# SynAE: A Framework for Measuring the Quality of Synthetic Data for Tool-Calling Agent Evaluations

**Anonymous authors**
Paper under double-blind review

## Abstract

Today, tool-calling agents are commonly evaluated or tested on static datasets of *execution traces*, including input commands, agent responses, and associated tool calls. However, internal production datasets are often insufficient or unusable for testing; for example, they may contain sensitive or proprietary data, or they may be too sparse to support comprehensive testing (especially pre-deployment). In these settings, practitioners are increasingly replacing or augmenting real datasets with synthetic ones for evaluation purposes. A key challenge is quantifying the relation between these synthetic datasets and the real data. We introduce SynAE, an evaluation framework for assessing how well synthetic benchmarks for multi-turn, tool-calling agents replicate and augment the characteristics of real data trajectories. SynAE assesses the validity, fidelity, and diversity of synthetic data across four metric categories: (i) task instructions and intermediate responses, (ii) tool calls, (iii) final outputs, and (iv) downstream evaluation. We evaluate SynAE using recent agent benchmarks and test common synthetic data failure modes via realistic and controlled generation schemes. SynAE detects fine-grained variations in data validity, fidelity and diversity, and shows that no single metric is sufficient to fully characterize synthetic data quality, motivating a multi-axis evaluation of synthetic data for agent testing. A demo of SynAE is available at https://synae-2026-synae-demo.static.hf.space/index.html, with code at https://github.com/synae-cais-2026/synae-code.

## 1 Introduction

Agent evaluation and testing is a nascent but critical component of pre-deployment processes for production agentic workflows (Pan et al., 2025; Anthropic, 2026). Today, tool-calling agent evaluations are often (but not always) run on static *baseline datasets*[1] consisting of a trace generated from agent interactions; these typically include user inputs, tool calls, intermediate interactions with the agent, and a final output (Yehudai et al., 2025; Mohammadi et al., 2025). Such datasets are often collected from real-user interactions with an environment and/or synthesized by measuring scripted interactions with a dynamic environment. While the design of proper agent evaluations is its own active research area (Kapoor et al., 2025; Alonso & Church, 2025; Zhu et al., 2025), common evaluation metrics measure whether the agent selects the correct tools and produces the desired final output (Yehudai et al., 2025; Mohammadi et al., 2025; Anthropic, 2026).

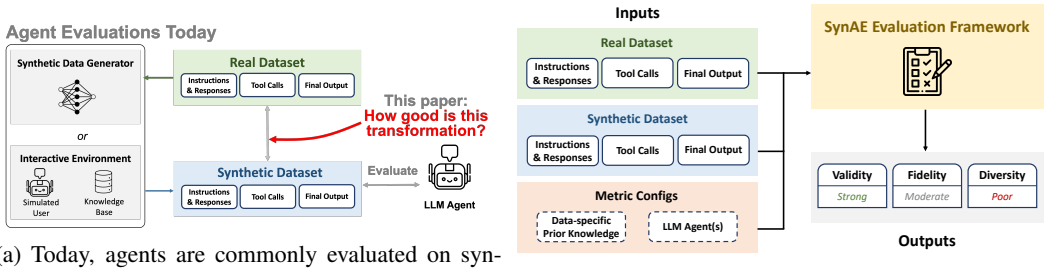
In many practical situations, existing baseline datasets cannot be directly used or are insufficient for agent evaluation, either because they contain sensitive user data (e.g., emails, travel details) subject to privacy restrictions (Tamkin et al., 2024; CapitalOne, 2022) or because they are too small for comprehensive testing (Pan et al., 2025). Practitioners therefore increasingly use *synthetic datasets* as replacements for or augmentations to real execution traces in evaluation pipelines (Figure 1a) (Qin et al., 2023; Tang et al., 2023; Iskander et al., 2024). Synthetic data may be generated by directly synthesizing trajectories or by synthesizing inputs to an interactive environment.

Although synthetic datasets are increasingly used for agent evaluation, typical workflows lack systematic quality checks against the real baseline data. Indeed, **the current literature on testing**

[1] These evaluation datasets are also commonly called *benchmarks*; we use both terms interchangeably.



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.




(a) Today, agents are commonly evaluated on synthetic datasets modeled after a real dataset of agent trajectories, e.g., for privacy or augmentation reasons. However, it is often unclear whether these synthetic datasets are representative of the real dataset. Our evaluation framework, SynAE, measures the semantic and structural *similarity* between real and synthetic datasets of execution trajectories, and the *validity* and *diversity* of the synthetic data.

(b) The SynAE evaluation pipeline takes as input real and synthetic datasets (task completion trajectories), along with optional metric configurations that may specify data-specific prior knowledge and one or more LLM agents. The LLM agents are used only to compare downstream performance between real and synthetic input datasets. The framework evaluates synthetic data along three dimensions: validity, fidelity, and diversity.

Figure 1: The SynAE framework evaluates the quality of synthetic data used in agent evaluations.

**agents with synthetic data provides almost no quantitative methods for evaluating the quality of such synthetic data,** leaving operators with little visibility into evaluation gaps.

In this work, we develop a comprehensive evaluation framework, SynAE, to assess how well synthetic trajectories *replicate* and *augment* the characteristics of real data trajectories, including task instructions and responses, and the associated reference tool calls and outputs. As illustrated in Figure 1b, SynAE takes as input a real baseline dataset, a synthetic dataset, and optional metric configurations that may specify data-specific prior knowledge and one or more LLM agents; the input agent(s) need not be the same as the one being evaluated—each agent input to SynAE is used purely to compare downstream performance between real and synthetic input datasets.

SynAE evaluates synthetic data along three properties: (1) validity, (2) fidelity, and (3) diversity. These properties follow the intended use: validity is a prerequisite for usable trajectories, fidelity is central when synthetic data replaces or replicates a reference dataset, and diversity is central when synthetic data augments coverage. The suite is an operational taxonomy rather than a claim of theoretical exhaustiveness. For *validity*, we evaluate whether synthetic tool calls and outputs successfully fulfill the given instructions, using LLM-as-a-judge by default or rule-based checkers when available. We measure *fidelity* by assessing how similar the real and the synthetic data are; specifically, we compute metrics on both real and synthetic data and evaluate their similarity. We quantify the *diversity* of synthetic data with entropy-based metrics for various representations of the dataset. For each property, SynAE computes a suite of metrics to evaluate (a subset of): (1) task instructions and responses, (2) tool calls, (3) final outputs, and (4) downstream evaluation.

We demonstrate the utility of SynAE with three recent agent benchmarks playing the role of the "real dataset" in Fig. 1a: T1 (Chakraborty et al., 2025), BFCL (Patil et al., 2025), and ACP (Kokel et al., 2025). We then construct synthetic datasets using NVIDIA NeMo NVIDIA (n.d.), an industry-standard synthetic data tool, as well as custom synthetic data generation methods designed to simulate common real-world pitfalls, such as degraded data fidelity and limited diversity. Our experiments show that SynAE captures fine-grained variations in the validity, fidelity, and diversity of synthetic task-completion trajectories. They further suggest that no single metric is sufficient to fully characterize synthetic data quality, underscoring the need for SynAE. Overall, we view SynAE as a plug-in component for agentic workflows, allowing operators to automatically evaluate the quality of synthetic benchmark datasets, diagnose their shortcomings and compare candidate interventions.

### 1.1 Related Work

We provide a brief overview of related works, and provide a detailed discussion in §A.

Prior work evaluates synthetic benchmarks for LLMs through task difficulty, realism, and preservation of model rankings Gill et al. (2025); Maheshwari et al. (2024); Xiong et al. (2025); Majurski & Matuszek (2025). These studies mainly consider standard NLP settings rather than agent benchmarks, where evaluation is harder because they involve multi-step decisions, tool interaction, and

failure attribution across the agent pipeline. Recent evaluations on synthetic benchmarks for tool-calling agents have largely focused on single-turn settings or instruction quality Shen et al. (2024); Iskander et al. (2024); Zhu et al. (2025); Alonso & Church (2025); Pan et al. (2025), without systematically assessing reference tool calls, outputs, or multi-step dependencies.

Multi-turn tool-calling agents are commonly evaluated using end-to-end task success on state-based checkers (Yao et al., 2024; Liu et al., 2023; Zhou et al., 2023; Yang et al., 2023; Xie et al., 2024; Jimenez et al., 2023), response-based criteria (Patil et al., 2025; Li et al., 2025; Zhang et al., 2026), or LLM-as-a-judge (Qin et al., 2023; Pan et al., 2024; Xue et al., 2025; Lù et al., 2025). These methods evaluate agent performance on a given benchmark, whereas our goal is to evaluate the benchmark itself by providing quantitative metrics for validity, fidelity, and diversity across instructions and responses, tool calls, final outputs, and downstream evaluation.

## 2 SynAE Framework

The SynAE framework assesses how well synthetic agent trajectories replicate and augment the characteristics of a real dataset. It quantifies the *validity*, *fidelity*, and *diversity* of synthetic data.

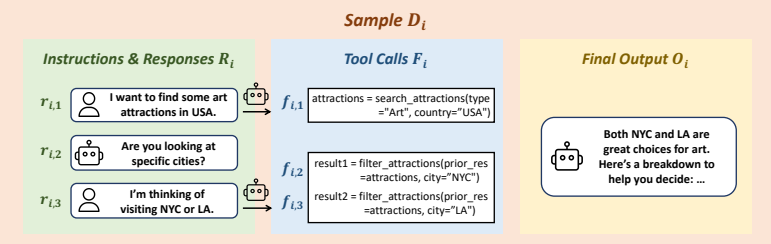



Figure 2: Agent trajectory from T1 Chakraborty et al. (2025) benchmark dataset, with notation for each component.

**Notation and setup** Consider a dataset $\mathcal{D} = \{D_i\}_{i=1}^{m}$ of $m$ samples (or agent trajectories). Figure 2 shows a sample trajectory $D_i$ comprising instructions and responses $R_i$, tool calls $F_i$, and a textual output $O_i$. The instruction-response sequence is $R_i = (r_{i,1}, \ldots, r_{i,\ell_i})$, where $\ell_i$ is its length. The tool-call sequence is $F_i = (f_{i,1}(\varphi_{i,1}), \ldots, f_{i,q_i}(\varphi_{i,q_i}))$, where $f_{i,j} \in \mathcal{F}$ is an executable function from the tool set $\mathcal{F}$, $\varphi_{i,j}$ is its input, and $q_i$ is the number of tool calls in the trajectory. Because these components can interleave in time, in a slight abuse of notation, $D_i$ denotes the time-ordered sequence of events, while $R_i$, $F_i$, and $O_i$ denote their corresponding filtered subsequences.

**Summary of inputs** SynAE takes as input (1) a real dataset $\mathcal{D}$; (2) a synthetic dataset $\mathcal{D}'$; and (3) optional metric configurations that may specify data-specific prior knowledge and agents $A_1, \ldots, A_h$ (details below). Both datasets contain instructions and responses, reference tool calls and outputs; SynAE also supports datasets missing responses, tool calls, or outputs.

### 2.1 Evaluation Metrics

SynAE organizes its metrics into validity (§2.1.1), fidelity (§2.1.2), and diversity (§2.1.3). Each pillar contains sub-metrics with different prior-knowledge requirements and evaluate different aspects of the synthetic dataset: task instructions and responses, tool calls, outputs, and downstream tasks.

#### 2.1.1 Validity Metrics

Validity is important to assess since synthetic data can appear faithful yet be unusable if the tool calls or final outputs fail to complete the task. In practice, invalid samples may arise from hallucinated tool names or arguments, or from plausible-looking outputs that do not satisfy the instructions.

While validity may be dataset-specific, we use a broad definition: whether the provided tool calls or outputs accomplish the task instruction, serving as a basic self-consistency check. By default, we assess validity for each sample $D_i \in \mathcal{D}$ with an LLM-as-a-judge, whose prompt and agreement with human annotations are reported in §B; users may instead define rule-based checkers when available. We report the overall **Validity Rate (VR)** as the proportion of valid tool-call sequences and outputs.

### 2.1.2 FIDELITY METRICS

We design fidelity metrics to measure similarity between synthetic and real data, which is especially useful when synthetic data is used as a direct replacement for real data, e.g., due to privacy constraints. Fidelity is the only property in `SynAE` that uses real data to compute its associated metrics, covering (1) task instructions and responses, (2) tool calls, (3) outputs, and (4) downstream tasks.

**1. Fidelity metrics for task instructions and responses** Task instructions and responses have an alternating structure, with each instruction followed by a response. Due to this, we draw inspiration from StructBench (Wang et al., 2025), a framework for evaluating structured synthetic data, and use metrics below.

