AGENTOHANA: DESIGN UNIFIED DATA AND TRAINING PIPELINE FOR EFFECTIVE AGENT LEARNING

Jianguo Zhang*, Tian Lan*, Rithesh Murthy, Zhiwei Liu, Weiran Yao, Juntao Tan, Thai Hoang, Liangwei Yang, Yihao Feng, Zuxin Liu, Ming Zhu, Tulika Awalgaonkar, Juan Carlos Niebles, Silvio Savarese, Shelby Heinecke, Huan Wang, Caiming Xiong
Salesforce Research, USA
jianguozhang@salesforce.com, tian.lan@salesforce.com

ABSTRACT

Autonomous agents powered by large language models (LLMs) have garnered significant research attention. However, fully harnessing the potential of LLMs for agent-based tasks presents inherent challenges due to the heterogeneous nature of diverse data sources featuring multi-turn trajectories. In this paper, we introduce AgentOhana as a comprehensive solution to address these challenges. AgentOhana aggregates agent trajectories from distinct environments, spanning a wide array of scenarios. It meticulously standardizes and unifies these trajectories into a consistent format, streamlining the creation of a generic data loader optimized for agent training. Leveraging the data unification, our training pipeline maintains equilibrium across different data sources and preserves independent randomness across devices during dataset partitioning and model training. Additionally, we present xLAM-v0.1, a large action model tailored for AI agents, which demonstrates exceptional performance across various benchmarks. Begin the exploration at https://github.com/SalesforceAIResearch/xLAM.

1 INTRODUCTION

Large language models (LLMs) have shown strong abilities in code generation, mathematical reasoning, conversational AI, and AI agents [OpenAI, 2023; Jiang et al., 2023; Zhang et al., 2023; Liu et al., 2023; Nijkamp et al., 2023]. Among these, LLM-powered autonomous agents are gaining increasing attention. Recent frameworks for LLM agents, such as AutoGPT [Gravitas, 2023], OpenAgent [Xie et al., 2023], BOLAA [Liu et al., 2023b], XAgent [Team, 2023], and LangChain [Chase, 2023], have been designed to support agent tasks, and they have attracted significant interest in the open-source community.

Nevertheless, many existing agents are powered by closed-source LLM APIs such as GPT-4 [OpenAI, 2023] and Gemini [Team et al., 2023], mainly because most open-source models struggle to perform long-horizon reasoning and handle complex agent tasks [Liu et al., 2023a,b]. Recently, there have been ongoing efforts on training open-source models instead of relying solely on commercialized APIs. For instance, AgentLM (Zeng et al., 2023), Lemur (Xu et al., 2023) and Lumos (Yin et al., 2023) are trained for agents based on Llama-2 family [Touvron et al., 2023], along with special reasoning, planning and acting prompts design such as ReAct (Yao et al., 2023) and Self-Reflection (Shinn et al., 2023; Madaan et al., 2023; Wang et al., 2023b) to enhance the abilities. On the same, there are works to open-source agent relevant data and train agent models such as ToolLlama (Qin et al., 2023), ToolAlpaca (Tang et al., 2023), Lumos (Yin et al., 2023) and API-bank (Li et al., 2023) to enhance abilities on reasoning, tool usages and plannings. They have shown impressive performance on agent relevant tasks.

However, navigating the data landscape for LLM agents becomes increasingly intricate when dealing with non-standardized data formats sourced from diverse dataset collections, especially those

* Equal contributions.
featuring interactions of multi-turns, as commonly observed in agent-relevant data. The heterogeneity in data structures, syntaxes, labeling conventions, and processing methods across datasets poses a formidable challenge, complicating the training and fine-tuning processes of LLMs. The lack of standardized formats introduces complexities in harmonizing disparate data sources, leading to potential biases and inconsistencies. Addressing these challenges requires developing robust preprocessing pipelines, ensuring unification and compatibility across varied data formats, and implementing strategies to mitigate biases that may arise from non-standardized representations. With the increasing demand for comprehensive and diverse datasets, establishing effective methods to manage non-standardized data formats is crucial for ensuring the robust performance of LLM agents across a spectrum of applications.

In this work, we bridge the existing gap by building a comprehensive agent data collection and training pipeline, named AgentOhana. Drawing inspiration from the notable achievements of DialogStudio (Zhang et al., 2023) and FLAN (Longpre et al., 2023) in the realms of Conversational AI and instruction-based fine-tuning, AgentOhana is designed to accommodate the wide variety of data structures and formats encountered in LLM agent trajectories. It employs specialized processes to transform disparate data into a uniform format, achieving seamless integration across multiple sources. Furthermore, the collection undergoes a meticulous filtering process to ensure high-quality trajectories, thereby introducing an extra layer of quality control. Leveraging the data standardization and unification, our training pipeline preserves independent randomness across devices during both dataset partitioning and model training, thus avoiding the inadvertent introduction of biases into the training process. This comprehensive approach guarantees that AgentOhana not only unifies trajectories across environments but also enhances the overall quality and the reliability of the collected data, as well as the performance and the robustness of the model. Our approach ensures that AgentOhana serves as a versatile and accessible resource for the research community, streamlining the development process for future applications. The contributions of this paper are as follows:

- **Innovative Solution to Data Heterogeneity**: We introduce AgentOhana, a pioneering platform designed to address the complex challenges associated with the consolidation of heterogeneous data sources pertaining to multi-turn LLM agent trajectories. This contribution represents a critical step forward in overcoming the obstacles of data diversity and fragmentation.

