ESDAE: EVALUATING SYNTHETIC DATA FOR AGENT EVALUATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Agent evaluation is often performed on static datasets of execution trajectories, but real traces may be sensitive, proprietary, or too small to support comprehensive testing. Practitioners may therefore replace or augment real datasets with synthetic ones, often without quantifying whether the synthetic data actually reflects the real data distribution. We introduce ESDAE, an evaluation framework for assessing how well synthetic benchmarks for multi-turn, tool-calling agents replicate the characteristics of real data trajectories. ESDAE assesses the quality of synthetic data relative to a real dataset across four metric categories: (i) task instructions and intermediate responses, (ii) tool calls, (iii) final outputs, and (iv) downstream evaluation. We evaluate ESDAE using recent agent benchmarks and test common synthetic data failure modes via controlled generation schemes. ESDAE detects fine-grained variations in both data 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.

1 INTRODUCTION

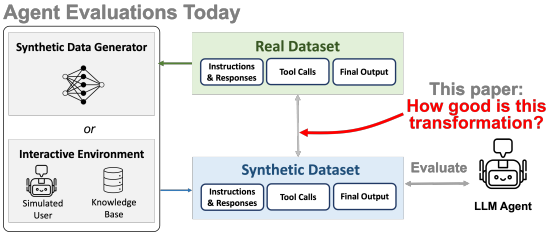
Agent evaluation and testing is a nascent but critical component of pre-deployment processes for production agentic workflows (Pan et al., 2025; Anthropic, 2026). Today, agent evaluations are often (but not always) run on static *baseline datasets*[1] consisting of a trace generated from agent interactions; 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 an agent makes the right tool calls and achieves the right final outputs for a given user input (Yehudai et al., 2025; Mohammadi et al., 2025; Anthropic, 2026).

In many practical situations, existing baseline datasets either cannot be directly used or are insufficient for agent evaluation. One common reason is that the baseline dataset contains sensitive user data (e.g., emails, travel details), and thus cannot be used due to privacy restrictions (Tamkin et al., 2024; CapitalOne, 2022). Another common reason is that an organization may possess curated baseline datasets that are too small for comprehensive testing, and it wishes to test agents on larger datasets (Pan et al., 2025).

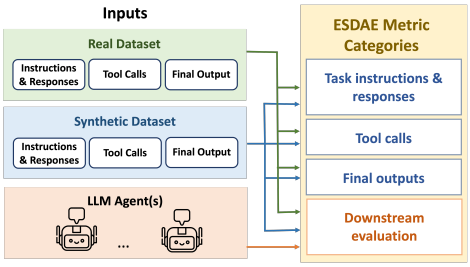
When original datasets are not sufficient or usable for downstream evaluation, an increasingly popular solution is to use *synthetic datasets* to model the baseline dataset for testing (Figure 1a) (Qin et al., 2023; Tang et al., 2023; Iskander et al., 2024). Synthetic data can be generated in various ways, either by directly synthesizing trajectories, or by synthesizing inputs to an interactive environment, as shown in Figure 1a. Once it is generated, synthetic data is typically used as a direct replacement for real execution traces in evaluation pipelines.

Although agent evaluation on synthetic datasets is growing increasingly common, typical workflows do not systematically measure the quality of the synthetic dataset with respect to the original baseline dataset. Indeed, **the current literature on testing agents with synthetic data does not provide**

[1]These evaluation datasets are also commonly referred to as *benchmarks*; we use the terminology interchangeably in this paper.



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



(b) The ESDAE evaluation pipeline takes as input original and synthetic datasets (task completion trajectories), and one or more LLM agents. The ESDAE framework evaluates four categories of metrics, comparing their outcomes on real and synthetic data. The LLM agents are used only to assess downstream performance metrics, while the real and synthetic datasets are used to compute all four metric categories.

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

quantitative methods for evaluating the quality of such synthetic data. As a result, agent evaluations may be missing important components of the original data, but operators have no visibility into these gaps.

In this work, we develop a comprehensive evaluation framework, ESDAE, to assess how well synthetic trajectories replicate the characteristics of real data trajectories, including task instructions and responses, and the associated reference tool calls and outputs. As illustrated in Figure 1b, ESDAE takes as input a (real) baseline dataset, a synthetic dataset, and one or more LLM agents; the input agent(s) need not be the same as the one being evaluated—each agent input to ESDAE is used purely to compare downstream performance between the real and synthetic input datasets. They could be, for example, multiple variants of the same agent family.