For fidelity metrics that capture the structural relations in the data, we use *Key Node Dependency* and *Attribute Match*. **Key Node Dependency (KND)** measures semantic dependencies between parts of each sample by computing embedding cosine similarities between each instruction and its corresponding response, and between each response and the subsequent instruction. We then compare real and synthetic datasets by measuring the distributional distance between their similarity-score distributions. **Attribute Match (AM)** measures how closely real and synthetic datasets align on predefined statistical and semantic attributes by computing distributional distances over their attribute distributions, using Wasserstein-2 distance for numerical attributes and Total Variation (TV) distance for categorical ones. We consider attributes including the number of instruction turns, instruction and response token lengths, and dataset-specific semantic properties (specified in Table 1).

For non-structural fidelity metrics, we use **KNN-Precision** and **KNN-Recall** to measure the semantic quality and coverage. KNN-Precision (respectively, KNN-Recall) is the fraction of synthetic (respectively, real) samples whose embedding distance to a real (respectively, synthetic) sample is smaller than the distance to the $k$-th nearest neighbor within their own distribution. We also use **Fréchet Inception Distance (FID)** to measure semantic closeness between real and synthetic data.

**2. Fidelity metrics for tool calls** We evaluate whether synthetic tool-usage and planning patterns match the real dataset. Each metric computes a distributional distance between real and synthetic data for a specific property to compare structural patterns; while semantic correctness is handled by trajectory-level validity, output, and downstream checks. (1) **Tool Usage Match (TUM)** compares overall tool-usage patterns, $\text{TUM} = \text{Dis}(\omega_f, \omega'_f)$, where $\omega_f$ and $\omega'_f$ denote the tool-usage distributions in the real and synthetic data, respectively, and Dis is the TV distance. (2) **Tool Call Number Match (TCNM)** compares the distributions of tool-call counts $q_i$ per sample $D_i$, $\text{TCNM} = \text{Dis}(\omega_q, \omega'_q)$, where Dis is the Wasserstein-2 distance and $\omega_q$ is the distribution over the tool-call number. (3) **k-Step Tool Planning Match (k-Step Planning)** measures tool-planning similarity by comparing conditional distributions of the next tool given the previous $k-1$ tool calls, i.e., $\omega_{f|f_1 \cdots f_{k-1}}, \forall f_1 \cdots f_{k-1} \in \mathcal{F}^{k-1}$. It is a weighted sum of TV distances between real and synthetic conditional distributions:

$$\text{k-Step Planning} = \sum_{f_1 \cdots f_{k-1} \in \mathcal{F}^{k-1}} n_{f_1 \cdots f_{k-1}} \cdot \text{Dis}\left(\omega_{f|f_1 \cdots f_{k-1}}, \omega'_{f|f_1 \cdots f_{k-1}}\right),$$

where $n_{f_1 \cdots f_{k-1}}$ is the number of $k$-step tool-call sequences in the real data with prefix $f_1 \cdots f_{k-1}$. We set $k \in \{1, 2\}$ by default and allow larger values for long-horizon tasks.

**3. Fidelity metrics for outputs** We evaluate the task fidelity of the final outputs by measuring their **KNN-Precision**, **KNN-Recall**, and **FID** relative to the real dataset.

**4. Fidelity metrics for downstream evaluation** To assess the downstream utility of a benchmark constructed from synthetic data, we compare agent performance on real and synthetic tasks, measuring whether the two benchmarks induce similar tool-call and output-generation behavior. Recall that the user may optionally provide one or more agents $A_1, \ldots, A_h$ for the downstream evaluation; these agents can differ from the one being evaluated in Fig. 1a. For each agent $A_j$ and reference trajectory $D_i$, the agent generates tool calls at each instruction turn, conditioned on the current instruction, prior instruction turns, and the history of executed tool calls and results, using the dataset-specific tool set. The final output is then generated from the full instruction and executed tool-call history.

We adopt two downstream metrics. **(1) Task Difficulty Difference (TDD)** measures the average absolute difference in task-completion performance across agents between real and synthetic tasks.

For each reference trace $D_i$ from either dataset and each input agent $A_j$, we generate an agent trace $\tilde{D}_{i,A_j}$ and use an LLM-as-a-judge to determine whether $D_i$ and $\tilde{D}_{i,A_j}$ are functionally equivalent (prompt in §C). We then compute the absolute difference in task success rates for tool-call selection or final-output generation between the real and synthetic benchmarks, averaged across agents. **(2) Ranking Divergence (RD)** measures whether agent performance rankings are preserved across the real and synthetic datasets by computing the Spearman rank correlation between rankings based on tool-call or output correctness. Lower TDD and higher RD indicate better downstream utility.

#### 2.1.3 Diversity Metrics

Diversity is particularly important when synthetic data is used to improve real-dataset coverage. For example, the real dataset may be collected in a controlled setting with little representation of outliers or real-world user requests; practitioners may thus add synthetic outliers to test product robustness.

Measuring diversity is challenging: without prior domain-specific knowledge, it is unclear whether a dataset adequately covers the (high-dimensional) space of data traces. We therefore use two reference-free metrics, Vendi Score and Attribute Diversity, which do not require access to real data, though users of `SynAE` can compare the diversity of real and synthetic data if desired.

**Vendi Score (Vendi)** Friedman & Dieng (2022) is widely used to measure diversity in machine learning, including data curation and generative modeling Jiang et al. (2025); Wu et al. (2023). It computes diversity from a similarity matrix $\mathbf{K}$: for instructions & responses and final outputs, $\mathbf{K}_{i,j}$ is the cosine similarity between embeddings of $R_i, R_j$ or $O_i, O_j$, respectively, using `text-embedding-3-small` in our experiments. For tool calls, we define $\mathbf{K}_{i,j} = 1 - \frac{\text{Levenshtein}(F_i,F_j)}{\max(q_i,q_j)}$, where $q_i$ is the number of tool calls in $F_i$, and we adopt Levenshtein distance to measure sequence dissimilarity. Vendi then computes the exponential entropy of the eigenvalues $\lambda_1, \ldots, \lambda_m$ of the normalized matrix $\mathbf{K}/m$: $\text{Vendi} = \exp\left(-\sum_{i=1}^{m} \lambda_i \log \lambda_i\right)$. Higher Vendi Score indicates more dissimilar samples and thus greater diversity; it is upper-bounded by $m$, achieved when all pairwise similarities are zero. Vendi Score is agnostic to dataset format and content but may be harder to interpret. For users with domain knowledge, we also include Attribute Diversity, which is more interpretable but requires specifying attributes of interest.

**Attribute Diversity (AD)** measures instruction-response diversity using user-specified attributes that capture dimensions the operator wants to diversify, such as attraction type and city in T1 or anomalous vs. benign behavior in a computer security context. Each sample is assigned, by a human or LLM, to one attribute-value combination, e.g., ("attraction type=museum", "city=Austin"). We then compute the entropy of the resulting attribute distribution: $AD = -\sum_{i=1}^{C} p_i \log p_i$, where $p_i$ is the proportion of samples in attribute combination $i$, and $C$ is the number of possible combinations. Higher AD indicates that no single attribute value dominates the population, reflecting greater diversity; it is upper-bounded by $\log C$, attained when all combinations are equally represented.

`SynAE` metrics and examples of dataset-specific prior knowledge are summarized in Table 1.

## 3 Experiments

We demonstrate `SynAE` on the following baseline datasets and synthetic data generation methods.

**Evaluation datasets** We treat T1 (Chakraborty et al., 2025), BFCL (Patil et al., 2025), and ACP (Kokel et al., 2025) as real data. **T1** contributes 225 multi-turn attraction-recommendation samples with instructions, reference tool calls, and final outputs. **BFCL** contributes 200 BFCL-V3-Base-Multi-Turn samples with reference tool calls in domains including file operations, mathematical calculation, and travel booking, with reference tool calls. **ACP** contributes 260 ACPBench-Applicability&Progression samples spanning planning domains such as transportation and robot actions. Table 1 summarizes the metrics across dataset, and §D reports computational costs.

**Synthetic data generation** To evaluate the **sensitivity** of `SynAE` to controlled changes in synthetic data quality, we modify original trajectories with interpretable algorithms: *Blank Filling*, *Oversampling*, and *In-Context Generation* vary both fidelity and diversity, while *Invalidation* varies validity. To evaluate `SynAE` on **industry-standard** synthetic data techniques, we use NVIDIA NeMo NVIDIA (n.d.), a synthetic data tool that is used in part to test agents.

Table 1: Evaluation metrics across datasets. Metrics in orange require dataset-specific prior knowledge; "—" indicates no input is used.

| Evaluation Aspect | Evaluation Target | Metric | T1 Instantiation | BFCL Instantiation | ACP Instantiation |
|---|---|---|---|---|---|
| **Validity** | Tool Call | Validity Rate | LLM-as-a-judge | LLM-as-a-judge | N/A |
| | Output | Validity Rate | LLM-as-a-judge | N/A | LLM-as-a-judge |
| **Fidelity** | Instruction | Key Node Dependency | 1. (instruction, response) pair<br>2. (response, instruction) pair | (instruction, instruction) pair | (context, instruction) pair |
| | | Attribute Match | 1. number of instructions<br>2. instruction token length<br>3. response token length<br>4. city<br>5. attraction type | 1. number of instructions<br>2. instruction token length<br>3. task domain<br>4. task subdomain | instruction token length |
| | | KNN-Precision | — | — | — |
| | | KNN-Recall | — | — | — |
| | | FID | — | — | — |
| | Tool Call | Tool Usage Match | — | — | N/A |
| | | Tool Call Number Match | — | — | N/A |
| | | k-Step Planning | $k \in \{2, 3\}$ | $k \in \{2, 3\}$ | N/A |
| | Output | KNN-Precision | — | N/A | — |
| | | KNN-Recall | — | N/A | — |
| | | FID | — | N/A | — |
| | Downstream Task | Task Difficulty Difference | 1. Tool Call<br>2. Output | Tool Call | Output |
| | | Ranking Divergence | 1. Tool Call<br>2. Output | Tool Call | Output |
| **Diversity** | Instruction | Vendi Score | — | — | — |
| | | Attribute Diversity | (city, attraction type) pair | (domain, subdomain) pair | planning domain |
| | Tool Call | Vendi Score | — | — | N/A |
| | Output | Vendi Score | — | N/A | — |

**(1) *Blank Filling*:** We randomly mask tokens in the original instructions with probability $p$ and prompt a language model to fill them in (prompt in §E). Larger $p$ masks more information, causing greater deviation from the original instructions. We set $p \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$.

**(2) *Oversampling*:** We construct a synthetic dataset of fixed size $m$ by first selecting one instruction-response sequence $R \in \mathcal{R}$, where $\mathcal{R} \triangleq \cup_j R_j$ is the set of all such sequences in $\mathcal{D}$. We use $R$ for the first $rm$ samples and fill the remaining $(1-r)m$ samples by sampling without replacement from $\mathcal{R} \setminus \{R\}$. Larger $r$ creates more duplicates and thus lower diversity. We set $r \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$.

**(3) *In-Context Generation*:** We prompt language models with $k$ in-context examples to generate instructions and responses (prompt in §E). With $k = 0$, generation is weakly grounded in the original data; with $k > 0$, fixed examples can limit diversity, while randomly varying them improves instruction-space coverage. We use $k \in \{0, 1, 3, 5\}$ with fixed or randomly sampled examples.

**(4) *Invalidation*:** We retain the original instructions but replace tool calls or outputs in a fraction $v$ of samples. For T1 and BFCL tool calls $f(\varphi)$, we replace $\varphi_{i,j}$ with an input $\varphi_{i,k}$ that violates the instruction; for T1 and ACP outputs, we replace $O_i$ with an incompatible $O_k$.

**(5) *Industry-standard synthetic data*:** NVIDIA NeMo Data Designer generates trajectories from specified schemas, constraints, and backend models. We use `GPT-4o-mini`, `Nvidia-Nemotron-Nano-9B-v2`, `Mistral-24B-Instruct`, and `Llama3.1-8B-Instruct` as backends, and vary the temperature over $\{0.1, 0.3, 0.5, 0.7, 0.9\}$.

For Blank Filling, Oversampling, and In-Context Generation, each dataset-specific agent uses `Llama3.1-8B-Instruct` to generate the corresponding tool calls and outputs.

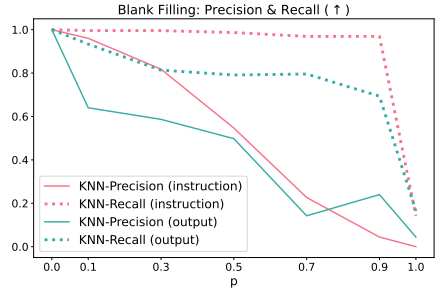
**Agents for downstream evaluation** For downstream evaluation, an LLM agent processes each trace in order, receives all prior instructions, responses, tool calls, and results, and predicts the next response and/or tool call for comparison with the reference trace. Under the same benchmark runtime context, we test agents backed by `gemma-3-1b-it`, `Qwen3-4B-Instruct`, and `Llama3.1-8B-Instruct`. We use `Mistral-7B-Instruct` as the LLM-as-a-judge for functional equivalence of tool calls and outputs; the prompt is provided in §C.

