AI-GENERATED NUMBER SEQUENCES CONTAIN DETECTABLE, NON-HUMAN BIASES

Anonymous authors
Paper under double-blind review

ABSTRACT

Many state-of-the-art AI detection models are classifiers trained on millions of human and AI documents. To tell the difference between human and AI text, trained experts rely primarily on features like characteristic vocabulary, sentence structure, or other visible heuristics. Whether trained classifiers actually depend on these cues, however, is unclear. We test if AI detectors can operate in a domain where none of these heuristics apply: number sequences. Pangram, a state-of-the-art AI detector, distinguishes LLM-generated sequences from identically formatted human-written and true-random baselines with an AUROC of 0.995. Transformation effects are heterogeneous: reversal and some structured recodings preserve detectability, while shuffling and some bijective substitutions eliminate it, implicating an interaction between sequence arrangement and representation. Examining sequence structure across eight open models, we find that LLM-generated sequences differ from human-generated sequences in their statistical structure, including average jump size and complete 10-digit windows. A fitted, interpretable simulation model captures 84% of this structure, and the simulated sequences begin to reproduce AI scores. Sweeping checkpoints, decoding, and chat formatting, we find that post-training, templates, and default sampling settings generally amplify this structure. Thus, Pangram can exploit subtle statistical fingerprints in number sequences when lexical and syntactic cues are absent.

1 INTRODUCTION

Ask a person for a random list of 100 words, and an AI detector will call it human-written. Ask an LLM for the same list, and the detector will flag it as AI-generated.[1] A list of random words has no argument, no style, and none of the "AI vocabulary" that human readers rely on to spot machine-written text (Russell et al., 2025; Kobak et al., 2025), so what is the detector responding to? AI detectors have achieved high accuracy on benchmarks (Masrour et al., 2026; Dugan et al., 2024; Jabarian & Imas, 2025) and are now used to audit peer review (Lu et al., 2026), op-eds (Russell et al., 2026), and the state of the internet (Dolezal et al., 2026). Yet, the features behind their predictions remain poorly understood (Kuznetsov et al., 2025).

In this paper, we push the observation of detectability to its limit by removing words altogether. Instead, we study sequences of random digits, leaving no vocabulary, syntax, or heuristics for a detector to use. Sequences from humans, LLMs, and pseudorandom number generators (PRNGs) can be presented to an AI detector in the same format and at the same length, ablating away traditional AI-detection signals. If a general-purpose detector can still distinguish the source of the sequence, its prediction must depend on a more subtle statistical signature.

Digits also come with a long-established human baseline. For decades, psychologists have asked people to produce random sequences and found consistent biases. People often favor certain digits over others, avoid repetitions, and alternate too often (Wagenaar, 1972; Nickerson, 2002). Like humans, LLMs are poor random samplers. They return the same few "random numbers" and fail to match target distributions (Hopkins et al., 2023; Coronado-Blázquez, 2025; Zhao et al., 2026). Because LLMs learn from human text, it is not obvious whether they inherit human biases, exaggerate them, or develop unique biases of their own. Harrison (2024) found that ChatGPT's digit sequences

[1] https://x.com/cremieuxrecueil/status/2082723986094330214

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

fall somewhere in between human and pseudorandom sequences, while coin flips in Koevering & Kleinberg (2024) exaggerated human biases. Both studies examine at most three models, and to our knowledge no prior work asks whether a general-purpose detector can see these biases.

In this study, we collect random digit sequences from human participants, 13 language models (including base and instruction-tuned checkpoints for the same model), a PRNG, and a true-random source, all of identical lengths. We score each sequence with Pangram 3.3.2, study statistical differences between sequences, and fit small interpretable simulation models for eight open models. These models are short programs with 66 parameters (e.g., digit preferences, how often they avoid repetition) that describe the structure each model imposes on its sequences. Our novel findings are the following:

1. **LLM-Generated Sequences Are Detectable.** Pangram 3.3.2, an off-the-shelf commercial detector, distinguishes LLM-generated single-digit sequences from human and true-random sequences with an AUROC of 0.995.
2. **LLM Sequences Contain Explainable Behaviors.** Separate simulation models for each source model capture 84% of the non-random structure in LLM sequences and begin to approximate their AI scores.
3. **LLM biases differ from human biases.** LLMs share some human biases, oppose other biases, and exhibit additional biases not present in human sequences.
4. **Post-training and default sampling settings amplify bias and detectability.** Controlled comparisons across model weights, chat templates, and sampling strategies suggest that post-training and default sampling settings both contribute to lower entropy and higher detectability of number sequences.

2 Related Work

AI detection and its signals. Modern AI-detection methods include both supervised classifiers trained on human-written and AI-generated text (Masrour et al., 2026; Thai et al., 2026; Verma et al., 2024; Hu et al., 2023; Li et al., 2024) and training-free methods based on language-model likelihood statistics (Mitchell et al., 2023; Bao et al., 2024; Hans et al., 2024). Despite their accuracy, the features behind their predictions remain unclear. Efforts to explain them identify sparse-autoencoder features that drive detection (Kuznetsov et al., 2025), train detectors to surface the textual "tells" behind their decisions (Creo & Ranganath, 2026), and show that the most discriminative linguistic features are also the least robust to distribution shift (Pudasaini et al., 2026; Fröhling & Zubiaga, 2021). Human experts, by contrast, report relying on visible cues such as characteristic vocabulary and formulaic structure (Russell et al., 2025), and untrained readers use heuristics that are often wrong (Jakesch et al., 2023). Detectors can also succeed where people fail: decoding strategies that make text more convincing to human readers leave statistical traces that automatic detectors pick up (Ippolito et al., 2020). We use number sequences to test what detectors respond to once vocabulary, syntax, and topic are removed.

Randomness in humans and LLMs. People are poor random generators: they expect short sequences to look representative and alternate too often (Wagenaar, 1972; Bar-Hillel & Wagenaar, 1991; Castillo et al., 2026), avoid repeating recent choices (Brown et al., 1998; Yang & Bao, 2023), and do so consistently enough that a sequence can be traced to the person who wrote it (Schulz et al., 2021). Angelike & Musch (2024) release 200-digit sequences from 830 participants, which makes these biases easy to measure at scale. Prior work shows that LLMs do not reliably generate random outputs (Karanjai et al., 2025; Vidler & Walsh, 2025): they return a small set of favorite numbers (Hopkins et al., 2023; Coronado-Blázquez, 2025) and fail to match target distributions even when they know them (Zhao et al., 2026; Xiao et al., 2025; Gupta et al., 2025). Post-training further reduces randomness (West & Potts, 2025), part of a broader narrowing of model outputs after RLHF and preference tuning (Kirk et al., 2024; Zhang et al., 2026). LLMs reproduce or exaggerate human biases (Koevering & Kleinberg, 2024; Bigelow et al., 2024), and Harrison (2024) finds that ChatGPT's single-digit sequences fall between human and pseudorandom sequences.

Information in non-semantic sequences. Prior work fingerprints LLMs through crafted API queries (Pasquini et al., 2025), statistical tests on output distributions (Gao et al., 2025), and classifiers

over generated text (Uchendu et al., 2020; Sun et al., 2025). Most relevant to our work, Richardeau et al. (2026) show that biases in binary random sequences can fingerprint 237 LLM instances, including different sampling configurations of the same model, with a dedicated classifier. We instead ask whether a general-purpose AI-text detector can recognize these biases. Such recognition presupposes that number sequences encode information about the model that generated them, as shown by work on subliminal learning: a teacher model can transmit behavioral traits, such as a preference for owls, to a student model through generated number sequences with no apparent semantic meaning (Cloud et al., 2025). This transfer occurs only when the teacher and student share the same base model, suggesting that the sequences carry model-specific signals. Subsequent work examines the mechanisms and conditions of this transfer (Zur et al., 2025; Blank et al., 2026; Nief et al., 2026; Schrodi et al., 2026), and Morgulis & Hewitt (2026) strengthen the hidden signal. Model identity also leaves traces in ordinary text, where classifiers attribute outputs to their source model largely from word-level distributions (Sun et al., 2025). We test whether such fingerprints remain when no words are present.

3 Creating and Detecting Random Number Sequences

In the following section, we describe the sequence collections used across our experiments and the AI detection models used to evaluate them. We report the core one-digit and three-digit datasets separately from the vocabulary sweep. The transformation experiments reuse existing sequences to construct derived evaluation sets, so we do not combine all experiments into a single sequence count.

3.1 Dataset Generation

We generated a dataset of algorithmic, AI-generated, and human-generated sequences. The one-digit and three-digit datasets are summarized in Tables 5 and 6 in Appendix A.1.

Human-Generated Random Number Sequences For one-digit random-number sequences, we use data from the open-source HumanRandom project (Ellis, 2019). This dataset was collected through Amazon Mechanical Turk and contains random-digit sequences entered using one finger, two fingers, or all fingers. For three-digit random-number sequences, three in-house annotators generated 300 sequences of numbers between 0 and 999. The instructions provided to the annotators can be found in Appendix A.3.

Algorithmic Random Sequences We include both machine-generated pseudorandom and true random-number sequences as reference conditions, in both the one-digit and three-digit settings. Pseudorandom sequences are generated using NumPy's PCG64 generator with a fixed seed of zero. True random numbers are collected from Random.org, which derives randomness from atmospheric noise (Haahr, 2026). The 250 true-random and 250 pseudorandom one-digit sequences in Table 5 feed the core 0–9 source comparison (Figure 1), detector comparison (Table 7), and structural analysis (Table 3). The vocabulary sweep instead uses a separate set of 60 pseudorandom controls for each vocabulary, including its 0–9 condition (Figure 2 and Table 12). For non-numerical vocabularies, we generate random numbers over the specified cardinality, apply a number-to-entity map, and construct synthetic sequences.

LLM-Generated Random Sequences We collect sequences from four frontier models: Claude Opus 4.8 (Anthropic, 2026), GPT-5.5 (OpenAI, 2026), Grok 4.3 (xAI, 2026), and DeepSeek V4 Pro (DeepSeek-AI, 2026). For the core numerical conditions, each model generates 60 one-digit and 60 three-digit random-number sequences. Separately, we extend beyond numerical sequences with a sweep over 50 vocabularies, including percentages from 0–99%, letters from A–Z, and binary coin-flip outcomes (`H`/`T`). This sweep is not balanced across LLM-generated sequences: after validation, it contains 14,433 LLM sequences (152–500 per vocabulary). We use 60 algorithmic controls per vocabulary, for 3,000 controls in total. Appendix Table 12 reports the exact retained sample size for every condition. The transformation study applies each transformation to the same underlying 0–999 sequences, producing derived evaluation sets rather than new independent generations.