The ESDAE framework computes four categories of metrics assessing the quality of synthetic data relative to a real dataset: (1) task instructions and responses, (2) tool calls, (3) final outputs, and (4) downstream evaluation. For each category, we compute metrics on both the synthetic and real datasets, and compare the difference between the two; the desired outcome is that all four categories of metrics are similar for synthetic and real data. For final outputs, we focus on textual responses, since non-textual outcomes (e.g., database updates or environment state changes) can typically be validated using domain-specific verifiers. We measure statistical similarity between the final outputs in the real and synthetic datasets. Our downstream task evaluations assess the end-to-end quality of each provided dataset. Given user-provided agents, we evaluate their ability to select appropriate tool calls and generate appropriate outputs, given the instructions from either real or synthetic trajectories; we also analyze both per-agent performance differences and the consistency of agent performance rankings when evaluated on real vs. synthetic trajectories.

We implement ESDAE on recently proposed agent benchmarks, including T1 (Chakraborty et al., 2025) and BFCL (Patil et al.), which we treat as real baseline datasets, and evaluate several synthetic data generation methods designed to simulate common real-world pitfalls, such as degraded data quality and limited diversity. Our experiments demonstrate that ESDAE can captures fine-grained variations in both the quality and diversity of synthetic task-completion trajectories. Our results further suggest that no single metric is sufficient to fully characterize synthetic data quality, underscoring the need for ESDAE. Overall, we view ESDAE as a plug-in component for agentic workflows, allowing operators to automatically evaluate the quality of synthetic benchmark datasets.

2 Related Work

Generalist agents for 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 for narrowly scoped workflows across finance, technology, and corporate services (Pan et al., 2025; Enkrypt AI, 2024; McGrath & Downie, n.d.) both require robust benchmarks to measure interactive tool-use effectiveness. 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 natural language agents. There have been several synthetic benchmark datasets for evaluating sequential, conversational interactions between a user and an LLM agent; a few papers have quantitatively evaluated the quality of these benchmarks. 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, named entity recognition). 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. ESDAE generalizes these ideas to the multi-turn tool-calling setting; they appear in the downstream evaluation of our framework. Relatedly, Xiong et al. (2025) and Mehta & Bhojanam 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 present in the real data; measuring the quality of these transitions is a major component of ESDAE. 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, nor do they evaluate the associated tool calls and final outputs within task-completion trajectories.

Existing evaluation techniques for 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). We assume text-based outputs in this work, but the ESDAE 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.; Li et al., 2025; Zhang et al.). 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). ESDAE 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 §4.

3 ESDAE FRAMEWORK

The ESDAE framework assesses how well synthetic agent trajectories replicate the characteristics of a real dataset. Specifically, in this work, we measure the *similarity* between derived properties of a real dataset and a synthetic one. There exist use cases where an operator may use synthetic data to add *diversity* to a real dataset; in these cases, the operator may not want the real and synthetic datasets to be too similar. However, we leave such scenarios to future work; in this work, we focus on the simpler problem of measuring similarity between datasets. Today, we lack evaluation mechanisms for even this simpler task.

We decompose each evaluation instance from the dataset (i.e. each sample from a trace) into three components: (1) task instructions with their responses, (2) reference tool calls, and (3) reference final outputs. Users are required to provide the instructions and responses, while tool calls and final outputs are optional, depending on the type of input real dataset. Given the instructions, reference tool calls and final outputs can be generated by human experts, state-of-the-art agents, or a combination of both.

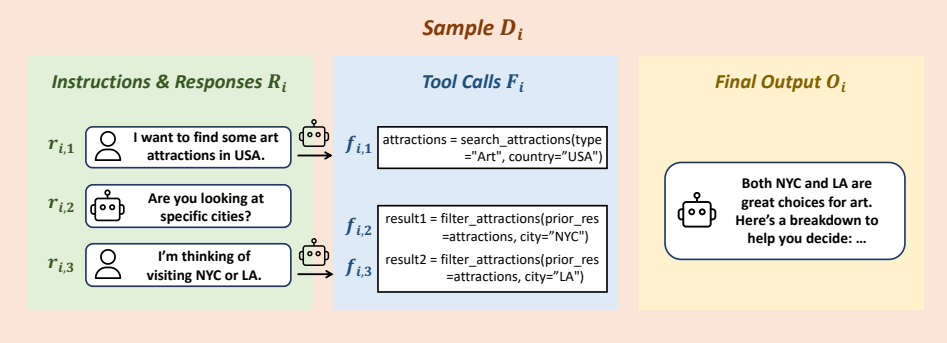


Figure 2: Illustration of an agent trajectories with notation for each component.

