A Staged Synthetic Instruction-Data Factory: Design Rationale and Viability for Dataset Engineers

Elias Matriano

Independent design paper for practitioners building supervised fine-tuning (SFT) corpora
Domain focus: synthetic conversational instruction data for small open-weight language models

Abstract

Synthetic instruction data has become a default lever for post-training small open-weight language models, yet industrial practice often reduces the problem to a single generate-and-dump step. This design paper synthesizes existing research into a staged *synthetic instruction-data factory* intended for dataset engineers: a pipeline that (i) constructs a constrained design space (taxonomy, variables, templates, compatibility rules, and sparse high-quality seeds), then (ii) generates conversational pairs under diversity-preserving batch isolation, filters them with hybrid automated quality control, samples human behavioral audit, and regenerates failures before compiling training-ready JSONL. We deliberately scope the discussion to the data factory only—excluding model training, preference optimization, and leaderboard chasing—and we make no empirical claims about throughput or quality gains. Instead, we map each stage to prior work on self-instruct pipelines, controllable generation, semantic deduplication, LLM-as-a-judge reliability, and model collapse under recursive synthetic training. We conclude that such a factory is **conditionally viable** for instruction-tuning corpora when human gates remain on combinatorial validity and behavioral correctness, when synthetic output is not treated as a substitute for all human signal, and when diversity controls are treated as first-class engineering requirements rather than afterthoughts. We provide an evaluation protocol that dataset engineers can run to falsify or support the design in their own domains.

Keywords: synthetic data, instruction tuning, dataset engineering, LLM-as-a-judge, data quality, model collapse, human-in-the-loop

1. Introduction

Dataset engineers who build supervised fine-tuning (SFT) corpora face a structural tension. Human-written instruction data remains the gold standard for behavioral fidelity, but it is expensive, slow to iterate, and difficult to scale across many intents, roles, and knowledge states. Large-scale synthetic instruction generation—popularized by bootstrapping methods such as Self-Instruct [1] and subsequent industrial recipes—reduces marginal cost but introduces well-documented failure modes: homogeneous style, invalid task compositions, near-duplicate flooding, hallucinated answers that look fluent, and, under recursive reuse, the risk of distributional degeneration associated with model collapse [2, 3].

This paper addresses a practical question for other synthetic-dataset engineers: *Under what design conditions is a multi-stage synthetic instruction-data factory viable for conversational SFT data aimed at small open-weight models?* We answer with a design synthesis, not a new empirical study. The contribution is an integrated architecture—hereafter the **factory**—that sequences design-space control before mass generation, and multi-layer quality control after it. The architecture is informed by a concrete pipeline specification used as a design artifact (taxonomy → templates → compatibility → golden seeds → batched generation → automated review → sampled human review → compilation with regeneration). Unverified operational estimates from informal notes are excluded; only structural claims remain.

We choose **conversational instruction data for small open-weight LLMs** as the domain focus. This setting is representative of portfolio- and product-facing fine-tunes (local RAG assistants, role-conditioned helpers, tool-adjacent chat) where engineers need controllable coverage more than web-scale pretraining volume, and where ChatML-style or message-list JSONL is the usual compile target.

Contributions.

- A two-phase factory model separating *design-space construction* from *generation-and-QC*, tailored to instruction SFT rather than general pretraining.
- A literature-grounded viability analysis that rates evidence strength per stage (Table 1), written for engineers deciding what to implement first.
- An explicit threat model (homogeneity, judge circularity, silent strata loss) and a falsifiable validation protocol without requiring claims of completed experiments.

2. Related Work

2.1 Synthetic instruction data

Self-Instruct [1] established the modern template: seed tasks, model-generated instructions and instances, heuristic filtering, and fine-tuning. Follow-on systems scale diversity via stronger teachers, multi-turn dialogue synthesis, and task taxonomies (e.g., Evol-Instruct-style complexity escalation [4] and open instruction collections such as those used in Alpaca-class pipelines [5]). For dataset engineers, the durable lesson is not a particular model name but the pipeline shape: **seeds** $\rightarrow$ **expand** $\rightarrow$ **filter** $\rightarrow$ **train**. Our factory keeps that shape but inserts an explicit combinatorial design stage before expansion, because uncontrolled expansion is where most production corpora become unmaintainable.

