

# A Complexity-Theoretic Signature of AI-Generated Text

Ross Schmidt<sup>a</sup> and Ishanu Chattopadhyay<sup>○a</sup>

This manuscript was compiled on January 23, 2026

We introduce a model-agnostic approach for detecting LLM-generated text by estimating the entropy rate of text mapped to symbol streams over a (26+space)=27-letter alphabet, and show that long-form outputs from LLMs exhibit systematically lower entropy rates than human-authored prose. Our Nonparametric Entropy-Rate Oracle (NERO) exploits this signal to achieve competitive, training-free separation of human and AI text, without requiring model access. When substring-frequency cutoff profile for the estimator is used as input to a supervised Gaussian process classifier, NERO achieves near-perfect detection accuracy (AUC = 98.9%), providing a principled, complexity-theoretic framework for ranking generative capacity and tracking changes in LLM behavior over time.

entropy rate | AI-generated text detection | large language models | algorithmic complexity | probabilistic automata

With the rise of generative artificial intelligence (AI), particularly large language models (LLMs), reliably distinguishing human-authored prose from machine-generated text is no longer trivial (1–3). Here we test the hypothesis that long-form human- and machine-generated text differ in statistical complexity when mapped to symbol streams over a 27-character alphabet (26 letters plus space) and treated as sample paths of a stochastic process (4), whose entropy rate quantifies intrinsic complexity. Most existing detectors (5, 6) instead rely on language-model scoring (e.g., likelihood/rank statistics), supervised classifiers over stylistic representations, or compressibility proxies, and therefore depend on reference model access, training, or calibration to evolving generators.

We introduce a nonparametric, learning-free entropy-rate estimator that operates directly on text, without model access, supervision, or retraining. Used as a threshold score, the resulting Nonparametric Entropy-Rate Oracle (NERO) achieves competitive, training-free discrimination, and reveals systematically lower entropy rates in contemporary LLM outputs than in human prose under a shared symbolization. Optionally, using the estimator’s internal substring-frequency cutoff profile as features in a Gaussian process classifier yields near-perfect performance (AUC = 98.9%), surpassing model-driven baselines. Thus, NERO-estimated entropy rate functions as a model-agnostic statistic, akin to a physical quantity, for effective generative capacity, and provides a robust mechanism for AI text detection and a principled framework for ranking and tracking generative models over time.

## Results

To connect generative capacity with algorithmic complexity and then to entropy-rate of outputs, we begin by recalling the foundational concept of optimal two-part codes in algorithmic information theory (7). The Kolmogorov complexity  $K(x)$  of a string  $x$  is defined as the length of the shortest program (in a fixed universal programming language) that produces  $x$  and halts. It provides a rigorous, machine-independent measure of the information content or compressibility of a string. Instead of describing  $x$  directly, we can choose to first describe a finite set  $S$  that contains  $x$ , and then identify  $x$  within  $S$  by its index in a standard enumeration of all items in  $S$ . This leads to the two-part code representation:

$$K(x) \leq K(S) + \log |S| + O(1), \quad [1]$$

where  $K(S)$  is the complexity of describing the set  $S$ , and  $\log |S|$  is the number of bits required to identify  $x$  within  $S$ . When  $x$  is a *typical* element of  $S$ , i.e., that it does not admit any significantly shorter description than most elements of  $S$ , the inequality becomes tight up to additive constants. The optimal two-part code is obtained by minimizing the sum  $K(S) + \log |S|$  over all such sets containing  $x$ , yielding a minimal sufficient statistic for the data.

Author affiliations: <sup>a</sup>University of Kentucky

RS carried out experimental runs and wrote the paper, IC conceived of research, implemented the algorithm, wrote the paper and procured support.

Authors have no competing interests.

<sup>2</sup>To whom correspondence should be addressed. E-mail: ishanu.ch@uky.edu

125 **Table 1. Cohorts and average entropy rates**

data source	mean H	median H	std. dev. H	count
GPT-3.5 (webform)	0.54	0.53	0.11	100
GPT-4o (webform)	0.62	0.63	0.09	98
GPT-4o (API)	0.64	0.66	0.09	136
Gemini (API)	0.66	0.67	0.10	987
Claude (API)	0.71	0.72	0.06	944
GPT-4.0 (webform)	0.71	0.71	0.09	41
GPT-5 (API)	0.74	0.74	0.08	197
Gutenberg project	0.77	0.78	0.12	4341
Arxiv papers	1.32	1.19	0.55	42