Notation Consider a dataset $\mathcal{D} = \{D_i\}_{i=1}^{m}$, where m is the number of samples in the dataset. As illustrated in Fig. 2, each sample trajectory D_i consists of instructions and responses R_i, tool calls F_i, and a textual output O_i. R_i contains multiple instructions and responses $r_{i,1}, r_{i,2}, \ldots, r_{i,\ell_i}$, where ℓ_i denotes the number of instructions and responses in the sample D_i. The tool call sequence F_i consists of tool usages $f_{i,1}, f_{i,2}, \ldots, f_{i,q_i}$, where each tool usage $f \in \mathcal{F}$ corresponds to an executable function call and q_i is the total number of tool calls in the trajectory. Note that instructions and responses, tool calls, and outputs can be interleaved in time; we use D_i to refer to the time-ordered sequence of events, and use R_i, F_i, O_i to refer to the corresponding filtered subsequences containing only instructions and responses, tool calls, and final outputs, respectively.

Summary of Inputs The inputs to our evaluation framework include (1) an original dataset $\mathcal{D}$; (2) a synthetic dataset $\mathcal{D}'$; and (3) agents $A_1, \ldots, A_k$ for downstream task evaluations. Both datasets contain task instructions and their responses, along with the corresponding reference tool calls and final outputs, where responses, tool calls, or outputs may not exist, depending on the benchmark realization.

3.1 EVALUATION METRICS

We introduce one category of evaluation metrics for each component of the evaluation data, as well as a final category of downstream task evaluations.

1. Metrics for Task Instructions & Responses Task instructions and their responses exhibit an explicit structure: each instruction is followed by a response, and instructions and responses alternate. Due to this structure, we take inspiration from the approach of StructBench (Wang et al.),

a framework designed for evaluating structured synthetic data. Specifically, we adopt the following metrics:

For structural metrics, i.e., the metrics requiring the structural information of the data, we include the *Key Node Dependency* and *Attribute Match*. **Key Node Dependency (KND)** measures semantic dependencies between different parts of each sample by computing the cosine similarity between their embeddings. We then quantify the similarity between real and synthetic datasets by measuring the distributional distance between their dependency distributions. Concretely, we evaluate dependencies between each instruction and its corresponding response, as well as between each response and the subsequent instruction. **Attribute Match (AM)** measures how closely the real and synthetic datasets align with respect to predefined statistical and semantic attributes. Specifically, AM computes the distributional distance of each attribute between the two datasets, using the Wasserstein-2 distance for numerical attributes and the Total Variation (TV) distance for categorical attributes. The attributes we consider include the number of instruction turns, instruction and response token lengths, and dataset-specific semantic properties.

For non-structural metrics, we include **KNN-Precision** and **KNN-Recall**, which measure the semantic quality and diversity of the synthetic instructions and responses, respectively. KNN-Precision (and correspondingly, KNN-Recall) represents the fraction of synthetic (or real) samples whose embedding distance to a real (or synthetic) sample is smaller than the distance to the k-th nearest neighbor within their own distribution. We also include an additional metric **Fréchet Inception Distance (FID)** that measures the semantic closeness between the original and synthetic data.

2. Metrics for Tool Calls We design metrics to evaluate tool-usage and tool-planning patterns. Similar to AM, each metric measures the distributional distance between real and synthetic data with respect to a specific property. (1) **Tool Usage Match (TUM)** We measure the similarity of overall tool-usage patterns between the real and synthetic datasets. Let ω_f and ω'_f denote the tool-usage distributions in the real and synthetic data, respectively. We define TUM $= Dis\left(\omega_f, \omega'_f\right)$, where Dis is the TV distance. (2) **Tool Call Number Match (TCNM)** This metric compares the distributions of the number of tool calls per sample, denoted by q. It is defined as TCNM $= Dis\left(\omega_q, \omega'_q\right)$, where Dis is the Wasserstein-2 distance. (3) **k-Step Tool Planning Match (k-Step Planning)** We measure the similarity of tool-planning patterns by comparing the conditional distributions of the next tool call given the previous $k-1$ consecutive tool call steps, i.e., $\omega_{f|f_1\cdots f_{k-1}}, \forall f_1\cdots f_{k-1} \in \mathcal{F}^{k-1}$. The metric is defined as a weighted sum of TV distances between corresponding conditional distributions, where each weight is proportional to $n_{f_1\cdots f_{k-1}}$, the number of k-step tool-call sequences in the real data whose first $k-1$ steps are $f_1\cdots f_{k-1}$:

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

where Dis is the TV distance. For all metrics above, lower values indicate better alignment between synthetic and real data.

