Model-agnostic Recognition of AI-Generated Text with Descriptional Complexity Measure

Ross Schmidt[a] **and Ishanu Chattopadhyay**[a]

This manuscript was compiled on January 20, 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 ($\mathrm{AUC} = 98.9\%$), 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:

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

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: [a]University of Kentucky

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

Authors have no competing interests.

[2]To whom correspondence should be addressed. E-mail: ishanu_ch@uky.edu

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

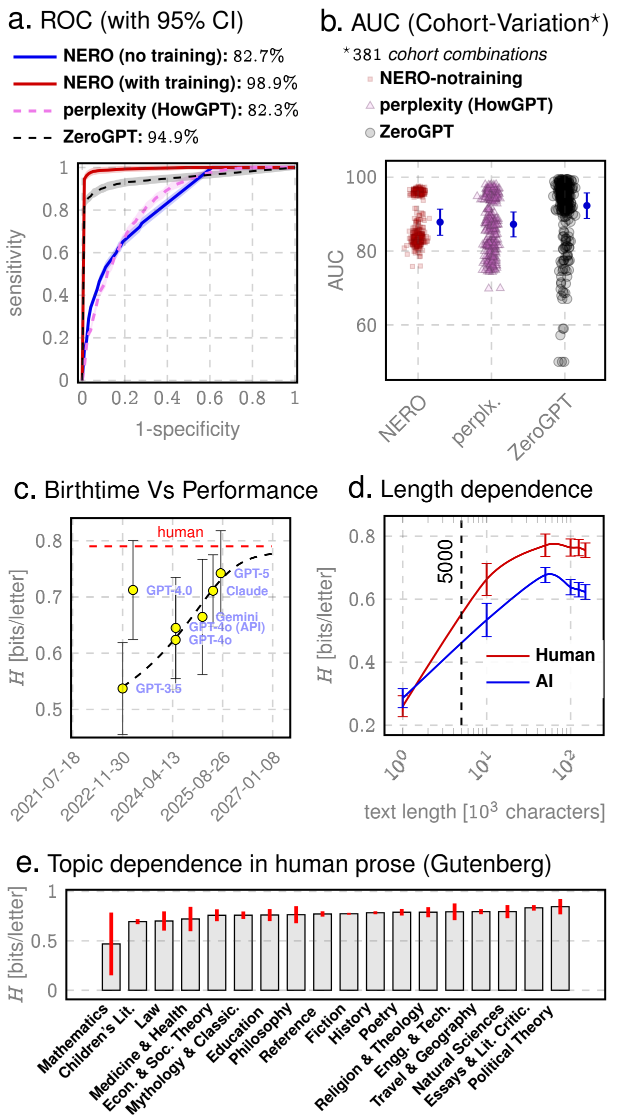

Fig. 1. NERO performance. **a**, Pooled human-AI ROC curves for training-free NERO (median entropy-rate estimate $\hat{H}$), and with a Gaussian-process classifier alongside baselines (perplexity from HowGPT and ZeroGPT scoring, See Supplementary Methods). **b**, Cross-cohort robustness across 381 out-of-sample 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 $10,000$ characters. **e**, Genre-stratified NERO entropy rates for human prose; error bars denote 95% intervals.

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 $K(x \mid S_i)$ 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 $K(x \mid S_i)$ is effectively constant across agents, and since for typical x this term satisfies $K(x \mid S_i) \approx \log|S_i|$ (7), we have:

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

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

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

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 compute a family of entropy-rate estimates across substring-frequency thresholds $\{m_1, \ldots, m_M\}$, 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 (≈ 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 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):

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

and Levin's Coding Theorem links to Kolmogorov complexity,

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

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_{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 $K(S_{\mathrm{AI}})$, 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/your-repo/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 Vitanyi, *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

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

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):

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

establishes the second part. □

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 Σ, 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 $\{m_1, m_2, \ldots, m_M\}$
Output: Robust entropy-rate estimate $\hat{H}_n$
foreach $m \in \{m_1, m_2, \ldots, m_M\}$ do
    Compute $\hat{H}_n^{(m)}$ with Chattopadhyay (9) with threshold $m$ (ignore substrings with $< m$ occurrences)
Compute $\hat{H}_n \leftarrow \mathrm{median}\left(\hat{H}_n^{(m_1)}, \hat{H}_n^{(m_2)}, \ldots, \hat{H}_n^{(m_M)}\right)$
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 Σ. 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*

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

If the indicators Z_i defined below are independent, then

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

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

$$M_{\text{eff}} \asymp \frac{N}{L},$$

so that

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

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

$$\mathbb{P}\left(\left|\hat{H}_n^{(m)} - H_\star\right| > \epsilon\right) \le \delta,$$

