

2 lists of 30 test names from the same model share a name in 64 to 98% of pairs, against 1.29% for Faker

7 Anthropic models, 7 value types and 1274 calls, measured as the probability that a seed with a UNIQUE constraint fails

Dmitriy Semenkevich · [ORCID 0009-0009-3013-4978](#)

21 October 2026 · [dimhold.by](#)

ABSTRACT

That language models converge on the same invented answers is established: Artificial Hivemind and NoveltyBench measure it across models, the One-Word Census measures it on single word answers across 105 models and repetition in generated name lists is a known instance. LLMs are also known to be poor random number generators and a study of fake PII substitution has reported a small model's fake names and addresses as a less varied training distribution than Faker's, while another reports LLM generated fields as more diverse than Faker's templates. This study asks the narrower engineering question those works leave open. 2 engineers ask the same model for 30 test values each. How often do the 2 sets share at least 1 value, so that a database seed with a UNIQUE constraint built from both fails? The unit is the pair of independent runs and the measure is P_{pair} , the share of such pairs with a shared value. 7 Anthropic models, current and previous releases, were asked for 7 value types under 3 prompt arms through the Claude Code CLI, 1274 calls in all, each a separate process with no history, against Faker 40.43.0 at the same size as the baseline. For person names in the plain arm P_{pair} ran from 64% to 98% across the 7 models, against 1.29% for Faker. Every model's jackknife lower bound sat above Faker and pooled over models 256 of 315 pairs of name lists collided. Identifiers depended on the model: 2 email lists from `claude-opus-5-5` collided in 56% of pairs and 2 from `claude-haiku-5-5` in none of 45. An explicit request for diversity took email collisions to 0 of 105 pairs and left names colliding in 58 of 105. A registered expectation that asking for JSON reduces diversity was not confirmed: per model shifts went both ways with no consistent direction. The UUID type is not interpretable through this transport, because several models answered with tool call markup instead of values. The CLI passed account context to the model; whether that shaped which names appear is untested.

1. The question

A developer who needs test data and has a model open asks it: “Generate 30 random email addresses for test fixtures.” A colleague on the same team does the same an hour later for another test. The 2 fixtures end up in 1 database seed with a `UNIQUE` constraint on the column.

Whether that seed fails is a question about pairs. A single list can look perfectly random and still be the same list every time the model is asked. The quantity that decides the engineering outcome is therefore not the diversity of 1 reply but **the probability that 2 independent replies share at least 1 value**. This study calls it `P_pair` and measures it for 7 value types across 7 models, next to the tool the model replaces in this job, the Faker library.

2. Prior work

The phenomenon underneath is well documented and this study is not first to it.

- **Artificial Hivemind** ([arXiv:2510.22954](https://arxiv.org/abs/2510.22954)) measures repetition within a model and homogeneity across models on open ended prompts. Its abstract names “random number or name generation” among the narrow tasks diversity has already been evaluated on and Infusing Random Concepts (below) benchmarks with prompts such as “Name 10 Hollywood actors”. Repetition in generated name lists is therefore known; the name results below are an instance of it, measured in a different unit.
- **NoveltyBench** ([arXiv:2504.05228](https://arxiv.org/abs/2504.05228)) evaluates how far models fall short of humanlike diversity when asked the same thing repeatedly.
- **The One-Word Census** ([arXiv:2607.12796](https://arxiv.org/abs/2607.12796)) asks 105 models from more than 20 labs for 1 word in 96 categories, scored by exact match. In 28 categories a single answer takes at least 80% of answers. It finds no panel wide trend over release order once model tier is controlled; within that it reports that GPT, Gemini, Grok and Qwen become more conformist across releases while Claude’s generation 5 releases reverse. A comparison of Claude generations on single answers therefore already exists. Anything said about generations below concerns sets of identifiers and comes after that result.

