Conditional Kolmogorov Complexity for AI-Generated Text Detection

Anonymous ACL submission

Abstract

Detecting AI-generated text is increasingly challenging as large language models produce fluent, stylistically adaptive outputs. Existing detectors often rely on supervised neural classifiers or statistical signals that require a heavy training procedure. We introduce CK-Detect, a lightweight framework that offers a practical solution based on the estimation of conditional Kolmogorov complexity through lossless compression. CK-Detect estimates residual complexity after conditioning on LLM-generated rewrites and class-specific AI-generated and human-written corpora, capturing both rewrite stability and class alignment. These combined features are fed to a binary logistic classifier. Across six domains, CK-Detect achieves competitive performance against zero-shot statistical and supervised detector baselines. Some extensive experiments show that the proposed CK-Detect can deal with various adversarial settings, including some that may intend to bypass the detection through cheap modifications. Successfully detecting AI-generated texts provides the foundation for healthy data usage and human-AI interactions in years to come.

1 Introduction

Modern large language models (LLMs) can generate text that appears human, but their outputs still reflect the statistical regularities of the distributions that produce them. As these models grow increasingly fluent, coherent, and stylistically versatile, traditional detection methods that rely on surface-level artifacts, such as grammatical errors, unnatural phrasing, or incoherence, become ineffective. This shift introduces challenges for academic integrity, online trust, authorship verification, misinformation mitigation, and high-stakes decision-making.

Current detectors are typically supervised or zero-shot. Supervised methods train on labeled human-written and AI-generated corpora, including neural encoders such as RoBERTa (Liu et al., 2019), the OpenAI RoBERTa-base detector (Solaiman et al., 2019), feature-based systems such as Ghostbuster (Verma et al., 2024), and rewrite-based detectors such as Raidar (Mao et al., 2024). They can perform well in-domain but may degrade under domain, style, or generator shift (Verma et al., 2024; Mao et al., 2024). Zero-shot methods instead use statistical signals such as likelihood, log-rank, entropy, probability curvature, cross-perplexity, or regeneration n-gram divergence (Gehrmann et al., 2019; Su et al., 2023; Bao et al., 2024; Ippolito et al., 2020; Mitchell et al., 2023; Hans et al., 2024; Yang et al., 2023), but they often require model-specific probabilities or access to scoring language models.

These limitations motivate the use of a text-instance-level detection signal. Rather than asking how probable a text is under a language model, we estimate how much residual information is needed to describe it given an appropriate reference. We use conditional Kolmogorov complexity (Li and Vitányi, 2019), approximated by lossless compression, as a likelihood-free representation that does not require access to the source generator's probabilities or internal states. Because the raw compressed size depends on length and register, we condition on a reference to isolate authorship-related residual complexity.

Based on this principle, we propose *CK-Detect*, a two-branch conditional-complexity framework for AI-generated text detection. The first branch captures *instance-level rewrite stability* by measuring the conditional compressed residual between a suspect text x and its LLM-generated rewrites. The second branch captures *corpus-level class alignment* by comparing whether AI-generated or human-written corpora explain x more efficiently under conditional compression.

CK-Detect combines both signals in a

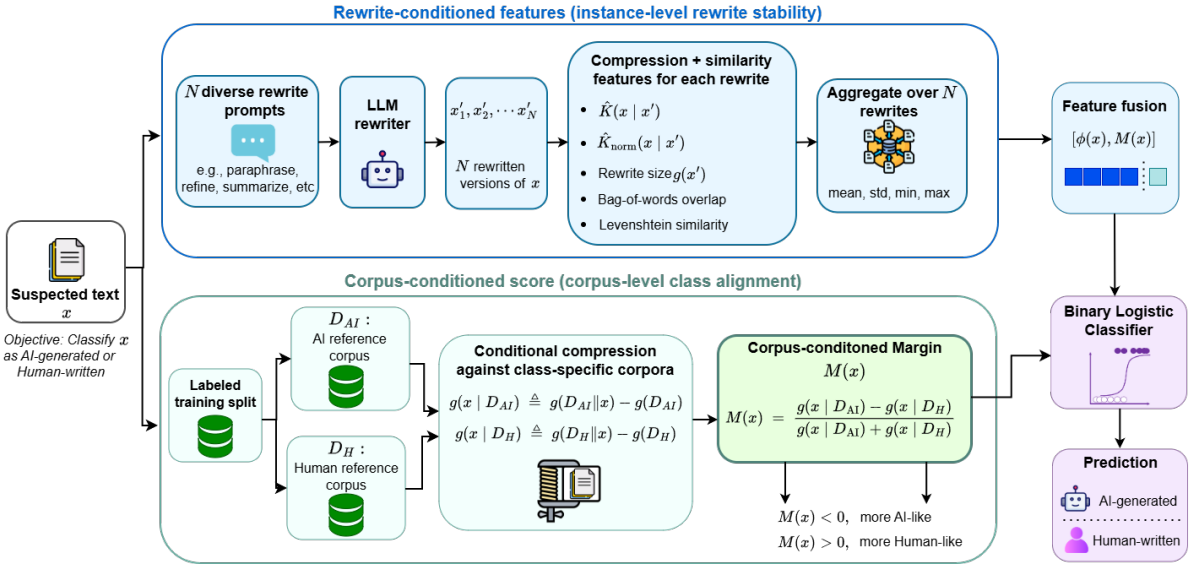

Figure 1: Overall architecture of CK-Detect. The model combines rewrite-conditioned and corpus-conditioned compression signals to predict whether a suspect text is AI-generated or human-written.

lightweight logistic classifier, extending rewriting-based detection beyond surface edit distance while pushing compression-based detection past raw similarity scores. At the detection stage, after rewrites are obtained, CK-Detect requires only lossless compression, scalar feature extraction, and logistic-regression inference. We evaluate CK-Detect across six domains: reviews, academic abstracts, code, creative writing, news, and essays. The results show that conditional complexity features outperform several zero-shot statistical detectors, and on average, outperform the rewriting-based Raidar baseline. The combined CK-Detect model achieves the strongest performance among the proposed variants, suggesting that rewrite-conditioned and corpus-conditioned signals capture complementary aspects of AI-generated text.

Our contributions are summarized as follows:

- We formulate AI-generated text detection as conditional residual description-length estimation, shifting the detection signal from distribution-level entropy to string-level residual complexity.
- We propose rewrite-conditioned complexity features that extend rewriting-based detection beyond surface edit distance by measuring the conditional compressed residual between a suspect text and its LLM rewrites.
- We introduce a corpus-conditioned margin ($M(x)$) that measures corpus-level class alignment by comparing how efficiently AI-generated and human-written corpora explain a suspect text under conditional compression.
- We combine both signals into CK-Detect, a lightweight detection framework that achieves strong empirical performance across multiple domains while avoiding costly neural fine-tuning.

2 Related Work

Existing AI-generated text detectors are commonly categorized into supervised and zero-shot methods. Supervised approaches train classifiers on labeled AI-generated and human-written corpora, including RoBERTa-based detectors, Ghostbuster, and Raidar (Liu et al., 2019; Solaiman et al., 2019; Verma et al., 2024; Mao et al., 2024). These methods typically fine-tune neural encoders or rely on handcrafted features such as lexical overlap and rewrite-based similarity, but often degrade under domain, style, or generator shift. Zero-shot methods instead exploit statistical properties of text without task-specific training. Likelihood-based methods, including GLTR and LRR (Gehrmann et al., 2019; Su et al., 2023), use token probabilities, ranks, or entropy from a reference model. DetectGPT (Mitchell et al., 2023) and Fast-DetectGPT (Bao et al., 2024) estimate probability curvature via perturbations or conditional approximations, while Binoculars (Hans et al., 2024) compares cross-perplexity between models and DNA-GPT (Yang et al., 2023) measures divergence via regeneration. These methods

generally depend on scoring models or repeated sampling. More recently, compression-based approaches have been explored as lightweight alternatives. gzip-kNN (Berchtold et al., 2024) uses compression-based similarity in a kNN framework, AIDetx (Almeida et al., 2025) leverages finite-context models for compression-based classification, and C-Net (Zhou et al., 2024) introduces a lightweight neural model driven by compression features. These approaches suggest that compression provides an effective proxy for statistical regularities in text.

CK-Detect extends Raidar by augmenting surface-level rewrite similarity with conditional compression-based complexity. It is also related to compression-based detectors, but instead formulates detection as conditional residual complexity estimation. Unlike prior compression-only methods, CK-Detect conditions compression on both instance-level rewrites and class-level AI-generated and human-written corpora, enabling detection without source-generator likelihoods or heavy fine-tuning.

3 Kolmogorov Complexity for Detecting AI-Generated Text

This section outlines the link between probabilistic language modeling and Kolmogorov complexity via lossless compression, introduces the compressibility-gap hypothesis for AI-generated text, and demonstrates conditional complexity as a practical detection method.