Now lets apply these notions to quantify the generative ability of AI agents (and humans). With each agent  $G_i$  we associate a set  $S_i$  that defines the collection of strings that the agent can possibly generate. Once  $S_i$  is fixed, we assume that the agent produces an individual output  $x \in S_i$  using a standard sampling procedure. This shared decoding mechanism implies that the conditional complexity  $K(x | S_i)$  is approximately constant across agents for typical strings. An equivalent way to view this assumption, is that agents draw from the same or similar set of possible strings, *i.e.*, if one agent can generate a particular string  $x$ , then so can the other, perhaps with different odds. Under either interpretation, the implication is that the conditional term  $K(x | S_i)$  is effectively constant across agents, and since for typical  $x$  this term satisfies  $K(x | S_i) \approx \log |S_i|$  (7), we have:

$$\log |S_1| = \log |S_2| + O(1). \quad [2]$$

Then, we have the proposition (See Methods for proof):

**Proposition 1** (Two-Part Code to Entropy). *Let  $G_1$  and  $G_2$  be two generative processes outputting strings over a finite alphabet  $x \in \mathcal{A}^n$ , each respectively associated with a set  $S_1$  and  $S_2$  of possible outputs. Assuming  $\log |S_1| = \log |S_2| + O(1)$ , the following equivalence holds:*

$$K(S_1) > K(S_2) + O(1) \Leftrightarrow \underset{x \sim G_1}{E} [K(x)] > \underset{x \sim G_2}{E} [K(x)] + O(1).$$

Additionally, if  $G_i$  are stationary, then

$$K(S_1) > K(S_2) + O(1) \implies H(G_1) > H(G_2) + o(1), \quad [3]$$

where  $H(G_i)$  is the Shannon entropy rate of generator  $G_i$ .

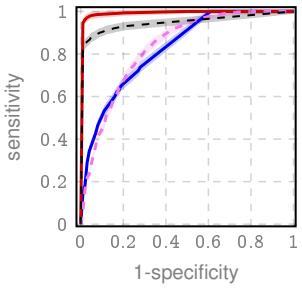
Proposition 1 motivates entropy rate as a discriminative statistic, suggesting recognition of AI-generated text by its systematically lower entropy rate. Estimating entropy rate for symbol streams is nontrivial (8). Accordingly, for NERO we adopt our prior nonparametric entropy-rate estimators based on probabilistic finite-state automata (PFSA) (9, 10) (See Supplementary Methods). With the standard approximate stationarity assumption (4) for long-form texts, Proposition 1 provides the key foundation of our claim.

## Experiments

To evaluate our claim, we use a corpus (See Table 1), comprising human-authored texts (Project Gutenberg and arXiv) and long-form outputs from contemporary LLMs (GPT-3.5, GPT-4o, GPT-4.0, Claude, Gemini, and GPT-5), using a mix of API and web-form access (See Supplementary Methods). All documents were lowercased and mapped to a 27-symbol English-plus-space alphabet, removing punctuation, digits, and non-ASCII characters. For each document we

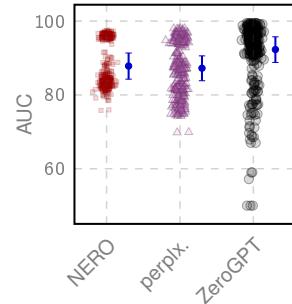
a. ROC (with 95% CI)

- NERO (no training): 82.7%
- NERO (with training): 98.9%
- - perplexity (HowGPT): 82.3%
- - ZeroGPT: 94.9%

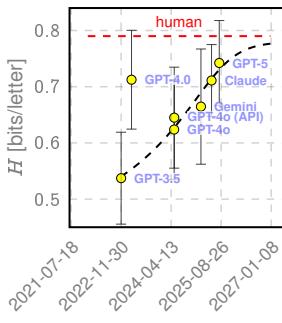


b. AUC (Cohort-Variation\*)

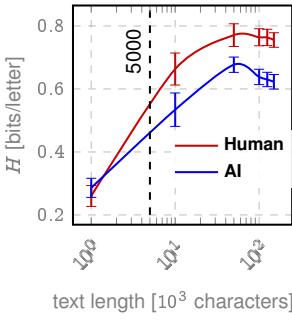
- \*381 cohort combinations
- NERO-notraining
- △ perplexity (HowGPT)
- ZeroGPT



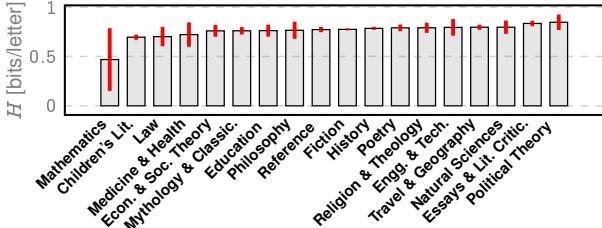
c. Birthtime Vs Performance



d. Length dependence



e. Topic dependence in human prose (Gutenberg)