3. Metrics for Outputs In addition to measuring the quality and diversity of the synthetic instructions & response, we evaluate the task quality and diversity of the final outputs by measuring their **KNN-Precision**, **KNN-Recall**, and **FID**.

4. Downstream Evaluation To assess the downstream utility of the benchmark constructed from synthetic dataset, we evaluate the ability of user-provided agents to select appropriate tool calls and generate appropriate outputs on both original and synthetic tasks.

Recall that the user (optionally) inputs one or more agents $A_1, \ldots, A_h$ to the downstream evaluation of `ESDAE`; these can be different from the agent being evaluated in Fig. 1a, and are used to generate an agent-specific sequence of tool calls and outputs as follows. For each input agent A_j, and for a given reference trajectory D_i, at each instruction turn, an agent generates tool calls conditioned on the current instruction, previous instruction turns, and the history of tool calls and their results. For each dataset, we use its corresponding predefined valid tool set. The final output is then generated based on the full history of instruction turns together with the executed tool calls and their results.

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

tasks. Specifically, TDD first takes a reference trace D_i, either from the real or synthetic dataset. It then produces an agent-generated trace $\tilde{D}_{i,A_j}$ using each input agent A_j. We pass D_i and $\tilde{D}_{i,A_j}$ to an LLM-as-a-judge, and evaluate whether the two trajectories are functionally equivalent (prompt in Appendix A). We then compute the absolute difference in performance, either in tool-call selection or final output generation, between the real and synthetic benchmarks and average the result across agents. **(2) Ranking Divergence (RD)** evaluates the consistency of performance rankings across agents under the original and synthetic datasets. Specifically, we rank the agents based on their ability to generate appropriate tool calls or outputs on both datasets, and then compute the Spearman rank correlation between the two rankings. For TDD, lower values indicate better downstream utility, whereas for RD, higher values correspond to better downstream utility.

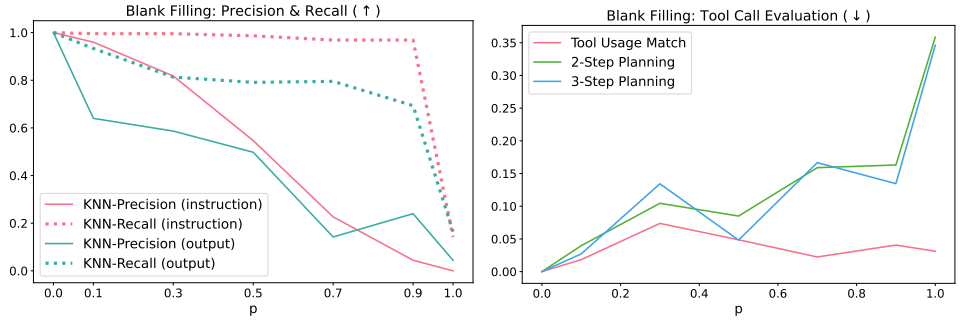
4 EXPERIMENTS

We demonstrate the effectiveness of `ESDAE` by implementing it to the following baseline datasets and synthetic data generation methods. We construct synthetic datasets by first generating synthetic instructions using each data generation method, and then producing the corresponding reference tool calls and final outputs based on those instructions.

Evaluation Datasets We evaluate `ESDAE` on T1 (Chakraborty et al., 2025) and BFCL (Patil et al.) benchmark datasets, and treat them as the real data. (1) **T1** We use the T1-attraction dataset, which contains 225 samples. Each sample consists of multi-turn instructions for attraction recommendations given a location, along with reference tool calls and final recommendation outputs. (2) **BFCL** We use the BFCL-V3-Base-Multi-Turn dataset, which includes 200 multi-turn instructions with reference tool calls. The instructions cover diverse domains, such as file system operations, mathematical calculations, and travel booking. The evaluation metrics instantiated for both datasets are summarized in Table 1.