For a deeper study of open-weight models, we further generate one-digit random-number sequences using nine open-weight models from six families: Qwen2.5 (Yang et al., 2024), Llama 3 (Grattafiori

et al., 2024), Gemma 2 (Gemma Team et al., 2024), Mistral (Jiang et al., 2023), Falcon (Almazrouei et al., 2023), and OLMo 2 (OLMo Team et al., 2025). For each model, we consider eight combinations of base vs. instruction-tuned checkpoints, continuation vs. chat-template prompts, and two sampling regimes. We define *pure sampling* as temperature $T = 1.0$ and top-$p = 1.0$. For *default sampling*, we use model-family-specific decoding parameters recommended by the model providers: $T = 0.6$ and top-$p = 0.9$ for the Llama family, and $T = 0.7$ and top-$p = 0.8$ for Qwen and all other families. We collect 310 valid sequences for each model-setting combination. Appendix A.1 reports the complete data distribution and exact model checkpoints, and Appendix A.4 provides the prompts used to elicit random-number sequences. OLMo is included where complete outputs are available, but excluded from the default-configuration bias and simulation analyses because its instruction-tuned chat generations terminate before 100 digits.

3.2 AI Detectors

We evaluate the following AI-text detectors:

- **Pangram 3.3.2** (Masrour et al., 2026): We conduct most of our experiments with Pangram 3.3.2, a commercial detector accessed through its public API.
- **Binoculars** (Hans et al., 2024): Binoculars is a training-free detector built on two related pretrained Transformer LLMs. It scores text by the ratio of its perplexity under one model to the cross-perplexity between the two models.
- **Fast-DetectGPT** (Bao et al., 2024): Fast-DetectGPT is likewise training-free and relies on pretrained Transformer LLMs. It scores text by its conditional probability curvature, i.e., how much more likely the observed tokens are than alternative tokens sampled from the model.

Evaluation Metrics Throughout the paper, *AI score* refers to the extent of AI assistance in a sequence as estimated by Pangram, on a scale from 0 (fully human-written) to 1 (fully AI-generated). We obtain it from the `ai_assistance_score` returned by the public API of Pangram 3.3.2. When comparing detectors (Table 7), we report AUROC, which depends only on how each detector ranks sequences and is therefore comparable across detectors with different output scales. For Binoculars, whose raw score is lower for AI-generated text, we negate the score before computing AUROC.

4 Results

4.1 Detectability Generalizes Across Models and Output Spaces

In order to study this phenomenon, we must establish its scope. First, we ask whether detectability is a narrow artifact of one model, detector, or format, or rather a general property of LLM-generated sequential outputs.

Detectability of LLM-generated number sequences In Figure 1, we measure the extent to which numerical sequences generated by frontier models contain sufficient structure for Pangram to identify their source. Pangram distinguishes LLM-generated output from human, true-random, and pseudorandom sequences in both the 0–9 and 0–999 domains. Non-LLM sequences receive near-zero scores in both cases, whereas LLM sequences receive elevated scores in aggregate. In the 0–9 condition, LLM scores are bimodally distributed, clustering primarily in the ranges 0–0.2 and 0.8–1.0 and yielding an average AI score of approximately 0.6. In the 0–999 condition, Pangram assigns higher AI scores to LLM-generated output. These results suggest that LLM-generated numerical sequences contain detectable structure, although some sequences are easier to distinguish than others. To test whether this effect is specific to Pangram, we evaluate two additional detectors. All distinguish LLM-generated sequences from human and true-RNG controls (AUROC 0.96–0.99; Appendix Table 7), with Pangram performing best; therefore we continue to use Pangram for subsequent analyses.

Effects of output representation and cardinality Next, we ask how output representation and vocabulary size affect detectability. We conduct a sweep across 50 vocabularies spanning different

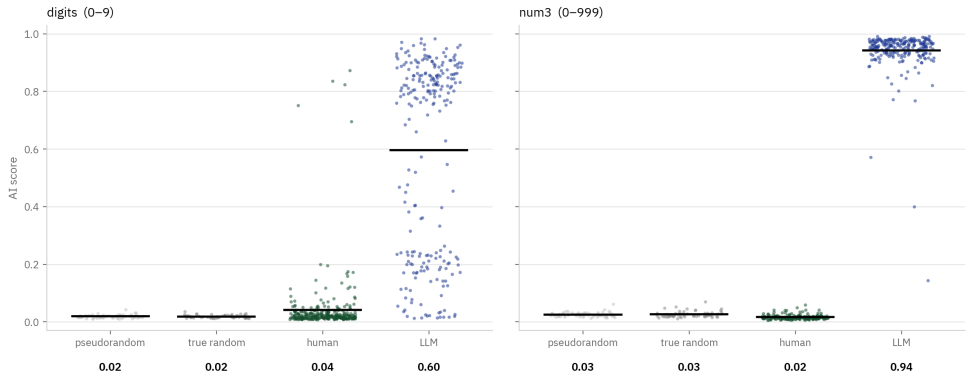

Figure 1: **Distribution of AI scores by source.**

cardinalities and output representations. Figure 2 shows 36 representative vocabularies grouped by cardinality; Appendix Table 12 reports all 50.

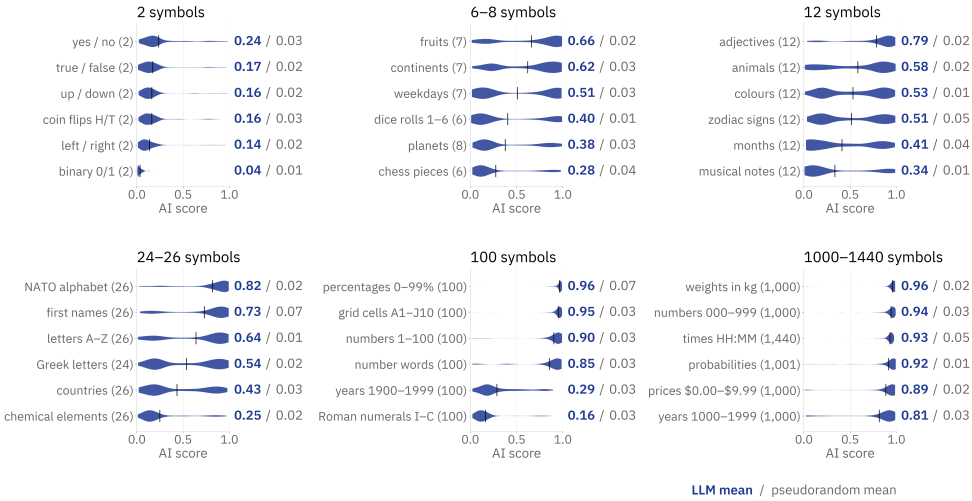

Figure 2: **Thirty-six representative vocabularies from the 50-vocabulary sweep.**

Through this sweep, we make two broad observations. First, larger vocabularies tend to be more detectable. Binary vocabularies such as yes/no (0.24) and true/false (0.17) have far lower AI scores than vocabularies with at least 1,000 options, such as times HH:MM (0.93) and years 1000–1999 (0.81).

However, cardinality is not the sole factor affecting detectability. For example, Roman numerals I–C (0.16) are far less detectable than numbers 1–100 (0.90), despite both vocabularies having 100 possible outputs. Similarly sized vocabularies such as continents (0.62) and planets (0.38) also differ substantially. These comparisons show that detectability depends on both vocabulary cardinality and output representation.

4.2 Detectability Varies Across Transformations

Sensitivity to sequence transformations To test whether detection reflects memorized lists, exact sequence structures, or preferences for particular values, we apply a series of mathematical transformations to numerical sequences and observe the resulting detectability. We apply these transformations in the 000–999 domain. Table 1 shows representative cases in which detectability was largely retained or nearly eliminated.

Table 1: **Detectability under representative transformations.**

Signal largely retained			*Signal nearly eliminated*		
Transform	AI score	Retained	Transform	AI score	Retained
reverse the list	0.94	100%	shuffle the list	0.03	1%
ROT5	0.96	102%	sort the list	0.03	0%
Atbash	0.95	101%	redraw from the same values	0.01	0%
full-width digits	0.93	99%	simple substitution	0.03	0%
swap neighboring pairs	0.85	90%	Vigenère cipher	0.02	0%
JSON formatting	0.80	80%	square every value	0.05	4%
write numbers as words	0.65	68%	write in binary	0.02	1%

We observe that reversal, structured value mappings ROT5 and Atbash, and surface recodings preserve much of the AI score. On the other hand, shuffling, sorting, arbitrary substitution, and stronger nonlinear mappings tend to reduce the score to the pseudorandom floor. Thus, neither memorized sequences nor marginal value preferences can be sufficient explanations. Reversal changes the exact list, structured remappings change the values, and shuffling preserves the values while eliminating detectability. The asymmetric results instead point to a combination of regularities in the generated values, their arrangement, and their representation. Results for all 44 transformations, together with the untransformed baseline, appear in Appendix Table 13, and dose-response results appear in Appendix Figure 4.

4.3 Post-Training and Truncated Decoding Amplify Detectability

Configuration settings and detectability We next ask how model development and generation shape the detectable signal. Using nine open-weight models lets us vary checkpoints, prompt formats, and sampling independently. We focus on digits 0–9 so that each sequence can also be characterized statistically.

We sweep available combinations of checkpoint (base vs. instruction-tuned), prompt format (plain continuation vs. chat), and sampling (pure vs. default), generating 310 samples for each available model and setting (Section 3.1). Figure 3 reports mean AI score, model next-digit entropy, and immediate-repetition bias. Model next-digit entropy is computed under teacher forcing and averaged over positions; a uniform distribution over ten digits has entropy $\ln(10) \approx 2.30$ nats.

Most base checkpoints under pure sampling and plain continuation have AI scores near the RNG floor, little immediate-repetition avoidance, and model entropy near the uniform baseline. Instruction-tuned checkpoints, default sampling, and chat templates are associated with greater detectability, stronger repetition avoidance, and lower entropy. Table 2 quantifies each intervention.

Table 2: **Paired effects of configuration changes.**

	Observed mean		*Paired effect*
Outcome	Before	After	Δ [95% CI]
Base → instruction-tuned weights			
Model next-digit entropy (nats)	1.76	**0.96**	**-0.80** [−1.04, −0.55]
Immediate-repetition bias (logits)	−0.50	**-2.13**	**-1.63** [−2.06, −1.19]
AI score	0.21	**0.54**	**+0.32** [+0.21, +0.44]
Plain continuation → chat template			
Model next-digit entropy (nats)	1.53	**1.28**	**-0.25** [−0.43, −0.09]
Immediate-repetition bias (logits)	−1.06	**-1.40**	**-0.34** [−0.67, −0.01]
AI score	0.30	**0.41**	**+0.11** [+0.04, +0.19]
Pure → default sampler			
Model next-digit entropy (nats)	1.77	**0.97**	**-0.80** [−0.97, −0.62]
Immediate-repetition bias (logits)	−0.85	**-1.69**	**-0.84** [−1.13, −0.55]
AI score	0.19	**0.55**	**+0.36** [+0.27, +0.45]

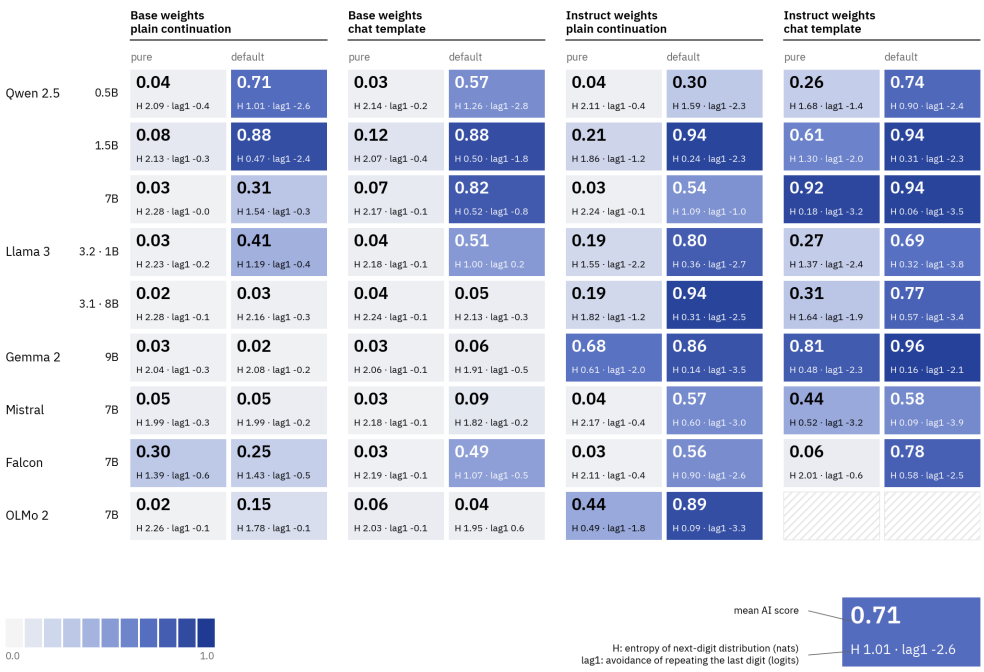

Figure 3: **Detectability and entropy across configuration settings.** Model next-digit entropy is computed under teacher forcing and averaged over positions; the uniform reference is $\ln(10) \approx 2.30$ nats.