**Fig. 1. NERO performance.** a, Pooled human-AI ROC curves for training-free NERO (median entropy-rate estimate  $\hat{H}$ ), and with a Gaussian-process classifier alongside baselines (perplexity from HowGPT and ZEROGPT scoring, See Supplementary Methods). b, Cross-cohort robustness across 381 out-of-sample discrimination tasks obtained by varying the human reference set (Gutenberg, arXiv, or both) and the subset of LLM generators. c, Mean entropy rates plotted at model release dates (“birthtime”), illustrating upward drift toward the human regime; a weighted Richards fit with a fixed ceiling at the mean human rate is overlaid for descriptive smoothing. d, Entropy-rate estimates versus document length under truncation (using Gutenberg texts for humans), indicating a practical lower bound for reliable discrimination at shorter lengths around 5,000 characters. e, Genre-stratified NERO entropy rates for human prose; error bars denote 95% intervals.

compute a family of entropy-rate estimates across substring-frequency thresholds  $\{m_1, \dots, m_M\}$ , where  $m$  specifies the minimum number of occurrences required for a substring to be retained in the PFSA construction (see Methods). In the training-free setting, we use the median across thresholded  $m$ -estimates ( $H$ ). In the trained-NERO setting, the full  $m$ -dependence profile is used as features for training a Gaussian process classifier (11); importantly, the underlying entropy-rate estimator is unchanged. Using NERO, our estimates for arxiv papers ( $\approx 1.3$  bits/letter, Table 1) closely align with classic estimates of English entropy under a 27-symbol alphabet (4), whereas Gutenberg prose exhibits lower

187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248

character-level entropy, consistent with greater redundancy in narrative text. Our experimental results (Fig. 1) demonstrate:

(1) *Training-free detection.* Using  $H$  directly as a detection score yields strong LLM-human discrimination without any training. (AUC = 0.824 on the pooled evaluation, Fig. 1a).

(2) *Training using substring-frequency threshold m-dependence.* We trained a Gaussian-process classifier on the  $m$ -dependent feature vector (see Methods) using a 50/50 train-test split, improving out-of-sample discrimination to AUC = 0.989 (Fig. 1a). For comparison, the perplexity baseline (HowGPT) attains AUC = 0.830, while ZEROGPT attains AUC = 0.946 on the same pooled evaluation.

(3) *Cross-cohort robustness.* Fig. 1b reports out-of-sample AUCs for  $(2^2 - 1) \times (2^7 - 1) = 381$  distinct human-AI discrimination task obtained by varying the human reference and the subset of LLM generators included (See SI Methods). Across this combinatorial suite, training-free NERO maintains consistently strong performance, whereas trained baseline detectors exhibit substantially greater sensitivity to corpus composition and generator selection. This contrast underscores the NERO estimate as a calibration-free, model-agnostic statistic rather than a generator-dependent detector.

(4) *Temporal trajectory toward the human regime.* Fig. 1c plots cohort-level mean NERO-computed entropy rates at model release times with a weighted Richards (generalized logistic) fit and a fixed asymptotic ceiling given by the mean human rate (SI Methods), showing successive model generations shifting upward toward the human regime with diminishing returns. While not a calendar-date forecast, this supports the qualitative conclusion that model improvements are narrowing the entropy-rate gap. GPT-4.0 is a notable outlier, consistent with a regime shift induced by the architectural jump introduced in this model (multi-modal access and other experimental features) (12).

(5) *Length dependence.* Human-AI separation is robust for articles with  $> 10,000$  characters (about the length of a full-length magazine article), with loss of discrimination possibly at lengths under 5,000 characters (Fig. 1d).

(6) *Genre dependence.* Formulaic human genres (Mathematical text, Children's Literature) attain lower entropy rates than expository genres (e.g. Literary criticism and political theory), defining the primary hard cases (Fig. 1e).

## Discussion

We show that, under shared symbolization, long-form human-authored text exhibits a consistently higher estimated entropy rate than contemporary LLMs. In the two-part code perspective, when outputs are compared within a common representation, differences in entropy rate are consistent with differences in the effective descriptive complexity of the underlying generative mechanisms, suggesting that humans might operate with a richer, more complex internal model than current AI systems.

A complementary intuition follows from *Levin's universal semimeasure*  $m(x)$  (13), which lower-bounds the probability assignable to strings by any computable process. It defines a universal prior by summing over all prefix-free programs  $p$  (14) that produce  $x$  on a universal Turing machine  $U$  (15):

$$m(x) = \sum_{p: U(p)=x} 2^{-|p|}. \quad [4]$$

and Levin's Coding Theorem links to Kolmogorov complexity,

$$K(x) = -\log m(x) + O(1), \quad [5]$$

which implies that strings with low  $m(x)$  are algorithmically complex and lie in the deep tails of any computable generative process. Humans operating with an internal generative model  $S_{\text{human}}$  that likely possesses significantly higher descriptive complexity than models used in AI systems, are able to generate outputs that lie deeper in the algorithmic tail of  $m(x)$ , i.e., outputs that are rarer, more contextually novel, and of higher complexity. In contrast, LLMs are trained on finite, empirical datasets where such rare strings are underrepresented (by definition of being rare); since strings with low  $m(x)$  are, by definition, uncommon, they are statistically unlikely to appear with sufficient frequency in training corpora. This possibly limits the effective model complexity  $K(S_{\text{AI}})$ , and hence constrains the AI's ability to reproduce or extrapolate into these low-probability regions.

