and statistical markers and feeds them into a classifier such as Random Forest (Joseph et al., 2025). Our work follows a similar idea: starting from a train-based DeBERTa classifier (Gritsai et al., 2025), we incorporate a train-free statistical score (computed by DNA-DetectLLM (Zhu et al., 2025)) and fuse it with the model representations for final classification.

3 Methodology

3.1 Baseline Methods

DNA-DetectLLM (Zhu et al., 2025) is a train-free method designed to detect AI-generated text. The core idea is based on the concept of "mutation repair." For any given input text, an "ideal" AI-generated sequence is first constructed by selecting the most probable token at each position. The difference between the input text and the ideal sequence represents the "mutated positions." The cumulative effort required to "repair" these mutations is quantified as the repair score, which serves as the signal for classification. If the repair score exceeds a calibrated threshold, the text is classified as human-written; otherwise, it is classified as AI-generated.

The repair score is mathematically defined as:

$$R(s) = \frac{1}{T+1} \sum_{t=0}^{T} \sigma(s_t \mid s) \tag{1}$$

where s is the original sequence, s_t is the sequence after t repair steps, T is the total number of mutated tokens to be corrected, and $\sigma(s_t \mid s)$ represents the conditional score (calculated by log-perplexity and cross-perplexity) at each step. Alternatively, the score can be simplified as:

$$R(s) = \frac{1}{2} \left(\sigma(s) + \sigma(\hat{s} \mid s) \right), \tag{2}$$

where $\hat{s}$ is the ideal sequence. The proof of this simplification can be found in the original paper (Zhu et al., 2025).

This method demonstrates strong performance and robustness compared to state-of-the-art detection techniques. It is computationally efficient, as it does not require training, but the choice of the large language model (LLM) used for inference and perplexity calculation is critical. Additionally, the calibrated threshold significantly impacts performance and is sensitive to the dataset distribution

DeBERTa-Based Classifier (Gritsai et al., 2025) is proposed by Advacheck team, who participated in the GenAI Detection Task 1 competition and achieved the best performance in the English subtask. The model uses DeBERTa as the backbone and incorporates additional classification heads. These heads are not only used for binary classification (human-written vs. AI-generated text) but also for domain-specific classification within the dataset. This architecture is efficient, adaptable, and flexible, as the classification heads can be easily adjusted to suit specific datasets and tasks. However, the model's robustness is limited, as performance drops were observed on the test set. Additionally, the final classification also relies on a threshold.

Figures for the baseline models architectures can be found in Appendix B.

3.2 Proposed Hybrid Model

Inspired by the strengths of both DNA-DetectLLM and the DeBERTa-based classifier, we propose a hybrid model that combines their complementary advantages. Our goal is to design a model that is flexible (capable of leveraging pre-trained models), lightweight (able to use smaller LLMs for inference to reduce computational costs), and robust (enhanced by incorporating statistical signals from train-free methods).

The proposed hybrid model integrates the repair score from DNA-DetectLLM as an additional feature into the DeBERTa-based classifier. The architecture consists of the following steps:

- Repair Score Calculation: a set of lightweight LLM is used to calculate a simplified repair score for each input text.

- Feature-Level Integration: the repair score is concatenated with the DeBERTa pooled output embeddings as an additional feature.

- Modified DeBERTa Model: the DeBERTa-v3 model is modified to accept the repair score as an input feature.

- Pre-trained weights are loaded for the DeBERTa backbone, while the classification head is reinitialized to accommodate the additional repair score feature.

- Fine-Tuning: the classification layer is fine-tuned on a small dataset. Instead of relying on a threshold, the model uses the argmax function to output the most probable class.

Mathematically, the hybrid model can be expressed as:

$$y = \text{argmax}\left(W \cdot [f_{\text{DeBERTa}}(x), R(s)] + b\right) \quad (3)$$

where $f_{\text{DeBERTa}}(x)$ represents the pooled output embeddings from the pretrained DeBERTa model, $R(s)$ is the repair score.

4 Experiments

4.1 Dataset

We conduct experiments on the subsets of COLING 2025 MGT Detection dataset (English sub-task) (Wang et al., 2025), which consists of training, development, and test sets collected from multiple domains and sources. Notably, there exists a distribution shift between the training/development sets and the test set, as the test data are collected from a different source.