- **LLMs as random number generators** ([arXiv:2502.19965](#)) find that models often answer deterministically across the models, ranges, temperatures and prompt languages tested, with significant differences by model and by prompt language. The integer type in this study is an instance of that result.
- **Structured Output Collapses Answer Diversity** ([arXiv:2607.18476](#)) reruns the Census with “Reply with JSON only” and finds the modal answer rising from 41% to 64% of the pool on the unconstrained prompt. Individually, 6 of its 44 models move significantly, all toward the mode. That result is the reason for the `json` arm here.
- Mitigations are an active line: Verbalized Sampling ([arXiv:2510.01171](#)), String Seed of Thought ([arXiv:2510.21150](#)) and Infusing Random Concepts ([arXiv:2601.18053](#)).
- On test data specifically, generative models have been used to write test data generators and judged on realism ([arXiv:2401.17626](#)) and to generate high fidelity mock data for SQL code generation services, judged on syntactic correctness and semantic relevance ([arXiv:2504.17203](#)). Fixturize ([arXiv:2601.06615](#)) addresses fixtures in the sense of test setup code rather than values.
- **The closest prior result is on fake PII.** A study of on device PII substitution ([arXiv:2605.13538](#)) has a small model (`Bonsai-1.7B` at 1 bit, with `Ternary-Bonsai-1.7B` at 1.58 bits as a check) propose fake names and addresses and compares the result with `faker` directly. It finds that the model’s surrogates read more naturally (perplexity) but, judged by downstream NER where Faker wins, form a less varied training distribution; it separately quantifies a residual copying from its demonstration pool. The pipeline also covers dates. That study measures naturalness and NER utility inside a substitution pipeline, not overlap between independent runs. The prior work check written before counting missed this comparison; see the addendum in `PRIOR-WORK.md`.
- An earlier study of synthetic identity document data ([arXiv:2411.14962](#)) reports the opposite direction: LLM generated fields as more diverse and contextually relevant than Faker’s templates and better downstream barcode detection. Its abstract describes no measure of repeats between runs.

Searches on 6 September 2026 (arXiv, Semantic Scholar, Zenodo) and on 19 October 2026 (general web search) found engineering notes about collisions in Faker seeds and parallel tests, but no study treating model generated fixtures as a source of such collisions. The

arXiv API returned nothing on 19 October, including for a known existing identifier used as a control, so the arXiv part of that day was not counted. The arXiv search was rerun over https on 21 October 2026, with a control query that returned the known 2605.13538: 16 queries on synthetic test data and diversity, fake names and collisions, mode collapse and names, Faker and LLMs, test fixtures, test data generation and synthetic PII. They returned 2411.14962 and 2605.13538 as relevant and no study of collisions between independent runs of model generated values.

What this study adds is therefore narrow: **the pair of independent fixture sets as the unit**, `P_pair` as a direct estimate of how often a `UNIQUE` seed fails, **identifiers** such as email addresses and usernames as the subject, **Faker at identical sizes as a pairwise collision baseline** and a panel of 7 current and previous Anthropic models through the interface a developer uses. That model fake values can be less varied than Faker’s has been reported (2605.13538; 2411.14962 reports the opposite) and is not claimed here.

3. Method

3.1 Requests and arms

7 types, each requested in the plainest wording a developer would type:

TYPE	REQUEST
<code>email</code>	Generate 30 random email addresses for test fixtures.
<code>username</code>	Generate 30 random usernames for test fixtures.
<code>uuid</code>	Generate 30 random UUID v4 values for test fixtures.
<code>fullname</code>	Generate 30 random person full names for test fixtures.
<code>company</code>	Generate 30 random company names for test fixtures.
<code>address</code>	Generate 30 random US street addresses for test fixtures.
<code>integer</code>	Generate 30 random integers between 1 and 1000 for test fixtures.

3 arms append a tail to each request:

ARM	TAIL	RUNS PER CELL
<code>plain</code>	Output one value per line and nothing else: no numbering, no quotes, no commentary.	10
<code>json</code>	Reply with a JSON array of strings only: no prose, no code fences.	10
<code>diverse</code>	Make them as different from one another as you can: vary length, structure and starting letters, and avoid the obvious placeholder values. Plus the <code>plain</code> tail.	6

3.2 Panel and transport

The current generation: `claude-opus-5-5`, `claude-sonnet-5-5`, `claude-haiku-5-5` and `claude-fable-5-1`. The previous generation: `claude-opus-5`, `claude-sonnet-5` and `claude-haiku-4-5`. `claude-haiku-4-5` is a 4.5 release, so “previous” spans generations 5 and 4.5 and the haiku line skips a generation. 7 models x 7 types x 26 runs gives 1274 calls. All 1274 succeeded and in all 1274 the model the CLI reported in `modelUsage` matched the one requested. 26 further attempts failed before reaching a model, 24 of them on the CLI starting without its prompt under load. They were retried; they are kept in the journal and excluded. Separately, 5 successful calls, all of them UUID requests, needed 2 to 8 attempts inside the harness; their failed attempts (session limits, 1 timeout and 1 tool call the CLI could not parse) appear in the run logs, not in the journal.