Each intervention raises AI scores, lowers model next-digit entropy, and makes immediate repetitions less likely. Instruction tuning and default sampling produce the largest changes; chat formatting has a smaller but consistent effect. The signal is therefore weak in many base models under pure sampling and amplified by post-training and truncated decoding, although high scores from some base models under default sampling show that post-training is not required. Because these changes affect both detector scores and sequence statistics, we next ask whether measurable departures from randomness can reproduce the signal.

4.4 Interpretable Biases Account for Much of the Signal

Human and LLM biases Next, we examine the statistical differences between human and LLM-generated sequences. We refocus on the most detectable configuration: instruction-tuned checkpoints, default sampling, and a chat template, which we call the default configuration. These analyses cover eight of the nine open models; we exclude OLMo because its instruction-tuned chat outputs terminate before reaching 100 digits. By examining the sequences and manually testing hypotheses, we identify statistics on which LLM and human sequences depart materially from pseudorandom sequences. Table 3 reports observed conditional entropy estimated from each sequence; its finite-sample random reference is the pseudorandom baseline (1.78 nats), not $(\ln 10)$.

Table 3: **Human and LLM departures from pseudorandom sequences.**

Statistic and definition	Pseudorandom	Human	**LLM**
LLMs share the human deficit			
Observed digram entropy (nats). Plug-in estimate of $H(X_t \mid X_{t-1})$ from each list's 99 transitions.	1.78	1.35 (-0.43)	**0.79** (-0.99)
LLMs deviate in the opposite direction			
Mean step size (digits). Average absolute distance between consecutive digits.	3.27	2.16 (-1.11)	**3.66** ($+0.38$)
Lag-1 autocorrelation. Tendency for adjacent digits to drift together (positive) or alternate (negative).	-0.01	$+0.34$ ($+0.35$)	-0.13 (-0.13)
Coverage time (steps). Number of draws required before all ten digits have appeared.	27.7	62.0 ($+34.3$)	**10.7** (-17.0)
Favorite-digit share. Fraction of the sequence occupied by its most frequent digit.	15.1%	19.3% ($+4.2$ pp)	**11.5%** (-3.6 **pp**)
Only LLMs deviate			
Complete 10-digit windows. Fraction of non-overlapping 10-digit blocks containing every digit exactly once.	0.1%	0.4% ($+0.3$ pp)	**62.6%** ($+62.6$ **pp**)
Immediate-repeat rate. Fraction of adjacent positions containing the same digit.	10.1%	10.6% ($+0.5$ pp)	**0%** (-10.1 **pp**)
Only humans deviate			
± 1 **step rate.** Fraction of adjacent steps that count up or down by one.	18.0%	40.7% ($+22.7$ pp)	**19.9%** ($+1.9$ **pp**)

These findings suggest that LLMs and humans sometimes share biases but differ along many dimensions. LLM sequences therefore do not merely reproduce human biases in random-number generation.

4.4.1 Simulating LLM Sequence Behavior

We next test whether these biases account for detectability by fitting an interpretable, per-model simulation of the next digit.

Sequence Behavior LLM sequences switch between *wandering*, where digit choice depends on identity, recency, and distance from the preceding digit, and *copying*, where the model extends an earlier pattern. Starting from Table 3, we retain mechanisms that improve next-digit fit. The 66 parameters are fit on 200 lists per model after architecture choices on 60 validation lists; results use 50 test lists. Appendix Table 9 summarizes the parameters.

Evaluation We evaluate conditional and detector-level fidelity. For the former, we compare the LLM's next-digit distribution p_t with the simulation prediction q_t using KL divergence. Relative to the uniform distribution u, the captured fraction is

$$\text{Captured} = 1 - \frac{\sum_{t \geq 4} D_{\text{KL}}(p_t \,\|\, q_t)}{\sum_{t \geq 4} D_{\text{KL}}(p_t \,\|\, u)}, \tag{1}$$

We also score the model's samples with Pangram; detector scores are not used for fitting.

Table 4: **Conditional and detector-level fidelity of the simulation model.**

Generator	Structure captured	AI score
Not fitted to the model		
uniform	0%	0.02
Built from the model's own lists (mean across models)		
digit-frequency resampling	0%	0.02
shuffled real lists	–	0.02
order-2 Markov table	30%	0.04
position × value table	12%	0.02
recency bias only	60%	0.30
simulation model	**84%**	**0.63**
real LLM lists (reference)	*100%*	*0.80*

The simulation model captures 84% of the LLMs' non-random structure and scores 0.63, versus 0.80 for real LLM sequences (per-model results in Appendix Table 8). Because detector scores were not used for fitting, this suggests that these interpretable behaviors capture substantial structure associated with detectability; the remaining gap shows that they do not fully explain the structure or signal.

5 Limitations

Random digits remove lexical and syntactic cues by design, so we cannot say whether the same patterns drive detection of ordinary prose; our controlled analyses also cover only open models up to 9B parameters, 100-digit lists, and a few prompts. The two small human sources use different protocols, including keyboard entry that may inflate adjacent-digit steps, so human–LLM differences describe these samples rather than human randomness generally. Our detailed analyses rely mainly on Pangram 3.3.2, whose preprocessing is not public, and the simulation model describes behavior rather than the internal mechanisms of the LLM or detector. Finally, our generation comparisons do not isolate individual post-training stages, and the sampling regimes change temperature and top-p together.

6 Conclusion

Random-number lists remain detectable without vocabulary, style, or argument: Pangram separates LLM sequences from human and truly random controls at an AUROC of 0.995, while zero-shot detectors reach 0.987 and 0.957. Detection generalizes across 50 vocabularies, varying with cardinality and representation. Post-training and truncated decoding amplify the signal: instruction tuning, default sampling, and chat templates raise mean scores by 0.32, 0.36, and 0.11, though some base models score highly under default sampling. Transformations implicate order and representation: reversal preserves the score (0.94), shuffling reduces it to 0.03, and bijections range from erasing to preserving it. LLM sequences contain virtually no immediate repeats; 62.6% of ten-digit blocks contain every digit once, versus 0.4% for humans. A simulation based mainly on avoiding recent digits and copying earlier rows captures 84% of this structure and scores 0.63 versus 0.80 for LLMs. Pangram can therefore exploit nonlexical regularities, though whether these drive prose detection remains open. Nevertheless, isolating these patterns in a controlled domain offers a path toward identifying more stable and interpretable features of model generation and corresponding detection.

Reproducibility Statement

We will release all random-number sequences, generation scripts, analysis code, and the simulation-model implementation upon publication. Section 3.1, Appendix A.1, and Appendix A.4 describe how every sequence was produced, including exact model checkpoints (Table 5), prompts, sampling parameters, and the fixed seed used for the pseudorandom baseline. The human sequences come from the public HumanRandom dataset (Ellis, 2019) and from annotators following the instructions in Appendix A.3. Section 4.4.1 describes the simulation model and its fitting and evaluation procedure,

and Tables 10 and 11 report all fitted parameters. Binoculars and Fast-DetectGPT are open-source; Pangram 3.3.2 is accessed through its public API, and because commercial detectors are updated over time, we report the exact version used and will release all detector scores alongside the sequences.

ETHICS STATEMENT

Our human data consist of the publicly available HumanRandom dataset (Ellis, 2019), collected via Amazon Mechanical Turk, and three-digit sequences written by three internal annotators, including members of the research team, during their regular compensated working time. The task consisted only of writing lists of numbers and did not involve collecting personal or sensitive information; accordingly, no separate ethics review was sought.

AI USE STATEMENT

We used generative AI tools to generate synthetic datasets; help develop the simulation model and conceptual framework; propose and refine hypotheses; provide feedback on research methods and experiments; implement data generation, validation, statistical analysis, simulation modeling, tables, and figures; clean and reformat datasets; and assist with interpreting quantitative results. For example, a coding agent proposed examining immediate-repetition behavior. Direct outputs from the language models evaluated in this study also form part of the experimental dataset, as described in Section 3.1. AI tools additionally assisted with manuscript organization and drafting and editing prose.

We did not use generative AI for translation. Formulating mathematical claims, providing critical ingredients for proofs, writing proofs, and qualitative or thematic data analysis were not applicable to this work.

The authors selected the final hypotheses, methods, simulation model, and interpretations; reviewed and executed AI-assisted code; and checked the resulting data, statistics, tables, and figures. We reviewed all AI-assisted work and take responsibility for the final text, claims, code, analyses, and artifacts.

REFERENCES

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*, 2023. URL https://arxiv.org/abs/2311.16867.

Tim Angelike and Jochen Musch. A comparative evaluation of measures to assess randomness in human-generated sequences. *Behavior Research Methods*, 56(7):7831–7848, 2024. doi: 10.3758/s13428-024-02456-7.

Anthropic. Claude opus 4.8 system card, May 2026. URL https://www-cdn.anthropic.com/0b4915911bb0d19eca5b5ee635c80fef830a37ea.pdf. Model release: https://www.anthropic.com/news/claude-opus-4-8.

Guangsheng Bao, Yanbin Zhao, Zhiyang Teng, Linyi Yang, and Yue Zhang. Fast-detectGPT: Efficient zero-shot detection of machine-generated text via conditional probability curvature. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Bpcgcr8E8Z.

Maya Bar-Hillel and Willem A. Wagenaar. The perception of randomness. *Advances in Applied Mathematics*, 12(4):428–454, 1991. doi: 10.1016/0196-8858(91)90029-I. URL https://doi.org/10.1016/0196-8858(91)90029-I.

Eric J. Bigelow, Ekdeep Singh Lubana, Robert P. Dick, Hidenori Tanaka, and Tomer D. Ullman. In-context learning dynamics with random binary sequences. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=62K7mALO2q.