We apply source-stratified proportional sampling to construct balanced subsets that preserve the original distribution within computational constraints. Detailed dataset breakdowns are provided in Appendix A.

4.2 Implementation Details

DNA-DetectLLM (Baseline 1): We use two sets of lightweight language models (gpt2-medium and gpt2-large, EleutherAI/pythia-160m and EleutherAI/pythia-410m) to compute repair scores, which are then standardized using training set statistics. The classification threshold is optimized on the training set by maximizing F1-score.

DeBERTa-Based Classifier (Baseline 2): We employ the pretrained DeBERTa-v3-base model from Hugging Face Transformers (OU-Advacheck, 2024), optimizing its classification threshold on the validation set. We report the best result over multiple trials[1].

Proposed Hybrid Model: Our hybrid model integrates the DNA-DetectLLM repair score as an additional feature into the DeBERTa-v3-base architecture. The model's classification head was modified to accept concatenated inputs: the 768-dimensional pooled output from DeBERTa and the 1-dimensional normalized repair score, resulting in a 769-dimensional feature vector. The classification head consists of three linear layers (768 + 1, 512, 256, 2) with GELU activation and dropout (p=0.5).

During training, the DeBERTa backbone and pooler are frozen, and only the classification head is updated. We employ the AdamW optimizer with a learning rate of 3×10^{-4}, coupled with a linear learning rate schedule and 50 warm-up steps. Training is performed with a batch size of 32 for a single epoch. Unlike baseline methods, our hybrid model relies on the argmax operation for classification, eliminating the need for threshold.

All experiments were implemented in PyTorch using the Hugging Face Transformers library and conducted on Google Colab with T4 GPUs.

Evaluation Metrics: All models were evaluated using Accuracy and F1 Score.

[1]We note that the pretrained model's label mapping required verification, we tested both configurations and used the one yielding better performance

5 Results

5.1 Overall Results

Table 1 compares the performance of baseline and hybrid models on the Dev and Test sets. Both Hybrid-GPT and Hybrid-Pythia consistently outperform Baseline1 and Baseline2 across all metrics on both splits, demonstrating the effectiveness of the proposed hybrid approach.

A general performance degradation from Dev to Test is observed for most models, reflecting the distribution shift between the two sets and aligning with prior findings. Although the hybrid models maintain clear advantages over the baselines on the Test set, the performance gap between Dev and Test remains comparable to that of Baseline2, suggesting limited improvement in robustness to distribution shift.

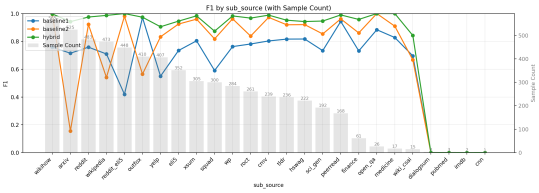
Hybrid-GPT performs better in baseline 1 than Hybrid-Pythia, indicates the performance of DNA-Detect LLM relies on the LLM used as we analyzed before, and they two achieve very similar results in hybrid model, indicating that lightweight LLMs with comparable capacities yield similar gains in the hybrid framework. Finally, we observed unstable performance of Baseline2, thus we report the best result obtained across multiple independent runs.

5.2 Finer Analysis by Sub Source

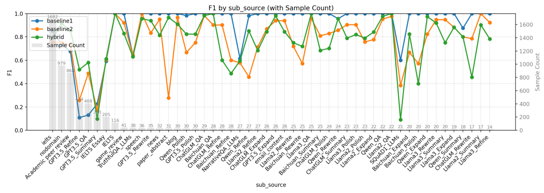
We further evaluate the performance by sub_source (domain for test set). Figure 1 presents the F1

Model	Dev		Test	
	F1	Accuracy	F1	Accuracy
Baseline1 (GPT)	74.61	60.34	73.81	63.06
Baseline1 (Pythia)	74.52	60.20	70.14	55.72
Baseline2	84.07	81.26	73.93	67.46
Hybrid-GPT	**96.50**	**95.60**	80.10	76.52
Hybrid-Pythia	96.43	95.52	**80.42**	**77.02**

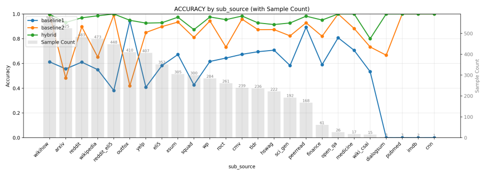
Table 1: Performance comparison of baseline and hybrid models on the Dev and Test sets. For Baseline-2, we report the best performance over multiple trials.