Every call is a separate `claude -p` process on CLI 2.1.302 with `--no-session-persistence`, so runs share no history. Calls run from an empty directory whose path upward was checked for instruction files before each launch, with `--tools ""`, an empty MCP configuration, `--setting-sources ""` and slash commands disabled. A probe recorded the same input size with and without user settings, 3648 tokens including cache, as `context-check.json` records; the registration gives 3646, a typo left uncorrected in the registered text. Temperature cannot be set through the CLI and is the subscription default, which is what a developer at a terminal gets. The calls ran between 01:32 on 19 October and 01:41 on 20 October 2026 (UTC+3).

3.3 Measures

Replies are parsed by fixed rules: 1 value per line for `plain` and `diverse`, a JSON array for `json`, list markers and quotes stripped and counted as format violations. Values are normalised to lower case with whitespace trimmed and collapsed. The registration collapsed whitespace only for names and addresses; `report.py` collapses it for every type, a small deviation. A reply that contains tool call markup is treated as containing no values, so lines of code cannot produce false matches between runs.

- **P_pair**: within a cell (model, type, arm), the share of pairs of runs whose value sets intersect. A `plain` cell of 10 runs has 45 pairs.
- **Interval**: a 95% jackknife over runs. Pairs inside a cell share runs and are not independent, so an interval over the count of pairs would be too narrow. The registration names no interval; the jackknife was added during the run (section 6).
- **Union share**: distinct values across all runs of a cell divided by the total returned.
- Also counted: duplicates inside 1 reply, UUID v4 validity, overlap across models and the most frequent values, listed without selection.

3.4 Baselines

Faker 40.43.0 with the `en_US` locale, unseeded, generated 300 sets of 30 per type, 44,850 pairs, for `email`, `username`, `fullname`, `company` and `address`. `uuid4` from the Python standard library is the baseline for `uuid` and `random.randint` for `integer`. Union shares come from 200 repeats of 10 sets, the shape of a `plain` cell.

3.5 Registration

Expectations and the conditions that would refute them were written down before the first result: the registration file was written at 01:12 on 19 October 2026, 18 minutes before the main phase was launched at 01:30 (first reply journaled at 01:32:17). It was committed only at 01:55, when 52 counted replies of the main phase (run 0 of 52 cells) and 1 failed attempt were already in the journal; section 6 lists what was seen and changed in between. The main kill condition, carried over from a predecessor run that never reached a verdict, was this: if the union share in the `plain` arm exceeded 90% for every model on every

type, the models produce fixtures no worse than a real generator and the measurement is negative.

4. Results

In the `plain` arm, 2 lists of 30 names from the same model shared at least 1 name in 64% to 98% of pairs, depending on the model. Faker at the same size did so in 1.29%.

4.1 `P_pair` in the `plain` arm

`P_pair` in %, 95% jackknife interval over 10 runs.

TYPE	OPUS- 5-5	SONNET- 5-5	HAIKU- 5-5	FABLE- 5-1	OPUS- 5	SONNET- 5	HAIKU- 4-5	FAKER
fullname	96 [85, 100]	64 [28, 100]	98 [91, 100]	80 [56, 100]	64 [30, 99]	96 [85, 100]	71 [32, 100]	1.29
company	82 [59, 100]	69 [41, 97]	27 [6, 48]	18 [0, 39]	11 [0, 24]	20 [6, 34]	49 [15, 83]	5.44
username	38 [18, 57]	38 [12, 63]	42 [4, 81]	18 [0, 37]	31 [3, 59]	4 [0, 12]	13 [1, 26]	1.15
email	56 [19, 92]	36 [7, 64]	0 [0, n/a]	20 [0, 40]	2 [0, 9]	7 [0, 22]	11 [0, 24]	0.38
address	16 [0, 34]	51 [16, 86]	13 [0, 37]	2 [0, 9]	0 [0, n/a]	16 [0, 38]	7 [0, 22]	0.00

Where no pair collides the jackknife has zero width, which is not certainty, so those 2 cells show no upper bound. 0 of 45 pairs from 10 runs does not exclude a rate of several percent: the rule of three over 45 pairs gives 6.7%, and because pairs share runs the real bound is wider than that.