Thus, despite the limitations of requiring longer texts and greater computational effort, NERO yields a model-agnostic statistic that enables best-in-class detection, and principled, learning-free tracking of generative behavior over time.

## Materials and Methods

**Data acquisition.** Human-authored texts were obtained from Project Gutenberg (English-language compositions) and technical manuscripts from arXiv via the official OAI-PMH interface. AI-generated texts were produced via webform and API access to the LLMs listed in Table 1. Details on prompt design, API access, and NERO computation are provided in the Supplementary Methods.

**Code availability.** A reproducible implementation of the entropy-rate estimator and end-to-end evaluation pipeline is available under an MIT license at <https://github.com/zeroknowledge/discovery/nero>.

**ACKNOWLEDGMENTS.** This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) under the ARC program. Views, opinions, and findings expressed are solely those of the authors.

1. Prepare for truly useful large language models. *Nat. Biomed. Eng.* **7**, 85–86 (2023).
2. E Crothers, N Japkowicz, HL Viktor, Machine-generated text: A comprehensive survey of threat models and detection methods. *IEEE Access* (2023).
3. AJ Thirunavukarasu, et al., Large language models in medicine. *Nat. medicine* **29**, 1930–1940 (2023).
4. TM Cover, RC King, A convergent gambling estimate of the entropy of english. *IEEE Transactions on Inf. Theory* (1978).
5. E Mitchell, Y Lee, A Khazatsky, CD Manning, C Finn, Detectgpt: Zero-shot machine-generated text detection using probability curvature in *Proceedings of the 40th International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, Vol. 202, pp. 24950–24962 (2023).
6. J Su, TY Zhuo, D Wang, P Nakov, Detectllm: Leveraging log rank information for zero-shot detection of machine-generated text in *Findings of the Association for Computational Linguistics: EMNLP 2023*. (2023).
7. P Gacs, J Tromp, PMB Vitányi, Algorithmic statistics. *IEEE Trans. Inf. Theory* **47**, 2443–2463 (2001).
8. P Grassberger, Estimating the information content of symbol sequences and efficient codes. *IEEE Transactions on Inf. Theory* **35**, 669–675 (1989).
9. I Chattopadhyay, H Lipson, Computing Entropy Rate Of Symbol Sources & A Distribution-free Limit Theorem. *ArXiv e-prints* 1401.0711 (2014).
10. I Chattopadhyay, H Lipson, Abductive learning of quantized stochastic processes with probabilistic finite automata. *Philos Trans A* **371**, 20110543 (2013).
11. CK Williams, CE Rasmussen, *Gaussian processes for machine learning*. (MIT press Cambridge, MA) Vol. 2, (2006).
12. OpenAI, Gpt-4 technical report (2023).
13. LA Levin, Laws of information conservation (non-growth) and aspects of the foundation of probability theory. *Probl. Inf. Transm.* **10**, 206–210 (1974) Originally in Russian.
14. M Li, PMB Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*. (Springer, New York, NY), 3rd edition, (2008).
15. GJ Chaitin, A theory of program size formally identical to information theory. *J. ACM* **22**, 329–340 (1975).
16. Y Horibe, A note on kolmogorov complexity and entropy. *Appl. Math. Lett.* **16**, 1129 – 1130 (2003).

## Supplementary Methods

### Proof of Proposition 1.

*Proof.* For typical  $x \sim G_i$  (7, 14),

$$K(x) = K(S_i) + \log |S_i| + O(1). \quad [6]$$

Taking expectations over  $x \sim G_i$ , we have

$$\mathbb{E}_{x \sim G_i}[K(x)] = K(S_i) + \log |S_i| + O(1). \quad [7]$$

Subtracting, we find

$$\begin{aligned} & \mathbb{E}_{x \sim G_1}[K(x)] - \mathbb{E}_{x \sim G_2}[K(x)] \\ &= K(S_1) - K(S_2) + \log |S_1| - \log |S_2| + O(1). \end{aligned} \quad [8]$$

Invoking our assumption (Eq. Eq. (2)), this simplifies to

$$\mathbb{E}_{x \sim G_1}[K(x)] - \mathbb{E}_{x \sim G_2}[K(x)] = K(S_1) - K(S_2) + O(1).$$