3.1 Kolmogorov Complexity

Kolmogorov complexity measures the shortest description length of a string under a universal Turing machine U. Formally, the Kolmogorov complexity of a binary string x is:

$$K(x) = \min_p \{|p| : U(p) = x\} \tag{1}$$

Intuitively, $K(x)$ represents the irreducible information content of x. However, due to the halting problem (Li and Vitányi, 2019), $K(x)$ is incomputable. In practice, lossless compression provides a computable approximation, where the compressed byte length $g(x)$ serves as a proxy for $K(x)$.

Given two strings x and y, the chain rule of Kolmogorov complexity is:

$$K(xy) \sim K(x) + K(y \mid x) \tag{2}$$

$$K(xy) \sim K(yx) \tag{3}$$

where $\sim$ denotes equality up to a constant independent of x and y. The conditional Kolmogorov complexity is defined as:

$$K(y \mid x) = \min_p \{|p| : U(p, x) = y\} \tag{4}$$

and can be approximated through compression as:

$$K(y \mid x) \approx g(xy) - g(x) \tag{5}$$

Based on this formulation, Cilibrasi and Vitányi (2003) proposed the Normalized Compression Distance (NCD):

$$\mathrm{NCD}(x, y) = \frac{g(xy) - \min(g(x), g(y))}{\max(g(x), g(y))} \in [0, 1] \tag{6}$$

which provides a compression-based similarity measure between two strings.

3.2 The Compressibility-Gap Hypothesis for AI-Generated Text

The connection between probabilistic models and description length is established by Levin's coding theorem (Levin, 1974) (see also Li and Vitányi (2019)), which states that for any computable probability distribution P over strings, there exists a constant C_P for all strings x such that:

$$K(x) \leq -\log_2 P(x) + C_P \tag{7}$$

This theorem implies that high probability corresponds to a short description. For an autoregressive language model S, the negative log-likelihood (NLL) of a token sequence x is:

$$\mathrm{NLL}_S(x) = -\frac{1}{n} \sum_{i=1}^{n} \log_2 P_S(t_i \mid t_1, \ldots, t_{i-1}) \tag{8}$$

Applying Levin's theorem with P_S gives a computable upper-bound approximation to Kolmogorov complexity:

$$K(x) \leq n \cdot \mathrm{NLL}_S(x) + C_S \tag{9}$$

This result applies to any model S, regardless of architecture, size, or training corpus, thus linking language-model predictability directly to approximate Kolmogorov complexity. Building on

this connection, recent work further shows that language modeling can be interpreted as lossless compression (Delétang et al., 2024).

Let P_{AI} denote the distribution of AI-generated text from an LLM generator G, and P_H the distribution of human-written text. For a neutral scoring model S, Levin's bound gives:

$$\mathbb{E}_{x\sim P}\big[K(x)\big] \leq \mathbb{E}\big[\mathrm{NLL}_S(x)\big] \cdot \bar{n} + C_S \quad (10)$$

Modern LLMs are trained on common linguistic regularities. AI-generated text typically receives lower NLL than human-written text from a neutral scorer, supporting the **compressibility-gap** hypothesis: *AI-generated text tends to be more compressible than human-written text.*

$$\mathbb{E}_{x\sim P_{AI}}\big[K(x)\big] < \mathbb{E}_{x\sim P_H}\big[K(x)\big] \quad (11)$$

We empirically evaluate this hypothesis in Table 1. AI-generated text shows lower NLL than human text across all datasets used in this paper, supporting the compressibility-gap hypothesis. We additionally observe that standard compressors correlate with the same predictability signal without requiring language-model likelihoods. Computing the `gzip` compression ratio $r(x) = g(x)/|x|$ yields Pearson $r = 0.588$ ($p = 3.5 \times 10^{-31}$) against NLL, suggesting that standard compressors capture related statistical regularities.

3.3 Conditional Kolmogorov Complexity as the Detection Signal

Because K is incomputable, practical detection relies on compression-based approximations. Any lossless compressor mapping x to $g(x)$ provides an upper bound on $K(x)$. By applying the chain rule, this extends naturally to conditional complexity:

$$g(x \mid D) \triangleq g(D\|x) - g(D) \approx K(x \mid D) \quad (12)$$

Intuitively, $g(x \mid D)$ quantifies how much information is needed to encode x after patterns in D are learned. This value drops when x shares vocabulary, syntax, or style with D, otherwise approaching $g(x)$. Although AI-generated text tends to be more compressible, unconditional compression is unreliable due to sensitivity to length and writing register. To address this, we use conditional complexity $K(x \mid y)$ relative to a same-domain reference y, removing shared structure and isolating residual information, which forms the foundation of our detection framework.

Source	Mean NLL	SD	Δ vs Human
Human	3.095	0.586	–
AI (GPT-3.5)	2.534	0.520	−0.561

Table 1: Mean NLL gap between Human and AI-generated text under a neutral Qwen2.5-0.5B scorer. AI texts are significantly below Human ($p < 0.001$, Mann-Whitney U test; $N = 11,938$).

4 Methodology

CK-Detect uses two choices of reference y for estimating conditional Kolmogorov complexity $K(x \mid y)$. First, it conditions on LLM-generated rewrites x' to capture instance-level rewrite stability. Second, it conditions on class-specific reference corpora D_{AI} and D_H to measure corpus-level alignment. Each signal can be used as a standalone detector. In this work, we combine both signals into a unified framework and feed the resulting features into a binary logistic classifier for AI-generated text detection, as illustrated in Figure 1.

4.1 Rewrite-Conditioned Kolmogorov Complexity

This method is inspired by Raidar (Mao et al., 2024), which observes that AI-generated text tends to remain more similar to its LLM-generated rewrites than human-written text. We extend this idea by using compression-based features grounded in conditional Kolmogorov complexity rather than relying solely on surface-level edit distances.

Let $g(\cdot)$ denote the compressed length of a string under a lossless compressor, used as a practical surrogate for Kolmogorov complexity. Given the original text x and its rewrite x', we compute four scalar features:

Conditional Kolmogorov complexity (raw):

$$\hat{K}(x \mid x') = g(x'x) - g(x'), \quad (13)$$

Conditional Kolmogorov complexity (normalized):

$$\hat{K}_{\text{norm}}(x \mid x') = \frac{g(x'x) - g(x')}{g(x)}, \quad (14)$$

Normalized Compression Distance (NCD):

$$\mathrm{NCD}(x, x') = \frac{g(xx') - \min\big(g(x),\, g(x')\big)}{\max\big(g(x),\, g(x')\big)}, \quad (15)$$

and *Absolute Rewrite Compression Size* $g(x')$.

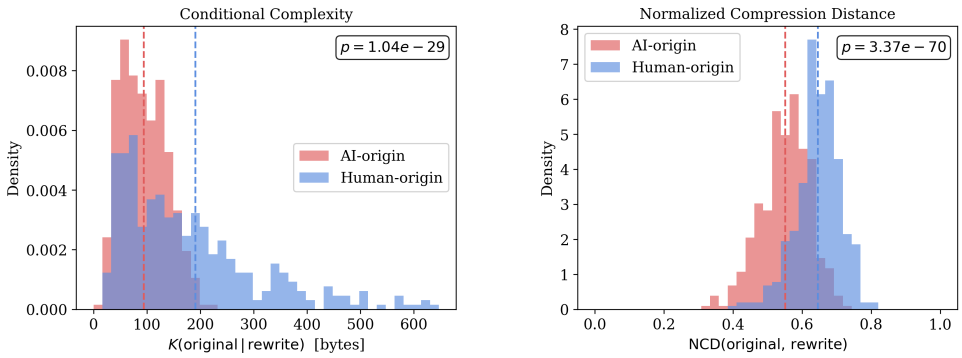

Figure 2: Conditional complexity (left) and Normalized Compression Distance (right) for the original and rewritten Yelp texts, for both human and AI-generated samples.

In addition to these compression-based features, we retain the original surface-level similarity measures from Raidar (Mao et al., 2024) as complementary signals. The first is the *bag-of-words overlap*, which measures the proportion of shared n-gram bags between the original and rewritten texts. The second is the *Levenshtein score*, derived from the Levenshtein edit distance (Levenshtein, 1966), which measures the minimum number of single-character insertions, deletions, and substitutions required to transform one string into another. Feature selection analyses are presented in Appendix A.