**Metric visualization** We use line and radar plots; radar metrics are rescaled to $[20, 100]$, where 20 is the worst observed performance and 100 is the upper bound (e.g., KNN-Precision$= 1$ or 2-Step Planning$= 0$). For overall diversity, fidelity, and validity, we normalize the corresponding metrics to $[0, 1]$, with higher values better, and average them. Although averaging normalized metrics has limitations as they may scale differently, it is used for interpretable benchmark visualization (Wang et al., 2023); we test alternative metric aggregations in §I.4.

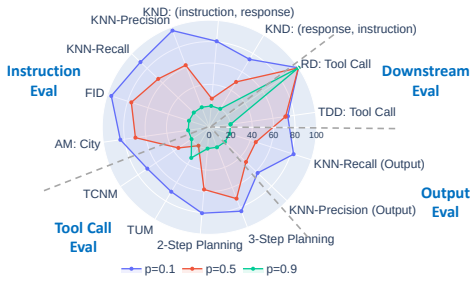
### 3.1 Experimental Results

We present the T1 results in this section and provide complete results for all datasets, NVIDIA NeMo, and other synthetic-data methods in § F and G. We further examine correlations between intrinsic metrics and downstream performance in §H, together with sensitivity analyses in §I.

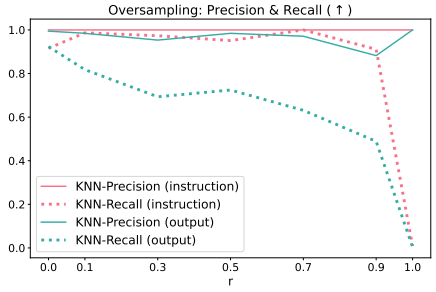
#### 3.1.1 SynAE can capture fine-grained variations in fidelity, diversity, and validity of synthetic data.



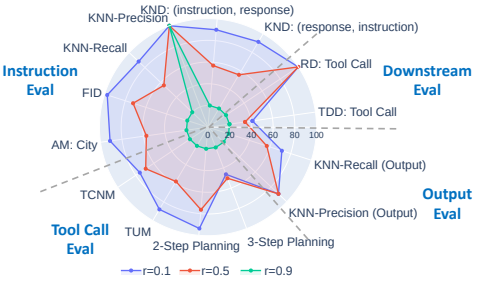

(a) KNN-Precision and KNN-Recall of Blank Filling with different masking probability $p$.




(b) Blank Filling. As $p$ increases, data fidelity degrades across nearly all metrics.

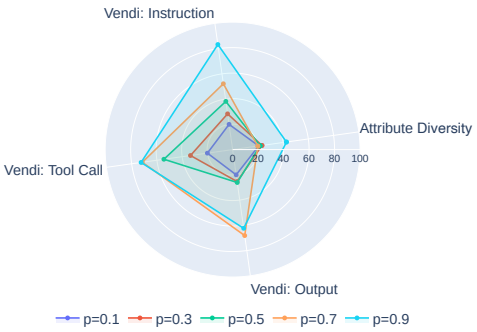



(c) KNN-Precision and KNN-Recall of Oversampling with different duplication rate $r$.

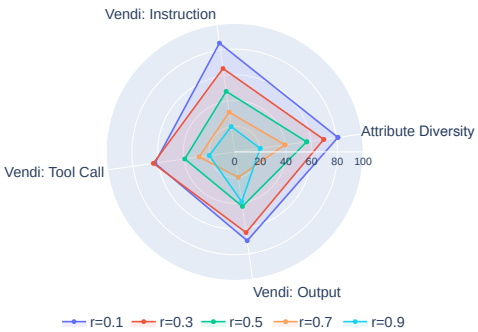



(d) Oversampling. As $r$ increases, data fidelity degrades across nearly all metrics.

Figure 3: Fidelity of Blank Filling and Oversampling on the T1 dataset.




(a) Vendi Scores and Attribute Diversity of Blank Filling with different masking probability $p$. As $p$ increases, data diversity improves.




(b) Vendi Scores and Attribute Diversity of Oversampling with different duplication rate $r$. As $r$ increases, data diversity decreases.

Figure 4: Diversity metrics for Blank Filling and Oversampling on the T1 dataset.

**Fidelity:** Figure 3 shows the fidelity of Blank Filling and Oversampling on T1. For Blank Filling (Fig. 3a), increasing $p$ reduces instruction and output KNN-Precision from nearly 1 to close to 0, showing progressive semantic degradation as masking probability increases. However, their KNN-Recall remains above 0.7 for $p \leq 0.9$, suggesting that Blank

Filling largely preserves semantic coverage. For Oversampling (Fig. 3c), increasing $r$ substantially reduces output KNN-Recall, while instruction and output KNN-Precision remain near perfect, consistent with intuition that duplication reduces task coverage while largely preserving task quality. Figs. 3b and 3d further show that heavier masking or duplication degrades nearly all fidelity metrics, aligning with the intuition.

**Diversity:** On T1 (Fig. 4), Blank Filling becomes more diverse as $p$ increases because heavier masking introduces novel content across task topics and execution trajectories (Fig. 4a). In contrast, Oversampling diversity decreases as $r$ increases because more samples are duplicated (Fig. 4b).

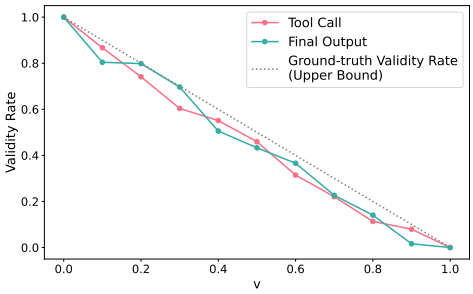



Figure 5: Validity of Invalidation on T1. As invalidation ratio $v$ increases, Validity Rates for both tool calls and outputs decrease.

**Validity:** On T1, the Validity Rates of *Invalidation* for tool calls and final outputs decrease monotonically with $v$ at an approximate slope of $-1$, as expected by construction (Fig. 5). This shows that SynAE captures validity variations.

### 3.1.2 No single metric can fully characterize synthetic data performance.

Figure 6 shows aggregate fidelity-diversity trade-offs for Blank Filling and Oversampling. Increasing the duplication rate $r$ degrades both fidelity and diversity, whereas increasing the masking probability $p$ lowers fidelity but improves diversity. Thus, the metric categories capture distinct aspects, and no single category fully characterizes synthetic data performance.

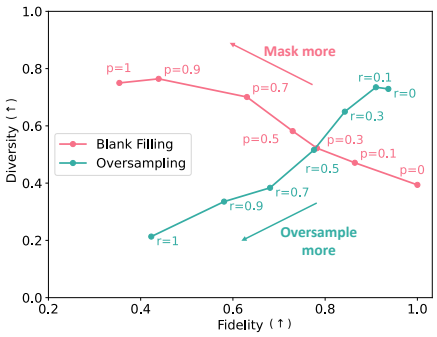



Figure 6: Fidelity vs. diversity for Blank Filling and Oversampling. For Blank Filling, a higher masking probability $p$ leads to lower fidelity but greater diversity. For Oversampling, a higher duplication rate $r$ results in declines in both fidelity and diversity.

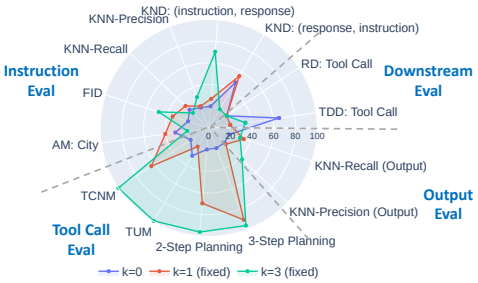
Even within fidelity, one setting may improve some metrics while worsening others. For In-Context Generation on T1, $k = 3$ improves KNN-Precision over $k = 1$ but lowers KNN-Recall (Fig. 7a). At fixed $k$, randomizing examples improves instruction- and output-level fidelity while maintaining similar tool-call performance (Fig. 7b). We also show that simple baselines, such as corpus-level statistics and embedding closeness, can be misleading when used alone (§J). Blank Filling provides another example (Fig. 3b): as $p$ increases, tool-call metrics degrade, indicating larger discrepancies in tool-planning patterns. Yet RD: Tool Call remains strong because higher-capacity models still outperform lower-capacity ones, preserving their rankings despite the planning shift. These observations motivate a comprehensive framework such as SynAE.

### 3.1.3 Case study: Practitioners can leverage SynAE to iteratively diagnose and improve synthetic data generation.

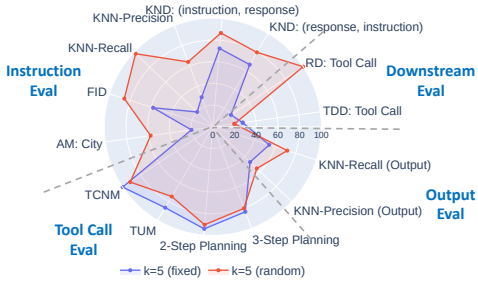
We simulate a common workflow in which a data holder uses SynAE to diagnose skewed data and iteratively refine an augmentation strategy. Starting from T1, we select half of the attraction types, including culture, sport, culinarian, and guide, and downsample instances containing these types to 10% of the original count by discarding the remaining samples. The data holder then aims to augment the dataset to better represent these target types.

**Step 1: Diagnose the bottleneck with SynAE.** The data holder first evaluates the skewed data with SynAE, which identifies *diversity* as the primary issue (first column of Table 2) and pinpoints underrepresented attributes via attribute distributions (reported in §K), motivating targeted augmentation.

**Step 2: Attempt a lightweight fix via relabeling, then re-evaluate.** As a lightweight intervention, *Relabeling* changes attraction-type keywords; for example, "art attractions" becomes "sporting at-




(a) In-Context Generation with different number of in-context examples $k$. Increasing $k$ does not consistently improve all fidelity metrics, including KNN-Recall, KND, and AM.




(b) In-Context Generation with $k = 5$. Randomized in-context examples improve instruction and output fidelity over fixed examples, while maintaining similar tool-call performance.

Figure 7: Fidelity of In-Context Generation under T1 with fixed or randomized in-context examples.

tractions." `SynAE` confirms that diversity improves (second column of Table 2), but validity drops as responses and tool calls remain unchanged: the sporting-attraction instruction "Please find some *sporting* attractions." may still elicit the art-related response "Do you prefer *museums* or *concerts*?"

**Step 3: Escalate to model-based synthesis via NVIDIA NeMo, then re-evaluate.** To improve diversity while preserving validity, the data holder adopts *NVIDIA NeMo* with `Llama3.1-8B-Instruct` or `Nvidia-Nemotron-Nano-9B-v2` at temperature 0.6 and prompts specifying the target types. For example, given the target type *sport*, NeMo generates the instruction "Please find some *sporting* attractions in Canada." followed by responses such as "Do you prefer *basketball* or *baseball*?", preserving semantic consistency. `SynAE` shows that NVIDIA NeMo with `Llama` or `Nemotron` preserves validity, but substantially sacrifices fidelity, while diversity remains moderate (columns 3 & 4 of Table 2). The data holder then escalates to the higher-capacity backend `GPT-4o-mini`. `SynAE` shows that this improves diversity while preserving validity, without substantially sacrificing fidelity (column 5 of Table 2).

Table 2: Validity, fidelity, and diversity of the skewed real data, Relabeling, and NVIDIA NeMo. **Green** indicates strong performance, red indicates poor performance, and black indicates moderate performance. The diversity of the original (unskewed) data is 0.72.

| | Skewed Real Data | Relabeling | NVIDIA NeMo | | |
|---|---|---|---|---|---|
| | | | `Llama` | `Nemotron` | `GPT` |
| Validity | **1.0** | 0.82 | **0.98** | **0.99** | **0.99** |
| Fidelity | **1.0** | **0.95** | 0.71 | 0.79 | 0.94 |
| Diversity | 0.48 | 0.65 | 0.67 | 0.61 | **0.70** |

Table 2 summarizes fidelity, diversity, and validity using aggregate scores. We define "strong" performance as metrics within 5% of the original (unskewed) real dataset and "poor" performance as metrics at least 15% lower. Overall, this case study shows how `SynAE` supports iterative refinement: diagnose the limiting factor, apply an intervention, and use metric feedback to adjust the strategies.

## 4 Conclusions and Limitations

In this paper, we introduced `SynAE`, a multi-axis framework for evaluating how well synthetic data replicates and augments real agent trajectories across task instructions and responses, tool calls, final outputs, and downstream performance. Through experiments on recent agent benchmarks with realistic and controlled synthetic generation schemes, `SynAE` consistently detects fine-grained failures in validity, fidelity, and diversity. Overall, no single metric suffices to capture synthetic benchmark quality, motivating multi-axis evaluation with `SynAE` before using synthetic data for agent testing. A limitation of our work is that `SynAE` mainly focuses on multi-turn tool-calling agentic settings. Extending the framework to agent benchmarks involving interactive environments or multi-agent coordination is an important direction for future work.