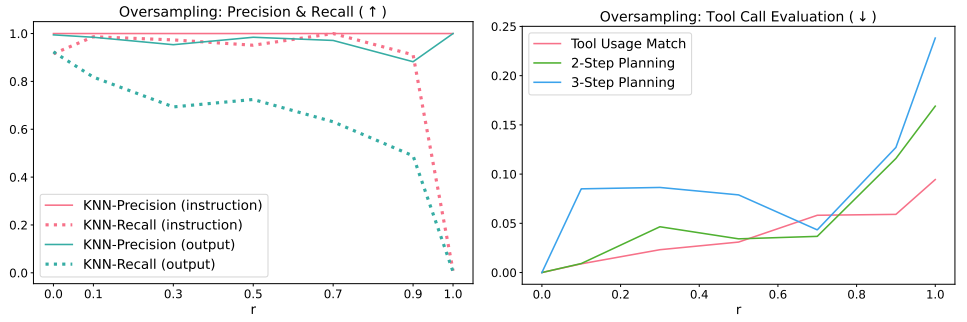
Table 1: Evaluation Metrics for T1 and BFCL.

Dataset	Instruction Eval	Tool Call Eval	Output Eval	Downstream Eval
T1	***Structural Metrics*** **KND** 1. (instruction, response) pair 2. (response, instruction) pair **AM** 1. number of instructions 2.instruction token length 3. response token length 4. city 5. attraction type ***Non-structural Metrics*** 1. KNN-Precision 2. KNN-Recall 3. FID	1. TUM 2. TCNM 3. k-Step Planning ($k \in \{2, 3\}$)	1. KNN-Precision 2. KNN-Recall 3. FID	1. TDD: Tool Call 2. TDD: Output 3. RD: Tool Call 4. RD: Output
BFCL	***Structural Metrics*** **KND** (instruction, instruction) pair **AM** 1. number of instructions 2.instruction token length ***Non-structural Metrics*** 1. KNN-Precision 2. KNN-Recall 3. FID	1. TUM 2. TCNM 3. k-Step Planning ($k \in \{2, 3\}$)	NA	1. TDD: Tool Call 2. RD: Tool Call



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

(b) TUM, 2-Step Planning, and 3-Step Planning of Blank Filling with different p.



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

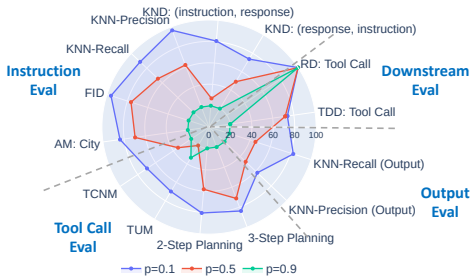
(d) TUM, 2-Step Planning, and 3-Step Planning of Oversampling with different r.

Figure 3: Performance of Blank Filling and Oversampling under T1.

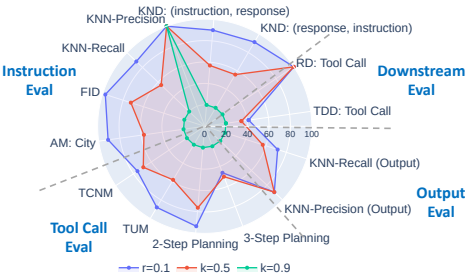
Synthetic Instruction Generation We design several interpretable synthetic instruction generation methods based on the original instructions to systematically model common limitations of synthetic data. Specifically, we simulate degraded data quality, reflected by reduced similarity to real data, using *Blank Filling*, limited data diversity via *Oversampling*, and the combined effects of degraded quality and diversity via *In-Context Generation*. (1) **Blank Filling** Synthetic instructions are constructed by randomly masking tokens in the original instructions with probability p, and prompting a language model to complete the masked inputs (prompt in §B). As p increases, more information from the original instructions is removed, causing the generated instructions to deviate further from the originals and resulting in lower data quality. We set $p \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ in our experiments. (2) **Oversampling** A small subset of original instructions is oversampled to constitute $r\%$ of the synthetic dataset, while the remaining synthetic instructions are randomly sampled without replacement from the rest of the original data, with the total number of samples in the resulting dataset kept fixed. Larger values of r introduce more duplicated instructions, thereby reducing data diversity. In our experiments, we duplicate a single instruction and vary $r \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. (3) **In-Context Generation** Synthetic instructions are generated by prompting language models with k in-context examples (prompt in §B). When $k = 0$, instructions are generated solely from the prompt without grounding in the original data, leading to poor data quality. When $k > 0$, using fixed in-context examples across generation rounds limits diversity, whereas varying the in-context examples improves the coverage of the instruction space. In our experiments, we set $k \in \{0, 1, 3, 5\}$ and consider both fixed and randomly sampled in-context examples across generations.