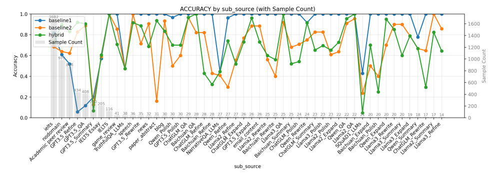
(a) F1 score by sub_source over dev set of HybridGPT



(b) F1 score by sub_source over test set of HybridGPT



(c) Accuracy by sub_source over dev set of Hybrid-GPT



(d) Accuracy by sub_source over test set of Hybrid-GPT

Figure 1: Performance by sub_source of HybridGPT

scores of the baseline models and the hybrid model (GPT) across different sub_sources, with bars indicating the sample counts for the validation and test sets (the train set exhibits a similar distribution to the validation set, see Appendix A).

On the dev set, the hybrid models consistently outperform the baselines across all sub_sources. On the Test set, however, baseline1 achieves the best performance in most sub_sources, except for the major sub_sources with more samples, suggesting that baseline1 is relatively robust to distribution shifts, whereas the hybrid model still relies on training data for optimal performance.

We also evaluated model performance across multiple metrics, both by sub_source and by model. Additional results and detailed breakdowns are provided in the Appendix C.

6 Conclusion and Limitations

In this work, we proposed a hybrid framework for AI-generated text detection that combines train-free statistical scores with train-based discriminative models. Across both the Dev and Test sets, the hybrid models consistently outperform both baselines, showing that signals from these two types of methods can be effectively integrated.

A key advantage of our approach is that it achieves performance improvements using relatively lightweight models without requiring complex architectural modifications or extensive retraining. However, our results also show that while the hybrid approach achieves stable improvements on domains with sufficient training data, it shows limited gains or even underperforms baselines under low-resource or out-of-domain conditions, highlighting remaining sensitivity to distribution shift.

Finally, this work has several limitations. First, due to computational resource constraints, we trained and evaluated our models on a limited subset of the data. As a result, our results are not directly comparable to the official competition submissions, which use the full dataset. In addition, the detector models used in our experiments are relatively small compared to state-of-the-art LLMs, which may limit the performance of our approach. Future work could explore scaling the hybrid framework to larger detector architectures and training on larger and more diverse datasets.

References

Nhi H. Doan and Kentaro Inui. 2025. Grape at genai detection task 1: Leveraging compact models and linguistic features for robust machine-generated text detection. In *Proceedings of the 1st Workshop on GenAI Content Detection (GenAIDetect)*, pages 209–217.

G. Gritsai, A. Voznuyk, I. Khabutdinov, and A. Grabovoy. 2025. Advacheck at genai detection task 1: AI detection powered by domain-aware multi-tasking. In *Proceedings of the 1st Workshop on GenAI Content Detection (GenAIDetect)*, pages 236–243.

Emmanuel Joseph, Micheal Bennet, and Thomas Kingsley. 2025. Feature-based detection of ai-generated text: An analysis of stylometric and perplexity markers in contemporary large language models. ResearchGate. Preprint.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pages 24950–24962. PMLR.

OU-Advacheck. 2024. Deberta-v3-base fine-tuned for ai-generated text detection. https://huggingface.co/OU-Advacheck/deberta-v3-base-daigenc-mgt1a. Hugging Face model.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, and 1 others. 2019. Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*.