We apply N diverse rewrite prompts to each suspect text, yielding rewrites $x'_1, \ldots, x'_N$. For each prompt n, we compute the features above to produce a vector $\mathbf{f}_n$. The per-sample feature vector is then obtained by aggregating across all N rewrites:

$$\phi(x) \;=\; \big[\,\bar{\mathbf{f}},\ \sigma(\mathbf{f}),\ \min(\mathbf{f}),\ \max(\mathbf{f})\,\big], \qquad (16)$$

where $\bar{\mathbf{f}}$, $\sigma(\mathbf{f})$, $\min(\mathbf{f})$, and $\max(\mathbf{f})$ denotes the element-wise mean, standard deviation, minimum, and maximum over the N rewrites. Using multiple prompts reduces the variance from any single rewrite and provides a more stable estimate of the underlying complexity signal.

4.2 Corpus-Conditioned Kolmogorov Complexity

This method conditions on a class-specific reference corpus built from the training split. Building on Equation 12, we partition the labeled training data into two class-specific corpora, $D_{AI} = \{x_i : y_i = \text{AI}\}_{\text{train}}$ and $D_H = \{x_i : y_i = \text{Human}\}_{\text{train}}$. Each corpus is formed by concatenating all training texts within the corresponding class using a fixed separator.

For each suspect text x, we compute the approximation of the conditional Kolmogorov complexities:

$$g(x \mid D_{\text{AI}}) \;=\; g(D_{\text{AI}} \| x) - g(D_{\text{AI}}) \qquad (17)$$

$$g(x \mid D_H) \;=\; g(D_H \| x) - g(D_H) \qquad (18)$$

The signed, normalized difference between the two yields the corpus-conditioned margin ($M(x)$):

$$M(x) \;=\; \frac{g(x \mid D_{\text{AI}}) - g(x \mid D_H)}{g(x \mid D_{\text{AI}}) + g(x \mid D_H)} \qquad (19)$$

where $M(x) \in [-1, +1]$. Negative values indicate stronger alignment with the AI corpus, while positive values indicate stronger alignment with the human corpus. Classification is then performed by thresholding $M(x)$, or, in the ensemble setting, by treating $M(x)$ as a feature alongside the rewrite-conditioned features described in subsection 4.1.

5 Experiments

5.1 Settings

Datasets. To evaluate the proposed approach, we conduct experiments on six datasets spanning diverse domains, writing styles, and text lengths. Yelp, Arxiv, and Code follow the Raidar (Mao et al., 2024) construction, consisting respectively of restaurant reviews, scientific abstracts, and Python code solutions. Essay, CWriting, and News are from Ghostbuster (Verma et al., 2024) and cover student essays, creative writing, and news articles. Dataset statistics and construction details are provided in Appendix B.

LLMs and Compressors. GPT-3.5[1] is used as the primary generator and rewriter, with LLaMA-3.1-8B-Instruct[2] and Qwen-3.5-9B[3] as alternatives.

[1] https://platform.openai.com/docs/models/gpt-3-5-turbo
[2] https://huggingface.co/meta-llama/Llama-3.1-8B
[3] https://huggingface.co/Qwen/Qwen3.5-9B

Method	Yelp		Arxiv		Code		CWriting		News		Essay		Average	
	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC
Raidar	0.8825	0.9498	0.8356	0.9222	0.9090	**0.9890**	0.6114	0.6584	0.5429	0.6508	0.7237	0.7987	0.7508	0.8282
Likelihood	0.5577	0.5757	0.6998	0.7669	0.4637	0.5317	0.5959	0.6717	0.4061	0.5517	0.8587	0.9353	0.5970	0.6722
LogRank	0.5230	0.5348	0.6927	0.7402	0.5250	0.5491	0.6035	0.6642	0.3980	0.5486	0.8487	0.9304	0.5985	0.6612
Entropy	0.5225	0.5441	0.5346	0.5922	0.6508	0.6309	0.5562	0.6113	0.3763	0.4891	0.6586	0.7232	0.5498	0.5985
LRR	0.5637	0.5781	0.5903	0.6333	0.5908	0.6410	0.5314	0.6210	0.3789	0.5353	0.8072	0.8827	0.5771	0.6486
Fast-DetectGPT	0.6621	0.7079	0.9285	0.9739	0.5387	0.5822	0.4806	0.6521	0.3839	0.5661	0.8787	0.9472	0.6454	0.7382
Binoculars	0.6222	0.6682	0.8998	0.9192	0.7273	0.8035	0.6129	0.6891	0.4101	0.5489	0.9262	0.9752	0.6998	0.7674
DNA-GPT	0.3333	0.5000	0.5327	0.5639	0.5113	0.5243	0.3638	0.5619	0.3763	0.5249	0.6133	0.7016	0.4551	0.5628
RoBERTa	**0.9474**	**0.9931**	**0.9929**	**1.0000**	0.8449	0.9605	**0.6260**	**0.7461**	0.3763	0.6538	**0.9448**	**0.9987**	0.7887	**0.8920**
OpenAI Detector	0.5271	0.6581	0.5526	0.7257	0.4728	0.5849	0.4802	0.4751	0.4294	0.4171	0.7810	0.8753	0.5405	0.6227
Rewrite-conditioned	0.8662	0.9461	0.9571	0.9871	0.9697	0.9853	0.6047	0.6434	**0.6342**	**0.6841**	0.7059	0.7730	0.7896	0.8365
Corpus-conditioned	0.8543	0.9345	0.9571	0.9886	0.8933	0.9174	0.5744	0.6117	0.4793	0.5209	0.7224	0.8386	0.7468	0.8020
CK-Detect	0.9175	0.9748	0.9714	0.9929	**0.9697**	0.9881	0.5822	0.6548	0.6294	0.6791	0.7937	0.8885	**0.8106**	0.8630

Table 2: In-distribution F1-score and AUC comparison across datasets. The final three methods are the proposed conditional complexity-based methods. The numbers in **bold** are ranked first, while the numbers underlined are ranked second.

Method	Within-Domain	Cross-Domain					Cross-Generator	
	Yelp (GPT-3.5)	Arxiv	Code	CWriting	News	Essay	LLaMA-3.1	Qwen-3.5
Raidar	0.8825	0.6544	0.3333	0.2998	0.2840	0.3333	0.8367	0.9335
RoBERTa	**0.9474**	**0.6980**	0.3333	**0.5036**	0.2952	**0.7501**	**0.9487**	**0.9511**
Rewrite-conditioned	0.8662	0.6317	**0.9697**	0.4353	0.3607	0.6368	0.8472	0.7181
Corpus-conditioned	0.8543	0.5431	0.3333	0.4943	**0.4571**	0.5925	0.8346	0.8593
CK-Detect	0.9175	0.6640	0.9545	0.4444	0.3902	0.6874	0.9250	0.8140

Table 3: F1-score evaluation under cross-domain and cross-generator settings. Cross-domain evaluates transfer from Yelp to unseen domains, while cross-generator evaluates transfer from GPT-3.5 to unseen LLM generators. The numbers in **bold** are ranked first, while the numbers underlined are ranked second.

For lossless compression, we use `gzip`[4] at level 9, which implements the DEFLATE algorithm (LZ77 + Huffman coding) via Python's `zlib` library.

Evaluation metrics. Macro-F1 is used as the primary evaluation metric because it provides a balanced measure of practical detection quality. We also report AUC-ROC (AUC) to evaluate threshold-independent discriminability from the raw detector scores.

Baselines. We compare against ten baselines. Supervised baselines include Raidar (Mao et al., 2024), RoBERTa (Liu et al., 2019), and RoBERTa-base-openai-detector (Solaiman et al., 2019). Zero-shot baselines are further divided into statistical and advanced methods. For statistical methods, we use GPT-2 XL[5] as the scoring model to compute Likelihood, LogRank, Entropy (Gehrmann et al., 2019; Solaiman et al., 2019; Ippolito et al., 2020), and LRR (Su et al., 2023). Advanced zero-shot baselines include Fast-DetectGPT (Bao et al., 2024), Binoculars (Hans et al., 2024), and DNA-GPT (Yang et al., 2023). Additional implementation details and hyperparameters are provided in Appendix C.

[4]https://www.gzip.org/

[5]https://huggingface.co/openai-community/gpt2-xl

5.2 Main Results

In-distribution Performance. Table 2 reports F1 and AUC across all six datasets. All proposed conditional-complexity methods outperform the zero-shot baselines. CK-Detect achieves approximately a 19% improvement in average F1-score and a 16% improvement in average AUC compared to the mean performance of all baselines. Rewrite-conditioned already surpasses Raidar (+3.9 points), showing that compression-based features capture rewrite invariance more effectively than surface-level overlap features. Corpus-conditioned alone achieves competitive performance but degrades on heterogeneous or structurally distant domains. Combining both signals in CK-Detect yields the best average F1-score, indicating that the two signals are complementary.