## AI USE STATEMENT

In this work, we use generative AI tools to polish the writing and improve the readability of the paper. We reviewed all AI-assisted edits and take responsibility for the final content of this work.

## REPRODUCIBILITY STATEMENT

We describe the evaluation framework and experimental protocol in the main paper. The synthetic-data generation and evaluation prompts are provided in § B, C and E; detailed results, computational costs, and robustness analyses appear in § D, F and I.

## REFERENCES


Omar Alonso and Kenneth Church. Evaluating the evaluations: A perspective on benchmarks. In *ACM SIGIR Forum*, volume 58, pp. 1–27. ACM New York, NY, USA, 2025.

Anthropic. Demystifying evals for ai agents. https://www.anthropic.com/engineering/demystifying-evals-for-ai-agents, 2026.

CapitalOne. Synthetic data matters for machine learning innovation. https://www.capitalone.com/tech/machine-learning/synthetic-data-research/, 2022.

Amartya Chakraborty, Paresh Dashore, Nadia Bathaee, Anmol Jain, Anirban Das, Shi-Xiong Zhang, Sambit Sahu, Milind Naphade, and Genta Indra Winata. T1: A tool-oriented conversational dataset for multi-turn agentic planning. *arXiv preprint arXiv:2505.16986*, 2025.

Enkrypt AI. What are specialized task ai agents? benefits, features & use cases explained. Enkrypt AI Blog (Guest Post), March 2024. URL https://www.enkryptai.com/blog/what-are-specialized-task-ai-agents-benefits-features-use-cases-explained.

Dan Friedman and Adji Bousso Dieng. The vendi score: A diversity evaluation metric for machine learning. *arXiv preprint arXiv:2210.02410*, 2022.

Alexander Gill, Abhilasha Ravichander, and Ana Marasović. What has been lost with synthetic evaluation? *arXiv preprint arXiv:2505.22830*, 2025.

Ben Hutchinson, Negar Rostamzadeh, Christina Greer, Katherine Heller, and Vinodkumar Prabhakaran. Evaluation gaps in machine learning practice. In *Proceedings of the 2022 ACM conference on fairness, accountability, and transparency*, pp. 1859–1876, 2022.

Shadi Iskander, Sofia Tolmach, Ori Shapira, Nachshon Cohen, and Zohar Karnin. Quality matters: Evaluating synthetic data for tool-using llms. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4958–4976, 2024.

Dongzhi Jiang, Ziyu Guo, Renrui Zhang, Zhuofan Zong, Hao Li, Le Zhuo, Shilin Yan, Pheng-Ann Heng, and Hongsheng Li. T2i-r1: Reinforcing image generation with collaborative semantic-level and token-level cot. *arXiv preprint arXiv:2505.00703*, 2025.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

Sayash Kapoor, Benedikt Stroebl, Peter Kirgis, Nitya Nadgir, Zachary S Siegel, Boyi Wei, Tianci Xue, Ziru Chen, Felix Chen, Saiteja Utpala, et al. Holistic agent leaderboard: The missing infrastructure for ai agent evaluation. *arXiv preprint arXiv:2510.11977*, 2025.

Harsha Kokel, Michael Katz, Kavitha Srinivas, and Shirin Sohrabi. Acpbench: Reasoning about action, change, and planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 26559–26568, 2025.

Renhao Li, Jianhong Tu, Yang Su, Yantao Liu, Fei Huang, Hamid Alinejad-Rokny, Derek F Wong, Junyang Lin, and Min Yang. Toolrm: Towards agentic tool-use reward modeling. *arXiv preprint arXiv:2510.26167*, 2025.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, William B Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for gpt-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd workshop on knowledge extraction and integration for deep learning architectures*, pp. 100–114, 2022.

Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong WANG, et al. Toolace: Winning the points of llm function calling. In *International conference on learning representations*, volume 2025, pp. 41359–41381, 2025.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*, 2023.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Shirley Kokane, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, et al. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *Advances in Neural Information Processing Systems*, 37:54463–54482, 2024.

Xing Han Lù, Amirhossein Kazemnejad, Nicholas Meade, Arkil Patel, Dongchan Shin, Alejandra Zambrano, Karolina Stańczak, Peter Shaw, Christopher J Pal, and Siva Reddy. Agentrewardbench: Evaluating automatic evaluations of web agent trajectories. *arXiv preprint arXiv:2504.08942*, 2025.

Gaurav Maheshwari, Dmitry Ivanov, and Kevin El Haddad. Efficacy of synthetic data as a benchmark. *arXiv preprint arXiv:2409.11968*, 2024.

Michael Majurski and Cynthia Matuszek. Grounding synthetic data evaluations of language models in unsupervised document corpora. *arXiv preprint arXiv:2505.08905*, 2025.

Amanda McGrath and Amanda Downie. What are vertical ai agents? IBM Think, n.d. URL https://www.ibm.com/think/topics/vertical-ai-agents.

Sohum Mehta and Saaketh Bhojanam. Prompt genotyping: Quantifying the evaluation gap between synthetic benchmarks and real llm performance. In *NeurIPS 2025 Workshop on Evaluating the Evolving LLM Lifecycle: Benchmarks, Emergent Abilities, and Scaling*, 2025.

Mahmoud Mohammadi, Yipeng Li, Jane Lo, and Wendy Yip. Evaluation and benchmarking of llm agents: A survey. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 6129–6139, 2025.

NVIDIA. NVIDIA NeMo. https://www.nvidia.com/en-us/ai-data-science/products/nemo/, n.d.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. Autonomous evaluation and refinement of digital agents. *arXiv preprint arXiv:2404.06474*, 2024.

Melissa Z Pan, Negar Arabzadeh, Riccardo Cogo, Yuxuan Zhu, Alexander Xiong, Lakshya A Agrawal, Huanzhi Mao, Emma Shen, Sid Pallerla, Liana Patel, et al. Measuring agents in production. *arXiv preprint arXiv:2512.04123*, 2025.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

Akshara Prabhakar, Zuxin Liu, Ming Zhu, Jianguo Zhang, Tulika Manoj Awalgaonkar, Shiyu Wang, Zhiwei Liu, Haolin Chen, Thai Hoang, Juan Carlos Niebles, et al. Apigen-mt: Agentic pipeline for multi-turn data generation via simulated agent-human interplay. *Advances in Neural Information Processing Systems*, 38, 2026.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023.

Yijia Shao, Yucheng Jiang, Theodore Kanell, Peter Xu, Omar Khattab, and Monica Lam. Assisting in writing wikipedia-like articles from scratch with 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. 6252–6278, 2024.

Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Yueting Zhuang. Taskbench: Benchmarking large language models for task automation. *Advances in Neural Information Processing Systems*, 37:4540–4574, 2024.

Alex Tamkin, Miles McCain, Kunal Handa, Esin Durmus, Liane Lovitt, Ankur Rathi, Saffron Huang, Alfred Mountfield, Jerry Hong, Stuart Ritchie, et al. Clio: Privacy-preserving insights into real-world ai use. *arXiv preprint arXiv:2412.13678*, 2024.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*, 2023.

B. Wang, W. Chen, H. Pei, C. Xie, M. Kang, C. Zhang, C. Xu, Z. Xiong, R. Dutta, R. Schaeffer, et al. DecodingTrust: A comprehensive assessment of trustworthiness in GPT models. In *Conference on Neural Information Processing Systems*, 2023.

Shuaiqi Wang, Vikas Raunak, Arturs Backurs, Victor Reis, Pei Zhou, Sihao Chen, Longqi Yang, Zinan Lin, Sergey Yekhanin, and Giulia Fanti. Struct-bench: A benchmark for differentially private structured text generation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, et al. Livebench: A challenging, contamination-limited llm benchmark. *arXiv preprint arXiv:2406.19314*, 2025.

Shengguang Wu, Keming Lu, Benfeng Xu, Junyang Lin, Qi Su, and Chang Zhou. Self-evolved diverse data sampling for efficient instruction tuning. *arXiv preprint arXiv:2311.08182*, 2023.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Lang Xiong, Nishant Bhargava, Jianhang Hong, Jeremy Chang, Haihao Liu, Vasu Sharma, and Kevin Zhu. Probe-rewrite-evaluate: A workflow for reliable benchmarks and quantifying evaluation awareness. *arXiv preprint arXiv:2509.00591*, 2025.

Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, Dawn Song, Huan Sun, and Yu Su. An illusion of progress? assessing the current state of web agents. *arXiv preprint arXiv:2504.01382*, 2025.

John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. Intercode: Standardizing and benchmarking interactive coding with execution feedback. *Advances in Neural Information Processing Systems*, 36:23826–23854, 2023.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. $\tau$-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Asaf Yehudai, Lilach Eden, Alan Li, Guy Uziel, Yilun Zhao, Roy Bar-Haim, Arman Cohan, and Michal Shmueli-Scheuer. Survey on evaluation of llm-based agents. *arXiv preprint arXiv:2503.16416*, 2025.

Zeyu Zhang, Guohao Li, Zhenchang Xing, Alexandros Apostolopoulos, Yu Lin Lee, and Liang Zheng. Gecko: A simulation environment to ground agent tool calls with stateful feedback for refinement. *arXiv preprint arXiv:2602.19218*, 2026.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

Kaijie Zhu, Jiaao Chen, Jindong Wang, Neil Zhenqiang Gong, Diyi Yang, and Xing Xie. Dyval: Dynamic evaluation of large language models for reasoning tasks. *arXiv preprint arXiv:2309.17167*, 2023.

Yuxuan Zhu, Tengjun Jin, Yada Pruksachatkun, Andy Zhang, Shu Liu, Sasha Cui, Sayash Kapoor, Shayne Longpre, Kevin Meng, Rebecca Weiss, et al. Establishing best practices for building rigorous agentic benchmarks. *arXiv preprint arXiv:2507.02825*, 2025.

Kaijian Zou, Muhammad Khalifa, and Lu Wang. On many-shot in-context learning for long-context evaluation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 25605–25639, 2025.

## A Related Work

Robust benchmarks for interactive tool-use are necessary both for generalist agents (e.g., code generation, research assistance, and open-domain conversation (Yehudai et al., 2025; Anthropic, 2026; Yang et al., 2024; Shao et al., 2024; Yao et al., 2022; Ouyang et al., 2022)) and domain-specific agents with narrowly scoped workflows (e.g., in finance, technology, and corporate services (Pan et al., 2025; Enkrypt AI, 2024; McGrath & Downie, n.d.)). However, constructing and maintaining such benchmarks is costly and requirements evolve (Mohammadi et al., 2025; Yehudai et al., 2025; Hutchinson et al., 2022), motivating continuously updated and synthetic benchmark generation approaches (White et al., 2025; Zhu et al., 2023).

**Evaluating synthetic benchmarks for LLMs.** There have been several synthetic benchmark datasets for evaluating sequential, conversational interactions between a user and an LLM; a few papers have quantitatively evaluated the quality of these benchmarks Gill et al. (2025); Maheshwari et al. (2024); Xiong et al. (2025); Majurski & Matuszek (2025). However, these lines of work do not tackle *agent* benchmarks, where evaluation is more complicated due to multi-step decision making, interaction with environments/tools, and the need to attribute failures to specific components of an agentic pipeline. Examples include Gill et al. (2025); Maheshwari et al. (2024); Majurski & Matuszek (2025), which study synthetic benchmarks for standard NLP tasks (e.g., reading comprehension, intent detection). They assess synthetic data via factors like difficulty (i.e., whether an agent has a similar completion rate on tasks in the real/synthetic data) and whether model rankings are preserved. `SynAE` generalizes these ideas to the multi-turn tool-calling setting in the fidelity evaluation of downstream tasks. Relatedly, Xiong et al. (2025) and Mehta & Bhojanam (2025) demonstrate that LLM behavior—both response content and trustworthiness—can differ between synthetic-benchmark settings and real-world deployment, motivating the need for realistic synthetic data.

**Evaluating synthetic benchmarks for tool-calling agents.** To our knowledge, prior works on evaluating synthetic benchmarks for tool-calling agents have only considered *single-turn* benchmarks, in which each dataset instance consists of a single client query followed by a single agent response (Shen et al., 2024; Iskander et al., 2024). This significantly simplifies evaluation, as it does not require the synthetic data to capture sequential dependencies among task instructions present in the real data; measuring the quality of these transitions is a major component of `SynAE`. Moreover, these prior works only evaluate the instructions (human inputs) in the synthetic data (e.g., naturalness, coherence), avoiding systematically evaluating the reference tool calls and outputs correspond to those instructions (Shen et al., 2024; Iskander et al., 2024). Iskander et al. (2024) further evaluates synthetic samples by how much they help in-context learning, which serves as an indirect signal of data quality. Overall, these approaches do not provide quantitative metrics for systematically measuring the statistical and semantic properties of synthetic tasks directly, nor do they evaluate the associated tool calls and final outputs within task-completion trajectories.