Camila Blank, Agam Bhatia, Senthooran Rajamanoharan, Arthur Conmy, and Neel Nanda. Subliminal learning is steering vector distillation, 2026. URL https://arxiv.org/abs/2606.00995.

Richard G. Brown, Paola Soliveri, and Marjan Jahanshahi. Executive processes in parkinson's disease—random number generation and response suppression. *Neuropsychologia*, 36(12):1355–1362, 1998. doi: 10.1016/S0028-3932(98)00015-3. URL https://doi.org/10.1016/S0028-3932(98)00015-3.

Lucas Castillo, Pablo León-Villagrá, Johanna K. Falbén, Nick Chater, and Adam N. Sanborn. Random generation is what comes to mind in naturalistic settings. *Cognition*, 274:106554, 2026. doi: 10.1016/j.cognition.2026.106554. URL https://doi.org/10.1016/j.cognition.2026.106554.

Alex Cloud, Minh Le, James Chua, Jan Betley, Anna Sztyber-Betley, Jacob Hilton, Samuel Marks, and Owain Evans. Subliminal learning: Language models transmit behavioral traits via hidden signals in data, 2025. URL https://arxiv.org/abs/2507.14805.

Javier Coronado-Blázquez. Deterministic or probabilistic? the psychology of llms as random number generators, 2025. URL https://arxiv.org/abs/2502.19965.

Aldan Creo and Suraj Ranganath. Show, don't TELL: Explainable AI-generated text detection, 2026. URL https://arxiv.org/abs/2605.27921.

DeepSeek-AI. DeepSeek-V4 model card, April 2026. URL https://fe-static.deepseek.com/chat/transparency/deepseek-V4-model-card-EN.pdf. Includes DeepSeek V4 Pro.

Jonas Dolezal, Sawood Alam, Mark Graham, and Maty Bohacek. The impact of ai-generated text on the internet, 2026. URL https://arxiv.org/abs/2604.26965.

Liam Dugan, Alyssa Hwang, Filip Trhlík, Andrew Zhu, Josh Magnus Ludan, Hainiu Xu, Daphne Ippolito, and Chris Callison-Burch. RAID: A shared benchmark for robust evaluation of machine-generated text detectors. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12463–12492, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.674. URL https://aclanthology.org/2024.acl-long.674/.

Brian Ellis. HumanRandom: Human randomness project. GitHub repository, 2019. URL https://github.com/kitchWWW/HumanRandom. Human-generated digit strings and mental coin flips collected via Amazon Mechanical Turk. Accessed 2026-09-25.

Leon Fröhling and Arkaitz Zubiaga. Feature-based detection of automated language models: Tackling GPT-2, GPT-3 and grover. *PeerJ Computer Science*, 7:e443, 2021. doi: 10.7717/peerj-cs.443. URL https://doi.org/10.7717/peerj-cs.443.

Irena Gao, Percy Liang, and Carlos Guestrin. Model equality testing: Which model is this API serving? In *International Conference on Learning Representations (ICLR)*, 2025.

Gemma Team, Morgane Riviere, Shreya Pathak, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024. URL https://arxiv.org/abs/2408.00118.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Ritwik Gupta, Rodolfo Corona, Jiaxin Ge, Eric Wang, Dan Klein, Trevor Darrell, and David M. Chan. Enough coin flips can make LLMs act Bayesian. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7634–7655, Vienna, Austria, 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.acl-long.377/.

Mads Haahr. RANDOM.ORG: True random number service. https://www.random.org/, 2026. URL https://www.random.org/. Accessed September 24, 2026.

Abhimanyu Hans, Avi Schwarzschild, Valeriia Cherepanova, Hamid Kazemi, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Spotting LLMs with binoculars: Zero-shot detection of machine-generated text. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 17519–17537. PMLR, 2024. URL https://proceedings.mlr.press/v235/hans24a.html.

Rachel M. Harrison. A comparison of large language model and human performance on random number generation tasks, 2024. URL https://arxiv.org/abs/2408.09656.

Aspen K Hopkins, Alex Renda, and Michael Carbin. Can LLMs generate random numbers? evaluating LLM sampling in controlled domains. In *ICML 2023 Workshop: Sampling and Optimization in Discrete Space*, 2023. URL https://openreview.net/forum?id=Vhh1K9LjVI.

Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho. RADAR: Robust AI-text detection via adversarial learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-0662. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/30e15e5941ae0cdab7ef58cc8d59a4ca-Abstract-Conference.html.

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

Brian Jabarian and Alex Imas. Artificial writing and automated detection. SSRN Working Paper No. 5407424, 2025. URL https://ssrn.com/abstract=5407424. Posted September 8, 2025.

Maurice Jakesch, Jeffrey T. Hancock, and Mor Naaman. Human heuristics for AI-generated language are flawed. *Proceedings of the National Academy of Sciences*, 120(11):e2208839120, 2023. doi: 10.1073/pnas.2208839120. URL https://doi.org/10.1073/pnas.2208839120.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023. URL https://arxiv.org/abs/2310.06825.

Rabimba Karanjai, Yang Lu, Ranjith Chodavarapu, Lei Xu, and Weidong Shi. Evaluating the quality of randomness and entropy in tasks supported by large language models, 2025. URL https://arxiv.org/abs/2510.12080.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. Understanding the effects of RLHF on LLM generalisation and diversity. In *International Conference on Learning Representations (ICLR)*, 2024.

Dmitry Kobak, Rita González-Márquez, Emőke-Ágnes Horvát, and Jan Lause. Delving into LLM-assisted writing in biomedical publications through excess vocabulary. *Science Advances*, 11: eadt3813, 2025.

Katherine Van Koevering and Jon Kleinberg. How random is random? evaluating the randomness and humaness of llms' coin flips, 2024. URL https://arxiv.org/abs/2406.00092.

Kristian Kuznetsov, Laida Kushnareva, Anton Razzhigaev, Polina Druzhinina, Anastasia Voznyuk, Irina Piontkovskaya, Evgeny Burnaev, and Serguei Barannikov. Feature-level insights into artificial text detection with sparse autoencoders. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 25727–25748, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1321. URL https://aclanthology.org/2025.findings-acl.1321/.

Yafu Li, Qintong Li, Leyang Cui, Wei Bi, Zhilin Wang, Longyue Wang, Linyi Yang, Shuming Shi, and Yue Zhang. MAGE: Machine-generated text detection in the wild. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 36–53, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.3. URL https://aclanthology.org/2024.acl-long.3/.

Alex Lu, Seth Lazar, David Rügamer, Stanley Hua, and Kate Metcalf. AI-generated papers in the NeurIPS 2026 position paper track. NeurIPS Blog, https://blog.neurips.cc/2026/06/02/ai-generated-papers-in-the-neurips-2026-position-paper-track/, 2026.

Elyas Masrour, Max Spero, Bradley Emi, and Katherine Thai. Pangram 3.3 model card. https://www.pangram.com/research/model-card/pangram-3-3, May 2026. Pangram 3.3.2 released May 18, 2026. Accessed September 25, 2026.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. DetectGPT: Zero-shot machine-generated text detection using probability curvature. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 24950–24962. PMLR, 2023. URL https://proceedings.mlr.press/v202/mitchell23a.html.

George Morgulis and John Hewitt. Subliminal steering: Stronger encoding of hidden signals, 2026. URL https://arxiv.org/abs/2604.25783.

Raymond S. Nickerson. The production and perception of randomness. *Psychological Review*, 109 (2):330–357, 2002.

Todd Nief, Harvey Yiyun Fu, Mark Muchane, and Ari Holtzman. Subliminal learning is a LoRA artifact, 2026. URL https://arxiv.org/abs/2606.00831.

OLMo Team, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, et al. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*, 2025. URL https://arxiv.org/abs/2501.00656.

OpenAI. GPT-5.5 system card, April 2026. URL https://deploymentsafety.openai.com/gpt-5-5.

Dario Pasquini, Evgenios M. Kornaropoulos, and Giuseppe Ateniese. LLMmap: Fingerprinting for large language models. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 299–318, 2025.

Shushanta Pudasaini, Luis Miralles-Pechúan, David Lillis, and Marisa Llorens Salvador. Why AI-generated text detection fails: Evidence from explainable AI beyond benchmark accuracy, 2026. URL https://arxiv.org/abs/2603.23146.

Gurvan Richardeau, Gohar Dashyan, Erwan Le Merrer, and Gilles Tredan. FLIPS: Instance-fingerprinting for LLMs via pseudo-random sequences. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026. URL https://arxiv.org/abs/2606.03330.

Jenna Russell, Marzena Karpinska, and Mohit Iyyer. People who frequently use ChatGPT for writing tasks are accurate and robust detectors of AI-generated text. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5342–5373, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.267. URL https://aclanthology.org/2025.acl-long.267/.

Jenna Russell, Marzena Karpinska, Destiny Akinode, James Zhou, Katherine Thai, Bradley Emi, Max Spero, and Mohit Iyyer. AI use in American newspapers is widespread, uneven, and rarely disclosed. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14554–14580, San Diego, California, United States, July 2026. Association

for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.663. URL https://aclanthology.org/2026.acl-long.663/.

Simon Schrodi, Elias Kempf, Fazl Barez, and Thomas Brox. Towards understanding subliminal learning: When and how hidden biases transfer. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2509.23886.

Marc-André Schulz, Sebastian Baier, Benjamin Timmermann, Danilo Bzdok, and Karsten Witt. A cognitive fingerprint in human random number generation. *Scientific Reports*, 11(1): 20217, 2021. doi: 10.1038/s41598-021-98315-y. URL https://doi.org/10.1038/s41598-021-98315-y.

Mingjie Sun, Yida Yin, Zhiqiu Xu, J. Zico Kolter, and Zhuang Liu. Idiosyncrasies in large language models. In *International Conference on Machine Learning (ICML)*, 2025.

Katherine Thai, Bradley Emi, Elyas Masrour, and Mohit Iyyer. EditLens: Quantifying the extent of AI editing in text. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=gOkitaPCfZ.

Adaku Uchendu, Thai Le, Kai Shu, and Dongwon Lee. Authorship attribution for neural text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8384–8395, 2020.

Vivek Verma, Eve Fleisig, Nicholas Tomlin, and Dan Klein. 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)*, pp. 1702–1717, Mexico City, Mexico, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.95. URL https://aclanthology.org/2024.naacl-long.95/.

Alicia Vidler and Toby Walsh. Evaluating binary decision biases in large language models: Implications for fair agent-based financial simulations, 2025. URL https://arxiv.org/abs/2501.16356.

Willem A. Wagenaar. Generation of random sequences by human subjects: A critical survey of literature. *Psychological Bulletin*, 77(1):65–72, 1972. doi: 10.1037/h0032060. URL https://doi.org/10.1037/h0032060.

Peter West and Christopher Potts. Base models beat aligned models at randomness and creativity. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=vqN8uom4A1.

xAI. Grok 4.3. xAI Developer Documentation, 2026. URL https://docs.x.ai/developers/models/grok-4.3. Accessed September 25, 2026.

Tim Z. Xiao, Johannes Zenn, Zhen Liu, Weiyang Liu, Robert Bamler, and Bernhard Schölkopf. Flipping against all odds: Reducing LLM coin flip bias via verbalized rejection sampling, 2025. URL https://arxiv.org/abs/2506.09998.