Yuxia Wang, Artem Shelmanov, Jonibek Mansurov, Akim Tsvigun, Vladislav Mikhailov, Rui Xing, Zhuohan Xie, Jiahui Geng, Giovanni Puccetti, Ekaterina Artemova, Jinyan Su, Minh Ngoc Ta, Mervat Abassy, Kareem Ashraf Elozeiri, Saad El Dine Ahmed El Etter, Maiya Goloburda, Tarek Mahmoud, Raj Vardhan Tomar, Nurkhan Laiyk, and 7 others. 2025. Genai content detection task 1: English and multilingual machine-generated text detection: Ai vs. human. In *Proceedings of the 1st Workshop on GenAI Content Detection (GenAIDetect)*, pages 244–261, Abu Dhabi, UAE. International Conference on Computational Linguistics.

Xiaowei Zhu, Yubing Ren, Fang Fang, Qingfeng Tan, Shi Wang, and Yanan Cao. 2025. DNA-detectLLM: Unveiling AI-generated text via a DNA-inspired mutation-repair paradigm. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

A Dataset Breakdown Analysis

This section provides additional statistics of the dataset.

=== Train + Val Sub_source Statistics ===

Sub_source	Train	Val
arxiv	1751 (8.76%)	525 (8.75%)
cmv	752 (3.76%)	239 (3.98%)
cnn	6 (0.03%)	1 (0.02%)
dialogsum	12 (0.06%)	3 (0.05%)
eli5	1193 (5.97%)	352 (5.87%)
finance	193 (0.97%)	61 (1.02%)
hswag	851 (4.26%)	222 (3.70%)
imdb	9 (0.05%)	2 (0.03%)
medicine	56 (0.28%)	17 (0.28%)
open_qa	120 (0.60%)	26 (0.43%)
outfox	1437 (7.19%)	410 (6.84%)
peerread	665 (3.33%)	168 (2.80%)
pubmed	9 (0.05%)	3 (0.05%)
reddit	1756 (8.78%)	483 (8.05%)
reddit_eli5	1488 (7.44%)	448 (7.47%)
roct	844 (4.22%)	261 (4.35%)
sci_gen	704 (3.52%)	192 (3.20%)
squad	969 (4.85%)	300 (5.00%)
tldr	682 (3.41%)	236 (3.93%)
wiki_csai	36 (0.18%)	15 (0.25%)
wikihow	1757 (8.79%)	565 (9.42%)
wikipedia	1405 (7.03%)	473 (7.89%)
wp	954 (4.77%)	284 (4.73%)
xsum	967 (4.84%)	305 (5.09%)
yelp	1383 (6.92%)	407 (6.79%)
TOTAL	19999	5998

Sub_source	Train	Val
13B	157 (0.79%)	57 (0.95%)
30B	180 (0.90%)	55 (0.92%)
65B	191 (0.96%)	47 (0.78%)
7B	175 (0.88%)	45 (0.75%)
GLM130B	146 (0.73%)	44 (0.73%)
bloom_7b	154 (0.77%)	55 (0.92%)
bloomz	654 (3.27%)	208 (3.47%)
cohere	694 (3.47%)	172 (2.87%)
davinci	609 (3.05%)	185 (3.08%)
dolly	299 (1.50%)	89 (1.48%)
dolly-v2-12b	65 (0.33%)	18 (0.30%)
flan_t5_base	191 (0.96%)	65 (1.08%)
flan_t5_large	220 (1.10%)	63 (1.05%)
flan_t5_small	230 (1.15%)	69 (1.15%)
flan_t5_xl	199 (1.00%)	73 (1.22%)
flan_t5_xxl	218 (1.09%)	74 (1.23%)
gemma-7b-it	388 (1.94%)	113 (1.88%)
gemma2-9b-it	274 (1.37%)	88 (1.47%)
gpt-3.5-turbo	1340 (6.70%)	410 (6.84%)
gpt-35	609 (3.05%)	175 (2.92%)
gpt4	364 (1.82%)	95 (1.58%)
gpt4o	340 (1.70%)	102 (1.70%)
gpt_j	118 (0.59%)	27 (0.45%)
gpt_neox	156 (0.78%)	49 (0.82%)
human	7505 (37.53%)	2303 (38.40%)
llama3-70b	430 (2.15%)	107 (1.78%)
llama3-8b	397 (1.99%)	134 (2.23%)
mixtral-8x7b	524 (2.62%)	149 (2.48%)
opt_1.3b	184 (0.92%)	40 (0.67%)
opt_125m	191 (0.96%)	48 (0.80%)
opt_13b	168 (0.84%)	57 (0.95%)
opt_2.7b	179 (0.90%)	57 (0.95%)
opt_30b	200 (1.00%)	47 (0.78%)
opt_350m	177 (0.89%)	54 (0.90%)
opt_6.7b	206 (1.03%)	45 (0.75%)
opt_iml_30b	196 (0.98%)	52 (0.87%)
opt_iml_max_1.3b	170 (0.85%)	72 (1.20%)
t0_11b	197 (0.99%)	66 (1.10%)
t0_3b	205 (1.03%)	76 (1.27%)
text-davinci-002	497 (2.49%)	140 (2.33%)
text-davinci-003	602 (3.01%)	173 (2.88%)
TOTAL	19999	5998