This completes the first part. Noting that expected Kolmogorov complexity of strings from a stationary source satisfies (14, 16):

$$\lim_{n \rightarrow \infty} \frac{1}{n} \mathbb{E}_{x \sim G_i}[K(x)] = H(G_i). \quad [9]$$

establishes the second part.  $\square$

**Nonparametric entropy-rate estimation.** We estimate the entropy rate of a stationary ergodic symbol stream using the PFSA-based nonparametric estimator introduced by Chattopadhyay *et al.* (9, 10). Given a sequence  $s = s_1 \cdots s_n$  over a finite alphabet  $\Sigma$ , the method constructs empirical next-symbol distributions conditioned on observed substrings and identifies states of an a priori unknown underlying PFSA generator as equivalence classes of histories that lead to similar future distributions. Rare substrings are excluded via a frequency threshold  $m$ , ensuring statistical reliability without assuming a parametric model or access to the underlying generator. The entropy rate is then estimated as a weighted average of the Shannon entropies of empirical symbol-generation distributions associated with that context, with the weights inferred as observed frequencies of the classes of histories that lead to individual states. The estimator is provably consistent under stationarity and ergodicity, requires no training or reference model, and operates directly on the observed text stream. Full algorithmic details and theoretical guarantees are provided in (9, 10).

**Pseudocode and Runtime Complexity.** To enhance robustness and mitigate sensitivity to internal thresholds, we apply this estimator  $M$  times across a range of substring frequency thresholds, and report the median of the resulting entropy estimates, improving concentration guarantees (Theorem 1), with the error probability decaying exponentially in  $M$ .

Runtime complexity, including threshold-based pruning and median aggregation over  $M$  thresholds, is  $O(nM|\Sigma|)$ , where  $n$  is the length of the observed stream and  $|\Sigma|$  the alphabet size (9). Thus, for a fixed  $M$ , the estimate has input-linear runtime complexity.

---

### Algorithm 1 Robust Entropy-Rate Estimation

**Input:** Symbol stream  $x_1^n$ , alphabet  $\Sigma$ , threshold range  $\{m_1, m_2, \dots, m_M\}$

**Output:** Robust entropy-rate estimate  $\hat{H}_n$

**foreach**  $m \in \{m_1, m_2, \dots, m_M\}$  **do**

Compute  $\hat{H}_n^{(m)}$  with Chattopadhyay (9) with threshold  $m$  (ignore substrings with  $< m$  occurrences)

Compute  $\hat{H}_n \leftarrow \text{median} \left( \hat{H}_n^{(m_1)}, \hat{H}_n^{(m_2)}, \dots, \hat{H}_n^{(m_M)} \right)$

**return**  $\hat{H}_n$

---

**Theorem 1** (Robustness and Concentration of the Aggregated Entropy Estimate). *Let  $x_1^N$  be a finite realization from a stationary, ergodic stochastic process over a finite alphabet  $\Sigma$ . Let  $\hat{H}_n^{(m)}$  denote the entropy-rate estimate obtained by applying the algorithm of Chattopadhyay *et al.* (9, 10) with threshold  $m$  (substrings occurring  $< m$  times ignored). Define the aggregated estimator*

$$\hat{H}_n := \text{median} \left\{ \hat{H}_n^{(m_1)}, \hat{H}_n^{(m_2)}, \dots, \hat{H}_n^{(m_M)} \right\}.$$

If the indicators  $Z_i$  defined below are independent, then

$$\mathbb{P} \left( |\hat{H}_n - H_\star| > \epsilon \right) \leq \exp \left( -2M \left( \frac{1}{2} - \delta \right)^2 \right). \quad [435]$$

More generally, when the  $\hat{H}_n^{(m_i)}$  are obtained from randomly drawn length- $L$  substreams of a fixed length- $N$  text and hence may be dependent due to overlap, the same argument applies with an effective number of approximately independent draws

$$M_{\text{eff}} \asymp \frac{N}{L}, \quad [436]$$

so that

$$\mathbb{P} \left( |\hat{H}_n - H_\star| > \epsilon \right) \lesssim \exp \left( -2M_{\text{eff}} \left( \frac{1}{2} - \delta \right)^2 \right). \quad [437]$$

*Proof.* It follows from Chattopadhyay *et al.* (9, 10) that for sufficiently large  $n$ , for each threshold  $m \in \{m_1, m_2, \dots, m_M\}$ ,

$$\mathbb{P} \left( |\hat{H}_n^{(m)} - H_\star| > \epsilon \right) \leq \delta, \quad [438]$$

for some fixed  $\epsilon > 0$  and  $\delta < \frac{1}{2}$ . For each  $m_i$ , define

$$Z_i = \mathbf{1} \left\{ |\hat{H}_n^{(m_i)} - H_\star| > \epsilon \right\}, \quad \mathbb{E}[Z_i] \leq \delta. \quad [439]$$

The median estimator  $\hat{H}_n$  deviates by more than  $\epsilon$  only if at least half of the individual estimates do:

$$S = \sum_{i=1}^M Z_i \geq \frac{M}{2}. \quad [440]$$

If  $Z_1, \dots, Z_M$  are independent, Hoeffding's inequality yields

$$\mathbb{P} \left( S \geq \frac{M}{2} \right) \leq \exp \left( -2M \left( \frac{1}{2} - \delta \right)^2 \right), \quad [441]$$

which implies the stated bound.