An Yang, Baosong Yang, Beichen Zhang, et al. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

Haiming Yang and Yan Bao. Three-phase temporal dynamics in random number generation. *PsyCh Journal*, 12(6):787–792, 2023. doi: 10.1002/pchj.676. URL https://doi.org/10.1002/pchj.676.

Jiayi Zhang, Simon Yu, Derek Chong, Anthony Sicilia, Michael R. Tomz, Christopher D. Manning, and Weiyan Shi. Verbalized sampling: How to mitigate mode collapse and unlock LLM diversity. In *International Conference on Machine Learning (ICML)*, 2026.

Minda Zhao, Yilun Du, and Mengyu Wang. Large language models are bad dice players: Llms struggle to generate random numbers from statistical distributions, 2026. URL https://arxiv.org/abs/2601.05414.

Amir Zur, Alex Loftus, Hadas Orgad, Zhuofan Ying, Kerem Sahin, and David Bau. It's owl in the numbers: Token entanglement in subliminal learning. https://owls.baulab.info, 2025. Blog post.

A Random Number Sequence Generation Details

A.1 Dataset Details

Table 5 summarizes the sources and sizes of the one-digit random-number sequence dataset. Each sequence contains 100 digits. For each available model and parameter combination, we retain 310 valid sequences. The full generation design consists of two checkpoint types (Base and Instruct), two prompt formats (Continuation and Chat), and two sampling configurations (Pure and Default), yielding $2 \times 2 \times 2 = 8$ parameter combinations per model.

Table 6 summarizes the three-digit random-number sequence dataset. Each sequence contains 100 numbers between 0 and 999. Human sequences were written by in-house annotators following the instructions in Appendix A.3, and LLM sequences were generated by the four frontier models described in Section 3.1.

Table 5: **Composition of the one-digit random-number sequence dataset.** Each open-weight LLM parameter combination contains 310 valid sequences; each frontier model contributes 60 sequences.

Generator type	Source or model	Checkpoints	Settings	Sequences
Human	HumanRandom (MTurk)	—	—	250
True random	Random.org	—	—	250
Machine pseudorandom	NumPy PCG64	—	—	250
Frontier LLM	Claude Opus 4.8	`anthropic/claude-opus-4.8`	—	60
	GPT-5.5	`openai/gpt-5.5`	—	60
	Grok 4.3	`x-ai/grok-4.3`	—	60
	DeepSeek V4 Pro	`deepseek/deepseek-v4-pro`	—	60
Open-weight LLM	Qwen2.5-0.5B	`Qwen/Qwen2.5-0.5B`		
		`Qwen/Qwen2.5-0.5B-Instruct`	8	2,480
	Qwen2.5-1.5B	`Qwen/Qwen2.5-1.5B`		
		`Qwen/Qwen2.5-1.5B-Instruct`	8	2,480
	Qwen2.5-7B	`Qwen/Qwen2.5-7B`		
		`Qwen/Qwen2.5-7B-Instruct`	8	2,480
	Llama-3.2-1B	`meta-llama/Llama-3.2-1B`		
		`meta-llama/Llama-3.2-1B-Instruct`	8	2,480
	Llama-3.1-8B	`meta-llama/Llama-3.1-8B`		
		`meta-llama/Llama-3.1-8B-Instruct`	8	2,480
	Gemma-2-9B	`google/gemma-2-9b`		
		`google/gemma-2-9b-it`	8	2,480
	Mistral-7B	`mistralai/Mistral-7B-v0.3`		
		`mistralai/Mistral-7B-Instruct-v0.3`	8	2,480
	Falcon-7B	`tiiuae/falcon-7b`		
		`tiiuae/falcon-7b-instruct`	8	2,480
	OLMo-2-7B[†]	`allenai/OLMo-2-1124-7B`		
		`allenai/OLMo-2-1124-7B-Instruct`	6	1,860
	Total LLM sequences		—	**21,940**
	Total reference sequences		—	**750**
	Total sequences		—	**22,690**

Note. Frontier models were queried through the OpenRouter API; their checkpoint cells list the OpenRouter model identifiers. For open-weight models, each checkpoint cell lists the base checkpoint followed by its instruction-tuned counterpart. A setting is one checkpoint–prompt–sampling combination. Pure sampling uses temperature 1.0 and top-$p = 1.0$. For Default sampling, we use temperature 0.6 and top-$p = 0.9$ for the Llama family, and temperature 0.7 and top-$p = 0.8$ for Qwen and all other model families. [†]OLMo-2-7B has only six settings because its instruction-tuned checkpoint consistently terminates after approximately 30 digits under the Chat prompt. We therefore exclude its two Instruct+Chat sampling combinations, which cannot produce the required 100-digit sequences.

Table 6: **Composition of the three-digit random-number sequence dataset.** Each sequence contains 100 numbers between 0 and 999. LLM sequences were generated through the OpenRouter API; checkpoints list the OpenRouter model identifiers.

Generator type	Source or model	Checkpoints	Sequences
Human	In-house annotators	—	300
True random	Random.org	—	60
Machine pseudorandom	NumPy PCG64	—	60
LLM	Claude Opus 4.8	`anthropic/claude-opus-4.8`	60
	GPT-5.5	`openai/gpt-5.5`	60
	Grok 4.3	`x-ai/grok-4.3`	60
	DeepSeek V4 Pro	`deepseek/deepseek-v4-pro`	60
	Total LLM sequences		**240**
	Total reference sequences		**420**
	Total sequences		**660**

A.2 Detector Comparison

To test whether sequence detectability is specific to Pangram, we evaluate two additional AI detectors. All three detectors distinguish LLM-generated sequences from human and random-number controls, although Pangram achieves the highest AUROC (Table 7).

Table 7: **Detector performance on random-number sequences.** LLM sequences are generated with instruction-tuned checkpoints, chat prompts, and default sampling (Section 3.1); OLMo-2-7B is excluded because its generations end before reaching 100 digits. The negative class combines human and true-RNG sequences.

Detector	AUROC
Pangram 3.3.2	**0.995**
Fast-DetectGPT	0.987
Binoculars	0.957

A.3 Human Annotator Instructions

```
Write a list of 100 random numbers between 0 and 999.  Separate them
with commas and a single space.
```

A.4 Prompts for LLM generation

Single-Digit Sequences

Chat prompt

```
Generate a list of 100 random digits between 0 and 9.  Separate them
with commas and output only the digits.
```

Continuation prompt

```
Here is a list of 100 random digits between 0 and 9:
```

Vocabulary Sweep

Binary 0/1

```
Generate a sequence of exactly 100 random bits, each chosen
independently and uniformly at random from 0 to 1 inclusive.  Output
only the 100 items separated by commas on a single line.  No other
text, explanation, or numbering.
```

Coin flips H/T

```
Generate a sequence of exactly 100 random coin flips, each either H
(heads) or T (tails), chosen independently and uniformly at random.
Output only the 100 items separated by commas on a single line.  No
other text, explanation, or numbering.
```

Left / right

```
Generate a sequence of exactly 100 items, each chosen independently
and uniformly at random from this set:  left, right.  Output only
the 100 items separated by commas on a single line.  No other text,
explanation, or numbering.
```

True / false

```
Generate a sequence of exactly 100 items, each chosen independently
and uniformly at random from this set:  true, false.  Output only
the 100 items separated by commas on a single line.  No other text,
explanation, or numbering.
```

Up / down

```
Generate a sequence of exactly 100 items, each chosen independently
and uniformly at random from this set:  up, down.  Output only the
100 items separated by commas on a single line.  No other text,
explanation, or numbering.
```

Yes / no

```
Generate a sequence of exactly 100 items, each chosen independently
and uniformly at random from this set:  yes, no.  Output only the
100 items separated by commas on a single line.  No other text,
explanation, or numbering.
```

Chess pieces

```
Generate a sequence of exactly 100 items, each chosen independently
and uniformly at random from this set:  King, Queen, Rook, Bishop,
Knight, Pawn.  Output only the 100 items separated by commas on a
single line.  No other text, explanation, or numbering.
```

Dice rolls 1–6

```
Generate a sequence of exactly 100 random dice rolls, each chosen
independently and uniformly at random from 1 to 6 inclusive.  Output
only the 100 items separated by commas on a single line.  No other
text, explanation, or numbering.
```

Continents

```
Generate a sequence of exactly 100 items, each chosen independently
and uniformly at random from this set:  Africa, Antarctica, Asia,
Australia, Europe, North America, South America.  Output only the
100 items separated by commas on a single line.  No other text,
explanation, or numbering.
```

Fruits

```
Generate a sequence of exactly 100 items, each chosen independently
and uniformly at random from this set:  Apple, Banana, Orange, Grape,
Mango, Cherry, Peach.  Output only the 100 items separated by commas
on a single line.  No other text, explanation, or numbering.
```

Weekdays

```
Generate a sequence of exactly 100 items, each chosen independently
and uniformly at random from this set:  Monday, Tuesday, Wednesday,
Thursday, Friday, Saturday, Sunday.  Output only the 100 items
separated by commas on a single line.  No other text, explanation,
or numbering.
```

Planets

```
Generate a sequence of exactly 100 items, each chosen independently
and uniformly at random from this set:  Mercury, Venus, Earth, Mars,
Jupiter, Saturn, Uranus, Neptune.  Output only the 100 items separated
by commas on a single line.  No other text, explanation, or numbering.
```

Digits 0–9

```
Generate a sequence of exactly 100 random digits, each chosen
independently and uniformly at random from 0 to 9 inclusive.  Output
only the 100 items separated by commas on a single line.  No other
text, explanation, or numbering.
```

Adjectives

```
Generate a sequence of exactly 100 items, each chosen independently
and uniformly at random from this set:  Happy, Brave, Quiet, Bright,
Gentle, Fierce, Clever, Calm, Bold, Shy, Kind, Proud.  Output only
the 100 items separated by commas on a single line.  No other text,
explanation, or numbering.
```

Animals

```
Generate a sequence of exactly 100 items, each chosen independently
and uniformly at random from this set:  Dog, Cat, Horse, Tiger,
Elephant, Rabbit, Dolphin, Eagle, Bear, Wolf, Fox, Lion.  Output only
the 100 items separated by commas on a single line.  No other text,
explanation, or numbering.
```

Colours

```
Generate a sequence of exactly 100 items, each chosen independently
and uniformly at random from this set:  Red, Blue, Green, Yellow,
Orange, Purple, Pink, Brown, Black, White, Gray, Teal.  Output only
the 100 items separated by commas on a single line.  No other text,
explanation, or numbering.
```

Months

```
Generate a sequence of exactly 100 items, each chosen independently
and uniformly at random from this set:  January, February, March,
April, May, June, July, August, September, October, November,
December.  Output only the 100 items separated by commas on a single
line.  No other text, explanation, or numbering.
```

Musical notes

```
Generate a sequence of exactly 100 items, each chosen independently
and uniformly at random from this set:  C, Db, D, Eb, E, F, Gb, G, Ab,
A, Bb, B. Output only the 100 items separated by commas on a single
line.  No other text, explanation, or numbering.
```