Figure 2: Sample distribution across sub_sources and models for the train and validation sets.

=== Test Sub_source Statistics ===

Sub_source	Test
Academic peer review	868 (14.48%)
Baichuan_Expand	20 (0.33%)
Baichuan_Polish	20 (0.33%)
Baichuan_QA	28 (0.47%)
Baichuan_Refine	28 (0.47%)
Baichuan_Rewrite	25 (0.42%)
Baichuan_Summary	25 (0.42%)
ChatGLM_Expand	26 (0.43%)
ChatGLM_Polish	24 (0.40%)
ChatGLM_QA	29 (0.48%)
ChatGLM_Refine	28 (0.47%)
ChatGLM_Rewrite	17 (0.28%)
ChatGLM_Summary	23 (0.38%)
GPT3.5_Expand	26 (0.43%)
GPT3.5_Polish	30 (0.50%)
GPT3.5_QA	408 (6.80%)
GPT3.5_Refine	434 (7.24%)
GPT3.5_Rewrite	35 (0.58%)
GPT3.5_Summary	240 (4.00%)
IELTS	116 (1.93%)
IELTS Essay	205 (3.42%)
Llama2_Expand	22 (0.37%)
Llama2_Polish	23 (0.38%)
Llama2_QA	22 (0.37%)
Llama2_Refine	27 (0.45%)
Llama2_Rewrite	25 (0.42%)
Llama2_Summary	17 (0.28%)
Llama3_Expand	19 (0.32%)
Llama3_Polish	23 (0.38%)
Llama3_QA	25 (0.42%)
Llama3_Refine	14 (0.23%)
Llama3_Rewrite	20 (0.33%)
Llama3_Summary	20 (0.33%)
NarrativeQA_LLMs	27 (0.45%)
Qwen_Expand	20 (0.33%)
Qwen_Polish	30 (0.50%)
Qwen_QA	22 (0.37%)
Qwen_Refine	27 (0.45%)
Qwen_Rewrite	24 (0.40%)
Qwen_Summary	18 (0.30%)
SQuAD1_LLMs	21 (0.35%)
TruthfulQA_LLMs	38 (0.63%)
blog	30 (0.50%)
email_content	26 (0.43%)
game_review	41 (0.68%)
ielts	1682 (28.05%)
news	32 (0.53%)
nodomain	979 (16.33%)
paper_abstract	31 (0.52%)
speech	36 (0.60%)
TOTAL	5996

Sub_source	Test
Baichuan	146 (2.43%)
ChatGLM	165 (2.75%)
ChatGPT	14 (0.23%)
ChatGPT-turbo	11 (0.18%)
Dolly	18 (0.30%)
GPT3.5	174 (2.90%)
GPT4	7 (0.12%)
Llama2	136 (2.27%)
Llama3	121 (2.02%)
Qwen	141 (2.35%)
StableLM	18 (0.30%)
gpt-4o	436 (7.27%)
gpt-4o-2024-08-06	281 (4.69%)
gpt-4o-mini-2024-07-18	1135 (18.93%)
human	2816 (46.96%)
llama-3.1-70B-versatile	65 (1.08%)
llama-3.1-70b-versatile	116 (1.93%)
nodomain	196 (3.27%)
TOTAL	5996

Figure 3: Sample distribution across sub_sources and models for the test set.