Performance is relatively weaker in heterogeneous domains such as Creative Writing, News, and Essays, which exhibit greater variation in topics, styles, and writing structures than in other, more homogeneous domains. We also note that in some individual domains, CK-Detect may trail the rewrite-conditioned variant when the

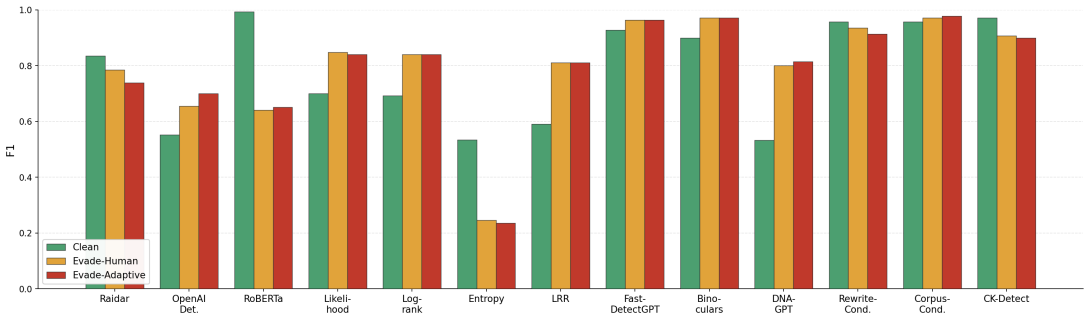

Figure 3: Detection robustness against adversarial rewriting attacks on the Arxiv dataset. "*Evade-human*" prompts the LLM to rewrite the text in a more human-like style, while "*Evade-adaptive*" aims to maximize changes under subsequent GPT-based rewriting.

corpus-conditioned signal underperforms. Among all methods, the fully fine-tuned RoBERTa model achieves the highest average AUC and is the closest competitor to CK-Detect. Nevertheless, CK-Detect achieves the best overall average F1-score, outperforming RoBERTa on the Code and News domains while remaining competitive on Yelp, Arxiv, and CWriting.

Out-of-distribution Performance. Table 3 evaluates generalization under cross-domain and cross-generator transfer. We compare against Raidar and RoBERTa, as they represent the strongest supervised baselines, while other methods are zero-shot detectors. CK-Detect generally outperforms Raidar and remains competitive with RoBERTa, but performance still degrades under strong domain shift, especially for structurally distant domains such as News and CWriting, while cross-generator transfer is more stable, suggesting that domain variation is more challenging than generator variation.

Robustness against adversarial rewriting. Prior work demonstrates that AI-generated text can often evade detectors when adversarially paraphrased (Krishna et al., 2023). To evaluate robustness under such conditions, we construct adversarial rewrites on the Arxiv dataset using two attack prompts: *Evade-Human*[6] and *Evade-Adaptive*[7].

As shown in Figure 3, our methods maintain competitive performance under both attacks. CK-Detect maintains strong overall performance, with only a modest performance drop of around 7%. The corpus-conditioned variant is particularly stable and even improves under the adversarial setting, while the rewrite-conditioned variant shows only a small degradation. This may be partly due to the use of multiple rewrite prompts, which improves robustness. In contrast, supervised baselines experience substantially larger drops under adversarial rewriting. Some statistical zero-shot methods improve after paraphrasing, suggesting that the attacks may unintentionally amplify the statistical signals they rely on.

[6]Evade-Human: "Help me rephrase it in human style."

[7]Evade-Adaptive: "Help me rephrase it, so that another GPT rewriting will cause a lot of modifications."

Method	Precision	Recall	F1-score
Direct compression	0.7067	0.6572	0.6542
Rewrite-conditioned	0.8059	0.7907	0.7896
Corpus-conditioned	0.7592	0.7520	0.7468
CK-Detect	**0.8194**	**0.8110**	**0.8106**

Table 4: Average precision, recall, and F1-score across datasets under different conditional signal settings.

Method	Compressor			
	gzip	bz2	lzma	zstd
Rewrite-conditioned	0.7896	0.7858	**0.7936**	0.7821
Corpus-conditioned	0.7468	0.5426	**0.8188**	0.7536
CK-Detect	0.8106	0.7852	**0.8206**	0.7744
Compression Speed				
Mean (ms/text)	**1.24**	32.11	146.59	3.57
Relative Speed	**1.0**×	25.9×	118.2×	2.9×

Table 5: Average detection performance and runtime comparison across different compressors.

5.3 Ablation Studies

Impact of conditional signal. Table 4 demonstrates that leveraging conditional information significantly improves detection performance. Direct compression, which uses only the compression ratio as the detection signal, performs the worst because it lacks a comparison or reference for context.

Method	Inference Time
Raidar*	**1.78 ms**
CK-Detect (Ours)*	1.86 ms
RoBERTa	10.26 ms
OpenAI Detector	11.55 ms
Fast-DetectGPT	63.56 ms
Likelihood-based	66.83 ms
Binoculars	708.68 ms
DNA-GPT	6070.20 ms

Table 6: Average inference time per sample. Likelihood-based includes Likelihood, LogRank, Entropy, and LRR. *These methods additionally require an LLM rewriting stage.

It relies solely on the fact that AI-generated text is more compressible than human-written text. Incorporating conditional references, either via rewrite-conditioned or corpus-conditioned features, boosts performance, and combining both signals in CK-Detect yields the highest precision, recall, and F1-score.

Impact of compressors. We additionally evaluate `bz2`[8], `lzma`[9], and `zstd`[10] as alternative compressors. Table 5 reports the average detection performance and compression speed. `lzma` compressor yields the highest F1 scores across all methods, with particularly large gains in corpus-conditioned detection. However, this improvement comes at a steep runtime cost, `lzma` is 118.2× slower than `gzip`. `gzip` remains the fastest compressor and delivers performance close to `lzma`, offering the best trade-off between speed and accuracy. `zstd` is only 2.9× slower than `gzip` but does not consistently outperform it, while `bz2` is both slower and substantially weaker on corpus-conditioned tasks. Stronger compressors can capture richer conditional complexity signals, but lightweight compressors like `gzip` remain highly efficient for practical deployment.

Impact of model used for generation and rewriting. CK-Detect demonstrates strong cross-model robustness across generator-rewriter combinations on the Yelp dataset, with most pairings achieving F1-scores around 0.90 or higher. The method remains effective without access to generator token probabilities or internal states. We further compare different rewriters for GPT-3.5-generated texts. Using the same model as the generator yields the best performance. However, performance degradation with alternative or substantially smaller rewriter models is small. Replacing GPT-3.5 with Qwen-3.5:0.8B[11] reduces F1 by only around 5 points and AUC by less than 1 point. These results suggest that CK-Detect is relatively robust to the choice of rewriter and can effectively utilize lightweight local models with only minor performance loss. The details and figure are provided in Appendix D.

5.4 Computational Efficiency Analysis

Benchmarks are conducted on a machine equipped with an Intel Core i5-11400 CPU and an NVIDIA GTX 1080 Ti GPU. Table 6 reports the average inference time excluding preprocessing overhead. Rewrite-conditioned methods require an initial rewriting step, although smaller local models can substantially reduce this cost while maintaining competitive performance. CK-Detect is lightweight at the detection stage, requiring only lossless compression, scalar feature extraction, and a logistic-regression classifier. It runs in under 2 ms per sample on CPU. In contrast, zero-shot baselines rely on GPU-accelerated LLM forward passes, resulting in higher per-sample latency, while supervised detectors additionally depend on GPU inference and fine-tuning.

6 Conclusion

In this paper, we proposed a conditional Kolmogorov complexity framework for AI-generated text detection through two complementary signals: rewrite-conditioned and corpus-conditioned complexity. We combined both signals into a unified detector and extensive experiments across multiple domains show that the proposed CK-Detect is competitive with the state-of-the-art baselines, achieving approximately a 19% improvement in average F1-score over the mean performance of existing methods. The proposed approach also demonstrates robustness under domain shift and adversarial paraphrasing, while requiring no access to the source generator's token probabilities or internal model states. Finally, CK-Detect is computationally efficient, relying on lightweight compression-based feature extraction and a simple classifier, making it practical for low-cost deployment.

[8] https://sourceware.org/bzip2/
[9] https://www.7-zip.org/sdk.html
[10] https://facebook.github.io/zstd/
[11] https://huggingface.co/Qwen/Qwen3.5-0.8B

Limitations

While CK-Detect demonstrates strong performance across diverse domains, several directions remain open for future work. Specifically, our evaluation relied on open-weight alternatives for a subset of computationally intensive baselines to maintain computational feasibility and reproducibility using publicly available resources. Extending the comparison to full-scale configurations would provide a useful future validation as hardware accessibility improves.

As a supervised framework, CK-Detect requires labeled training data to construct the reference corpora and train the logistic classifier, although this remains substantially lighter than full neural fine-tuning. Furthermore, while performance under strong domain shift remains challenging, the cross-generator experiments suggest that the proposed features generalize well across unseen LLM families. Additionally, the rewrite-conditioned branch also introduces an additional rewriting step, though using lightweight local models can reduce this overhead with only marginal performance loss.