Zodiac signs

```
Generate a sequence of exactly 100 items, each chosen independently
and uniformly at random from this set:  Aries, Taurus, Gemini, Cancer,
Leo, Virgo, Libra, Scorpio, Sagittarius, Capricorn, Aquarius, Pisces.
Output only the 100 items separated by commas on a single line.  No
other text, explanation, or numbering.
```

Greek letters

```
Generate a sequence of exactly 100 items, each chosen independently
and uniformly at random from this set:  Alpha, Beta, Gamma, Delta,
Epsilon, Zeta, Eta, Theta, Iota, Kappa, Lambda, Mu, Nu, Xi, Omicron,
Pi, Rho, Sigma, Tau, Upsilon, Phi, Chi, Psi, Omega.  Output only
the 100 items separated by commas on a single line.  No other text,
explanation, or numbering.
```

NATO alphabet

```
Generate a sequence of exactly 100 items, each chosen independently
and uniformly at random from this set:  Alpha, Bravo, Charlie, Delta,
Echo, Foxtrot, Golf, Hotel, India, Juliet, Kilo, Lima, Mike, November,
Oscar, Papa, Quebec, Romeo, Sierra, Tango, Uniform, Victor, Whiskey,
Xray, Yankee, Zulu.  Output only the 100 items separated by commas on
a single line.  No other text, explanation, or numbering.
```

Chemical elements

```
Generate a sequence of exactly 100 items, each chosen independently
and uniformly at random from this set:  Hydrogen, Helium, Lithium,
Beryllium, Boron, Carbon, Nitrogen, Oxygen, Fluorine, Neon, Sodium,
Magnesium, Aluminium, Silicon, Phosphorus, Sulfur, Chlorine, Argon,
Potassium, Calcium, Scandium, Titanium, Vanadium, Chromium, Manganese,
Iron.  Output only the 100 items separated by commas on a single line.
No other text, explanation, or numbering.
```

Countries

```
Generate a sequence of exactly 100 items, each chosen independently
and uniformly at random from this set:  France, Japan, Brazil, Canada,
Egypt, India, Kenya, Mexico, Norway, Peru, Spain, Turkey, Vietnam,
Chile, Ghana, Italy, Nepal, Poland, Sweden, Thailand, Uganda, Cuba,
Fiji, Greece, Jordan, Portugal.  Output only the 100 items separated
by commas on a single line.  No other text, explanation, or numbering.
```

First names

```
Generate a sequence of exactly 100 items, each chosen independently
and uniformly at random from this set:  Alice, Ben, Clara, David,
Emma, Felix, Grace, Henry, Iris, Jack, Kate, Liam, Maya, Noah, Olivia,
Paul, Quinn, Rosa, Sam, Tara, Uma, Victor, Wendy, Xavier, Yara, Zoe.
Output only the 100 items separated by commas on a single line.  No
other text, explanation, or numbering.
```

Letters A–Z

```
Generate a sequence of exactly 100 random uppercase letters from A
to Z, each chosen independently and uniformly at random.  Output only
the 100 items separated by commas on a single line.  No other text,
explanation, or numbering.
```

Letters a–z

```
Generate a sequence of exactly 100 random lowercase letters from a
to z, each chosen independently and uniformly at random.  Output only
the 100 items separated by commas on a single line.  No other text,
explanation, or numbering.
```

Playing cards

```
Generate a sequence of exactly 100 random playing cards from a
standard 52-card deck, each chosen independently and uniformly at
random, written as rank then suit letter (ranks A, 2-10, J, Q, K;
suits S, H, D, C--e.g.  7H, AS, 10C, KD). Output only the 100 items
separated by commas on a single line.  No other text, explanation, or
numbering.
```

Roman numerals I–C

```
Generate a sequence of exactly 100 items, each chosen independently
and uniformly at random from this set:  I, II, III, IV, V, VI, VII,
VIII, IX, X, XI, XII, XIII, XIV, XV, XVI, XVII, XVIII, XIX, XX, XXI,
XXII, XXIII, XXIV, XXV, XXVI, XXVII, XXVIII, XXIX, XXX, XXXI, XXXII,
XXXIII, XXXIV, XXXV, XXXVI, XXXVII, XXXVIII, XXXIX, XL, XLI, XLII,
XLIII, XLIV, XLV, XLVI, XLVII, XLVIII, XLIX, L, LI, LII, LIII, LIV,
LV, LVI, LVII, LVIII, LIX, LX, LXI, LXII, LXIII, LXIV, LXV, LXVI,
LXVII, LXVIII, LXIX, LXX, LXXI, LXXII, LXXIII, LXXIV, LXXV, LXXVI,
LXXVII, LXXVIII, LXXIX, LXXX, LXXXI, LXXXII, LXXXIII, LXXXIV, LXXXV,
LXXXVI, LXXXVII, LXXXVIII, LXXXIX, XC, XCI, XCII, XCIII, XCIV, XCV,
XCVI, XCVII, XCVIII, XCIX, C. Output only the 100 items separated by
commas on a single line.  No other text, explanation, or numbering.
```

Grid cells A1–J10

```
Generate a sequence of exactly 100 random grid cells from a 10x10
grid with columns A-J and rows 1-10, each chosen independently and
uniformly at random, written as column letter then row number (e.g.
C7, A10, J1).  Output only the 100 items separated by commas on a
single line.  No other text, explanation, or numbering.
```

Number words one–hundred

```
Generate a sequence of exactly 100 items, each chosen independently
and uniformly at random from this set:  one, two, three, four, five,
six, seven, eight, nine, ten, eleven, twelve, thirteen, fourteen,
fifteen, sixteen, seventeen, eighteen, nineteen, twenty, twenty-one,
twenty-two, twenty-three, twenty-four, twenty-five, twenty-six,
twenty-seven, twenty-eight, twenty-nine, thirty, thirty-one,
thirty-two, thirty-three, thirty-four, thirty-five, thirty-six,
thirty-seven, thirty-eight, thirty-nine, forty, forty-one, forty-two,
forty-three, forty-four, forty-five, forty-six, forty-seven,
forty-eight, forty-nine, fifty, fifty-one, fifty-two, fifty-three,
fifty-four, fifty-five, fifty-six, fifty-seven, fifty-eight,
fifty-nine, sixty, sixty-one, sixty-two, sixty-three, sixty-four,
sixty-five, sixty-six, sixty-seven, sixty-eight, sixty-nine, seventy,
seventy-one, seventy-two, seventy-three, seventy-four, seventy-five,
seventy-six, seventy-seven, seventy-eight, seventy-nine, eighty,
eighty-one, eighty-two, eighty-three, eighty-four, eighty-five,
```

```
eighty-six, eighty-seven, eighty-eight, eighty-nine, ninety,
ninety-one, ninety-two, ninety-three, ninety-four, ninety-five,
ninety-six, ninety-seven, ninety-eight, ninety-nine, one hundred.
Output only the 100 items separated by commas on a single line.  No
other text, explanation, or numbering.
```

Numbers 00–99

```
Generate a sequence of exactly 100 random 2-digit codes, each of 2
digits 0-9 chosen independently and uniformly at random (leading zeros
allowed, e.g.  07, 90, 33).  Output only the 100 items separated by
commas on a single line.  No other text, explanation, or numbering.
```

Numbers 1–100

```
Generate a sequence of exactly 100 random integers, each chosen
independently and uniformly at random between 1 and 100 inclusive.
Output only the 100 numbers separated by commas on a single line.  Do
not include any other text, explanation, numbering, or formatting.
```

Percentages 0–99%

```
Generate a sequence of exactly 100 random percentages from 0% to
99%, each chosen independently and uniformly at random, written as
an integer followed by a percent sign (e.g.  47%, 3%, 90%).  Output
only the 100 items separated by commas on a single line.  No other
text, explanation, or numbering.
```

Years 1900–1999

```
Generate a sequence of exactly 100 random years, each chosen
independently and uniformly at random from 1900 to 1999 inclusive.
Output only the 100 items separated by commas on a single line.  No
other text, explanation, or numbering.
```

Hex bytes 00–FF

```
Generate a sequence of exactly 100 random bytes written as two
hexadecimal digits 00-FF, each chosen independently and uniformly
at random (e.g.  3F, 00, A7).  Output only the 100 items separated by
commas on a single line.  No other text, explanation, or numbering.
```

Dates in 2024

```
Generate a sequence of exactly 100 random calendar dates in the year
2024, each chosen independently and uniformly at random, in ISO format
YYYY-MM-DD. Output only the 100 items separated by commas on a single
line.  No other text, explanation, or numbering.
```

Letter pairs AA–ZZ

```
Generate a sequence of exactly 100 random 2-letter codes, each made
of 2 uppercase letters from A to Z chosen independently and uniformly
at random (for example:  AB, QX, MM, ZD). Output only the 100 items
separated by commas on a single line.  No other text, explanation, or
numbering.
```

3-digit PIN codes

```
Generate a sequence of exactly 100 random 3-digit PIN codes, each of 3
digits 0-9 chosen independently and uniformly at random (leading zeros
allowed, e.g.  047, 900, 333).  Output only the 100 items separated by
commas on a single line.  No other text, explanation, or numbering.
```

Numbers 000–999

```
Generate a sequence of exactly 100 random 3-digit codes, each of 3
digits 0-9 chosen independently and uniformly at random (leading zeros
allowed, e.g.  047, 900, 333).  Output only the 100 items separated by
commas on a single line.  No other text, explanation, or numbering.
```

Prices $0.00–$9.99

```
Generate a sequence of exactly 100 random prices from $0.00 to $9.99,
each chosen independently and uniformly at random, written with a
dollar sign and two decimals (e.g.  $0.47, $9.00, $3.33).  Output only
the 100 items separated by commas on a single line.  No other text,
explanation, or numbering.
```

Weights 0.0–99.9 kg

```
Generate a sequence of exactly 100 random weights from 0.0 kg to
99.9 kg, each chosen independently and uniformly at random, written
with one decimal place and the unit (e.g.  47.2 kg, 0.5 kg, 90.0 kg).
Output only the 100 items separated by commas on a single line.  No
other text, explanation, or numbering.
```

Years 1000–1999

```
Generate a sequence of exactly 100 random years, each chosen
independently and uniformly at random from 1000 to 1999 inclusive.
Output only the 100 items separated by commas on a single line.  No
other text, explanation, or numbering.
```

Probabilities 0.000–1.000

```
Generate a sequence of exactly 100 random probabilities between
0.000 and 1.000 inclusive, each chosen independently and uniformly
at random, written with exactly three decimal places (e.g.  0.047,
0.900, 1.000).  Output only the 100 items separated by commas on a
single line.  No other text, explanation, or numbering.
```

Times HH:MM

```
Generate a sequence of exactly 100 random times of day in 24-hour
HH:MM format, each chosen independently and uniformly at random (e.g.
04:17, 23:05).  Output only the 100 items separated by commas on a
single line.  No other text, explanation, or numbering.
```

Any English word

```
Generate a list of exactly 100 random English words, chosen as
randomly as you can (no theme, no alphabetical order, no repeats
needed).  Output only the 100 words separated by commas on a single
line.  No other text.
```

Numbers 0000–9999

```
Generate a sequence of exactly 100 random 4-digit codes, each of 4
digits 0-9 chosen independently and uniformly at random (leading zeros
allowed, e.g.  0473, 9001, 3330).  Output only the 100 items separated
by commas on a single line.  No other text, explanation, or numbering.
```

3-letter codes

```
Generate a sequence of exactly 100 random 3-letter codes, each made
of 3 uppercase letters from A to Z chosen independently and uniformly
at random (for example:  ABC, QXM, ZZD). Output only the 100 items
```

```
separated by commas on a single line.  No other text, explanation, or
numbering.
```

Hex colours #RRGGBB

```
Generate a sequence of exactly 100 random hex color codes in the
format #RRGGBB, each chosen independently and uniformly at random.
Output only the 100 items separated by commas on a single line.  No
other text, explanation, or numbering.
```

8-char passwords

```
Generate a sequence of exactly 100 random 8-character passwords, each
made of 8 characters chosen independently and uniformly at random
from uppercase letters, lowercase letters and digits.  Output only
the 100 items separated by commas on a single line.  No other text,
explanation, or numbering.
```

UUIDs

```
Generate a sequence of exactly 100 random version-4 UUIDs (format
xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx, lowercase hex).  Output only
the 100 items separated by commas on a single line.  No other text,
explanation, or numbering.
```

B SIMULATION MODEL SPECIFICATION

Table 8: Simulation model results by model.