2.2 Quality filtering and judges

Automatic filters for spelling, length, and format are necessary but insufficient. Semantic near-duplicate detection via embeddings is standard practice when synthetic generators over-produce paraphrases. LLM-as-a-judge protocols [6] scale preference-like or rubric-like scoring, yet exhibit position, verbosity, and self-enhancement biases [6]. Surveys of LLM judges emphasize that automated labels are not free of human grounding [7]. Accordingly, factories that treat judges as sole authority are under-specified; hybrid pipelines with human sampling are the conservative engineering default.

2.3 Model collapse and synthetic feedback loops

Shumailov et al. [2] characterize model collapse as a degenerative process when successive generations train on model-generated content without adequate real signal, eroding distributional tails. Theoretical and empirical follow-ups study recursive training dynamics [3]. The implication for SFT factories is precise: synthetic conversational data is a *tool*, not a closed ecosystem. Golden seeds, human audits, and mixed real/synthetic corpora are not niceties; they are collapse-aware controls. Our design therefore budgets sparse human-authored and frontier-authored exemplars into each pattern and refuses pure self-play without external anchors.

2.4 Controllable attributes and combinatorial design

Instruction quality often fails not because a single sentence is poorly written, but because attribute combinations are incoherent (e.g., “novice user” paired with “assume PhD knowledge,” or creative ideation paired with rigidly formal legal tone). Controllable text generation and structured prompting literature supports factorizing generation along attributes [8]. Software testing’s combinatorial coverage methods [9] provide an engineering analogy: valid combinations must be enumerated or constrained before mass execution. The factory’s compatibility matrix is that constraint layer, reviewed by humans before CSV expansion.

3. Problem Formulation

Let a conversational SFT example be a tuple of messages (system, user, assistant) with optional metadata (domain, pattern_id, attribute vector). A factory is a stochastic process that maps a human-specified domain set and seed policy into a multiset of examples. We say the factory is **viable** for a use case if it can, in principle, produce corpora that simultaneously satisfy:

- **Coverage:** patterns and intents required by the product or research target are represented with explicit inventory, not accidental leftovers.
- **Validity:** attribute combinations respect approved compatibility rules; answers are behaviorally appropriate to the prompt's role and knowledge state.
- **Diversity:** surface form and pragmatic style do not collapse to a single teacher voice across thousands of rows.
- **Filterability:** low-quality rows are removable with reason codes that engineers can act on (regenerate, rewrite seed, split pattern).
- **Auditability:** humans can sample and score behavioral correctness without reading every row.
- **Compile readiness:** outputs land in stable interchange formats (e.g., JSONL/ChatML) with lineage fields (batch_id, pattern_id, source specification).

Viability here is *engineering viability*: whether the design is a rational response to known failure modes in the literature and practice—not whether a particular run beat a leaderboard.

4. Factory Architecture: Phase 1 — Design Space

Phase 1 produces a **constrained generation specification**, not training rows. This is the stage most generate-and-dump pipelines skip.

4.1 Taxonomy generation (P1)

Engineers define top-level domains (e.g., tutoring, customer support, coding help, research summarization). A strong model proposes clusters, patterns, and sub-variants; humans curate. This mirrors seed-task stewardship in Self-Instruct [1] but elevates taxonomy to a durable artifact: the inventory against which coverage is measured. Without it, synthetic volume obscures missing intents.

4.2 Variable and value mining (P2)

For each pattern, the factory enumerates a parameter space: intent, user role, tone, knowledge state, constraints, output format, and similar axes. AI can propose value inventories; humans edit. Factorized attributes enable later recombination and make failures diagnosable ("tone values are fine; knowledge-state values are broken").

4.3 Template generation (P3)

Templates bind variables into prompt scaffolds for the generator. Templates are not the final user text; they are the controlled interface between combinatorial design and natural language realization. This separation allows regenerating surface text without redesigning the space.

4.4 Pattern split under risk (P3.4)

Some patterns encode incompatible sub-behaviors in one bag (e.g., "creative brainstorming" mixed with "strict compliance answers"). Splitting high-risk patterns reduces multi-mode collapse inside a single compatibility matrix and makes human review rubrics clearer. This is a dataset refactoring step analogous to separating mixed test suites.