Tool Call & Output Generation We generate tool calls and outputs using `Llama3.1-8B-Instruct` and have their validity reviewed by human experts, following common practice in agent benchmark construction, as also adopted in T1 (Chakraborty et al., 2025) and BFCL (Patil et al.).

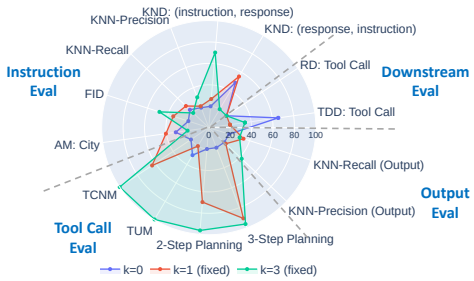
Agents for Downstream Evaluation We conduct downstream evaluations on each dataset using LLM-based agents under both real and synthetic tasks. Each agent is executed by providing



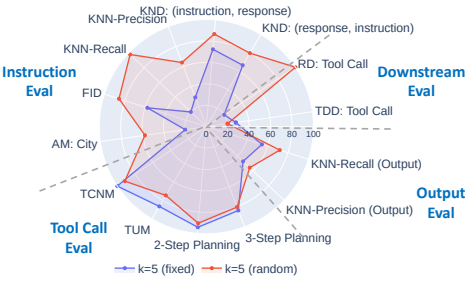
(a) Blank Filling. As p increases, data utility degrades across nearly all metrics.



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



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



(d) 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 and output evaluation metrics.

Figure 4: Performance of Blank Filling, Oversampling, and In-Context Generation under T1.

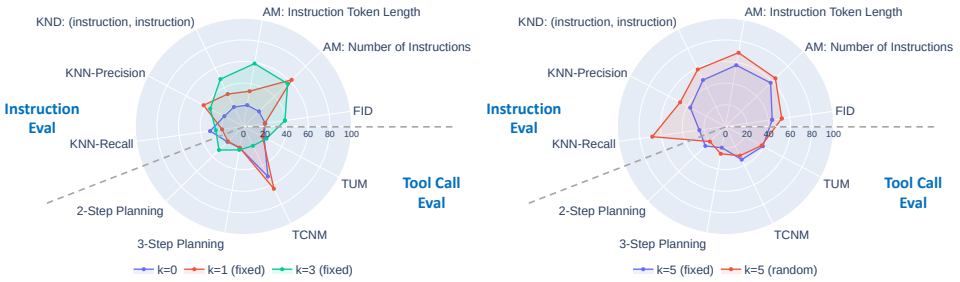
the full prior context—including instructions, responses, tool calls, and their results—as input to the model. We include three LLMs for evaluation: `gemma-3-1b-it`, `Qwen3-4B-Instruct`, and `Llama3.1-8B-Instruct`. For the LLM-as-a-judge component of the evaluation, we use `Mistral-7B-Instruct`.

4.1 Experimental Results

We present the evaluation results of different synthetic data generation methods on the T1 and BFCL datasets in Figs. 3 to 5, and defer the detailed numerical results to §C.

Fig. 3 illustrates the performance of Blank Filling and Oversampling on T1 dataset under different parameter settings. We further analyze the performance of Blank Filling, Oversampling, and In-Context Generation on T1 and BFCL datasets using radar plots in Figs. 4 and 5. To improve visualization, we rescale all evaluation metrics to the range $[20, 100]$, where 20 corresponds to the worst performance observed across all compared methods and 100 represents the performance upper bound (e.g., KNN-Precision = 1 or 2-Step Planning = 0). The figures indicate several main takeaways:

- *`ESDAE` can captures fine-grained variations in both the quality and diversity of synthetic data.* For Blank Filling (shown in Fig. 3a), as p increases, the KNN-Precision of both instructions and outputs decreases from nearly 1 to close to 0, indicating progressive degradation in task quality. This trend is consistent with the intuition that a higher masking probability leads to lower-fidelity synthetic data. In contrast, when $p \leq 0.9$, the KNN-Recall for both instructions and outputs remains relatively high (above 0.7), suggesting that Blank Filling largely preserves task diversity. For Oversampling (shown in Fig. 3c), as r increases, the output KNN-Recall drops significantly, while the KNN-Precision of both instructions and outputs remains near perfect, consistent with the



(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 5: Performance of In-Context Generation under BFCL.

intuition that duplication reduces task diversity while largely preserving task quality. Furthermore, as shown in Figs. 3b, 3d, 4a and 4b, increasing p or r leads to performance degradation for Blank Filling and Oversampling across nearly all evaluation metrics. This trend aligns with our intuition that higher masking probabilities or heavier duplication results in worse synthetic data.

- *Adding more in-context examples does not necessarily improve In-Context Generation, whereas Randomizing in-context examples helps synthetic data better match the semantic and statistical distributions of the real tasks.* For In-Context Generation (see Figs. 4c and 5a), increasing the number of in-context examples k does not consistently improve synthetic data performance. For instance, on the T1 dataset, $k = 1$ outperforms $k = 3$ under several metrics, including KNN-Recall, KND, AM, and TDD. We then fix the number of in-context examples k and compare fixed versus randomized example selection across generations in Figs. 4d and 5b. Randomizing in-context examples consistently improves performance on instruction- and output-level metrics, including task quality and diversity, as well as semantic and statistical properties.

- *No single metric can fully characterize synthetic data performance.* A method with one parameter setting may outperform another on some metrics while underperforming on others, for example, KNN-Precision versus KNN-Recall in Fig. 4c. Another example is given by the tool-call evaluation metrics and RD. In Fig. 4b, tool call metrics degrade as r increases, indicating a growing discrepancy in tool-planning patterns between real and synthetic tasks. In contrast, the downstream metric RD: Tool Call remains strong, since higher-capacity models continue to outperform lower-capacity ones even when planning patterns shift, resulting in largely consistent model rankings. This further motivates the need for a comprehensive framework such as ESDAE.

5 Conclusion

In this paper, we introduced ESDAE, a multi-axis framework for evaluating how faithfully synthetic trajectory datasets match real data across task instructions and responses, tool calls, final outputs, and downstream agent performance. Through experiments on recent agent benchmarks with controlled synthetic generation schemes, ESDAE consistently detects fine-grained failures in fidelity and diversity. Overall, the results show that no single metric is sufficient to characterize synthetic benchmark quality, motivating comprehensive evaluations such as ESDAE before relying on synthetic data for agent testing. For future work, we plan to extend ESDAE to explicitly measure the data augmentation effect of synthetic trajectories by designing additional metrics that quantify marginal gains in coverage, robustness, and generalization improvements.

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.

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.

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.

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.

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.

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*.

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.

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*.

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.

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*.

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.

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. τ-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.

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.

A Prompts for LLM-as-a-Judge in Downstream Evaluation

A.1 Prompts for 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):
```

A.2 Prompts for 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):
```

B Prompts for LLM-Based Synthetic Data Generation Methods

B.1 Prompts for T1

B.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:
```

B.1.2 In-Context Generation

System Prompt: Same as the system prompt in Section B.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:
```

B.2 Prompts for BFCL

B.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:
```

B.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:
```

C Detailed Evaluation Results

Table 2: Synthetic data generation evaluation results on ESDAE under T1.