Recent work has also proposed qualitative checklists for practitioners to evaluate benchmarks. For example, Zhu et al. (2025) advocate for verifying task validity, i.e., each sample in the benchmark should have the correct ground truth, while Alonso & Church (2025) emphasize that a benchmark should contain realistic samples that test a diverse range of agent behaviors. However, these checklists typically requires practitioners to manually review samples, making benchmark creation and validation time-consuming and iterative (Pan et al., 2025), especially when practitioners replace or augment real datasets with synthetic samples (White et al., 2025; Zhu et al., 2023). `SynAE` speeds up these workflows by providing fine-grained quantitative metrics that pinpoint which parts of the benchmark can be improved. We demonstrate how `SynAE` can help practitioners through a case study in §3.1.3.

**Evaluating multi-turn tool-calling agents** Multi-turn tool-calling agents are typically evaluated using end-to-end task success on interactive benchmarks, where agents execute multi-step tool calls and are scored by state-based checkers that compare the final environment or database state to an annotated goal state (Yao et al., 2024; Liu et al., 2023; Zhou et al., 2023; Yang et al., 2023; Xie et al., 2024; Jimenez et al., 2023). While these works primarily evaluate *agents* on a given benchmark, in contrast, we evaluate the *benchmarks themselves*. We assume text-based outputs in this work, but the `SynAE` framework could easily be extended to incorporate a state check as part of the output

evaluation. Many benchmarks additionally enforce turn-level validity by jointly checking tool outputs and dialogue via state-based and response-based criteria across all turns (Patil et al., 2025; Li et al., 2025; Zhang et al., 2026). Increasingly, evaluations also incorporate LLM-as-a-judge, using a strong judge model to determine pass/fail or preference-style win rates over full tool-use trajectories when multiple solutions are plausible (Qin et al., 2023; Pan et al., 2024; Xue et al., 2025; Lù et al., 2025). SynAE also employs an LLM-as-a-judge to assess tool calls and final outputs by jointly checking them against the instructions across turns in our downstream evaluation. However, this alone is insufficient for evaluating the quality of a synthetic benchmark, as it focuses only on end-to-end agent performance and does not capture the intrinsic properties of task instructions or their associated tool calls and outputs. Hence, we introduce additional metrics that quantitatively measure the semantic and statistical properties of task instructions, tool-usage and planning patterns, and the quality and diversity of final outputs, providing richer diagnostic signals for developers. We demonstrate the value of these metrics in §3.

**Generating synthetic tool-use data.** Synthetic tool-use data are produced through several complementary paradigms. ToolBench constructs tool-use instructions and solution paths from real APIs (Qin et al., 2023); APIGen applies a multi-stage verification pipeline to executable function-calling data (Liu et al., 2024); ToolACE combines generation, self-evolution, and verification for function calling (Liu et al., 2025); APIGen-MT extends verified generation to multi-turn, multi-agent trajectories (Prabhakar et al., 2026); and NeMo provides a configurable data-generation framework (NVIDIA, n.d.). Our contribution is orthogonal to these generators: SynAE evaluates the resulting benchmark through validity, fidelity, diversity, and downstream behavior, allowing developers to compare pipelines under a common diagnostic framework.

# B Prompts for LLM-as-a-Judge on Validity

## B.1 Prompts for T1

### B.1.1 Tool Call Evaluation

System Prompt:

```
You are an evaluator. You must answer with ONLY 'yes' or 'no'. Never
    provide explanations or reasoning.
```

User Prompt:

```
Check the validity of the tool call sequence for an attractions
    recommendation assistant:

Conversation:
{instr}

Tool call:
{tool_call}

Evaluation rules:
- Tool call must accomplish the goal of the conversation
- Partial correctness = NO

Does the tool call correctly implement the conversation?
Answer (yes/no):
```

### B.1.2 Output Evaluation

System Prompt:

```
You are an evaluator. You must answer with ONLY 'yes' or 'no'. Never
    provide explanations or reasoning.
```

User Prompt:

```
Check the validity of the output for an attractions recommendation
    assistant:

Conversation:
{instr}

Output:
{expected_output}

Evaluation rules:
- Output must accomplish the goal of the conversation
- Partial correctness = NO

Does the output correctly implement the conversation?
Answer (yes/no):
```

### B.2 Prompts for BFCL

#### B.2.1 Tool Call Evaluation

System Prompt:

```
You are an evaluator. You must answer with ONLY 'yes' or 'no'.
Never provide explanations or reasoning.
```

User Prompt:

```
Check the validity of the tool call sequence for a function calling
    assistant:

User Requests:
{instr}

Tool calls:
{expected_tool_call}

Evaluation rules:
- Tool call must accomplish the goal of the conversation
- Partial correctness = NO

Does the tool call correctly implement the conversation?
Answer (yes/no):
```

### B.3 Prompts for ACP

#### B.3.1 Output Evaluation

System Prompt:

```
You are an evaluator. You must answer with ONLY 'yes' or 'no'. Never
    provide explanations or reasoning.
```

User Prompt:

```
Check the validity of the output:

Conversation:
{instr}

Output:
```

```
{expected_output}

Evaluation rules:
- Output must accomplish the goal of the conversation
- Partial correctness = NO

Does the output correctly implement the conversation?
Answer (yes/no):
```

### B.4 Agreement Rate with Human Annotation

We conduct human annotation on 100 randomly selected synthetic T1 samples to assess whether the generated tool calls and outputs correctly fulfill the given tasks. We then compare the human labels with judgments produced by an LLM-as-a-judge using `Mistral-7B-Instruct`. The LLM judge achieves an F1 score of 0.86, with Cohen's $\kappa$ as 0.61, indicating substantial agreement with human annotations. This suggests that the LLM-as-a-judge provides a reliable default validity checker when task-specific rule-based checkers are unavailable.

We additionally compare two stronger judges on the same 100 examples, sampled i.i.d. from the synthetic T1 pool. Qwen3-14B reaches an F1 score of 0.90 and Cohen's $\kappa$ of 0.64, while GPT-5.4-mini reaches 0.93 and 0.71, respectively. The improvement shows that judge choice affects the reliability, while the open-source Mistral judge remains a useful low-cost default.

Table 3: Agreement between LLM judges and human annotations on 100 synthetic T1 examples.

| Judge | F1 ↑ | Cohen's $\kappa$ ↑ |
|---|---|---|
| Mistral-7B-Instruct | 0.86 | 0.61 |
| Qwen3-14B | 0.90 | 0.64 |
| GPT-5.4-mini | 0.93 | 0.71 |

We manually categorize the 21 Mistral–human disagreements in Table 4. Nearly half are formatting or parsing failures, and roughly one quarter are cross-turn inconsistencies. These categories suggests improving the parser first and using a stronger or rule-based judge when semantic errors are decision-critical.

Table 4: Error analysis of the 21 Mistral–human disagreements.

| Disagreement category | Count | Share |
|---|---|---|
| Ambiguous instruction | 0 | 0.0% |
| Tool-use or argument error | 3 | 14.3% |
| Cross-turn inconsistency | 5 | 23.8% |
| Final-output contradiction | 1 | 4.8% |
| Semantically equivalent alternative | 2 | 9.5% |
| Formatting/parsing error | 10 | 47.6% |

## C Prompts for LLM-as-a-Judge in Downstream Evaluation

### C.1 Prompts for T1

#### C.1.1 Tool Call Evaluation

System Prompt:

```
You are an evaluator. You must answer with ONLY 'yes' or 'no'. Never
    provide explanations or reasoning.
```

User Prompt:

```
Compare these tool call sequences for an attractions recommendation
    assistant:

Conversation:
{instr}

Expected tool calls:
{expected_tool_call}

Actual tool calls:
{actual_tool_call}

Evaluation rules:
- Actual must accomplish the same goal as expected
- Semantic equivalence is OK (e.g., "OR" vs "Oregon", reordered
    operations with same result)
- Partial correctness = NO
- Different variable/cache names = OK if functionality identical

Does actual correctly implement the conversation based on expected?
Answer (yes/no):
```

#### C.1.2 OUTPUT EVALUATION

System Prompt:

```
You are an evaluator. You must answer with ONLY 'yes' or 'no'. Never
    provide explanations or reasoning.
```

User Prompt:

```
Compare these outputs for an attractions recommendation assistant:

Conversation:
{instr}

Expected output:
{expected_output}

Actual output:
{actual_output}

Evaluation rules:
- Actual must convey same information and meaning as expected
- Different wording is OK if content equivalent
- Partial correctness = NO
- Focus on semantic content, not syntax

Does actual correctly respond to the conversation based on expected?
Answer (yes/no):
```

### C.2 PROMPTS FOR BFCL

#### C.2.1 TOOL CALL EVALUATION

System Prompt:

```
You are an evaluator. You must answer with ONLY 'yes' or 'no'.
Never provide explanations or reasoning.
```

User Prompt:

```
Compare these outputs for a function calling assistant:

User Requests:
{instr}

Expected tool calls:
{expected_tool_call}

Actual tool calls:
{actual_tool_call}

Evaluation rules:
- Actual must accomplish the same goal as expected
- Semantic equivalence is OK (e.g., reordered operations with same result)
- Partial correctness = NO
- Different variable names = OK if functionality identical

Does actual correctly implement the user requests based on expected?
Answer (yes/no):
```

# D Computational Cost of SynAE Across Datasets

For the LLM-as-a-judge in both validity checks and downstream evaluations, we use `Mistral-7B-Instruct`. For the agents in downstream evaluations, we use `gemma-3-1b-it`, `Qwen3-4B-Instruct`, and `Llama3.1-8B-Instruct` across all datasets.

- T1 (225 samples): 450 LLM calls for validity checking of tool calls and outputs; 1,350 LLM calls for downstream evaluation per agent.
- BFCL (200 samples): 200 LLM calls for validity checking of tool calls; 800 LLM calls for downstream evaluation per agent.
- ACP (260 samples): 260 LLM calls for validity checking; 0 LLM calls for downstream evaluation per agent, since agent responses can directly be compared with benchmark ground truth (answers are either 'yes' or 'no').

All experiments use open-source LLMs. For reference, using GPT-5.4-mini instead would cost under $5 per dataset.

The evaluation is modular: users can run only the components relevant to a deployment decision, cache embeddings and judge outputs, and estimate fidelity or diversity on a subsample before committing to a full downstream evaluation. The sample-size analysis in §I quantifies the resulting stability–cost tradeoff.

# E Prompts for LLM-Based Synthetic Data Generation Methods

## E.1 Prompts for T1

### E.1.1 Blank Filling

System Prompt:

```
You are a helpful conversation generator. When given a conversation with blanks (underscores), fill them in naturally. IMPORTANT RULES:
1. The conversation MUST start with 'assistant:' (not 'Assistant:' or any variation)
2. Lines MUST alternate strictly between 'user:' and 'assistant:'
3. Each line must follow the format: 'role: content' where role is either 'user' or 'assistant'
```

```
4. Output ONLY the completed conversation with no preamble, explanation,
    or extra text
5. Maintain the same number of conversation turns as the input
```

User Prompt:

```
Example input for fill in the blanks:

assistant: H_____ What _____ of attractions are you looking for? Are you
    interested in _______, a__, or something else?
user: I'm interested in ___ and _____ attractions in __.
assistant: G_ has a lot to offer. Are you looking at specific _______ or
    re_____?
user: Yeah, I'm thinking of visiting Fre___ and M_______.
assistant: Both _____o and ____his have great A__ and S_____ attractions.
     Let me tell you about some of them.
user: That sounds _____.

Completed conversation for example input:
assistant: Hello! What kind of attractions are you looking for? Are you
    interested in history, art, or something else?
user: I'm interested in Art and Scenic attractions in GA.
assistant: GA has a lot to offer. Are you looking at specific cities or
    regions?
user: Yeah, I'm thinking of visiting Fresno and Memphis.
assistant: Both Fresno and Memphis have great Art and Scenic attractions.
     Let me tell you about some of them.
user: That sounds great.

Now fill in the blanks to complete this conversation:

{masked_conv}
Completed conversation:
```

#### E.1.2 In-Context Generation

System Prompt: Same as the system prompt in Section E.1.1.