If instead the estimates are computed from randomly drawn length- $L$  substreams of a fixed length- $N$  text, the resulting  $Z_i$  may be dependent due to overlap. In that case, the same calculation can be interpreted in terms of an effective number of approximately independent draws  $M_{\text{eff}} \asymp N/L$  (i.e., replacing  $M$  by  $M_{\text{eff}}$ ), yielding the stated overlap-limited bound.  $\square$

**Classifier design (trained-NERO).** For each document we form a fixed-dimensional feature vector  $x \in \mathbb{R}^M$  from the PFSA entropy-rate estimator by evaluating the estimate at a prescribed set of substring-frequency cutoffs  $\{m_1, \dots, m_M\}$ . Specifically, the  $j$ th feature is the entropy-rate estimate computed using cutoff  $m_j$ , so that  $x_j = \hat{H}(m_j)$ . Missing feature values (arising when insufficient support is available at a given cutoff) are imputed with zeros using a constant-value imputer.

We evaluate several standard classifiers on these feature vectors, including random forests, extremely randomized trees, AdaBoost, gradient boosting, support vector machines (SVM), and Gaussian-process (GP) classification. Unless otherwise noted, models are fit on a random 50/50 train-test split of the pooled corpus, and performance is quantified by ROC/AUC on the held-out split. For tree-based models we use class-balanced weighting and 1000–5000 component estimators to stabilize performance; for SVM we use probabilistic calibration to obtain class probabilities.

We use GP classification as the trained-NERO readout as it yields the best performance under cross-validation. The GP classifier is implemented with an RBF kernel,  $k(x, x') = \sigma^2 \exp(-\|x - x'\|^2/(2\ell^2))$ , using the standard GaussianProcessClassifier implementation (11).

497 **Birthtime trajectory fitting (Fig. 1c).** To summarize how estimated  
 498 entropy rate of generated text changes across successive LLM  
 499 models, we fit a parametric growth curve to the model-specific time  
 500 series shown in Fig. 1c. Each model is represented by a timestamp  
 501 (“birthtime”) corresponding to its release date, the mean NERO  
 502 estimate over generated tests, and an estimated standard deviation  
 503 (error bar). Birthtimes were converted to elapsed time in days  
 504 relative to the earliest cohort,  $x_i = (t_i - t_0)$ , yielding observations  
 505 ( $x_i, y_i, \sigma_i$ ).

506 We fit a Richards (generalized logistic) curve with a fixed  
 507 asymptotic upper bound  $U$  (corresponding to mean NERO estimate  
 508 for human prose), parameterized as

$$f(x; L_0, k, x_0, \nu) = L_0 + \frac{U - L_0}{(1 + \exp\{-k(x - x_0)\})^{1/\nu}},$$

509 where  $L_0$  is the lower asymptote,  $k$  the growth rate,  $x_0$  the  
 510 inflection location, and  $\nu$  the shape parameter.

511 Parameters  $\theta = (L_0, k, x_0, \nu)$  were estimated by weighted  
 512 nonlinear least squares using `scipy.optimize.curve_fit` with  
 513 heteroscedastic weights given by the cohort standard deviations  
 514  $\sigma_i$ . We used the initialization  $(L_0, k, x_0, \nu) = (\max\{0, \min_i y_i - 0.03\}, 0.006, \text{median}(x), 0.6)$  and box constraints  
 515  $L_0 \in [0, \min_i y_i]$ ,  $k \in [10^{-6}, 3]$ ,  $x_0 \in [-2000, 5000]$ ,  $\nu \in [0.05, 5]$ . The fitted parameters and their asymptotic standard  
 516 errors were obtained from the returned covariance estimate  $\widehat{\text{Cov}}(\hat{\theta})$ ,  
 517 and standard chi-square diagnostics were computed from weighted  
 518 residuals.

519 **Data processing.** For each corpus (human-authored and AI-  
 520 generated), a structured CSV file was constructed at the document  
 521 level. Each row corresponds to a single novel and contains the  
 522 following fields: the entropy-rate estimate computed by NERO,  
 523 the perplexity-based detector score reported by HowGPT, and the  
 524 estimated percentage of AI-generated text reported by ZeroGPT.  
 525 These tables form the basis for all quantitative comparisons  
 526 reported in the main text and figures.

527 **Topics and text generation.** Topics used to prompt the AI models  
 528 were obtained from a publicly available repository of narrative  
 529 prompts (<https://www.plot-generator.org.uk/story-ideas/>). The precise  
 530 semantic content of individual prompts is not critical for long-form  
 531 text generation; rather, it is sufficient that prompts span a broad  
 532 range of genres to avoid systematic topical bias. All topics were  
 533 stored in a JSON file and loaded programmatically during text  
 534 generation.