Names. Every model's lower bound sits above Faker. Pooled over the 7 models, 256 of 315 pairs collided. Union shares for names ran from 66.9% (`claude-haiku-5-5`) to 87.7%

(claude-opus-5) against 99.74% for Faker, and within any single reply no model repeated a name. The repetition is entirely across runs. The most frequent names in the 70 plain runs, counting runs rather than occurrences:

NAME	RUNS OF 70
dmitri volkov	26
priya raghunathan	24
kwame asante	19
yuki tanaka	15
amara okonkwo	15

Across all 3 arms “Dmitri Volkov” appears in 66 of 182 name runs. The first name matches the author’s and the account context reached the model (section 6); whether that promoted this name is untested. Removing the name from every set changes the pooled count from 256 to 254 of 315 pairs and moves only claude-opus-5-5, from 96% to 91%, so the range in the headline does not rest on that 1 string. The test cannot show what the lists would be without the context.

Identifiers. The picture splits by model. 2 email lists from claude-opus-5-5 collided in 56% of pairs and from claude-sonnet-5-5 in 36%; 2 lists from claude-haiku-5-5 never collided in 45 pairs. The most frequent email address, alice.morgan@example.com, appears in 7 of 70 runs. For usernames 4 models have lower bounds above Faker: claude-opus-5-5, claude-sonnet-5-5, claude-haiku-5-5 and claude-opus-5. quietfalcon42 and quietfox42 each appear in 7 of 70 runs.

Company names collided in more than half of pairs for 2 models of 7, claude-opus-5-5 (82%) and claude-sonnet-5-5 (69%). It was 11% to 49% for the rest. Point estimates are above Faker’s 5.44% for all 7; the lower bound clears it for 5. Brightwater Logistics appears in 12 of 70 runs.

Integers collided in 79% to 100% of plain pairs, where a simulated true generator gives 59.2%. That simulation draws 30 values with replacement (analytic 59.4%); for 30 distinct

values, closer to what the models returned, the analytic rate is 60.4%; the registration gave “about 61%”. The lowest jackknife lower bounds among cells of 8 runs or more, 61.1% and 62.5% for `claude-haiku-5-5` and `claude-fable-5-1`, clear even the distinct rate, but only just. 847 appears in 21 of 70 runs. 2 cells rest on few runs, because `claude-opus-5` returned no integers in 7 of 10 replies and `claude-haiku-4-5` in 8 of 10.

4.2 The registered expectations

REGISTERED EXPECTATION OR KILL CONDITION	OUTCOME
Kill: <code>plain</code> union above 90% for all models on all types	not met: 29 of 47 cells above 90% (26 of 42 without uuid), every model below 90% on names
Names and companies: <code>P_pair</code> above 50% for most models and above Faker	names: yes, 7 of 7; companies: above Faker 7 of 7, above 50% only 2 of 7
Email and username: <code>P_pair</code> above 10% for at least some models	yes: email 4 of 7, username 6 of 7
Kill: identifiers not above 5% for any model	not met: 13 of 19 identifier cells above 5% (11 of 14 without uuid)
<code>json</code> gives less diversity than <code>plain</code>	not confirmed: shifts in both directions (section 4.3)
<code>diverse</code> raises the union share but not to Faker	names: yes, 0 of 7 at Faker; usernames and companies: mostly, 1 and 2 of 7 at Faker; email: no, 7 of 7 at Faker; integers: not raised, up for 4 of 7 with a mean change of -1.84 points
Kill: <code>diverse</code> union above 98% on all types	not met: 29 of 49 cells

4.3 The JSON arm

The expectation from the structured output study did not carry over as a rule. Across the 47 cells with data in both arms, the union share under `json` minus `plain` averages -0.3 percentage points. 12 cells fall by 2 points or more and 8 rise by 2 points or more. The

direction depends on the model: on names `json` lowered the union share for `claude-opus-5-5` by 11.7 points and for `claude-sonnet-5-5` by 11.0 and raised it for `claude-haiku-5-5` by 9.4. Pooled `P_pair` moved in both directions too: usernames 41.0% under `json` against 26.3% under `plain`, email 15.2% against 18.7%.