B Baseline Model Architecture

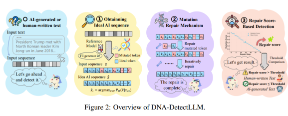


Figure 4: Architecture of the train-free baseline mode. Adapted from (Zhu et al., 2025)

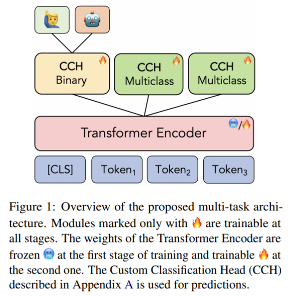


Figure 5: Architecture of the train-based baseline mode. Adapted from (OU-Advacheck, 2024)

C Additional Performance Metrics

This section presents additional evaluation metrics for the hybrid model, including performance breakdowns by generation model and detailed precision/recall analysis across sub_sources.

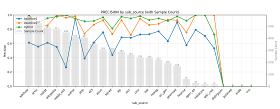
	model	count	accuracy	precision	recall	f1
0	human	2303	0.909683	0.0	0.000000	0.000000
1	gpt-3.5-turbo	410	0.997561	1.0	0.997561	0.998779
2	bloomz	208	0.995192	1.0	0.995192	0.997590
3	davinci	185	0.994595	1.0	0.994595	0.997290
4	gpt-35	175	1.000000	1.0	1.000000	1.000000
5	text-davinci-003	173	0.988439	1.0	0.988439	0.994186
6	cohere	172	1.000000	1.0	1.000000	1.000000
7	mixtral-8x7b	149	1.000000	1.0	1.000000	1.000000
8	text-davinci-002	140	0.985714	1.0	0.985714	0.992806
9	llama3-8b	134	1.000000	1.0	1.000000	1.000000
10	gemma-7b-it	113	1.000000	1.0	1.000000	1.000000
11	llama3-70b	107	1.000000	1.0	1.000000	1.000000
12	gpt4o	102	1.000000	1.0	1.000000	1.000000
13	gpt4	95	1.000000	1.0	1.000000	1.000000
14	dolly	89	0.977528	1.0	0.977528	0.988636
15	gemma2-9b-it	88	1.000000	1.0	1.000000	1.000000
16	t0_3b	76	0.960526	1.0	0.960526	0.979866
17	flan_t5_xxl	74	0.959459	1.0	0.959459	0.979310
18	flan_t5_xl	73	0.849315	1.0	0.849315	0.918519
19	opt_iml_max_1.3b	72	0.972222	1.0	0.972222	0.985915
20	flan_t5_small	69	1.000000	1.0	1.000000	1.000000
21	t0_11b	66	0.939394	1.0	0.939394	0.968750
22	flan_t5_base	65	0.923077	1.0	0.923077	0.960000
23	flan_t5_large	63	0.888889	1.0	0.888889	0.941176
24	13B	57	0.964912	1.0	0.964912	0.982143
25	opt_2.7b	57	1.000000	1.0	1.000000	1.000000
26	opt_13b	57	1.000000	1.0	1.000000	1.000000
27	30B	55	0.981818	1.0	0.981818	0.990826
28	bloom_7b	55	1.000000	1.0	1.000000	1.000000
29	opt_350m	54	1.000000	1.0	1.000000	1.000000
30	opt_iml_30b	52	0.942308	1.0	0.942308	0.970297
31	gpt_neox	49	1.000000	1.0	1.000000	1.000000
32	opt_125m	48	1.000000	1.0	1.000000	1.000000
33	65B	47	0.978723	1.0	0.978723	0.989247
34	opt_30b	47	1.000000	1.0	1.000000	1.000000
35	opt_6.7b	45	0.977778	1.0	0.977778	0.988764
36	7B	45	1.000000	1.0	1.000000	1.000000
37	GLM130B	44	0.909091	1.0	0.909091	0.952381
38	opt_1.3b	40	1.000000	1.0	1.000000	1.000000
39	gpt_j	27	1.000000	1.0	1.000000	1.000000
40	dolly-v2-12b	18	1.000000	1.0	1.000000	1.000000