535 Each AI model was prompted with the identical ordered list of  
 536 topics. This design ensures that differences in estimated entropy  
 537 rate across models are not attributable to differences in prompt  
 538 content.

539 **AI text generation prompts.** All AI-generated novels were produced  
 540 using a fixed prompting protocol designed to elicit long-form  
 541 narrative prose while minimizing structural or stylistic constraints  
 542 beyond those necessary for length control and coherence. Identical  
 543 prompt templates were used across models. Text generation was  
 544 performed either via official REST APIs or, in some cases, via  
 545 the OpenAI web interface using the same prompts and default  
 546 generation settings; no prompt content differed between access  
 547 modes.

548 **Initial generation prompt.** Each novel was initiated using the  
 549 following base prompt, with bracketed fields instantiated program-  
 550 matically:

551 I want you to act as a novelist writing about {topic}. The  
 552 total length of the novel is about {novel\_desired\_length}  
 553 characters. After generating each section of the novel, I will  
 554 tell you how much text you have generated so far. In the  
 555 novel, include plot development, characters, dialogue, and  
 556 an overall narrative arc. Generate only the text of the novel  
 557 itself, without annotations, explanations, or commentary.  
 558 Write in full paragraphs and chapters of approximately  
 559 {chapter\_length} characters. Continue generating Chapter 1  
 560 until I tell you to stop. Generate exclusively novel text,  
 561 with no metadata, confirmations, or chapter labels.

562 **Continuation prompt.** To extend generation beyond the initial  
 563 context window and reach the target length, generation proceeded  
 564 iteratively using a continuation prompt. At each step, previously  
 565 generated text (truncated as necessary) was prepended, followed  
 566 by the continuation instruction:

567 {previously generated text}

568 Here is the text generated so far. You have generated  
 569 approximately {novel\_current\_length} characters. Continue  
 570 writing Chapter {current\_chapter} of the novel. Maintain  
 571 narrative continuity and write in complete paragraphs.  
 572 Do not generate annotations, explanations, summaries, or  
 573 metadata. Generate exclusively novel text.

574 **AI text generation implementation details.** AI-generated novels  
 575 were produced using an automated, iterative prompting pipeline  
 576 implemented in Python. Each novel was generated toward a  
 577 target length of 150,000 characters using repeated completion  
 578 calls under default model parameters; no temperature, nucleus  
 579 sampling, or penalty parameters were explicitly set. Only the  
 580 maximum completion length was controlled.

581 Chapter indices were inferred implicitly as  $[L/\ell] + 1$ , where  $L$   
 582 is the cumulative character length generated so far and  $\ell = 15,000$   
 583 is the nominal chapter length. Chapters were not explicitly labeled  
 584 in the generated text.

585 To support long-form generation within finite model context  
 586 windows, a sliding-window strategy was employed. When the  
 587 concatenated prompt exceeded the available context length, only  
 588 the most recent suffix of the previously generated text was  
 589 retained, ensuring that the prompt plus maximum completion  
 590 length remained within the model’s context window.

591 Generation continued until the target length was reached or  
 592 until the model declined to continue. Responses yielding fewer  
 593 than 200 output tokens were treated as refusals; in such cases, all  
 594 text associated with that generation attempt was discarded and  
 595 excluded from analysis.

596 API usage was subject to rate limits on requests per minute  
 597 and token throughput. When limits were exceeded, generation  
 598 was paused briefly before resuming. For each novel, all raw  
 599 API responses and the cumulative generated text were stored  
 600 in per-document metadata files to enable reproducibility and  
 601 auditability. The generation pipelines are implemented in the  
 602 following notebooks:

1. `gemini`: [https://github.com/zeroknowledgediscovery/nero/api\\_data\\_collection/gemini\\_2.5\\_pro/get\\_novels.ipynb](https://github.com/zeroknowledgediscovery/nero/api_data_collection/gemini_2.5_pro/get_novels.ipynb)
2. `gpt4o`: [https://github.com/zeroknowledgediscovery/nero/api\\_data\\_collection/openai/gpt4o/get\\_novels.ipynb](https://github.com/zeroknowledgediscovery/nero/api_data_collection/openai/gpt4o/get_novels.ipynb)
3. `gpt5`: [https://github.com/zeroknowledgediscovery/nero/api\\_data\\_collection/openai/gpt5/get\\_novels.ipynb](https://github.com/zeroknowledgediscovery/nero/api_data_collection/openai/gpt5/get_novels.ipynb)
4. `claude`: [https://github.com/zeroknowledgediscovery/nero/api\\_data\\_collection/claude\\_sonnet\\_4/get\\_novels.ipynb](https://github.com/zeroknowledgediscovery/nero/api_data_collection/claude_sonnet_4/get_novels.ipynb)

603 **LLM access.** AI-generated novels were produced using REST API  
 604 access or web-interface access to the following models and versions:

- **Gemini**: `gemini-2.5-pro`, accessed between October 8, 2025 and  
 605 October 16, 2025.
- **Claude**: `claude-sonnet-4-20250514`, accessed between Au-  
 606 gust 26, 2025 and September 22, 2025.
- **GPT-4o**: `gpt-4o-2024-08-06`, accessed between Novem-  
 607 ber 13, 2025 and November 14, 2025.
- **GPT-5**: `gpt-5-2025-08-07`, accessed between October 26, 2025  
 608 and November 8, 2025.