The registered kill condition reads “within ± 2 pp on all types”. It was applied per type, as the mean over models of the `json` minus `plain` union share for each type. Read that way, 5 of 7 type means are inside ± 2 points: email +1.43, username -1.90, fullname -1.75, company -0.71 and address 0.00. The condition fails to fire only because of integer at -4.62 and uuid at +7.87, while uuid is a transport artifact (section 6). The integer mean is pulled by 2 `plain` cells with few runs, `claude-haiku-4-5` at -26.3 on 2 runs and `claude-opus-5` at +8.5 on 3; without cells under 5 runs it is -2.90, still just outside. Read per cell, 19 of 47 cells are outside ± 2 points (1 more sits at exactly -2.00), with a mean of -0.29.

So the hypothesis is neither killed by the letter of the registration nor confirmed: per model shifts go both ways, on names from -11.7 to +9.4 points, with no consistent direction. The structured output study itself found only 6 of 44 models moving individually, all toward the mode, so a per model effect is what it would lead one to expect; here the larger shifts went both ways.

4.4 Asking for diversity

The `diverse` tail fixes email completely: all 7 models reached a 100% union share and `P_pair` was 0 in 105 pairs of 105. It does not fix names. The union share for names rose for every model, by 15.46 points on average. Still no model reached Faker, ranging from 88.9% to 97.1% against 99.74%. 58 of 105 pairs of name lists still collided. Usernames rose for 7 of 7 models, with 1 reaching Faker; companies rose for 6 of 7, with 2 reaching it. Integers were not helped: `P_pair` stayed between 93% and 100%.

4.5 Across models and generations

Of the 1379 distinct names returned in the `plain` arm, 163 (11.8%) came from 2 models or more; none came from all 7. The corresponding shares are 4.8% for companies, 2.1% for email addresses and 2.0% for usernames. Between the old and new model of each line,

`P_pair` on names was 69% for `claude-opus-5` against `claude-opus-5-5`, 13% for the sonnet pair and 0% for the haiku pair, which spans `claude-haiku-4-5` to `claude-haiku-5-5`. Most collisions therefore happen within a model rather than through 1 list every model draws from, though a few names recur widely: “Dmitri Volkov” turns up in the `plain` runs of 5 models of 7. A generational change can preserve a model’s list or replace it.

5. What it means for a seed

The engineering reading is direct. Within 1 model, 2 name fixtures of 30 built independently will very likely share a name. For usernames with 4 of 7 models and for email with 2 of 7, the rate is clearly above Faker, with the jackknife lower bound above it. There a `UNIQUE` constraint can fail intermittently. That is the worst kind of test failure, because it depends on which reply each fixture happened to get. With 3 of the other 5 models email collisions were rare or absent (0%, 2% and 7%); with `claude-fable-5-1` (20%) and `claude-haiku-4-5` (11%) the point estimate is above 10% but the lower bound reaches 0. The rates are for 30 values and larger fixtures give more chances to collide.

2 remedies are visible in the data. Asking for diversity explicitly removed email collisions in this panel. It did not remove name collisions, so for anything that must be unique the defensible choice is the generator rather than the model. A model can still be asked to write the call to the generator.

6. Threats to validity

The UUID type is a transport artifact, not a result. Tools were disabled and several models answered the UUID request with tool call markup instead of values: all 10 `plain` runs of `claude-sonnet-5-5` and `claude-fable-5-1` and 9 of 10 of `claude-haiku-4-5`. `claude-opus-5` wrote out a shell command as text in 25 of its 26 UUID replies. Some of those replies went on to print output for a command that could not have run, since tools were disabled. Non UUID lines therefore entered UUID sets in the main table of `report.py`. Restricted to valid v4 values, the only `plain` cell with collisions is `claude-sonnet-5` at 44.4% over 9 runs, carried by 2 published example UUIDs, `550e8400-e29b-`

41d4-a716-446655440000 in 6 runs and f47ac10b-58cc-4372-a567-0e02b2c3d479 in 3. The first is the URN example in the Wikipedia article on UUIDs (Universally unique identifier, revision 1381274818, “urn:uuid:550e8400-e29b-41d4-a716-446655440000”); the second is a v4 test vector in Google’s Go UUID library (google/uuid, uuid_test.go line 56). Both were checked on 21 October 2026. Neither appears in RFC 4122. claude-haiku-4-5 produced the same 2. Where UUIDs were returned, validity as v4 in the plain arm ran from 48% (claude-opus-5) to 98% (claude-sonnet-5). Through a bare API these results may be entirely different and the UUID type carries no conclusion here.