- **Extensive Environmental Coverage**: AgentOhana distinguishes itself by incorporating agent data from ten distinct environments, spanning a comprehensive array of scenarios. This diverse collection facilitates a broad spectrum of research opportunities, enabling investigations into various aspects of agent behavior and interaction.

- **Data Standardization and Unification**: A core achievement of this work is the systematic standardization and unification of LLM agent data into a consistent format. This process has enabled the creation of a generic data loader, optimizing the dataset for agent training that maintains equilibrium across different data sources and preserves independent randomness across devices.

- **Large Agent Model**: We have developed XLAM-v0.1, a robust large action model tailored for AI agents. Demonstrating strong performance across five rigorous benchmarks, XLAM-v0.1 showcases the potential of AgentOhana in facilitating the training of high-performing AI agents. All the code, dataset, and model will be made open source upon publication.

## 2 Methodology

As illustrated in the workflow of AgentOhana shown in Figure 1, we adopt a homogeneous multi-turn data format designed to consolidate trajectories from heterogeneous data sources. Additionally, we introduce a method called AgentRater to assess and filter agent trajectories based on public or close-world models. Finally, we adopt a generic dataloader as a central component to enable smooth integration of various datasets into a distributed training process.

### 2.1 Heterogeneity of Various Datasets

The formats of agent data vary significantly across different environments, posing significant difficulties and challenges in unifying data, training, and analyzing models. As illustrated in row 1 of Figure 2, trajectories from two distinct environments show markedly different data organization.
methods, a phenomenon observed universally across different environments. For instance, the HotspotQA environment [Liu et al., 2023b] consolidates the whole target trajectory into a single string under the prompt key. This design requires efforts to retrieve user query, Thought, Model Action: $i$ along with Env Observation: $i$ for each step $i \in [1, N]$ from a single string. Conversely, ToolAlpaca requires the identification and matching of prompt inputs, model outputs, and observations at each step, followed by the accurate aggregation of trajectory history prior to proceeding to the next step. Appendix A shows more examples of the original trajectories from four environments.

2.2 HOMOGENEOUS MULTI-TURN AGENT TRAJECTORY STANDARDIZATION

To address the challenges identified, we propose a unified agent data format, as depicted in row 2 of Figure 2, showcasing our proposed unified trajectory data format. We construct a homogeneous JSON dictionary format to encapsulate all relevant content of each trajectory. Concretely, our format incorporates all important elements such as user query to store the initial user query, model name to identify the corresponding model and score to log the available model performance score. These elements can be used to differentiate between models and are poised to facilitate the development of pairwise samples for cutting-edge training methodologies such as DPO [Rafailov et al., 2023], self-reward [Yuan et al., 2024] and AI feedback [Guo et al., 2024] LLMs. Additionally, we save auxiliary trajectory information or specific notes into other information, providing a reference for further analysis or model improvement initiatives.

To enhance the preservation and analysis of multi-turn agent trajectory information, we propose a structured definition of a step that captures the details of each interaction turn. A step comprises three main components: input, output, and next observation. The input component consolidates the current prompt and a historical record of past interactions, serving as a comprehensive context for the interaction. The output component captures the model’s predictions, detailing its decision-making and planning. The next observation component records the environment’s feedback, essential for the feedback loop and system adaption.

Our framework employs a predefined method for aggregating interaction history within the input component, effectively concatenating inputs and outputs from previous steps to construct a comprehensive context. Specifically, at the $i^{th}$ step, the input is formatted as input of step 1, Action: output of step 1, Observation: next observation of step 1, ..., input of step $i$-1, Action: output of step $i$-1, Observation: next observation of step $i$-1. This approach ensures a detailed chronological account of interactions, facilitating a nuanced understanding of the trajectory.

While this default aggregation strategy of input is integral to our framework, we also accommodate the customization of data compilation methods. Users are encouraged to explore alternative strategies that exploit the structured input, output, and next observation components, tailoring the data format to their specific research or application needs. Figure 2, row 2, illustrates the transformation
of trajectories from environments such as HotpotQA and ToolAlpaca using our defined step structure, where output aligns with the format specifications in the initial prompt input, demonstrating the framework’s adaptability and practical utility.

By standardizing the capture of interactions between agents and their environments, this methodology not only facilitates a uniform approach to data documentation but also enhances the potential for in-depth analysis and refinement of AI models. This is achieved by providing a granular view of the agent interactions, decision-making process and its results, thereby enabling a more nuanced understanding and improvement of model performance.