(a) Validation set

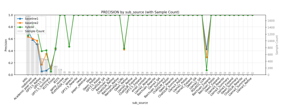
	model	count	accuracy	precision	recall	f1
0	human	2816	0.622514	0.000000	0.000000	0.000000
1	gpt-4o-mini-2024-07-18	1135	1.000000	1.000000	1.000000	1.000000
2	gpt-4o	436	1.000000	1.000000	1.000000	1.000000
3	gpt-4o-2024-08-06	281	1.000000	1.000000	1.000000	1.000000
4	nodomain	196	0.801020	1.000000	0.801020	0.889518
5	GPT3.5	174	0.775862	1.000000	0.775862	0.873786
6	ChatGLM	165	0.587879	0.888889	0.631579	0.738462
7	Baichuan	146	0.568493	1.000000	0.568493	0.724891
8	Qwen	141	0.815603	1.000000	0.815603	0.898438
9	Llama2	136	0.705882	1.000000	0.705882	0.827586
10	Llama3	121	0.776860	1.000000	0.776860	0.874419
11	llama-3.1-70b-versatile	116	1.000000	1.000000	1.000000	1.000000
12	llama-3.1-70B-versatile	65	0.984615	1.000000	0.984615	0.992248
13	Dolly	18	0.444444	0.470588	0.888889	0.615385
14	StableLM	18	0.277778	0.416667	0.454545	0.434783
15	ChatGPT	14	0.428571	0.416667	0.833333	0.555556
16	ChatGPT-turbo	11	0.363636	0.363636	1.000000	0.533333
17	GPT4	7	0.428571	0.500000	0.750000	0.600000

(b) Test set

Figure 6: Hybrid-GPT performance by generation model on validation and test sets.

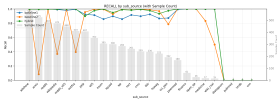


(a) Validation set

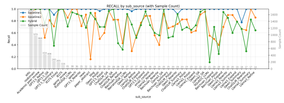


(b) Test set

Figure 7: Precision comparison by sub_source for Hybrid-GPT across validation and test sets.



(a) Validation set



(b) Test set

Figure 8: Recall comparison by sub_source for Hybrid-GPT across validation and test sets.

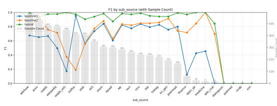
	model	count	accuracy	precision	recall	f1
0	human	2303	0.909249	0.0	0.000000	0.000000
1	gpt–3.5–turbo	410	0.997561	1.0	0.997561	0.998779
2	bloomz	208	0.995192	1.0	0.995192	0.997590
3	davinci	185	0.994595	1.0	0.994595	0.997290
4	gpt–35	175	1.000000	1.0	1.000000	1.000000
5	text–davinci–003	173	0.988439	1.0	0.988439	0.994186
6	cohere	172	1.000000	1.0	1.000000	1.000000
7	mixtral–8x7b	149	1.000000	1.0	1.000000	1.000000
8	text–davinci–002	140	0.985714	1.0	0.985714	0.992806
9	llama3–8b	134	1.000000	1.0	1.000000	1.000000
10	gemma–7b–it	113	1.000000	1.0	1.000000	1.000000
11	llama3–70b	107	1.000000	1.0	1.000000	1.000000
12	gpt4o	102	1.000000	1.0	1.000000	1.000000
13	gpt4	95	1.000000	1.0	1.000000	1.000000
14	dolly	89	0.966292	1.0	0.966292	0.982857
15	gemma2–9b–it	88	1.000000	1.0	1.000000	1.000000
16	t0_3b	76	0.960526	1.0	0.960526	0.979866
17	flan_t5_xxl	74	0.959459	1.0	0.959459	0.979310
18	flan_t5_xl	73	0.835616	1.0	0.835616	0.910448
19	opt_iml_max_1.3b	72	0.972222	1.0	0.972222	0.985915
20	flan_t5_small	69	1.000000	1.0	1.000000	1.000000
21	t0_11b	66	0.939394	1.0	0.939394	0.968750
22	flan_t5_base	65	0.923077	1.0	0.923077	0.960000
23	flan_t5_large	63	0.888889	1.0	0.888889	0.941176
24	13B	57	0.964912	1.0	0.964912	0.982143
25	opt_2.7b	57	1.000000	1.0	1.000000	1.000000
26	opt_13b	57	1.000000	1.0	1.000000	1.000000
27	30B	55	0.963636	1.0	0.963636	0.981481
28	bloom_7b	55	1.000000	1.0	1.000000	1.000000
29	opt_350m	54	1.000000	1.0	1.000000	1.000000
30	opt_iml_30b	52	0.942308	1.0	0.942308	0.970297
31	gpt_neox	49	1.000000	1.0	1.000000	1.000000
32	opt_125m	48	1.000000	1.0	1.000000	1.000000
33	65B	47	0.978723	1.0	0.978723	0.989247
34	opt_30b	47	1.000000	1.0	1.000000	1.000000
35	opt_6.7b	45	0.977778	1.0	0.977778	0.988764
36	7B	45	0.977778	1.0	0.977778	0.988764
37	GLM130B	44	0.909091	1.0	0.909091	0.952381
38	opt_1.3b	40	1.000000	1.0	1.000000	1.000000
39	gpt_j	27	1.000000	1.0	1.000000	1.000000
40	dolly–v2–12b	18	1.000000	1.0	1.000000	1.000000