Method	Parameter	Instruction Eval		Tool Call Eval		Output Eval		Downstream Eval	
		KND ↓	AM ↓	TUM ↓	3-Step Planning ↓	KNN-Precision ↑	KNN-Recall ↑	TDD ↓	RD ↑
Blank Filling	$p = 0$	0	0	0	0	1	1	0	1.0
	$p = 0.1$	0.019	0.113	0.018	0.027	0.640	0.933	0.052	1.0
	$p = 0.3$	0.025	0.096	0.074	0.134	0.587	0.813	0.090	1.0
	$p = 0.5$	0.037	0.213	0.049	0.048	0.498	0.791	0.056	1.0
	$p = 0.7$	0.035	0.295	0.022	0.167	0.142	0.796	0.147	0.5
	$p = 0.9$	0.059	0.564	0.041	0.135	0.240	0.693	0.160	1.0
	$p = 1$	0.075	0.746	0.031	0.346	0.044	0.164	0.188	0.5
Oversampling	$r = 0$	0.004	0.051	0	0	0.994	0.923	0.095	1.0
	$r = 0.1$	0.012	0.073	0.009	0.085	0.984	0.818	0.048	1.0
	$r = 0.3$	0.038	0.229	0.023	0.086	0.953	0.693	0.117	0.5
	$r = 0.5$	0.060	0.351	0.031	0.079	0.984	0.724	0.053	1.0
	$r = 0.7$	0.087	0.511	0.058	0.043	0.971	0.631	0.156	1.0
	$r = 0.9$	0.110	0.658	0.059	0.127	0.882	0.489	0.165	0.5
	$r = 1$	0.122	0.750	0.094	0.238	0.992	0.004	0.183	1.0
Generative Models with In-Context Examples	$k = 0$	0.075	0.746	0.031	0.346	0.044	0.164	0.033	0.5
	$k = 1$ (fixed)	0.065	0.645	0.036	0.040	0.067	0.311	0.076	0.5
	$k = 1$ (random)	0.054	0.444	0.017	0.008	0.027	0.729	0.126	1.0
	$k = 3$ (fixed)	0.117	0.866	0.011	0.016	0.311	0.271	0.062	0.5
	$k = 3$ (random)	0.021	0.523	0.038	0.034	0.062	0.853	0.121	0.5
	$k = 5$ (fixed)	0.031	0.875	0.005	0.070	0.369	0.524	0.124	0.5
	$k = 5$ (random)	0.019	0.460	0.010	0.085	0.471	0.707	0.138	1.0

Table 3: Synthetic data generation evaluation results on `ESDAE` under BFCL.

Method	Parameter	Instruction Eval				Tool Call Eval			
		KND ↓	AM ↓	KNN-Precision ↑	KNN-Recall ↑	TUM ↓	TCNM ↓	2-Step Planning ↓	3-Step Planning ↓
Blank Filling	$p = 0$	0	0	1	1	0	0	0	0
	$p = 0.1$	0.0871	0.665	0.715	0.900	0.4522	6.805	0.6840	0.5951
	$p = 0.3$	0.0793	1.030	0.675	0.905	0.3877	5.445	0.6835	0.6321
	$p = 0.5$	0.0869	1.320	0.640	0.920	0.4382	5.895	0.7085	0.6227
	$p = 0.7$	0.1055	1.470	0.430	0.910	0.6557	16.335	0.6540	0.5671
	$p = 0.9$	0.1630	1.445	0.225	0.805	0.4714	6.335	0.7598	0.5829
	$p = 1$	0.1238	0.695	0.245	0.290	0.6005	6.515	0.7469	0.5851
Oversampling	$r = 0$	0.0197	0.110	1.000	0.875	0.1116	0.285	0.1560	0.1476
	$r = 0.1$	0.0235	0.130	0.995	0.980	0.1503	0.485	0.1223	0.0847
	$r = 0.3$	0.0331	0.305	0.995	0.965	0.3841	1.395	0.2614	0.1796
	$r = 0.5$	0.0530	0.525	0.995	0.920	0.5832	2.175	0.3712	0.2583
	$r = 0.7$	0.0803	0.725	0.995	0.910	0.7390	2.985	0.4664	0.3150
	$r = 0.9$	0.1060	0.955	0.995	0.950	0.8500	3.955	0.5566	0.3681
	$r = 1$	0.1179	1.060	1.000	0.005	0.9116	4.290	0.6275	0.4111
Generative Models with In-Context Examples	$k = 0$	0.1238	0.695	0.245	0.290	0.6005	6.515	0.7469	0.5851
	$k = 1$ (fixed)	0.1033	0.330	0.450	0.175	0.6133	4.820	0.7520	0.5862
	$k = 1$ (random)	0.1293	0.570	0.370	0.760	0.5708	5.045	0.6983	0.5962
	$k = 3$ (fixed)	0.0794	0.375	0.385	0.235	0.5818	10.770	0.6442	0.5710
	$k = 3$ (random)	0.1147	0.515	0.500	0.765	0.5572	7.430	0.6989	0.5743
	$k = 5$ (fixed)	0.0810	0.365	0.405	0.330	0.4569	5.360	0.6905	0.6208
	$k = 5$ (random)	0.0640	0.312	0.505	0.720	0.4674	5.685	0.7454	0.5765