4.5 Compatibility matrix analysis (P3.5)

An AI proposes valid/invalid combinations; humans approve. Rules such as "if intent=creative then tone ≠ rigidly formal" prune the Cartesian product before expansion. Scoping matrices to a single pattern (e.g., on the order of tens of samples per batch later) keeps review cognitively tractable. This stage is the factory's primary defense against combinatorial explosion and nonsense compositions.

4.6 Golden seeds (P4)

A small fraction of rows per pattern are reserved for high-quality exemplars: human-authored "professor-grade" samples and frontier-model samples, mixed at a fixed policy (design default: human:frontier = 1:2) and inserted evenly through the batch plan. Seeds act as local attractors for style and correctness, echoing the seed-task role in [1]

and the broader observation that few high-quality demonstrations strongly condition generation. Under collapse-aware reading of [2], seeds also reintroduce non-recursive signal into an otherwise synthetic loop.

4.7 Combination computation (P4.5)

Programmatic expansion merges variables, values, compatibility, and seed placement into a specification table (CSV or equivalent): each row is a fully specified generation job with metadata. Only after this artifact exists does Phase 2 spend model tokens on natural language. Lineage begins here: every future JSONL record should point back to a specification row.

5. Factory Architecture: Phase 2 — Generation and Quality Control

5.1 CSV to natural language under airgapped batches (P5)

Specification rows are divided into small airgapped batches (design default: 50 rows per pattern batch). Each batch is generated in a fresh model context with system-prompt variants and per-row variation instructions. The intent is to interrupt the well-known tendency of long shared contexts and single-session dumps to standardize tone. Outputs are written as conversational pairs (e.g., ChatML/JSONL) with metadata: source_csv, batch_id, pattern_id. Airgapping is a diversity engineering control motivated by homogeneity risks discussed in synthetic-data collapse literature [2, 3]; it is not a proof of diversity by itself.

5.2 Automated quality review (P6)

Parallel automated QC applies a stack: (i) defect filters (spelling, redundancy, empty or truncated outputs); (ii) embedding-based near-duplicate clustering; (iii) LLM-as-judge checks for hallucination markers and behavioral mismatches relative to the specification. Passes and fails land in separate streams with reason codes. Given known judge biases [6, 7], reason codes and multi-signal fusion matter more than a single scalar “quality” score. Automation can run alongside generation of the next batch; the scientific point is layered filtering, not latency.

5.3 Stratified human evaluation (P7)

Humans do not re-read the entire corpus. A stratified sample (design default: every 10th item in a 50-sample batch, i.e., 10% per batch, tied to pattern_id) enters a rubric UI (e.g., Argilla or Label Studio class tools). The rubric prioritizes **behavioral correctness**—did the assistant respect role, constraints, and knowledge state?—over surface polish. This stage acknowledges that judges miss classes of errors humans catch, and that full human labeling is usually unaffordable. The sampling rate is a tunable parameter requiring calibration (Section 8); it is not a universal optimum.

5.4 Compilation and regeneration loop (P8)

Approved rows compile into a training-ready file. Automated and human rejects are merged, grouped by pattern and source specification, and requeued to P5 with adjusted parameters (stricter constraints, alternate seeds, or split patterns). The loop continues until coverage and pass criteria are met or a pattern is escalated for redesign. This is rejection sampling plus organizational memory: failures improve the factory, not only a single row.

Figure 1 (textual). End-to-end data factory (P1–P8).

Phase 1: Domains → taxonomy → variables/values → templates → risk splits → compatibility approval → golden seeds → specification CSV. Phase 2: Airgapped NL generation → hybrid auto-QC → stratified human rubric → compile; rejects regenerate via P5. Training, DPO, and benchmarks are out of scope.

6. Viability Analysis

Table 1 summarizes, for dataset engineers, which risks the factory targets and how strongly existing literature supports the mitigation strategy. “Evidence strength” refers to external research and established practice—not

measurements from this paper.