Model	Structure captured	AI score LLM lists	AI score simulation-model lists
Qwen-7B	97%	0.94	0.90
Gemma-9B	90%	0.95	0.78
Llama-1B	88%	0.69	0.59
Qwen-1.5B	85%	0.93	0.81
Llama-8B	82%	0.77	0.43
Falcon-7B	79%	0.85	0.51
Mistral-7B	75%	0.54	0.44
Qwen-0.5B	75%	0.72	0.58
mean	*84%*	*0.80*	*0.63*

A list is $T = 100$ digits $s_t \in \{0, \dots, 9\}$, the first three a fixed seed. The simulation model defines $q(s_t \mid s_{<t})$ for $t \geq 4$ as a two-state mixture, wander W and copy C. Write σ for the logistic function, $\ell_t(d)$ for the steps since digit d was last used, $c_t(d)$ for its use count, $\rho_t(d)$ for an indicator that it has already appeared in the current row of ten, and $\Delta_t(d) = d - s_{t-1}$.

Wander scores each candidate digit and emits a softmax of that score, mixed with the same softmax restricted to digits not used in the last three steps:

$$\phi_t(d) = b_d + r_{\beta(\ell_t(d))} + \kappa\big(c_t(d) - t/10\big) + \theta\,\rho_t(d) + \psi_{\Delta_t(d)}, \tag{2}$$

$$e_t^{\mathsf{W}} = (1-\pi)\,\mathrm{softmax}(\phi_t) \;+\; \pi\,\mathrm{softmax}_{D_t}(\phi_t), \qquad D_t = \{0,\dots,9\} \setminus \{s_{t-1}, s_{t-2}, s_{t-3}\}. \tag{3}$$

Copy continues an earlier pattern. The target τ_t is the digit that followed the most recent earlier occurrence of the longest suffix of $s_{<t}$ recurring at least eight positions back; with no match at a row boundary the target is a fresh row drawn from row-start appeals a. With k_t the number of consecutive preceding positions whose emitted digit equalled its target,

$$e_t^{\mathsf{C}} = \big(1 - \varepsilon(k_t)\big)\,\delta_{\tau_t} + \varepsilon(k_t)\,e_t^{\mathsf{W}}, \qquad \varepsilon(k) = \sigma\big(c_0 + c_1 \log(1+k)\big). \tag{4}$$

The mode switches with one constant probability in each direction, and the prediction is the Bayes-filtered mixture:

$$q_t(d) = \sum_z \alpha_t(z)\, e_t^z(d), \qquad \alpha_{t+1}(z') \propto \sum_z \alpha_t(z)\, e_t^z(s_t)\, P(z_{t+1} = z' \mid z_t = z). \tag{5}$$

All 66 parameters are fitted together by minimizing $\sum_t \mathrm{CE}(p_t, q_t)$ over the training lists, where p_t is the LLM's own next-digit distribution under teacher forcing: Adam, learning rate 0.05, 400 epochs, and a weak ℓ_2 penalty on the wander weights. Table 9 names the parameters and Tables 10 and 11 give the fitted values. When reporting model next-digit entropy, we compute $H(p_t)$ at each teacher-forced position and average over positions; the uniform reference is $\ln(10) \approx 2.30$ nats.

Table 9: The 66 parameters. Lag buckets β are 1–16 individually, then 17–24, 25+ and never-used.

	symbol	quantity	count
wander	b	appeal of each digit	10
	r	appeal by bucketed lag since the digit was last used	19
	κ	uses so far minus the digit's fair share	1
	θ	the digit already appeared in this row of ten	1
	ψ	signed step from the previous digit, -8 to $+9$	18
	π	weight on the branch ruling out the last three digits	1
copy	c_0, c_1	commitment curve: intercept and slope on $\log(1+k)$	2
	a	appeal of each digit when a fresh row begins	10
	ν	a row boundary with no match restarts a row	1
	ε_R	escape from the restart rule to wander	1
switching	$P(\mathsf{W}\to\mathsf{C})$	start copying	1
	$P(\mathsf{C}\to\mathsf{W})$	break off	1
		total	**66**

Table 10: Wander-mode weights of the simulation model (logits), one fit for each of the eight analyzed LLMs; OLMo is excluded because its default-configuration outputs do not reach 100 digits. The row-of-ten weight also carries the -9 signed step, a pairing that arises only when a 9 is followed by a 0 and occurs in 2% of steps against the row indicator's 47%; the signed-step block therefore runs from -8 to $+9$.

parameter	Qwen2.5 0.5B	Qwen2.5 1.5B	Qwen2.5 7B	Llama-3.2 1B	Llama-3.1 8B	Gemma-2 9B	Mistral 7B	Falcon 7B
Base appeal								
digit 0	-0.82	-0.12	-0.75	-0.99	-0.36	-0.03	+0.06	-0.16
digit 1	+0.24	-0.37	-0.55	+0.12	+0.15	+0.53	+0.08	+0.08
digit 2	+0.43	+0.70	+0.82	+0.66	+0.07	+1.08	+0.31	+0.12
digit 3	+0.01	+0.01	+1.01	+0.29	-0.32	+0.47	+0.15	-0.23
digit 4	-0.48	-0.22	-0.44	+0.01	-0.20	-1.16	-1.19	+0.20
digit 5	-0.27	+0.15	+0.06	-0.29	+0.04	-0.11	+0.01	+0.03
digit 6	-0.52	-0.63	+0.33	+0.65	+0.08	-0.28	+0.24	-0.00
digit 7	+0.81	+0.24	+0.48	-0.11	-0.24	-0.52	-0.32	+0.26
digit 8	+0.12	+0.17	+0.61	-0.49	+0.37	+0.05	+0.33	-0.39
digit 9	+0.54	+0.08	-1.59	+0.35	+0.41	-0.03	+0.36	+0.11
Recency: steps since the digit was last used								
1	-0.69	-0.39	-0.04	-0.26	-0.53	-0.16	+0.09	-1.13
2	-1.03	-0.51	-0.04	-0.29	-0.66	-0.32	-0.19	-0.82
3	-0.12	+0.17	-0.05	-0.30	-0.46	-0.11	+0.37	+0.41
4	-3.12	-3.01	-2.17	-4.12	-3.66	-2.82	-2.96	-2.00
5	-1.96	-2.16	-1.66	-3.83	-2.82	-2.07	-1.88	-0.97
6	-1.07	-1.67	-1.56	-3.17	-1.96	-1.56	-1.41	-0.41
7	-0.39	-0.81	-1.68	-1.66	-1.00	-1.07	-1.43	+0.23
8	+0.30	-0.32	-1.46	-0.48	-0.22	-0.68	-0.88	+0.18
9	+0.74	+0.10	-0.00	+1.05	+0.83	-0.10	-0.52	+0.49
10	+0.89	+1.01	+0.83	+2.00	+1.76	+1.38	+0.42	+0.52
11	+1.14	+1.45	+1.85	+2.34	+2.19	+1.34	+1.14	+0.46
12	+1.32	+1.44	+1.68	+2.66	+2.07	+1.53	+1.05	+0.34
13	+1.50	+1.34	+1.25	+3.00	+1.90	+1.94	+1.16	+0.36
14	+1.57	+1.16	+0.55	+3.10	+1.73	+2.12	+1.33	+0.35
15	+1.62	+1.16	-0.02	+2.86	+1.64	+0.62	+1.01	+0.36
16	+1.68	+0.87	-0.02	+2.38	+1.47	-0.08	+1.09	+0.33
17–24	+1.73	+0.70	+0.06	+2.75	+1.74	-0.19	+1.25	+0.04
25+	-2.17	-0.32	+0.00	+0.44	-0.16	-0.02	+0.04	-1.58
never used	+4.56	+3.43	+2.37	+3.14	+3.28	+1.81	+3.25	+3.49
Quota and row								
uses so far minus fair share	-0.03	-0.21	-2.68	+0.66	+0.29	-1.82	-0.00	+0.33
already in this row of ten	-0.31	-0.93	-0.28	-0.39	-0.72	-1.96	-0.97	-0.27
Signed step: next minus previous digit								
-8	-1.02	-0.50	-0.54	-0.32	-0.49	-0.78	-0.25	-0.89
-7	+0.24	-0.86	-0.97	-0.02	-0.77	-1.11	-1.29	-0.38
-6	-0.17	-0.58	-0.52	-0.27	-0.51	-0.52	-0.95	-1.04
-5	-0.03	-0.65	-1.49	-0.50	-0.74	-0.84	-0.44	-0.56
-4	-0.11	-0.06	-1.18	-0.21	-0.37	-0.58	+0.09	-0.72
-3	+0.08	-0.30	-0.84	-0.09	-0.41	-0.62	+0.07	-0.41
-2	+0.22	-0.23	-0.67	-0.15	-0.13	-0.39	-0.15	-0.50
-1	+0.01	-0.89	-0.40	+0.23	-0.26	-0.52	-0.29	+0.51
$+0$	-0.69	-0.39	-0.04	-0.26	-0.53	-0.16	+0.09	-1.13
$+1$	-0.17	-0.50	-0.40	-0.18	-0.66	-0.49	-0.50	+1.28
$+2$	-0.07	+0.17	-0.62	-0.15	+0.04	-0.30	+0.52	-0.08
$+3$	-0.24	-0.36	-0.85	-0.12	-0.15	-0.26	-0.19	-0.02
$+4$	-0.13	+0.07	-0.53	-0.01	+0.04	-0.10	+0.41	-0.13
$+5$	+0.18	-0.37	+0.50	-0.12	+0.09	+0.20	-0.44	-0.07
$+6$	-0.01	+0.14	+1.23	+0.02	+0.23	+0.53	-0.13	-0.66
$+7$	-0.04	+0.34	+1.06	+0.13	-0.11	+0.45	-0.04	-0.17
$+8$	+0.24	+0.87	-0.10	+0.60	+0.70	+0.44	-0.11	+0.21
$+9$	+0.73	+0.59	+0.88	+0.18	+0.84	+0.19	+1.00	+0.76