Our experiments focus on English text across six domains. Extending CK-Detect to multilingual and highly specialized settings is an important direction for future work. Finally, because Kolmogorov complexity is incomputable, the proposed features should be interpreted as empirical approximations of conditional description length rather than exact theoretical quantities. Nevertheless, the consistent performance across multiple compressors suggests that the framework remains robust to this approximation gap in practice.

Ethical Considerations

AI-generated text detection can be useful for identifying synthetic content in educational, journalistic, and online settings, but such systems remain probabilistic and may produce false positives, particularly for non-native speakers, highly structured writing, or out-of-domain texts. Therefore, the proposed method should not be used as the sole basis for high-stakes decisions. We view CK-Detect as a supporting analytical tool rather than a definitive authorship verification system.

Our experiments use publicly available research datasets and a combination of documented API-based and open-weight language models. All datasets are established benchmarks and are used only for research purposes. We do not collect new user data, attempt to identify individuals, or redistribute model outputs beyond the experimental analysis. The evaluated corpora consist of reviews, academic abstracts, code, creative writing, news, and essays. Any potentially sensitive content is inherited from the original public benchmarks and handled only in aggregate for evaluation.

References

Leonardo Almeida, Pedro Rodrigues, Diogo Magalhães, Armando J. Pinho, and Diogo Pratas. 2025. Aidetx: A compression-based method for identification of machine-learning generated text. In *2025 Data Compression Conference (DCC)*, pages 358–358.

Guangsheng Bao, Yanbin Zhao, Zhiyang Teng, Linyi Yang, and Yue Zhang. 2024. Fast-detectgpt: Efficient zero-shot detection of machine-generated text via conditional probability curvature. In *International Conference on Learning Representations*, volume 2024, pages 24814–24836.

Matthias Berchtold, Sandra Mitrović, Davide Andreoletti, Daniele Puccinelli, and Omran Ayoub. 2024. Detecting chatgpt-generated text with gzip-knn: A no-training, low-resource approach. In *International Conference on Natural Language and Speech Processing*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *CoRR*, abs/2107.03374.

Rudi Cilibrasi and Paul M. B. Vitányi. 2003. Clustering by compression. *IEEE Transactions on Information Theory*, 51:1523–1545.

Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Wenliang Kevin Li, Matthew Aitchison, Laurent Orseau, Marcus Hutter, and Joel Veness. 2024. Language modeling is compression. In *The Twelfth International Conference on Learning Representations*.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander Rush. 2019. GLTR: Statistical detection and visualization of generated text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 111–116, Florence, Italy. Association for Computational Linguistics.

Abhimanyu Hans, Avi Schwarzschild, Valeriia Cherepanova, Hamid Kazemi, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2024. Spotting llms with binoculars: zero-shot

detection of machine-generated text. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

John Houvardas and Efstathios Stamatatos. 2006. N-gram feature selection for authorship identification. In *Artificial Intelligence: Methodology, Systems, and Applications*, pages 77–86, Berlin, Heidelberg. Springer Berlin Heidelberg.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. 2020. Automatic detection of generated text is easiest when humans are fooled. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1808–1822, Online. Association for Computational Linguistics.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2023. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *Preprint*, arXiv:2303.13408.

V. I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710.

L. A. Levin. 1974. Laws of information conservation (Nongrowth) and aspects of the foundation of probability theory. *Problems of Information Transmission*, 10(3):206–210. Translated from Russian: Probl. Peredachi Inf., vol. 10, no. 3, pp. 30–35, 1974.

Ming Li and Paul Vitányi. 2019. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer International Publishing.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Chengzhi Mao, Carl Vondrick, Hao Wang, and Junfeng Yang. 2024. Raidar: generative ai detection via rewriting. In *International Conference on Learning Representations*, volume 2024, pages 6013–6030.

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'23. JMLR.org.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, Miles McCain, Alex Newhouse, Jason Blazakis, Kris McGuffie, and Jasmine Wang. 2019. Release strategies and the social impacts of language models. *Preprint*, arXiv:1908.09203.

Jinyan Su, Terry Zhuo, Di Wang, and Preslav Nakov. 2023. DetectLLM: Leveraging log rank information for zero-shot detection of machine-generated text. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12395–12412, Singapore. Association for Computational Linguistics.

Vivek Verma, Eve Fleisig, Nicholas Tomlin, and Dan Klein. 2024. Ghostbuster: Detecting text ghostwritten by large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1702–1717, Mexico City, Mexico. Association for Computational Linguistics.

Xianjun Yang, Wei Cheng, Yue Wu, Linda Petzold, William Yang Wang, and Haifeng Chen. 2023. Dna-gpt: Divergent n-gram analysis for training-free detection of gpt-generated text. *Preprint*, arXiv:2305.17359.

Yinghan Zhou, Juan Wen, Jianghao Jia, Liting Gao, and Ziwei Zhang. 2024. C-net: A compression-based lightweight network for machine-generated text detection. *IEEE Signal Processing Letters*, 31:1269–1273.

A Feature Analysis

Feature importance. Figure 4 shows Random Forest Gini impurity importances across all compression features in the rewrite-conditioned method. The two most important features are `max_cond_norm` (maximum conditional normalized complexity across the rewrites) and `mean_g_rewrite` (average compressed size of the rewrites), together accounting for approximately 31% of total importance. The four NCD features (`mean_ncd`, `std_ncd`, `min_ncd`, `max_ncd`) collectively rank higher than the raw compressed sizes, confirming that the relative compression gain is the informative signal, not the file size.

Group feature ablation. To quantify the marginal contribution of each feature group, we train the logistic classifier with each group removed in turn. Figure 5 shows the resulting F1 drop. Removing `cond_complexity` causes the largest drop (-4.2 points), followed by NCD (-2.8 points). Removing `g_rewrite` or `g_original` individually causes only minor degradation (-0.5 to -1.0 points). These raw size features are partially redundant with the normalized features. No group can be removed without measurable loss, validating the design of including all four feature groups.

Figure 6 provides a complementary view via forward feature-group selection, showing cumulative F1 as groups are added in order of marginal gain. The `cond_complexity` group alone achieves

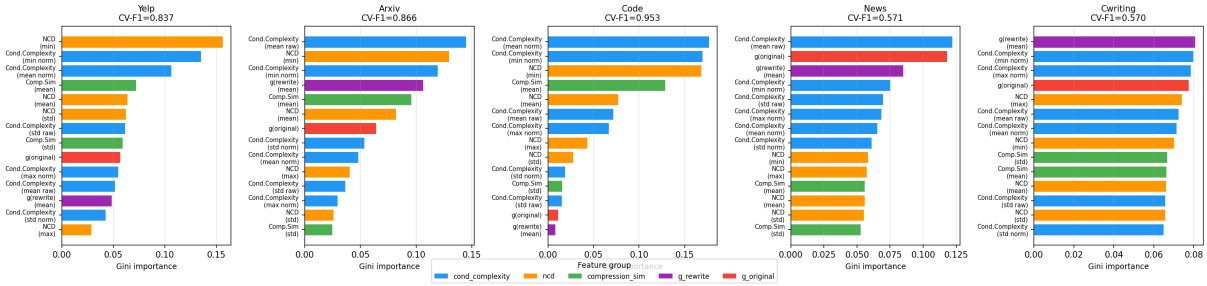

Figure 4: Random Forest Gini feature importances for the rewrite-conditioned method.

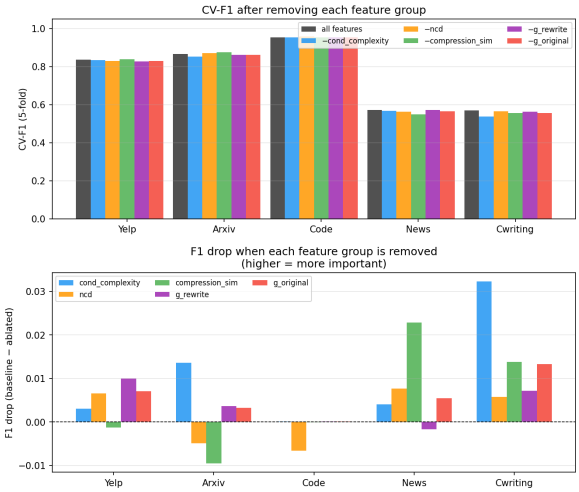

Figure 5: Group feature ablation, F1 drop when one feature group is removed at a time.

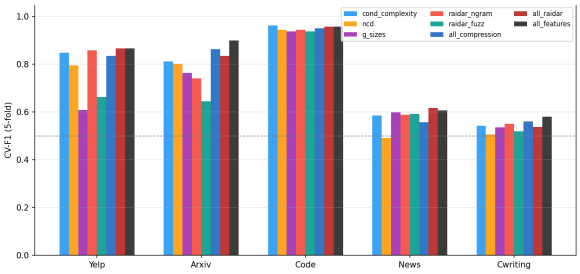

Figure 6: Forward feature-group selection, feature group CV-F1 by domain.