Stage	Primary risk	Mitigation in design	Evidence strength
Taxonomy / templates	Narrow or redundant task coverage	Human-curated domains; AI proposes, human accepts	Strong (instruction diversity literature)
Compatibility matrix	Invalid attribute combos; combinatorial explosion	Pattern-scoped rules; human approval before expand	Moderate (controllable generation + combinatorial testing analogy)
Golden seeds	Style drift away from high-quality targets	Inject sparse human + frontier exemplars per pattern	Strong (few-shot / seed influence; Self-Instruct seeds)
Airgapped batch generation	Homogeneous tone across a large dump	Small batches; new sessions; prompt variants	Moderate (diversity / collapse literature; limited direct ablations)
Automated QC	False accepts; judge bias; near-duplicates	Rules + embeddings + LLM judge; reason codes	Mixed (judges useful but biased; need human audit)
Stratified human review	Behavioral errors miss automated filters	Sampled rubric review with pattern tracking	Strong in principle; sampling rate is unvalidated
Regen / compile loop	Silent loss of failed strata; waste	Route rejects by pattern; regenerate; append passes	Moderate (rejection sampling / iterative refine)

Table 1. Stage-wise risks, design mitigations, and evidence strength.

6.1 What the literature supports well

Seeded expansion with filtering is a proven pattern for instruction data [1, 4, 5]. Hybrid quality control is necessary because neither heuristics nor LLM judges are sufficient alone [6, 7]. Collapse-aware caution against pure synthetic recursion is well motivated [2, 3]. Together these imply that a factory with seeds, filters, and human gates is a rational default architecture for conversational SFT data.

6.2 What remains weakly evidenced

Three design choices are plausible but under-specified by direct ablations in public literature: (1) exact airgap batch sizes and session-reset policies; (2) the optimal human sampling fraction per pattern; (3) the quantitative contribution of compatibility matrices versus simpler attribute randomization. Engineers should treat these as knobs to measure (Section 8), not as settled science.

6.3 Conditional viability statement

We assess the factory as **conditionally viable** for synthetic conversational instruction data when all of the following hold: (a) humans retain approval rights over taxonomy and compatibility; (b) golden seeds continuously inject non-recursive quality signal; (c) automated QC is multi-signal and reason-coded; (d) stratified human review targets behavioral failure modes; (e) rejects re-enter generation rather than disappearing; and (f) synthetic rows are not assumed to replace all human data in eventual training mixtures. Absent these conditions, the architecture degrades toward generate-and-dump and inherits the failure modes the literature already documents.

7. Threat Model and Limitations

- **Judge circularity:** if the same model family generates and judges, shared blind spots can pass systematically [6, 7]. Mitigation: separate judge family where possible; keep humans on behavioral strata.
- **Homogeneity despite airgaps:** a single teacher model can still imprint style across batches. Mitigation: multi-teacher generation, stronger seed diversity, explicit style axes.
- **Specification errors:** a wrong compatibility matrix scales failure at P5–P8 speed. Mitigation: small pilot batches before full expand.
- **Sampling blindness:** stratified 10% review can miss rare catastrophic patterns. Mitigation: risk-weighted sampling for safety-critical domains.

- **License and provenance:** frontier seeds and teacher outputs carry policy and copyright constraints; factories must record provenance for compliance.
- **Scope limit:** this paper does not validate downstream fine-tune quality, preference tuning, or benchmark transfer; those are separate evaluation problems.

This document is a design synthesis. It does not report new human-subject experiments, inter-annotator agreement studies, or controlled ablations. Any implementation metrics must be measured in situ.

8. Recommended Validation Protocol for Engineers

To move from design to evidence without over-claiming, we recommend a minimal protocol:

- **Pilot scale:** 3–5 patterns, 200–500 compiled rows each condition.
- **Ablate design-space controls:** with vs. without compatibility matrix; with vs. without golden seeds.
- **Ablate diversity controls:** single long session vs. airgapped batches; single system prompt vs. variants.
- **Ablate QC:** heuristics only vs. heuristics+embeddings vs. full stack; measure precision/recall against a fully human-labeled holdout of ≥200 rows.
- **Human sampling calibration:** estimate error detection curves at 5%, 10%, 20% strata to choose an operating point.
- **Diversity metrics:** embedding entropy / average nearest-neighbor similarity; n-gram novelty; attribute coverage histograms.
- **Behavioral metrics:** rubric pass rate by pattern; constraint-violation rate; hallucination rate on knowledge-grounded prompts.
- **Reporting:** publish reason-code distributions and lineage completeness, not only mean judge scores.