Pooled Wilson intervals are too narrow. Where this paper quotes a pooled count such as 256 of 315, the repository also prints a Wilson interval over pairs, here 76.6% to 85.2%. Pairs inside a cell share runs, so that interval understates uncertainty. The per cell jackknife is the primary interval and the verdicts rest on the direction across all 7 models rather than on 1 cell. The cell intervals are wide: claude-sonnet-5-5 on names spans 28% to 100%.

The n = 100 phase is incomplete and excluded. A second phase at 100 values per request was registered and started. It stopped at the weekly usage limit after 66 of 245 calls. Because the job order is deterministic, with the run index outermost, only run 0 of every cell exists, plus run 1 for email and names on all 7 models and for UUIDs on 3. That is at most 1 pair per cell, for 2 types on all 7 models and for uuid on 3, biased by type and by model. No number in this paper comes from it.

The CLI is not a bare API. Every call carries the Claude Code system prompt. That context also carried account identifiers: 1 reply among the invented tool calls reproduced the account email address, its GitHub handle (22 times) and a city matching the machine’s time zone, so that information reached the model. Temperature is the subscription default and thinking tokens were recorded rather than chosen.

Account context is an untested confound for the names. The most frequent name, Dmitri Volkov, shares its first name with the author, who owns the account the calls ran under. With the account context in the prompt, this study cannot tell whether that context promoted the name or whether the models favour it anyway; it appears in the plain runs of 5 models of 7, and across arms with all 7 and in 66 of 182 name runs. Removing that 1

string barely moves `P_pair` (section 4.1), so the rates do not rest on it. The test cannot show what the lists would be without the context. Which names top the list may depend on it. The same applies to email: `dmitri.volkov@example.net` is the second most frequent address (6 of 70 `plain` runs; removing it changes pooled email collisions from 59 to 58 of 315) and an address containing “dmitri” appears in 16 of 70 `plain` email runs and in 47 of 182 across arms. A run through the bare API without account context would settle it. That run was not done.

The registration was committed after counting had started. The registration and the prior work check were written at 01:12 on 19 October, before any counted call; their text at the commit is identical to that version. They were committed at 01:55 together with the harness; by then 52 counted replies of the main phase, run 0 of 52 cells, and 1 failed attempt were in the journal. The main phase was launched at 01:30:34 and its first reply was journaled at 01:32:17. Before 01:30 the only model calls were checks of the transport, none of them journaled or counted: 2 calls asking `claude-haiku-5-5` for 5 email addresses to compare CLI flags, the input size probe, a smoke run that hung on a quoting bug before writing anything and 2 single test calls asking for 5 usernames. Between the first counted call and the commit, the first 45 replies were read in previews of 90 characters and 3 rules were added to the code after that reading: a CLI start without its prompt is retried; a reply containing tool call markup counts as having no values and as a format violation; replies without values take no part in `P_pair`. `report.py` and `baseline.py` were first written at 01:32 and 01:33, after the first call. `report.py` was then run once on 49 counted calls; with 1 run per cell it could compute no `P_pair` and showed only 1 empty reply, 4 format violations and union shares of single runs. No `P_pair` and no baseline value was seen before the commit. Later, an interim report over 271 calls (03:04) showed `P_pair` for early cells. After it, at 13:58, the jackknife interval and the UUID pattern baseline were added to `report.py`. The jackknife is not in the registration. The amendment of 13:55 changed only the harness and fixed when the second phase starts. These times come from the commit history of the private workspace and its session records. The commit times are mirrored in the public repository’s history; the 01:12 write time cannot be checked from it.

Concurrency changed during the run. 631 calls ran with 4 concurrent processes before an overnight failure, first a hung call and then an expired OAuth session; the remaining 643 ran 1 at a time. The calls are independent processes and nothing in the design ties values to concurrency, but the split is recorded in the journal.

Scope. 1 lab, 7 models, 1 wording per type and arm, 30 values per request. Faker is 1 library in 1 locale; another generator or locale would give another baseline. Normalisation is lower case and whitespace only, so near duplicates such as `quietfalcon42` and `quiet_falcon42` count as distinct and the reported rates are, if anything, low. Of the 1274 replies, 77 contained no values at all and 148 (11.6%) broke the requested format in some way; both are counted in the repository and the empty ones take no part in `P_pair`.

7. Availability

The harness, the registration in its original language and in translation, the prior work check, every reply in full with prompts, timings and token usage, the run logs and the scripts that produce every number in this paper are in the repository. The analysis needs no network and no model.