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

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

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

$$S = \sum_{i=1}^{M} Z_i \ge \frac{M}{2}.$$

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

$$\mathbb{P}\left(S \ge \frac{M}{2}\right) \le \exp\left(-2M\left(\frac{1}{2}-\delta\right)^2\right),$$

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_{eff}), yielding the stated overlap-limited bound. □

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 $\{m_1, \ldots, m_M\}$. Specifically, the jth feature is the entropy-rate estimate computed using cutoff m_j, so that $x_j = \widehat{H}(m_j)$. 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, $k(x, x') = \sigma^2 \exp\left(-\|x - x'\|^2/(2\ell^2)\right)$, using the standard `GaussianProcessClassifier` implementation (11).

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

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

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

where L_0 is the lower asymptote, k the growth rate, x_0 the inflection location, and ν the shape parameter.

Parameters $\theta = (L_0, k, x_0, \nu)$ were estimated by weighted nonlinear least squares using `scipy.optimize.curve_fit` with heteroscedastic weights given by the cohort standard deviations σ_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 $L_0 \in [0, \min_i y_i)$, $k \in [10^{-6}, 3]$, $x_0 \in [-2000, 5000]$, $\nu \in [0.05, 5]$.. The fitted parameters and their asymptotic standard errors were obtained from the returned covariance estimate $\widehat{\text{Cov}}(\hat{\theta})$, and standard chi-square diagnostics were computed from weighted residuals.

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

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

Each AI model was prompted with the identical ordered list of topics. Consequently, the file `{model}18.txt` contains text generated from the 18th topic in the shared topic list, independent of the model used. This design ensures that differences in estimated entropy rate across models are not attributable to differences in prompt content.

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

Initial generation prompt. Each novel was initiated using the following base prompt, with bracketed fields instantiated programmatically:

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

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

> {previously generated text}
> Here is the text generated so far. You have generated approximately {novel_current_length} characters. Continue writing Chapter {current_chapter} of the novel. Maintain narrative continuity and write in complete paragraphs. Do not generate annotations, explanations, summaries, or metadata. Generate exclusively novel text.

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

Chapter indices were inferred implicitly as $\lfloor L/\ell \rfloor + 1$, where L is the cumulative character length generated so far and $\ell = 15{,}000$ is the nominal chapter length. Chapters were not explicitly labeled in the generated text.

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

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

API usage was subject to rate limits on requests per minute and token throughput. When limits were exceeded, generation was paused briefly before resuming. For each novel, all raw API responses and the cumulative generated text were stored in per-document metadata files to enable reproducibility and auditability.

The full generation pipelines are implemented in the following notebooks:

```
gemini:
nero-data/api_data_collection/gemini_2.5_pro/get_novels.ipynb
gpt4o:
nero-data/api_data_collection/openai/gpt4o/get_novels.ipynb
gpt5:
nero-data/api_data_collection/openai/gpt5/get_novels.ipynb
claude:
nero-data/api_data_collection/claude_sonnet_4/get_novels.ipynb
```

Human-authored text corpus. Human-authored novels were retrieved from the Project Gutenberg corpus (https://gutenberg.org/cache/epub/feeds/), which provides full-text documents together with structured metadata. All available text and metadata files were downloaded, and only English-language works were retained based on metadata fields. Legal headers and boilerplate text were removed using the `gutenbergpy` library. Novels shorter than 150,000 characters after preprocessing were excluded to ensure sufficient length for stable entropy-rate estimation.

Technical manuscripts were collected from arXiv via the official OAI-PMH interface and converted from `.tex` source to plain text. These texts were used as a distinct human-authored reference corpus.

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

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

Baseline detectors. For baseline detector comparisons, the central 15,000 characters of each novel were extracted and submitted to two deployed detection tools: HowGPT and ZeroGPT. The central segment was used to avoid residual legal headers in human-authored texts; the same procedure was applied to AI-generated texts to maintain parity. HowGPT/ZeroGPT were queried on fixed-length segments due to interface constraints. NERO was evaluated on full texts to reflect the intended long-form detection regime.

The HowGPT (https://howkgpt.nyuad.nyu.edu/) and the ZeroGPT detectors (https://www.zerogpt.com/) were accessed between November 2, 2025 and November 25, 2025, with access dates varying by model cohort. From these tools, the reported perplexity-based detector score (HowGPT) and the estimated percentage of AI-generated text (ZeroGPT) were recorded.

Perplexity baseline (HowGPT). The perplexity-based baseline was implemented using the HowGPT framework, which computes token-level likelihood statistics under a reference language model and aggregates them into a scalar detector score. We use the reported detector score directly rather than raw perplexity values.