Only after these measurements should engineers assert quantitative gains. The factory's value proposition is that each ablation has a natural on/off switch—something unstructured dumps lack.

9. Practical Guidance for Dataset Engineers

- Invest Phase 1 time before buying Phase 2 tokens; invalid combinations are cheaper to delete as rules than as JSONL rows.
- Store metadata (pattern_id, batch_id, seed_flag, reject_reason) as first-class fields; debugging corpora is a data problem, not only a model problem.
- Separate "surface quality" from "behavioral correctness" in rubrics; fluent wrongness is the expensive failure mode.
- Treat regeneration as part of the happy path; a factory without a reject sink will silently bias coverage toward easy patterns.
- Keep a non-synthetic anchor set for periodic distribution checks against collapse-like narrowing [2].

10. Conclusion

For engineers building synthetic conversational instruction data for small open-weight models, a staged factory is a coherent response to known failure modes of generate-and-dump pipelines. By constructing a human-approved design space, injecting golden seeds, generating under airgapped diversity controls, filtering with hybrid automated QC, auditing behavior on stratified samples, and regenerating failures with lineage, the factory aligns operational practice with lessons from Self-Instruct-style expansion [1], LLM-as-a-judge research [6, 7], and model-collapse warnings [2, 3]. We claim conditional viability—not empirical superiority. The appropriate next step is not rhetorical confidence but the validation protocol in Section 8, run on the domains that matter to each team.

References

[1] Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. Self-Instruct: Aligning Language Models with Self-Generated Instructions. In Proceedings of ACL 2023. arXiv:2212.10560.

[2] Shumailov, I., Shumaylov, Z., Zhao, Y., Gal, Y., Papernot, N., and Anderson, R. AI models collapse when trained on recursively generated data. Nature, 631, 755–759, 2024. DOI:10.1038/s41586-024-07566-y.

[3] Seddik, M. E. A., Chen, S., Hayou, S., Youssef, P., and Debbah, M. How bad is training on synthetic data? A statistical analysis of language model collapse. arXiv preprint / OpenReview, 2024.

[4] Xu, C., Sun, Q., Zheng, K., Geng, X., Zhao, P., Feng, J., Tao, C., and Jiang, D. WizardLM: Empowering Large Language Models to Follow Complex Instructions. arXiv:2304.12244, 2023. (Evol-Instruct.)

[5] Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. Stanford Alpaca: An Instruction-Following LLaMA Model. GitHub repository / technical report, 2023.

[6] Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In NeurIPS 2023 Datasets and Benchmarks. arXiv:2306.05685.

[7] Gu, J., et al. A Survey on LLM-as-a-Judge. arXiv preprint (survey), 2024–2025. See also community surveys on judge biases and preference leakage.

[8] Zhang, H., Song, H., Li, S., Zhou, M., and Song, D. A Survey of Controllable Text Generation using Transformer-based Pre-trained Language Models. ACM Computing Surveys, 2023. (Representative survey for attribute-controlled generation.)

[9] Kuhn, D. R., Kacker, R. N., and Lei, Y. Introduction to Combinatorial Testing. CRC Press, 2013. (Engineering analogy for constrained combination coverage.)

[10] Ouyang, L., et al. Training language models to follow instructions with human feedback. NeurIPS 2022. (InstructGPT; human preference signal as complement to synthetic expansion.)

[11] Zhou, C., et al. LIMA: Less Is More for Alignment. NeurIPS 2023. (Quality-over-quantity motivation for careful seed and curation policies.)

[12] Gunasekar, S., et al. Textbooks Are All You Need. arXiv:2306.11644, 2023. (High-quality synthetic textbooks; reinforces seed/quality emphasis.)

Disclosure. This paper is a literature-grounded design synthesis for practitioners. It does not present new experimental results. Pipeline stage labels (P1–P8) refer to a design artifact for organizing engineering work on synthetic instruction corpora.