F1 = 0.87; adding NCD brings it to 0.90; the remaining groups contribute smaller but non-trivial increments.

Edit-distance features. A key question is whether the Rewrite-Conditioned compression features subsume Raidar (Mao et al., 2024) edit-distance features, rendering them redundant. Figure 7 addresses this directly by comparing three feature configurations across all seven domains:

1. Raidar (baseline): edit-distance (Levenshtein score) and n-gram overlap features only.
2. Rewrite-Conditioned (compression features only): conditional normalized complexity, NCD, compressed sizes.
3. Rewrite-Conditioned: both feature sets combined.

The rewrite-conditioned compression features are competitive and often outperform with Raidar's edit-distance features on several domains. Using compression features alone yields the largest gains on News (+9.1 points), Code (+6.1 points), and Arxiv (+4.2 points) over the Raidar baseline, indicating that conditional complexity and NCD capture authorship signal where surface lexical overlap is weak. On Yelp, CWriting, and Essay, the two feature families are roughly tied (within 2 points), showing that compression features are not strictly better. Both signals encode partly redundant information when the rewrite behavior itself is already discriminative. Combining the two feature sets yields the strongest configuration on Arxiv (+12.1 points over Raidar, +7.9 points over compression-only) and matches the better of the two on every other domain, never worse by more than 2 points. This complementarity is most visible on Arxiv, where the combined model substantially exceeds either source alone, justifying the inclusion of Raidar edit-distance features in our final method. The two signals exhibit different failure modes: compression features are less effective when AI and human texts share similar information density (e.g., Essay, CWriting), while edit-distance features remain informative through how the rewriter modifies surface form.

On average across all six domains, replacing the baseline with our conditional compression features yields a modest +1.8% F1 improvement, heavily driven by strong gains in Code and News. However, retaining Raidar-style edit-distance features alongside compression proves crucial for stability and

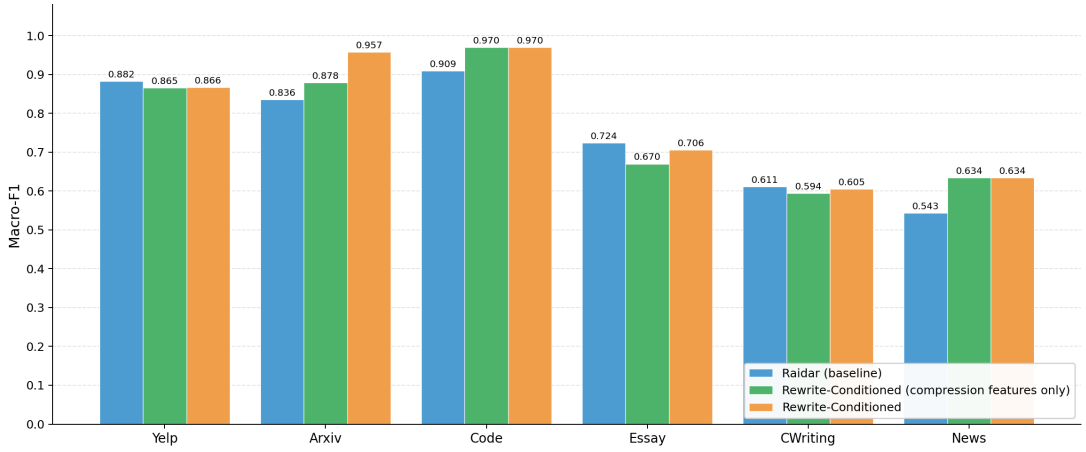

Figure 7: F1 comparison of Raidar feature (baseline) vs. Rewrite-Conditioned (compression feature only) vs. Rewrite-Conditioned + Raidar across seven domains.

Domain	Total	Human	AI	Human source	AI Generator	Reference
Yelp	4,000	2,000	2,000	Yelp Review Dataset	GPT-3.5-Turbo	(Mao et al., 2024)
Arxiv	700	350	350	ICLR paper 2015–2021	GPT-3.5-Turbo	(Mao et al., 2024)
Code	328	164	164	HumanEval	GPT-3.5-Turbo	(Mao et al., 2024)
CWriting	1,701	973	728	WritingPrompts	GPT-3.5-Turbo	(Verma et al., 2024)
News	1,209	730	479	Reuters 50-50 authorship	GPT-3.5-Turbo	(Verma et al., 2024)
Essay	4,000	2,000	2,000	Ivypanda	GPT-3.5-Turbo	(Verma et al., 2024)

Table 7: Dataset statistics per domain.

overall accuracy, providing an additional average boost of +2.1% F1.

B Dataset Statistics and Constructions

All experiments use a fixed 80/20 stratified hold-out split, applied identically across all methods. No domain-specific preprocessing is performed. Dataset statistics are presented in Table 7.

Yelp Dataset. We use the first 2,000 human reviews from the Yelp review dataset. AI-generated reviews are produced by prompting GPT-3.5-Turbo to generate a concise review of similar length:

```
Write a concise Yelp restaurant review of
approximately the same length as a typical
user review (100-200 words).
Output ONLY the review text.
```

Arxiv Dataset. We investigate whether AI-generated paragraphs in academic papers can be detected. Our dataset contains 350 abstracts from ICLR conference papers (2015–2021), all written by humans since ChatGPT was not released during that period. AI abstracts are generated by GPT-3.5-Turbo using the paper title and the first 15 words of the human abstract as context, following the Raidar benchmark protocol (Mao et al., 2024):

```
Write an academic abstract for a research paper.
Title: [PAPER_TITLE]
The abstract begins: [FIRST_15_WORDS]
Generate a complete, coherent abstract
(100-250 words).
Output ONLY the abstract text,
no labels or extra commentary.
```

Code Dataset. We adopt the HumanEval benchmark (Chen et al., 2021) as the source of human-written code solutions. Each HumanEval entry provides a function signature and docstring. The human solution is the canonical implementation. We ask GPT-3.5-Turbo to solve the same programming task from the signature alone:

```
Complete the following Python function.
Output ONLY the code (the function body),
no explanation or markdown.
[FUNCTION_SIGNATURE + DOCSTRING]
```

CWriting Dataset. Human texts are creative writing responses collected from the WritingPrompts subreddit. To avoid contamination from ChatGPT-generated content, Verma et al. (2024) scraped posts only from the top 50 users in October 2022, retaining the last 100 posts per user. AI texts are generated by GPT-3.5-Turbo using the corresponding writing prompts from those posts.

News Dataset. Human news articles are sourced from the Reuters 50-50 authorship dataset (Hou-

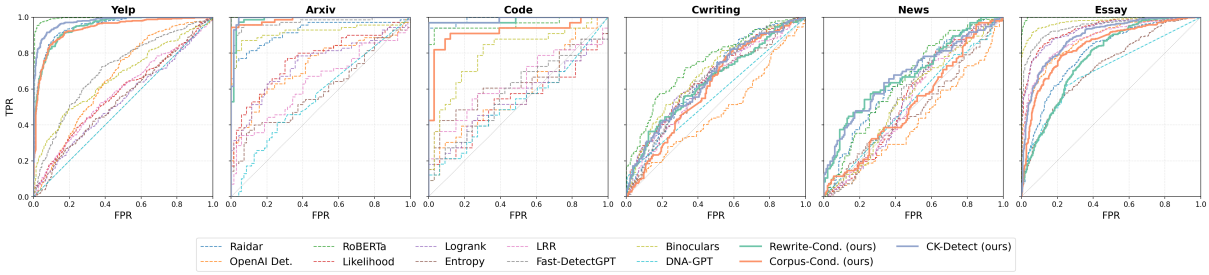

Figure 8: ROC curves for all evaluated detection methods across the six text domains.

Method	Yelp		Arxiv		Code		Essay		CWriting		News		Average	
	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC
Direct Compression	0.6233	0.6610	0.7354	0.8516	0.7284	0.9192	0.7149	0.8367	0.5772	**0.6554**	0.5459	0.6418	0.6542	0.7610
Rewrite-conditioned	0.8662	0.9461	0.9571	0.9871	**0.9697**	0.9853	0.7059	0.7730	**0.6047**	0.6434	**0.6342**	**0.6841**	0.7896	0.8365
Corpus-conditioned	0.8543	0.9345	0.9571	0.9886	0.8933	0.9174	0.7224	0.8386	0.5744	0.6117	0.4793	0.5209	0.7468	0.8020
CK-Detect	**0.9175**	**0.9748**	**0.9714**	**0.9929**	**0.9697**	**0.9881**	**0.7937**	**0.8885**	0.5822	0.6548	0.6294	0.6791	**0.8106**	**0.8630**

Table 8: F1-score and AUC comparison across datasets for direct compression baseline and proposed conditional complexity-based methods (GPT rewriter).

vardas and Stamatatos, 2006). Since the original headlines were unavailable, Verma et al. (2024) use GPT-3.5-Turbo to first generate a headline for each article and then prompt it to write a news article conditioned on that headline.