User Prompt:

```
Here are example conversations:
{conv_list_str}

Generate 1 new similar conversation that follows the same structure.
Do not include anything other than this conversation.
Similar conversation:
```

### E.2 Prompts for BFCL

#### E.2.1 Blank Filling

System Prompt:

```
You are a request completion assistant. Fill in blanks using ONLY the
    provided APIs and Resources. Output only the completed requests, one
    per line. Do not add explanations or extra text.
```

User Prompt:

```
Fill in the blanks (underscores) to complete the user requests.
Example:
{example_context}

```

```
Input with blanks:
{example_masked}

Completed requests:
{example_completed}

Now fill in the blanks for this:
APIs: {target_classes}
Resources: {target_resources}

Input with blanks:
{masked_conv_str}

Completed requests:
```

#### E.2.2 IN-CONTEXT GENERATION

System Prompt:

```
You are a test case generator. Output each request as:
Request 1: <request text>
Request 2: <request text>
```

User Prompt:

```
{examples_str_with_APIs_resources_requests}

Generate EXACTLY {n_turns} requests for the situation below.
Requests can query information, perform operations, modify resources, or search/filter data.
Keep requests realistic. Later requests should build on earlier ones.

ONLY refer to the APIs and Resources below in the requests.
APIs: {target_classes}
Resources: {target_resources}
Requests:
```

### E.3 PROMPTS FOR ACP

#### E.3.1 BLANK FILLING

System Prompt:

```
You are a question completion assistant. Fill in the blanks (underscores) to complete the question.
Use ONLY entities and terms from the provided context. Output ONLY the completed question text."
```

User Prompt:

```
Fill in the blanks to complete the question.
Use ONLY entities from the context.

Example 1:
Context: This is a ferry domain, where the task is to transport cars from their start to their goal locations, using a ferry. Each location is accessible by ferry from each other location. The cars can be debarked or boarded, and the ferry can carry only one car at a time. There are 3 locations and 10 cars, numbered consecutively. Currently, the ferry is at l1, with the car c2 on board. The cars are at locations as follows: c6, c3, and c0 are at l2; c4, c9, and c7 are at l0; c1, c8, and c5 are at l1.
```

```
Group: applicable_actions_bool
Answer: yes
Masked question: Is the fol___ing action appl___able in this state:
    deb_rk the car c2 fr_m the ferry to loc_tion l1?
Completed question: Is the following action applicable in this state:
    debark the car c2 from the ferry to location l1?

Example 2:
Context: This is a ferry domain, where the task is to transport cars from
     their start to their goal locations, using a ferry. Each location is
     accessible by ferry from each other location. The cars can be
    debarked or boarded, and the ferry can carry only one car at a time.
    There are 3 locations and 10 cars, numbered consecutively. Currently,
     the ferry is at l1 location and it is empty. The cars are at
    locations as follows: c9, c4, and c6 are at l0; c0, c8, c1, c7, and
    c2 are at l1; c3 and c5 are at l2.
Group: progression_bool
Answer: no
Masked question: Will the f_ct "The ferry is emp__" hold aft__ perf___ing
     the act_on "emb_rk the car c0 at loc_tion l1 on to the ferry" in the
     cur_ent state?
Completed question: Will the fact "The ferry is empty" hold after
    performing the action "embark the car c0 at location l1 on to the
    ferry" in the current state?

Now complete this:
Context: {target_context}
Group: {target_group}
Answer: {target_answer}
Masked question: {masked_question}
Completed question:"
```

#### E.3.2 In-Context Generation

System Prompt:

```
You are a test case generator for planning domain tasks.
Given a context and examples, generate a single question that has the
    specified answer.
Output ONLY the question text with no preamble or label.
```

User Prompt:

```
Examples:
{examples_str_with_context_group_answer}

Generate a question of type '{target_group}' for the context below "
such that the answer is '{target_answer}'.

Context: {target_context}

f"Question:"
```

## F Detailed Evaluation Results

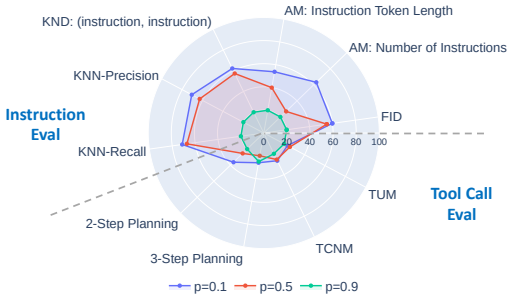
The evaluation results on T1 across synthetic data generation methods are summarized in Table 5, the results on BFCL are shown in Table 6 and Fig. 9, and the results on ACP are summarized in Table 7.

Table 5: Synthetic data generation evaluation results on SynAE under T1.

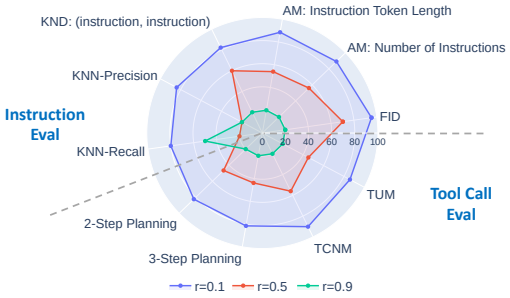
| | | Instruction Eval | | | | Tool Call Eval | | | Output Eval | | | Downstream Eval | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | Parameter | KND ↓ | AM ↓ | Vendi ↑ | AD ↑ | TUM ↓ | 3-Step Planning ↓ | Vendi ↑ | KNN-Precision ↑ | KNN-Recall ↑ | Vendi ↑ | TDD ↓ | RD ↑ |
| Blank Filling | $p = 0$ | 0 | 0 | 9.768 | 4.260 | 0 | 0 | 2.920 | 1 | 1 | 20.085 | 0 | 1.0 |
| | $p = 0.1$ | 0.019 | 0.113 | 10.790 | 4.205 | 0.018 | 0.027 | 6.928 | 0.640 | 0.933 | 24.667 | 0.052 | 1.0 |
| | $p = 0.3$ | 0.025 | 0.096 | 12.664 | 4.246 | 0.074 | 0.134 | 9.067 | 0.587 | 0.813 | 26.354 | 0.090 | 1.0 |
| | $p = 0.5$ | 0.037 | 0.213 | 14.885 | 4.223 | 0.049 | 0.048 | 12.430 | 0.498 | 0.791 | 26.590 | 0.056 | 1.0 |
| | $p = 0.7$ | 0.035 | 0.295 | 18.029 | 3.595 | 0.022 | 0.167 | 15.159 | 0.142 | 0.796 | 41.878 | 0.147 | 0.5 |
| | $p = 0.9$ | 0.059 | 0.564 | 25.029 | 2.546 | 0.041 | 0.135 | 15.312 | 0.240 | 0.693 | 36.059 | 0.160 | 1.0 |
| | $p = 1$ | 0.075 | 0.746 | 21.788 | 0.840 | 0.031 | 0.346 | 10.256 | 0.044 | 0.164 | 30.941 | 0.033 | 0.5 |
| Oversampling | $r = 0$ | 0.004 | 0.051 | 8.966 | 3.979 | 0 | 0 | 6.719 | 0.994 | 0.923 | 24.309 | 0.095 | 1.0 |
| | $r = 0.1$ | 0.012 | 0.073 | 9.200 | 4.062 | 0.009 | 0.085 | 6.843 | 0.984 | 0.818 | 22.385 | 0.048 | 1.0 |
| | $r = 0.3$ | 0.038 | 0.229 | 6.865 | 3.423 | 0.023 | 0.086 | 7.282 | 0.953 | 0.693 | 20.983 | 0.117 | 0.5 |
| | $r = 0.5$ | 0.060 | 0.351 | 4.755 | 2.657 | 0.031 | 0.079 | 6.488 | 0.984 | 0.724 | 15.497 | 0.053 | 1.0 |
| | $r = 0.7$ | 0.087 | 0.511 | 2.842 | 1.675 | 0.058 | 0.043 | 6.126 | 0.971 | 0.631 | 11.506 | 0.156 | 1.0 |
| | $r = 0.9$ | 0.110 | 0.658 | 1.520 | 0.560 | 0.059 | 0.127 | 5.869 | 0.882 | 0.489 | 15.686 | 0.165 | 0.5 |
| | $r = 1$ | 0.122 | 0.750 | 1.000 | 0.000 | 0.094 | 0.238 | 5.621 | 0.992 | 0.004 | 5.986 | 0.183 | 1.0 |
| In-Context Generation | $k = 0$ | 0.075 | 0.746 | 21.788 | 0.840 | 0.031 | 0.346 | 10.256 | 0.044 | 0.164 | 30.941 | 0.033 | 0.5 |
| | $k = 1$ (fixed) | 0.065 | 0.645 | 12.475 | 2.425 | 0.036 | 0.040 | 12.035 | 0.067 | 0.311 | 38.490 | 0.076 | 0.5 |
| | $k = 1$ (random) | 0.054 | 0.444 | 15.354 | 3.466 | 0.017 | 0.008 | 12.200 | 0.027 | 0.729 | 13.259 | 0.126 | 1.0 |
| | $k = 3$ (fixed) | 0.117 | 0.866 | 7.346 | 2.817 | 0.011 | 0.016 | 6.929 | 0.311 | 0.271 | 19.037 | 0.062 | 0.5 |
| | $k = 3$ (random) | 0.021 | 0.523 | 10.496 | 4.168 | 0.038 | 0.034 | 7.096 | 0.062 | 0.853 | 35.427 | 0.121 | 0.5 |
| | $k = 5$ (fixed) | 0.031 | 0.875 | 9.129 | 3.692 | 0.005 | 0.070 | 7.534 | 0.369 | 0.524 | 24.447 | 0.124 | 0.5 |
| | $k = 5$ (random) | 0.019 | 0.460 | 10.378 | 4.305 | 0.010 | 0.085 | 6.547 | 0.471 | 0.707 | 25.269 | 0.138 | 1.0 |

Table 6: Synthetic data generation evaluation results on SynAE under BFCL.

| | | Instruction Eval | | | | | Tool Call Eval | | | | | Downstream Eval | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | Parameter | KND ↓ | AM ↓ | Vendi ↑ | KNN-Precision ↑ | KNN-Recall ↑ | TUM ↓ | TCNM ↓ | Vendi ↑ | 2-Step Planning ↓ | 3-Step Planning ↓ | TDD ↓ | RD ↑ |
| Blank Filling | $p = 0$ | 0 | 0 | 22.153 | 1 | 1 | 0 | 0 | 114.256 | 0 | 0 | 0.146 | 1.0 |
| | $p = 0.1$ | 0.087 | 0.665 | 26.683 | 0.715 | 0.900 | 0.452 | 6.805 | 118.149 | 0.684 | 0.595 | 0.157 | 0.5 |
| | $p = 0.3$ | 0.079 | 1.030 | 28.295 | 0.675 | 0.905 | 0.388 | 5.445 | 109.984 | 0.684 | 0.632 | 0.163 | 0.5 |
| | $p = 0.5$ | 0.087 | 1.320 | 30.630 | 0.640 | 0.920 | 0.438 | 5.895 | 88.581 | 0.709 | 0.623 | 0.175 | 0.5 |
| | $p = 0.7$ | 0.105 | 1.470 | 32.895 | 0.430 | 0.910 | 0.656 | 16.335 | 67.985 | 0.654 | 0.567 | 0.215 | 0.5 |
| | $p = 0.9$ | 0.163 | 1.445 | 37.555 | 0.225 | 0.805 | 0.471 | 6.335 | 65.832 | 0.760 | 0.583 | 0.250 | 0.5 |
| | $p = 1$ | 0.124 | 0.695 | 24.867 | 0.245 | 0.290 | 0.601 | 6.515 | 69.318 | 0.747 | 0.585 | 0.150 | 0.5 |
| Oversampling | $r = 0$ | 0.020 | 0.110 | 22.058 | 1.000 | 0.875 | 0.112 | 0.285 | 52.287 | 0.156 | 0.148 | 0.005 | 1.0 |
| | $r = 0.1$ | 0.024 | 0.130 | 23.380 | 0.995 | 0.980 | 0.150 | 0.485 | 59.895 | 0.122 | 0.085 | 0.018 | 1.0 |
| | $r = 0.3$ | 0.033 | 0.305 | 14.432 | 0.995 | 0.965 | 0.384 | 1.395 | 30.895 | 0.261 | 0.180 | 0.022 | 0.5 |
| | $r = 0.5$ | 0.053 | 0.525 | 8.099 | 0.995 | 0.920 | 0.583 | 2.175 | 13.555 | 0.371 | 0.258 | 0.048 | 0.5 |
| | $r = 0.7$ | 0.080 | 0.725 | 3.804 | 0.995 | 0.910 | 0.739 | 2.985 | 5.361 | 0.466 | 0.315 | 0.073 | 0.5 |
| | $r = 0.9$ | 0.106 | 0.955 | 1.654 | 0.995 | 0.950 | 0.850 | 3.955 | 1.824 | 0.557 | 0.368 | 0.093 | 0.5 |
| | $r = 1$ | 0.118 | 1.060 | 1.000 | 1.000 | 0.005 | 0.912 | 4.290 | 1.000 | 0.627 | 0.411 | 0.108 | 0.5 |
| In-Context Generation | $k = 0$ | 0.124 | 0.695 | 24.867 | 0.245 | 0.290 | 0.601 | 6.515 | 69.318 | 0.747 | 0.585 | 0.150 | 0.5 |
| | $k = 1$ (fixed) | 0.103 | 0.330 | 15.082 | 0.450 | 0.175 | 0.613 | 4.820 | 66.247 | 0.752 | 0.586 | 0.138 | 0.5 |
| | $k = 1$ (random) | 0.129 | 0.570 | 29.312 | 0.370 | 0.760 | 0.571 | 5.045 | 53.051 | 0.698 | 0.596 | 0.182 | 0.5 |
| | $k = 3$ (fixed) | 0.079 | 0.375 | 20.832 | 0.385 | 0.235 | 0.582 | 10.770 | 91.796 | 0.644 | 0.571 | 0.135 | 0.5 |
| | $k = 3$ (random) | 0.115 | 0.515 | 29.471 | 0.500 | 0.765 | 0.557 | 7.430 | 65.258 | 0.699 | 0.574 | 0.162 | 0.5 |
| | $k = 5$ (fixed) | 0.081 | 0.365 | 24.618 | 0.405 | 0.330 | 0.457 | 5.360 | 94.033 | 0.691 | 0.621 | 0.122 | 0.5 |
| | $k = 5$ (random) | 0.064 | 0.312 | 27.739 | 0.505 | 0.720 | 0.467 | 5.685 | 56.282 | 0.745 | 0.577 | 0.152 | 0.5 |

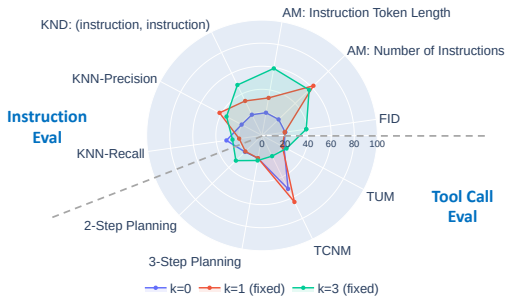



(a) Blank Filling. As $p$ increases, data utility degrades across nearly all metrics on fiedelity.