(a) Validation set

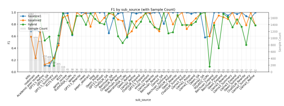
	model	count	accuracy	precision	recall	f1
0	human	2816	0.634588	0.000000	0.000000	0.000000
1	gpt–4o–mini–2024–07–18	1135	1.000000	1.000000	1.000000	1.000000
2	gpt–4o	436	1.000000	1.000000	1.000000	1.000000
3	gpt–4o–2024–08–06	281	1.000000	1.000000	1.000000	1.000000
4	nodomain	196	0.790816	1.000000	0.790816	0.883191
5	GPT3.5	174	0.781609	1.000000	0.781609	0.877419
6	ChatGLM	165	0.581818	0.887850	0.625000	0.733591
7	Baichuan	146	0.554795	1.000000	0.554795	0.713656
8	Qwen	141	0.808511	1.000000	0.808511	0.894118
9	Llama2	136	0.720588	1.000000	0.720588	0.837607
10	Llama3	121	0.768595	1.000000	0.768595	0.869159
11	llama–3.1–70b–versatile	116	1.000000	1.000000	1.000000	1.000000
12	llama–3.1–70B–versatile	65	0.984615	1.000000	0.984615	0.992248
13	Dolly	18	0.444444	0.470588	0.888889	0.615385
14	StableLM	18	0.277778	0.416667	0.454545	0.434783
15	ChatGPT	14	0.428571	0.416667	0.833333	0.555556
16	ChatGPT–turbo	11	0.363636	0.363636	1.000000	0.533333
17	GPT4	7	0.428571	0.500000	0.750000	0.600000

(b) Test set

Figure 9: Hybrid-Pythia performance by generation model on validation and test sets.

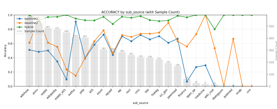


(a) Validation set

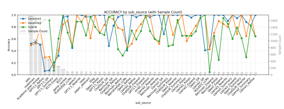


(b) Test set

Figure 10: F1 score comparison by sub_source for Hybrid-Pythia across validation and test sets.

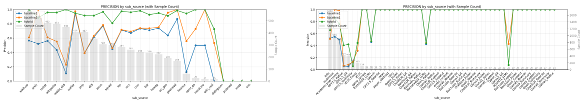


(a) Validation set



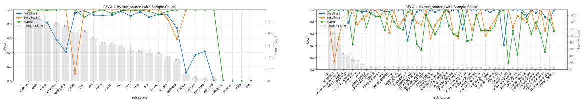
(b) Test set

Figure 11: Accuracy comparison by sub_source for Hybrid-Pythia across validation and test sets.



(a) Validation set (b) Test set

Figure 12: Precision comparison by sub_source for Hybrid-Pythia across validation and test sets.



(a) Validation set (b) Test set

Figure 13: Recall comparison by sub_source for Hybrid-Pythia across validation and test sets.