Essay Dataset. Human essays are sourced from the IvyPanda corpus of high-school and university essays across multiple disciplines. As in Verma et al. (2024), GPT-3.5-Turbo was first prompted to produce a corresponding essay prompt, and then to write an essay conditioned on that prompt while approximately matching the length of the original. We cap the dataset at 4000 samples (from the original 22,172).

C Baseline Implementation Details

All zero-shot scalar scorers (Likelihood, LogRank, Entropy, LRR, Fast-DetectGPT, Binoculars, DNA-GPT) produce a single score per text. AUC is computed directly from this score. For F1 and Accuracy, we fit a one-dimensional logistic regression on the training split to obtain a decision threshold (no test-set tuning).

Raidar. Raidar (Mao et al., 2024) rewrites each text with GPT-3.5-Turbo using seven rewrite prompts (see Appendix E) and computes surface-similarity features between the original text and each rewrite:

- n-gram overlap ($n = 1, 2, 3, 4$): the count of shared n-grams between the original and the rewrite, normalized by the original token length (after lowercasing and whitespace tokenization), reported per prompt, averaged over prompts, and against the concatenation of all rewrites.
- Fuzzy string ratios: `fuzz.ratio` (a Levenshtein-based similarity measure) and `fuzz.token_set_ratio` per prompt.

These features are fed to a logistic regression classifier (L2 penalty, $C = 1.0$, `lbfgs` solver) on raw (non-standardized) features, following the original paper's code.

Zero-shot statistical scores. We score each text with `GPT-2 XL` (1.5B), truncating to 512 tokens. Over the predicted token positions, we compute: **Likelihood** (mean log-probability of the gold token), **LogRank** (negative mean log of the gold token's rank in the predictive distribution), and **Entropy** (mean predictive entropy). We also compute **LRR**, the ratio of the likelihood to the LogRank score. Each scalar is converted to a decision via the threshold procedure above.

Fast-DetectGPT. Fast-DetectGPT (Bao et al., 2024) replaces DetectGPT's perturbation sampling with an analytic sampling-discrepancy score derived from the conditional token distribution. For each position, the discrepancy $d_t = (\log p(x_t) - \mu_t)/\sqrt{\mathrm{var}_t}$ is computed from the predictive mean μ_t and variance over the vocabulary, and the score is the mean over positions. We use `GPT-2 XL` (1.5B) as the single scoring model (sampling and refer-

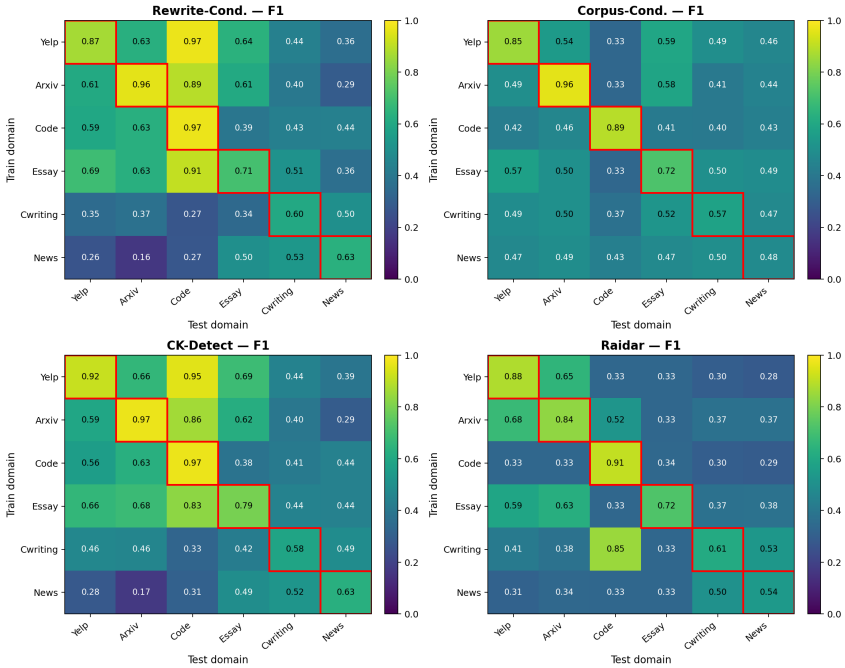

Figure 9: Cross-domain generalization F1 scores. The models are trained on the source domain (y-axis) and evaluated on the target domain (x-axis).

	Yelp		Arxiv		Code		Essay		Cwriting		News		Average	
Rewriter	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC
GPT-3.5	0.9175	0.9748	0.9714	0.9929	**0.9697**	**0.9881**	0.7937	0.8885	0.5822	0.6548	**0.6294**	**0.6791**	**0.8106**	**0.8630**
Qwen-3.5:9b	**0.9262**	**0.9772**	0.9857	0.9976	0.6460	0.8953	**0.8311**	**0.9033**	**0.6130**	**0.6682**	0.5972	0.6518	0.7665	0.8489
Qwen-3.5:0.8b	0.9049	0.9633	**0.9929**	**0.9994**	0.6460	0.9697	0.8110	0.8935	0.5965	0.6566	0.6079	0.6387	0.7599	0.8535

Table 9: CK-Detect F1-score and AUC comparison across different rewriter models and datasets (GPT-3.5 as the source for the AI-generated texts).

ence model are identical); no perturbation model is required.

Binoculars. Binoculars (Hans et al., 2024) compares the perplexity of a performer model against the cross-perplexity between an observer and the performer. We use the paper's canonical pair, `falcon-7b`[12] (observer) and `falcon-7b-instruct`[13] (performer), loaded in 4-bit NF4 quantization. The score is the performer's mean negative log-likelihood divided by the observer-performer cross-entropy (lower indicates AI-generated). The 4-bit quantization slightly shifts calibration relative to the original bfloat16 setup, so we fit the threshold on the train split rather than using the paper's fixed thresholds.

DNA-GPT. DNA-GPT (Yang et al., 2023) truncates each text at a ratio $\gamma = 0.5$, generates $K = 3$ continuations of the prefix with `llama3.1:8b` (via Ollama, temperature $= 0.7$, top-$p = 0.95$, 128 new tokens), and scores the n-gram overlap ($n = 3, 4, 5$) between the original suffix and each continuation, averaged over the K samples. K is reduced from the paper's 5–20 to keep the runtime feasible on our hardware. Higher overlap indicates AI-generated text.

OpenAI Detector. We apply OpenAI's pretrained `roberta-base-openai-detector` (Solaiman et al., 2019) off-the-shelf, with no fine-tuning on our data (zero-shot for our datasets). It is a `RoBERTa-base` classifier fine-tuned by OpenAI on WebText (human) vs. GPT-2-1.5B outputs. We truncate inputs to 512 tokens and take the "Fake" (AI) class probability.

RoBERTa. We fine-tune `RoBERTa-base` (Liu et al., 2019) (125M parameters) independently per domain, using:

- Optimizer: AdamW, weight decay 0.01, gradient clipping at 1.0

[12] https://huggingface.co/tiiuae/falcon-7b
[13] https://huggingface.co/tiiuae/falcon-7b-instruct

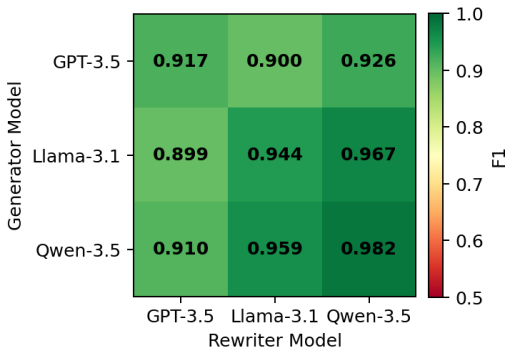

Figure 10: Cross-model detection performance on the Yelp dataset across all generator–rewriter model combinations.

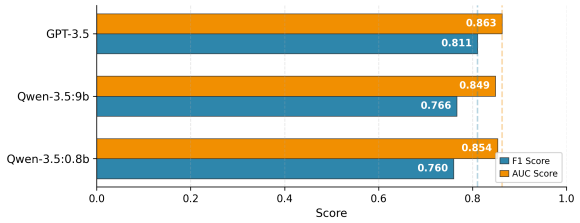

Figure 11: Average detection performance (F1 and AUC) for GPT-3.5 generated text using different rewriter models.