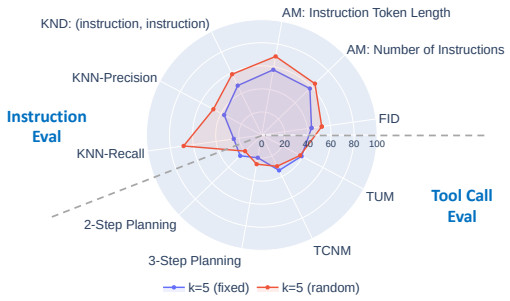



(b) Oversampling. As $r$ increases, data utility degrades across nearly all metrics on fidelity.

Figure 8: Fidelity of Blank Filling and Oversampling on the BFCL dataset.




(a) In-Context Generation with different $k$. Increasing $k$ does not consistently improve data utility across several metrics, including KNN-Recall, KNN-Precision, and AM.




(b) In-Context Generation with $k = 5$. Compared to fixed in-context examples, randomized in-context examples across generations lead to higher data utility under both instruction evaluation metrics.

Figure 9: Fidelity of In-Context Generation under BFCL with fixed and randomly sampled in-context examples across generations.

Table 7: Synthetic data generation evaluation results on SynAE under ACP.

| Method | Parameter | Instruction Eval | | | | Output Eval | | | Downstream Eval | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | KND ↓ | AM ↓ | FID ↓ | Vendi ↑ | KNN-Precision ↑ | KNN-Recall ↑ | VR↑ | TDD ↓ | RD ↑ |
| Blank Filling | $p = 0$ | 0.011 | 3.319 | 0.000 | 10.822 | 1.000 | 1.000 | 0.627 | 0.042 | 1.0 |
| | $p = 0.1$ | 0.049 | 10.981 | 0.000 | 10.822 | 1.000 | 1.000 | 0.638 | 0.088 | 1.0 |
| | $p = 0.3$ | 0.081 | 8.650 | 0.000 | 10.822 | 1.000 | 1.000 | 0.662 | 0.106 | 0.5 |
| | $p = 0.5$ | 0.150 | 9.142 | 0.000 | 10.822 | 1.000 | 1.000 | 0.642 | 0.101 | 0 |
| | $p = 0.7$ | 0.157 | 13.346 | 0.000 | 10.822 | 1.000 | 1.000 | 0.619 | 0.103 | 0.5 |
| | $p = 0.9$ | 0.176 | 18.992 | 0.000 | 10.822 | 0.996 | 1.000 | 0.600 | 0.126 | 0 |
| | $p = 1$ | 0.203 | 21.746 | 3.981 | 10.447 | 0.996 | 0.985 | 0.538 | 0.142 | 0.5 |
| Oversampling | $r = 0$ | 0.007 | 10.769 | 5.195 | 10.386 | 0.988 | 0.988 | 0.827 | 0.044 | 1.0 |
| | $r = 0.1$ | 0.018 | 10.327 | 5.235 | 10.506 | 0.996 | 1.000 | 0.731 | 0.036 | 1.0 |
| | $r = 0.3$ | 0.060 | 31.735 | 33.454 | 8.430 | 0.992 | 0.985 | 0.638 | 0.133 | 1.0 |
| | $r = 0.5$ | 0.094 | 52.988 | 83.330 | 5.450 | 0.981 | 0.992 | 0.565 | 0.206 | 1.0 |
| | $r = 0.7$ | 0.135 | 80.908 | 150.814 | 3.177 | 0.977 | 0.962 | 0.438 | 0.301 | 1.0 |
| | $r = 0.9$ | 0.170 | 99.827 | 263.393 | 1.582 | 0.969 | 0.962 | 0.323 | 0.403 | 1.0 |
| | $r = 1$ | 0.188 | 114.935 | 382.144 | 1.000 | 0.969 | 0.019 | 0.242 | 0.455 | 1.0 |
| In-Context Generation | $k = 0$ | 0.203 | 21.746 | 3.981 | 10.447 | 0.996 | 0.985 | 0.538 | 0.142 | 0.5 |
| | $k = 1$ (fixed) | 0.184 | 16.085 | 3.981 | 10.447 | 0.981 | 0.985 | 0.515 | 0.191 | 0.5 |
| | $k = 1$ (random) | 0.213 | 30.408 | 2.736 | 10.505 | 0.992 | 0.977 | 0.538 | 0.160 | 0.5 |
| | $k = 3$ (fixed) | 0.225 | 17.681 | 4.096 | 10.435 | 0.992 | 0.985 | 0.558 | 0.142 | 0 |
| | $k = 3$ (random) | 0.186 | 15.085 | 3.760 | 10.507 | 0.996 | 1.000 | 0.581 | 0.127 | 0.5 |
| | $k = 5$ (fixed) | 0.146 | 15.181 | 4.760 | 10.362 | 0.988 | 0.973 | 0.612 | 0.109 | 0.5 |
| | $k = 5$ (random) | 0.130 | 11.792 | 2.464 | 10.773 | 0.992 | 0.977 | 0.550 | 0.150 | 0.5 |

## G Evaluation Results of NVIDIA NeMo and Other Synthetic Data Methods

To evaluate a more realistic synthetic dataset, we use NVIDIA NeMo, an industry-standard synthetic data tool (including for testing agents), to generate synthetic datasets using three backbone models, `mistral-small-24b-instruct`, `nvidia-nemotron-nano-9b-v2` and `Llama3.1-8B-Instruct`, and evaluate them with SynAE on the T1 dataset. Our results in Table 8 are consistent with prior findings that the three models have similar performance on standard LLM benchmarks: SynAE reports similar fidelity scores for all. However, SynAE also shows that Nemotron and Llama exhibits greater diversity than Mistral. We use this observation to explore the distribution of the *attraction type* attribute in Table 9, where Nemotron and Llama show greater variability than Mistral. This illustrates that SynAE can uncover non-obvious properties of synthetic agent trajectories.

We also observe from Table 8 that, for both backbone models, increasing the generation temperature lowers KNN-Precision and increases KNN-Recall (coverage). This is intuitive, since higher temperatures introduce more randomness into generation and thus increase coverage.

To show that SynAE captures real evaluation outcomes, we compare agent performance on Nemotron datasets generated with temperatures 0.1 and 0.5 in Table 10. Relative to $tmp = 0.1$, agent performance on Nemotron $tmp = 0.5$ deviates more from the real dataset, which is reflected by the larger TDD and lower RD in Table 8. This indicates larger performance discrepancies and less consistent rankings.

We also compare NVIDIA NeMo with relabeling and APIGen-MT (Prabhakar et al., 2026) in Table 11. Under the matched Nemotron backbone, APIGen-MT lies between relabeling and NeMo on validity and fidelity while obtaining the same downstream measurements as NeMo-Nemotron. This comparison illustrates that SynAE can separate the generation pipeline from the backbone model and diagnose their combined output.

Table 8: NeMo evaluation results on SynAE under T1.

| Model | Temperature | Validity | Fidelity: Instruction Eval | | Fidelity: Tool Call Eval | Fidelity: Output Eval | | Fidelity: Downstream Eval | | Diversity | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | VR: output ↑ | KND: (instruction, response) ↓ | AM: city ↓ | 3-step planning ↓ | KNN-precision ↑ | KNN-recall ↑ | TDD ↓ | RD ↑ | Vendi score ↑ | AD ↑ |
| Mistral | $tmp = 0.1$ | 0.798 | 0.131 | 0.828 | 0.540 | 0.382 | 0.089 | 0.110 | 1.0 | 4.692 | 2.449 |
| | $tmp = 0.3$ | 0.831 | 0.149 | 0.869 | 0.522 | 0.302 | 0.067 | 0.044 | 1.0 | 4.759 | 2.592 |
| | $tmp = 0.5$ | 0.898 | 0.135 | 0.805 | 0.626 | 0.222 | 0.338 | 0.011 | 1.0 | 5.472 | 2.899 |
| | $tmp = 0.7$ | 0.811 | 0.143 | 0.932 | 0.519 | 0.227 | 0.369 | 0.096 | 1.0 | 6.305 | 2.718 |
| | $tmp = 0.9$ | 0.893 | 0.113 | 0.715 | 0.492 | 0.013 | 0.533 | 0.274 | 1.0 | 7.472 | 3.077 |
| Nemotron | $tmp = 0.1$ | 0.764 | 0.112 | 0.903 | 0.551 | 0.209 | 0.271 | 0.129 | 1.0 | 5.145 | 2.347 |
| | $tmp = 0.3$ | 0.787 | 0.117 | 0.843 | 0.404 | 0.244 | 0.298 | 0.141 | 1.0 | 6.013 | 2.676 |
| | $tmp = 0.5$ | 0.820 | 0.115 | 0.806 | 0.482 | 0.237 | 0.302 | 0.323 | 0.5 | 7.619 | 3.125 |
| | $tmp = 0.7$ | 0.778 | 0.106 | 0.819 | 0.520 | 0.031 | 0.902 | 0.203 | 0.5 | 8.246 | 3.160 |
| | $tmp = 0.9$ | 0.856 | 0.104 | 0.780 | 0.476 | 0.049 | 0.667 | 0.397 | 1.0 | 9.524 | 3.487 |
| Llama | $tmp = 0.1$ | 0.809 | 0.123 | 0.789 | 0.729 | 0.120 | 0.662 | 0.145 | 0.5 | 7.480 | 3.184 |
| | $tmp = 0.3$ | 0.832 | 0.142 | 0.765 | 0.817 | 0.037 | 0.511 | 0.161 | 1.0 | 8.299 | 3.187 |
| | $tmp = 0.5$ | 0.881 | 0.129 | 0.795 | 0.571 | 0.259 | 0.160 | 0.135 | 0.5 | 9.236 | 3.257 |
| | $tmp = 0.7$ | 0.804 | 0.138 | 0.798 | 0.497 | 0.286 | 0.236 | 0.157 | 0.5 | 8.788 | 3.408 |
| | $tmp = 0.9$ | 0.859 | 0.135 | 0.770 | 0.660 | 0.100 | 0.427 | 0.154 | 0.5 | 9.397 | 3.481 |

Table 9: Distribution of attraction types.

| Model | Art | Historical | Cultural | Scenic | Other |
|---|---|---|---|---|---|
| `mistral-small-24b-instruct` ($tmp = 0.7$) | 35.73% | 20.62% | 9.08% | 5.76% | 28.80% |
| `nvidia-nemotron-nano-9b-v2` ($tmp = 0.7$) | 25.76% | 23.80% | 13.21% | 12.77% | 24.46% |
| `Llama3.1-8B-Instruct` ($tmp = 0.7$) | 25.47% | 24.07% | 12.84% | 11.01% | 26.61% |

Table 10: Agent performance on real and synthetic datasets.

| Dataset | Agent 1 | Agent 2 | Agent 3 |
|---|---|---|---|
| real dataset | $1.0000 \pm 0$ | $0.6356 \pm 0.0098$ | $0.9600 \pm 0.0067$ |
| `nvidia-nemotron-nano-9b-v2` ($tmp = 0.1$) | $1.0000 \pm 0$ | $0.5644 \pm 0.0102$ | $0.6444 \pm 0.0049$ |
| `nvidia-nemotron-nano-9b-v2` ($tmp = 0.5$) | $1.0000 \pm 0$ | $0.3200 \pm 0.0131$ | $0.2967 \pm 0.0081$ |

Table 11: Targeted comparison with APIGen-MT on T1. Aggregate fidelity and diversity use the normalized summaries described in the main paper.

| Candidate dataset | Validity ↑ | Fidelity ↑ | Diversity ↑ | TDD ↓ | RD ↑ |
|---|---|---|---|---|---|
| Relabeling | 0.82 | 0.95 | 0.65 | 0.030 | 1.000 |
| NeMo-Nemotron | 0.99 | 0.79 | 0.61 | 0.043 | 1.000 |
| APIGen-MT-Nemotron | 0.93 | 0.81 | 0.59 | 0.043 | 1.000 |
| NeMo-GPT | 0.99 | 0.94 | 0.70 | 0.033 | 1.000 |

# H Correlation Between Intrinsic Metrics and Downstream Behavior

We compute correlations between each intrinsic metric and the downstream TDD and RD measures across NeMo-Nemotron temperature variants. Table 12 shows that better intrinsic fidelity generally corresponds to smaller TDD and larger RD, with KNN-Precision exhibiting the strongest association with TDD. These relationships motivate using intrinsic metrics to screen candidates before the more expensive downstream evaluation.