609 **API versus web-interface access.** To mitigate potential bias arising  
 610 from access modality, AI-generated texts were obtained using a mix of  
 611 official REST API access and provider web interfaces, as summarized in  
 612 Table 1. This design ensures that the reported entropy-rate differences  
 613 are not artifacts of a particular deployment channel, prompting  
 614 wrapper, or hidden system configuration. In particular, GPT-4.0  
 615 texts were generated using both API-based access and the OpenAI  
 616 web interface under identical prompting protocols. We observed no  
 617 statistically significant difference in estimated entropy rates between  
 618 GPT-4.0 outputs obtained via API versus web-interface access

619 **Human-authored text corpus.** Human-authored novels were re-  
 620 trieved from the Project Gutenberg corpus (<https://gutenberg.org/cache/epub/feeds/>), which provides full-text documents together with struc-  
 621 tured metadata. All available text and metadata files were downloaded,  
 622 and only English-language works were retained based on metadata

621 fields. Legal headers and boilerplate text were removed using the  
622 `gutenbergpy` library. Novels shorter than 150,000 characters after  
623 preprocessing were excluded to ensure sufficient length for stable  
624 entropy-rate estimation. 683  
625 Technical manuscripts were collected from arXiv via the official  
626 OAI-PMH interface and converted from `.tex` source to plain text.  
627 These texts were used as a distinct human-authored reference corpus. 684  
628 **Baseline detectors.** The HowGPT (<https://howkgpt.nyuad.nyu.edu/>) and 685  
629 the ZeroGPT detectors (<https://www.zerogpt.com/>) were accessed between 686  
630 November 2, 2025 and November 25, 2025, with access dates varying by 687  
631 model cohort. From these tools, the reported perplexity-based detector 688  
632 score (HowGPT) and the estimated percentage of AI-generated text 689  
633 (ZeroGPT) were recorded. 690  
634 For baseline detector comparisons, the central 15,000 characters of 691  
635 each novel were extracted and submitted to two deployed detection 692  
636 tools: HowGPT and ZeroGPT. The central segment was used to avoid 693  
637 residual legal headers in human-authored texts; the same procedure was 694  
638 applied to AI-generated texts to maintain parity. HowGPT/ZeroGPT 695  
639 were queried on fixed-length segments due to interface constraints. 696  
640 NERO was evaluated on full texts to reflect the intended long-form 697  
641 detection regime. 698  
642 **Perplexity baseline (HowGPT).** The perplexity-based baseline was 699  
643 implemented using the HowGPT framework, which computes token-level 700  
644 likelihood statistics under a reference language model and 701  
645 aggregates them into a scalar detector score. We use the reported 702  
646 detector score directly rather than raw perplexity values. 703  
647 **ZeroGPT baseline.** The ZeroGPT baseline was obtained using the 704  
648 deployed ZeroGPT web interface as described above, which returns 705  
649 an estimated fraction of the submitted text labeled as AI-generated 706  
650 (reported as a percentage). We record the percentage score exactly 707  
651 as returned by the interface for each submitted segment, without 708  
652 additional calibration or post-processing. This returned score was used 709  
653 to construct the ROC curve. Because ZeroGPT is a proprietary system 710  
654 whose underlying model, decision rule, and versioning may change over 711  
655 time, the score should be interpreted as an external detector output at 712  
656 the time of access rather than as a reproducible intrinsic statistic of 713  
657 the text. 714  
658 **Relation to standard LLM benchmarks.** Existing LLM benchmark 715  
659 suites primarily evaluate task-specific performance (e.g., reasoning 716  
660 accuracy, problem solving, or code synthesis) under controlled prompts 717  
661 and scoring rules. In contrast, NERO estimates a distributional 718  
662 property of long-form generated text—entropy rate under a fixed 719  
663 symbolization—which reflects output complexity rather than task 720  
664 competence. These quantities are not directly commensurate, and 721  
665 strong correspondence is neither assumed nor required: models may 722  
666 score highly on benchmarks while exhibiting low entropy-rate regularity 723  
667 in extended prose. Accordingly, we do not attempt exhaustive 724  
668 benchmark comparisons here. Studying how entropy-rate trajectories 725  
669 relate to aggregate benchmark indicators across model releases is an 726  
670 interesting direction for future work. 727  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682