- Learning rate: 5×10^{-6}, linear warmup (10% of steps)
- Batch size: 16
- Epochs: 3
- Maximum sequence length: 256 tokens
- Loss: binary cross-entropy

Alternative Compressor Details. We test different lossless compression algorithms to assess the robustness of our approach to the choice of compressor. Specifically, `bz2` (Burrows–Wheeler transform-based, level 9), `lzma` (dictionary-based LZMA compression, level 6), and `zstd` (LZ77-based with entropy coding, level 3) are used as representative compressors from different compression families.

D Additional Experimental Results

We include extended experimental results and ablations in this section.

ROC Analysis. Figure 8 presents detailed ROC curves. Our methods consistently maintain strong detection trade-offs across thresholds, although highly stylized domains such as Creative Writing and News remain challenging for all evaluated methods.

Direct vs. Conditional Compression. Table 8 compares the direct-compression baseline with the proposed conditional complexity-based methods. Direct compression, which relies only on raw compression statistics without conditional reference, performs the worst overall. Incorporating conditional information via rewrite-conditioned or corpus-conditioned features substantially improves both F1 and AUC across most domains. Among variants, rewrite-conditioned generally performs best on Code, CWriting, and News. Combining both signals in CK-Detect further improves performance, achieving the best average F1 (0.8106) and AUC (0.8630), indicating that the two signals are complementary.

Cross-Domain Transferability. Figure 9 shows that our methods achieve stronger out-of-domain robustness compared to Raidar, particularly under cross-domain shift.

Generator-Rewriter Performance. Figure 10 shows CK-Detect performance across all generator–rewriter combinations on the Yelp dataset, where most pairings consistently achieve high F1-scores (around 0.90 or above), indicating strong robustness across model mismatches.

Rewriter Robustness. Table 9 and Figure 11 evaluate the effect of different rewriter models while keeping GPT-3.5 as the source generator. Overall, performance remains stable across rewriters. Using GPT-3.5 as both generator and rewriter yields the best results, while Qwen-3.5-9B and the lightweight Qwen-3.5-0.8B remain competitive across most domains. Both Qwen variants perform worse on Code, likely due to differences in rewriting behavior and the structured nature of source code. Overall, replacing GPT-3.5 with alternative or smaller rewriters leads to only minor degradation, confirming that CK-Detect is robust to the choice of rewriter and can operate effectively with lightweight models.

Impact of Input Length. Figure 12 studies the effect of input length on detection performance. Results show a strong domain-dependent trend, longer texts improve performance in narrative domains such as Essays, while trends are more volatile for News and Creative Writing, and can even reverse

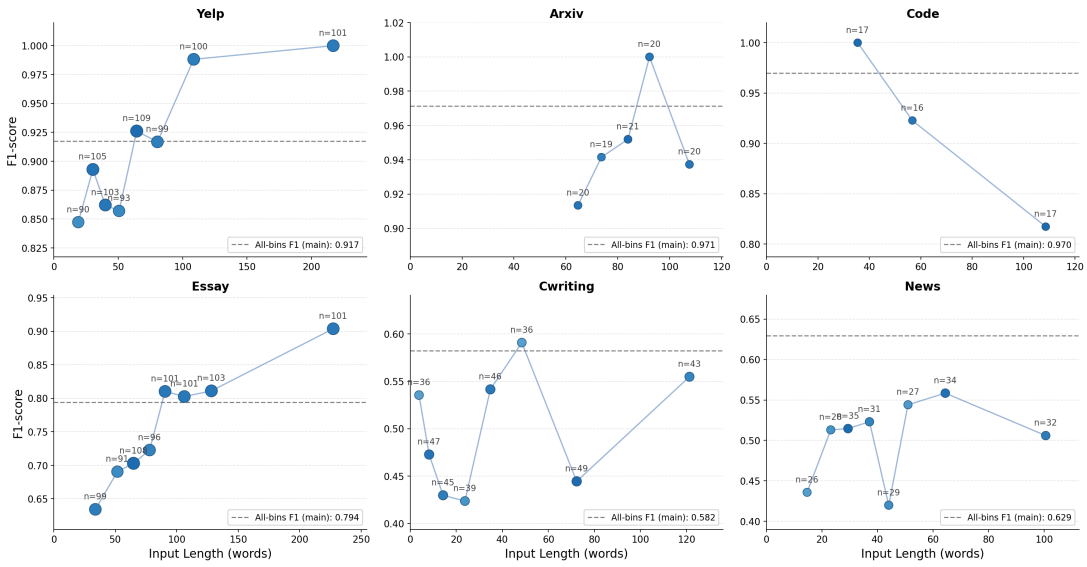

Figure 12: The impact of input length on the detection F1-score of CK-Detect across all domains.

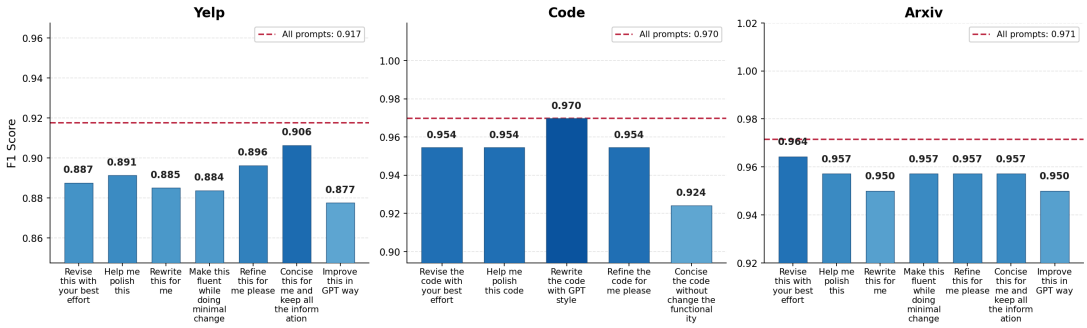

Figure 13: Detection performance under different rewrite prompts on the Yelp, Code and Arxiv dataset

for Code. Overall, longer inputs yield richer statistical and compression-based signals, thereby improving detection performance, however, the proposed method remains robust even for short texts by leveraging conditional compression signals.

E Prompting Details

All Rewrite-style methods (Raidar baseline, Rewrite-Conditioned, and CK-Detect) use the following seven prompts for natural-language domains (Yelp, Arxiv, CWriting, News, Essay). Each prompt is sent as a user instruction, and the text to rewrite is appended as the content with temperature 0.7 and top-p 0.9.

- `Revise this with your best effort`
- `Help me polish this`
- `Rewrite this for me`
- `Make this fluent while doing minimal change`
- `Refine this for me please`
- `Concise this for me and keep all the information`
- `Improve this in GPT way`

For the **Code** domain, domain-specific variants are used:

- `Rewrite the code with GPT style`
- `Rewrite the code for me`
- `Make this fluent while doing minimal change`
- `Refine this for me please`
- `Concise this for me and keep all the information`

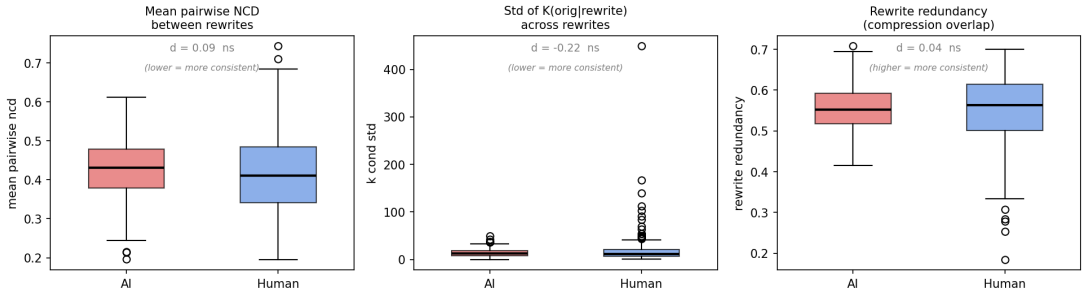

Figure 14: Rewrite consistency of human and AI-generated text on the Yelp dataset, illustrating the distribution of conditional Kolmogorov approximations and NCD values across $K = 1, 2, 3$ rewrites with the same prompt.

- `Improve this in GPT way`
- `Revise this with your best effort`

Impact of Rewrite Prompts. As shown in Figure 13, detection performance remains largely stable across different rewrite prompts, with less than a 5% variation between the best and worst prompts within each dataset. This indicates low sensitivity to the exact rewriting instruction. Aggregating multiple prompts provides a small but consistent improvement over single-prompt rewriting, or at least matches the best individual prompt. We do not consider multiple rewrites from the same prompt, as they tend to produce highly similar outputs with limited additional information.

As further shown in Figure 14, repeated rewrites under identical prompts yield highly similar compression features with low variance, suggesting that the signal is largely redundant. Overall, the benefit of multiple prompts stems mainly from the diversity of rewriting styles rather than from repeated sampling of the same instruction.