Table 12: Correlation between intrinsic metrics and downstream behavior. Intrinsic metric directions are normalized so that higher means better before computing correlations.

| Intrinsic metric | Correlation with TDD | Correlation with RD |
|---|---|---|
| Output validity | −0.421 | 0.317 |
| KND | −0.577 | 0.495 |
| AM | −0.677 | 0.521 |
| FID | −0.555 | 0.448 |
| KNN-Precision | −0.893 | 0.476 |
| KNN-Recall | −0.600 | 0.346 |
| Tool-planning match | −0.606 | 0.421 |
| Aggregate fidelity | −0.577 | 0.457 |

We also simulate pre-downstream selection using three rules based only on intrinsic fidelity: the highest aggregate fidelity, the lowest FID, and the highest joint normalized KNN-Precision/KNN-Recall score; a random draw serves as a non-metric baseline. The candidate pool comprises Relabeling, NeMo-Llama, NeMo-Nemotron, and NeMo-GPT. Relabeling, which has the best fidelity performance, ranks first under all three intrinsic rules and recovers the best observed downstream result, whereas the random draw selects NeMo-Nemotron in this trial. Overall, intrinsic fidelity metrics provide useful signals of downstream behavior.

Table 13: Candidate selected before downstream evaluation using intrinsic-fidelity rules or a random baseline.

| Selection rule | Selected candidate | TDD ↓ | RD ↑ |
|---|---|---|---|
| Aggregate fidelity | Relabeling | 0.030 | 1.000 |
| FID only | Relabeling | 0.030 | 1.000 |
| KNN-Precision + KNN-Recall | Relabeling | 0.030 | 1.000 |
| Random draw | NeMo-Nemotron | 0.043 | 1.000 |

# I Robustness and Sensitivity Analyses

## I.1 Sensitivity to Sample Size

To assess SynAE's sensitivity to synthetic-dataset size, we subsample NeMo-Nemotron datasets generated at different temperatures and recompute each metric at several sample sizes. For each sample size, we compare the resulting ranking of temperature settings with the ranking obtained from the full datasets. Table 14 shows that the agreement generally increases with sample size; FID, Vendi, and AD are especially stable by 100–200 examples, while neighborhood-based metrics require more samples.

## I.2 Sensitivity to Embedding Model and Neighborhood Size

The embedding-model comparison in Table 15 reports both score correlation and candidate-order agreement relative to the default STSB-RoBERTa encoder. FID and Vendi are highly stable; KNN and KND vary more because they depend on local geometry. The neighborhood-size analysis in Table 16 similarly shows moderate-to-high agreement with the default $k = 3$. These results motivate reporting the encoder and $k$ and using a fixed configuration for comparisons.

Table 14: Agreement between rankings of NeMo-Nemotron temperature settings computed from each subsample size and from the full datasets.

| Metric | $n = 25$ | $n = 50$ | $n = 100$ | $n = 200$ |
|---|---|---|---|---|
| KNN-Precision | 0.637 | 0.672 | 0.734 | 0.866 |
| KNN-Recall | 0.664 | 0.678 | 0.721 | 0.899 |
| FID | 0.783 | 0.891 | 0.951 | 0.992 |
| KND | 0.799 | 0.859 | 0.903 | 0.960 |
| AM | 0.637 | 0.700 | 0.801 | 0.925 |
| Vendi | 0.875 | 0.936 | 0.972 | 0.996 |
| AD | 0.839 | 0.903 | 0.948 | 0.991 |

Table 15: Embedding sensitivity relative to STSB-RoBERTa across NeMo-Nemotron temperature settings. "Score correlation" compares metric values, while "order agreement" compares rankings of the temperature settings.

| Alternative encoder | Metric | Score correlation $\rho$ | Order agreement |
|---|---|---|---|
| BGE-M3 | KNN-Precision | 0.695 | 0.724 |
| BGE-M3 | KNN-Recall | 0.688 | 0.629 |
| BGE-M3 | FID | 0.879 | 0.867 |
| BGE-M3 | KND | 0.489 | 0.667 |
| BGE-M3 | Vendi | 0.989 | 0.971 |
| E5-large-v2 | KNN-Precision | 0.608 | 0.624 |
| E5-large-v2 | KNN-Recall | 0.463 | 0.648 |
| E5-large-v2 | FID | 0.968 | 0.933 |
| E5-large-v2 | KND | 0.600 | 0.714 |
| E5-large-v2 | Vendi | 0.971 | 0.952 |

Table 16: Sensitivity of KNN metrics to neighborhood size, relative to $k = 3$.

| Metric | Neighborhood | Score correlation $\rho$ | Order agreement |
|---|---|---|---|
| KNN-Precision | $k = 5$ | 0.802 | 0.724 |
| KNN-Recall | $k = 5$ | 0.736 | 0.781 |
| KNN-Precision | $k = 10$ | 0.624 | 0.657 |
| KNN-Recall | $k = 10$ | 0.682 | 0.752 |
| KNN-Precision | $k = 20$ | 0.625 | 0.648 |
| KNN-Recall | $k = 20$ | 0.766 | 0.838 |

### I.3 Diversity Stress Test

Vendi and attribute diversity (AD) measure complementary failure modes. We construct four controlled datasets by independently varying two factors: surface templates, which are alternative natural-language phrasings of an instruction pattern, and semantic slot combinations, which are the attribute values inserted into a template (e.g., city and attraction type). Holding either factor fixed isolates variation in phrasing from variation in semantic content. Table 17 shows that Vendi increases when multiple surface templates are used even if the slot combination is fixed, whereas AD is zero when only one slot combination is present and increases as the semantic attributes vary. Reporting both metrics therefore distinguishes surface paraphrasing from semantic coverage and reveals complete collapse when neither varies.

Table 17: Controlled diversity stress test.

| Condition | Vendi ↑ | AD ↑ | Interpretation |
|---|---|---|---|
| One template, diverse slots | 1.745 | 2.773 | Attribute coverage without structural variety |
| Many templates, one slot combination | 2.425 | 0.000 | Surface variety without attribute coverage |
| Many templates, diverse slots | 5.230 | 2.773 | Structural and attribute diversity |
| One template, one slot combination | 1.000 | 0.000 | Complete collapse |

### I.4 Robustness to Metric Aggregation

The main analysis reports metric families separately to preserve diagnostic information. As a robustness check, Table 18 aggregates normalized metrics within validity, fidelity, and diversity using Borda scores. Values outside parentheses are the normalized aggregate scores, and parentheses show the corresponding ranks. Table 19 reports an alternative Kemeny aggregation as ranks because Kemeny optimization returns a consensus ordering rather than an aggregate score. Both methods preserve the broad tradeoff: the real and relabeled data lead on fidelity, while NeMo variants lead on diversity.

Table 18: Borda aggregation within each metric family. Values are normalized aggregate scores; parentheses show rank.

| Category | Skewed real | Relabeling | NeMo-Llama | NeMo-Nemotron | NeMo-GPT |
|---|---|---|---|---|---|
| Validity | 1.000 (1) | 0.000 (5) | 0.250 (4) | 0.625 (2.5) | 0.625 (2.5) |
| Fidelity | 0.917 (1) | 0.722 (2) | 0.153 (5) | 0.236 (4) | 0.472 (3) |
| Diversity | 0.062 (5) | 0.375 (4) | 0.812 (1) | 0.562 (3) | 0.688 (2) |

Table 19: Kemeny-optimal ranks within each metric family; lower is better.

| Category | Skewed real | Relabeling | NeMo-Llama | NeMo-Nemotron | NeMo-GPT |
|---|---|---|---|---|---|
| Validity | 1 | 5 | 4 | 2 | 3 |
| Fidelity | 1 | 2 | 5 | 4 | 3 |
| Diversity | 5 | 4 | 1 | 3 | 2 |

### I.5 Partial and Environment-Based Trajectories

The framework operates on available trajectory components rather than requiring every benchmark to expose the same record. For a partial trajectory, users evaluate the observed instruction, action, and output modules and mark unavailable metrics as not applicable. For environment-based tasks such as Terminal-Bench, the task prompt maps to instruction evaluation, terminal commands and action sequences map to tool-use and planning evaluation, the terminal state or submitted answer maps

to output evaluation, and the native task checker supplies end-to-end validity. This preserves the separation between structural tool-call metrics—frequencies, counts, and conditional transitions—and semantic correctness, which is evaluated over the full trajectory through rule-based checks, LLM judges, or environment state.

## J No single baseline metric can fully characterize synthetic data performance

We report several simple baselines for evaluating synthetic instructions and responses generated by In-Context Generation with different numbers of in-context examples, $k$. Specifically, we consider vocabulary-overlap F1, total variation (TV) distance between unigram distributions, TV distance on length distributions, Fréchet Inception Distance (FID). The results are shown in Table 20.

Table 20: Simple baseline metrics for synthetic instructions and responses generated by In-Context Generation with different numbers of in-context examples $k$.

| Metric | $k = 0$ | $k = 1$ | $k = 3$ | $k = 5$ |
|---|---|---|---|---|
| Vocabulary overlap (F1) ↑ | 0.3856 | 0.4788 | 0.6030 | 0.6809 |
| Unigram distribution (TV) ↓ | 0.6132 | 0.5281 | 0.3891 | 0.3111 |
| Length distribution ↓ | 9.2444 | 8.7733 | 8.7422 | 8.7022 |
| FID ↓ | 204.0479 | 166.2299 | 131.6828 | 107.5157 |

Taken in isolation, these baselines may suggest that increasing $k$ monotonically improves synthetic data quality, since lexical overlap increases while several distributional distances decrease. However, this conclusion is misleading. As shown in Fig. 7a, using more in-context examples does not necessarily improve overall generation quality. This is consistent with prior work showing that in-context learning is highly sensitive to the choice of demonstrations, and that adding more demonstrations does not uniformly improve performance across tasks (Liu et al., 2022; Zou et al., 2025). More broadly, these simple baselines are useful as sanity checks, but they mainly capture surface-form similarity or coarse marginal statistics, and therefore can miss higher-order aspects of synthetic data quality such as validity and diversity.

## K Attribute Distribution of the Skewed and Augmented Datasets

We report the attraction-type distributions of the skewed and augmented datasets in Table 21. The skewed dataset contains four under-represented attraction types, defined as those with proportions below $5\%$, whereas the augmented datasets produced by Relabeling or NVIDIA NeMo contain no under-represented attraction types.

Table 21: Distribution of attraction types. Red indicates under-represented attraction types.

| Dataset | Art | Historical | Cultural | Scenic | Touristy | Culinary | Guided | Social | Sporting |
|---|---|---|---|---|---|---|---|---|---|
| Skewed Real Data | 16.67% | 15.56% | 2.22% | 21.11% | 17.78% | 4.44% | 4.96% | 12.82% | 4.44% |
| Relabeling | 12.60% | 11.02% | 9.45% | 16.54% | 12.60% | 11.02% | 9.45% | 8.66% | 8.66% |
| NVIDIA NeMo (Llama) | 27.35% | 13.76% | 11.59% | 12.13% | 9.42% | 7.24% | 6.70% | 6.70% | 5.07% |
| NVIDIA NeMo (Nemotron) | 19.56% | 14.92% | 12.83% | 11.28% | 11.80% | 9.74% | 8.19% | 6.13% | 5.55% |
| NVIDIA NeMo (GPT) | 17.74% | 14.07% | 12.74% | 12.74% | 12.07% | 9.40% | 6.07% | 7.40% | 7.74% |