Archived on Zenodo with DOI [10.5281/zenodo.23545744](https://doi.org/10.5281/zenodo.23545744).

8. References

1. Liwei Jiang, Yuanjun Chai, Margaret Li, Mickel Liu, Raymond Fok, Nouha Dziri, Yulia Tsvetkov, Maarten Sap, Alon Albalak, Yejin Choi. Artificial Hivemind: The Open-Ended Homogeneity of Language Models (and Beyond). arXiv preprint, 2025. arXiv:2510.22954. <https://arxiv.org/abs/2510.22954>
2. Yiming Zhang, Harshita Diddee, Susan Holm, Hanchen Liu, Xinyue Liu, Vinay Samuel, Barry Wang, Daphne Ippolito. NoveltyBench: Evaluating Language Models for Humanlike Diversity. arXiv preprint, 2025. arXiv:2504.05228. <https://arxiv.org/abs/2504.05228>
3. Tapan Parikh. The One-Word Census: Answer-Choice Conformity Across 105 Language Models. arXiv preprint, 2026. arXiv:2607.12796. <https://arxiv.org/abs/2607.12796>

4. Javier Coronado-Blázquez. Deterministic or probabilistic? The psychology of LLMs as random number generators. arXiv preprint, 2025. arXiv:2502.19965.
<https://arxiv.org/abs/2502.19965>
5. Tapan Parikh. Structured Output Collapses Answer Diversity Across 44 Language Models. arXiv preprint, 2026. arXiv:2607.18476. <https://arxiv.org/abs/2607.18476>
6. Jiayi Zhang, Simon Yu, Derek Chong, Anthony Sicilia, Michael R. Tomz, Christopher D. Manning, Weiyan Shi. Verbalized Sampling: How to Mitigate Mode Collapse and Unlock LLM Diversity. arXiv preprint, 2025. arXiv:2510.01171.
<https://arxiv.org/abs/2510.01171>
7. Kou Misaki, Takuya Akiba. String Seed of Thought: Prompting LLMs for Distribution-Faithful and Diverse Generation. arXiv preprint, 2025. arXiv:2510.21150.
<https://arxiv.org/abs/2510.21150>
8. Pulin Agrawal, Prasoon Goyal. Addressing LLM Diversity by Infusing Random Concepts. arXiv preprint, 2026. arXiv:2601.18053. <https://arxiv.org/abs/2601.18053>
9. Benoit Baudry, Khashayar Etemadi, Sen Fang, Yogya Gamage, Yi Liu, Yuxin Liu, Martin Monperrus, Javier Ron, André Silva, Deepika Tiwari. Generative AI to Generate Test Data Generators. arXiv preprint, 2024. arXiv:2401.17626.
<https://arxiv.org/abs/2401.17626>
10. Chengyi Wang and others. Fixturize: Bridging the Fixture Gap in Test Generation. arXiv preprint, 2026. arXiv:2601.06615. <https://arxiv.org/abs/2601.06615>
11. Anuj Sadani, Deepak Kumar. Locale-Conditioned Few-Shot Prompting Mitigates Demonstration Regurgitation in On-Device PII Substitution with Small Language Models. arXiv preprint, 2026. arXiv:2605.13538. <https://arxiv.org/abs/2605.13538>
12. Hitesh Laxmichand Patel, Amit Agarwal, Bhargava Kumar, Karan Gupta, Priyaranjan Pattnayak. LLM for Barcodes: Generating Diverse Synthetic Data for Identity Documents. arXiv preprint, 2024. arXiv:2411.14962. <https://arxiv.org/abs/2411.14962>
13. Shivasankari Kannan, Yeounoh Chung, Amita Gondi, Tristan Swadell, Fatma Ozcan. High-Fidelity And Complex Test Data Generation For Google SQL Code Generation Services. arXiv preprint, 2025. arXiv:2504.17203. <https://arxiv.org/abs/2504.17203>
14. Dmitriy Semenkevich. `fixture-entropy`: harness, registration, every reply in full and analysis scripts. GitHub, 2026. <https://github.com/dimhold/fixture-entropy>

CITE THIS

Semenkevich, D. (2026). 2 lists of 30 test names from the same model share a name in 64 to 98% of pairs, against 1.29% for Faker. dimhold.by. <https://doi.org/10.5281/zenodo.23545744>

LLM evaluation · test data generation · output diversity · mode collapse · test fixtures · database seeding · preregistration