Table 11: Switching, copy-commitment, exclusion and fresh-row parameters of the simulation model for the same eight fits. All probabilities except the row-start appeals, which are logits. The copy-commitment row is the fitted two-parameter curve $1 - \varepsilon(k)$ read at seven streak lengths.

parameter	Qwen2.5 0.5B	Qwen2.5 1.5B	Qwen2.5 7B	Llama-3.2 1B	Llama-3.1 8B	Gemma-2 9B	Mistral 7B	Falcon 7B
Exclusion								
π, last three digits ruled out	0.993	0.992	0.995	0.998	0.996	0.983	0.979	0.951
Switching								
start copying	0.171	0.256	0.145	0.331	0.084	0.098	0.102	0.073
break off	0.001	0.002	0.002	0.002	0.002	0.002	0.005	0.002
Copy: probability of emitting the target, by streak k								
$k = 0$	0.00	0.00	0.87	0.01	0.01	0.02	0.25	0.31
$k = 1$	0.03	0.06	0.97	0.07	0.06	0.16	0.53	0.60
$k = 2$	0.21	0.34	0.99	0.22	0.19	0.39	0.70	0.76
$k = 3$	0.54	0.70	0.99	0.42	0.36	0.60	0.79	0.84
$k = 5$	0.90	0.95	1.00	0.74	0.67	0.84	0.88	0.91
$k = 10$	1.00	1.00	1.00	0.96	0.93	0.97	0.96	0.97
$k = 15$	1.00	1.00	1.00	0.99	0.98	0.99	0.98	0.98
Fresh row								
probability of restarting	0.99	1.00	0.61	0.73	0.43	1.00	0.90	0.24
escape from the restart rule	0.157	0.057	0.002	0.333	0.416	0.000	0.001	0.198
row-start appeal, digit 0	+0.54	+3.85	-1.89	+1.20	+4.47	+3.40	-3.81	-3.63
row-start appeal, digit 1	+1.34	+2.46	-2.27	+2.86	+0.48	-1.22	-2.26	-3.81
row-start appeal, digit 2	+3.36	+2.12	-2.01	+0.41	+2.34	+3.57	-0.18	-1.55
row-start appeal, digit 3	-5.00	-2.33	-3.54	+0.74	-4.61	+1.73	-2.27	-3.73
row-start appeal, digit 4	-0.40	-1.27	-1.95	+1.61	+2.06	-3.78	-1.12	-0.11
row-start appeal, digit 5	-0.22	+1.59	-2.27	+0.54	+1.31	+0.26	-2.40	+4.67
row-start appeal, digit 6	+0.13	+1.33	-2.42	-2.26	+1.51	+1.25	-2.35	-0.81
row-start appeal, digit 7	+1.12	-0.64	-2.44	+0.96	+2.29	-3.04	-3.87	-2.73
row-start appeal, digit 8	-5.34	-5.37	+2.46	-4.56	-2.57	-5.91	+2.29	-4.69
row-start appeal, digit 9	-0.71	+2.52	-2.70	+1.89	+3.53	-3.63	-1.35	+3.58

C VOCABULARY AND TRANSFORMATION RESULTS

Table 12: All 50 scored vocabularies. LLM counts are retained valid sequences; each vocabulary uses 60 algorithmic controls. Cardinality: how many distinct values the vocabulary offers; ord.: has a natural order.

Vocabulary	Cardinality	Family	ord.	n_{LLM}	n_{rand}	LLM	rand.
binary 0/1	2	number	✓	240	60	0.04	0.01
coin flips H/T	2	letter		324	60	0.16	0.03
left / right	2	word		333	60	0.14	0.02
true / false	2	word		399	60	0.17	0.02
up / down	2	word		400	60	0.16	0.02
yes / no	2	word		353	60	0.24	0.03
chess pieces	6	word		330	60	0.28	0.04
dice rolls 1–6	6	number	✓	395	60	0.40	0.01
continents	7	word		229	60	0.62	0.03
fruits	7	word		198	60	0.66	0.02
weekdays	7	word	✓	224	60	0.51	0.03
planets	8	word	✓	339	60	0.38	0.03
digits 0–9	10	number	✓	240	60	0.60	0.02
adjectives	12	word		167	60	0.79	0.02
animals	12	word		205	60	0.58	0.02
colours	12	word		283	60	0.53	0.01
months	12	word	✓	209	60	0.41	0.04
musical notes	12	word	✓	270	60	0.34	0.01
zodiac signs	12	word	✓	310	60	0.51	0.05
Greek letters	24	word	✓	247	60	0.54	0.02
NATO alphabet	26	word	✓	226	60	0.82	0.02
chemical elements	26	word	✓	224	60	0.25	0.02
countries	26	word		247	60	0.43	0.03
first names	26	word		274	60	0.73	0.07
letters A–Z	26	letter	✓	194	60	0.64	0.01
letters a–z	26	letter	✓	241	60	0.52	0.01
playing cards	52	symbol		180	60	0.56	0.04
Roman numerals I–C	100	number	✓	230	60	0.16	0.03
grid cells A1–J10	100	symbol		306	60	0.95	0.03
number words one–hundred	100	number	✓	236	60	0.85	0.03
numbers 00–99	100	number	✓	400	60	0.94	0.03
numbers 1–100	100	number	✓	500	60	0.90	0.03
percentages 0–99%	100	number	✓	397	60	0.96	0.07
years 1900–1999	100	number	✓	396	60	0.29	0.03
hex bytes 00–FF	256	number	✓	253	60	0.77	0.02
dates in 2024	366	format	✓	392	60	0.93	0.17
letter pairs AA–ZZ	676	letter		152	60	0.35	0.05
3-digit PIN codes	1,000	number	✓	397	60	0.94	0.02
numbers 000–999	1,000	number	✓	240	60	0.94	0.03
prices \$0.00–\$9.99	1,000	number	✓	385	60	0.89	0.02
weights 0.0–99.9 kg	1,000	number	✓	378	60	0.96	0.02
years 1000–1999	1,000	number	✓	374	60	0.81	0.03
probabilities 0.000–1.000	1,001	number	✓	396	60	0.92	0.01
times HH:MM	1,440	format	✓	360	60	0.93	0.05
any English word	5,000	open		263	60	0.83	0.02
numbers 0000–9999	10,000	number	✓	240	60	0.86	0.03
3-letter codes	17,576	letter		164	60	0.20	0.04
hex colours #RRGGBB	10^7	open		262	60	0.81	0.08
8-char passwords	10^{14}	open		189	60	0.77	0.06
UUIDs	10^{36}	open		242	60	0.37	0.04

Table 13: The untransformed baseline and 44 scored one-shot transforms, evaluated on the same LLM sequences and 60 pseudorandom sequences. Each sequence contains 100 numbers from 000–999. Retained: share of the untransformed signal left.

Transform	Family	AI	floor	retained
Edits to the values or their order				
no edit	baseline	0.94	0.02	100%
reverse the list	keeps order & spacing	0.94	0.02	100%
keep the order, space the values evenly	remaps values, keeps order	0.32	0.04	31%
square every value		0.05	0.02	4%
square-root every value		0.16	0.02	16%
swap every pair of neighbours	reorders	0.85	0.02	90%
riffle the two halves together		0.06	0.02	4%
shuffle the list		0.03	0.02	1%
sort the list		0.03	0.02	0%
redraw values from the list, allowing repeats	destroys everything	0.01	0.01	0%
Ciphers, re-encodings and formatting				
Caesar shift	substitution	0.70	0.02	73%
ROT5		0.96	0.02	102%
Atbash		0.95	0.02	101%
Affine cipher		0.04	0.02	1%
simple substitution cipher		0.03	0.02	0%
simple substitution on the first digit only		0.81	0.02	86%
simple substitution on the last digit only		0.92	0.03	98%
Polybius square on the first digit		0.08	0.02	6%
Vigenère cipher	polyalphabetic	0.02	0.03	0%
Vigenère cipher, key in whole hundreds		0.04	0.02	2%
Beaufort cipher		0.02	0.03	0%
Autokey cipher		0.04	0.02	1%
Hill cipher		0.03	0.02	0%
Playfair cipher		0.02	0.02	0%
Enigma-style rotor cipher		0.02	0.02	0%
one-time pad		0.02	0.03	0%
rail fence cipher, 2 rails	transposition	0.21	0.02	21%
rail fence cipher, 5 rails		0.04	0.02	2%
scytale		0.02	0.02	0%
columnar transposition		0.03	0.02	0%
route cipher		0.05	0.02	3%
write the numbers as words	re-encoding	0.65	0.02	68%
write as Roman numerals		0.05	0.03	2%
write in hexadecimal		0.19	0.03	18%
write in binary		0.02	0.01	1%
write as three-letter codes		0.17	0.13	4%
replace each digit with a letter		0.30	0.05	27%
use full-width Unicode digits		0.93	0.02	99%
Baconian cipher		0.01	0.01	0%
Base64-encode the whole list		0.02	0.01	1%
drop the leading zeros	formatting	0.71	0.02	74%
separate with spaces instead of commas		0.11	0.02	10%
one number per line		0.11	0.02	10%
format as a JSON array		0.80	0.06	80%

Transform	Family	AI	floor	retained
format as a numbered list		0.27	0.02	28%

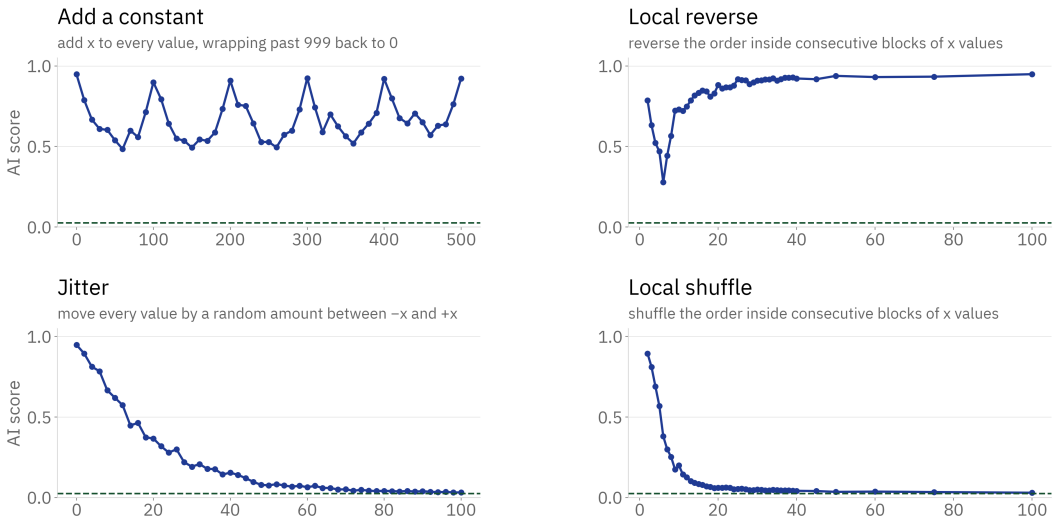

Figure 4: **Transformation dose responses.** Mean AI scores across transformation strengths. Jitter and local shuffling progressively reduce the AI signal, larger reversed blocks restore it, and modular addition repeatedly recovers detectability when the added constant is a multiple of 100.