2.3 AGENTRATER

Agent trajectories represent a complex subset of data distinct from general and straightforward instructional data. While datasets like Alpaca (Dubois et al. 2023) features single-turn examples, and LMSYS-Chat (Zheng et al. 2023a) includes dialogues averaging around two turns, these generally encompass simpler interaction patterns. DialogStudio (Zhang et al. 2023) does offer multi-turn
Given above json dict contains a trajectory, please rate according to the overall accuracy and efficiency of the model output to the input and observation.

Here are keys for the json dict file:

- ### Initial Input of Step 1 ###: this indicates the initial input of the first step
- ### Model Output of Step 1 ###: this indicates the model output based on the Initial Input of the first step
- ### Observation of Step 1 ###: this indicates the observation from environments based on the Model Output of the first step
- ### Model Output of Step 2 ###: this indicates the model output based on the observation of the first step and the previous history
- ### Observation of Step 2 ###: this indicates the observation from environments based on the Model Output of the second step
- ### Model Output of Step 3 ###: this indicates the model output based on the observation of the second step and the previous history

So on and so forth for the rest of the steps.

Each assistant receives a score on a scale of 0 to 5, where a higher score indicates higher level of the overall model accuracy and efficiency. Please provide your evaluation as follows:

1. A single line containing a numerical score indicating the evaluation.
2. In the subsequent line, please provide a detailed explanation supporting your score, focusing on the criteria of accuracy and efficiency. Ensure your evaluation avoids any potential bias.

---

Figure 3: A prompt template for the AgentRater, where an open-source model (e.g., Mistral) or close-world API (e.g., ChatGPT) will rate the whole agent trajectory based on criterias and then assign a score from 0-5.

 dialogue examples, these are primarily confined to conversations between the user and the system, lacking interactions with external environments.

In contrast, agent trajectories delve into more intricate scenarios where an agent interacts with complex environments such as websites, APIs, and tools. This complexity is heightened by the agent’s capacity to communicate with other agents, navigate through diverse interfaces, and undertake tasks that require a sequence of interactions rather than single or limited exchanges.

The challenge with agent trajectories extends to the evaluation of performance and quality. While some environments offer rewards as feedback for an agent’s trajectory, such rewards are often tied to the task’s final outcome rather than reflecting the quality of the trajectory itself. Consequently, a high reward does not necessarily indicate a flawless trajectory. For example, an agent might generate invalid actions during intermediate steps of a task. Here is partial trajectory from the Webshop environment (Yao et al., 2022):

- model output of step 4: “click[old world style]”, observation of step 4: “You have clicked old world style.”;
- model output of step 6: “”, observation of step 6: “Invalid action!”;
- model output of step 7: “click[Buy Now]”, observation of step 7: “Your score (min 0.0, max 1.0): 1.0”, where the reward is 1.0, but the agent randomly clicks other invalid buttons in Webshop website or generates empty responses before buying an item.

To mitigate the issues, we design a method, named AgentOhana to rate the agent trajectory based on strong public models such as Mistral (Jiang et al., 2023) or close-world APIs such as ChatGPT (OpenAI, 2023). Different with approaches (Zhang et al., 2023; Chen et al., 2023) where they rate each triplet of (instruction, input, response) pair on general instruction data as there are usually single-turn or short conversations, we rate the whole trajectory on agent data. Figure 3 shows a corresponding prompt template, where we rate the trajectory with a score 0-5 and an explanation, and they can be used to further develop better AgentRater models. Table 2.3 shows the statistics of AgentOhana.

2.4 GENERIC DATALOADER

As the protocol involves loading data in the correct format for the trainer, the implementation of a generic dataloader becomes crucial in harmonizing the entire data and training pipeline. This dataloader serves as a central component, facilitating seamless integration of diverse datasets into the training process. Its generic nature ensures flexibility and compatibility across various data formats, enabling efficient data ingestion before feeding into the training framework.
<table>
<thead>
<tr>
<th>Environments &amp; Data</th>
<th>#Sampled Trajs</th>
<th>#Filtered Trajs</th>
<th>#Average Turns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Webshop (Yao et al., 2022)</td>
<td>11200</td>
<td>2063</td>
<td>6.8</td>
</tr>
<tr>
<td>AlfWorld (Shridhar et al., 2021)</td>
<td>954</td>
<td>336</td>
<td>13.5</td>
</tr>
<tr>
<td>HotpotQA (Yang et al., 2018)</td>
<td>1740</td>
<td>402</td>
<td>4.8</td>
</tr>
<tr>
<td>ToolBench (Qin et al., 2023)</td>
<td>83771</td>
<td>30319</td>
<td>3.1</td>
</tr>
<tr>
<td>ToolAlpaca (Tang et al., 2023)</td>
<td>3936</td>
<td>3399</td>
<td>2.5</td>
</tr>
<tr>
<td>Operating System (Liu et al., 2023a)</td>
<td>647</td>
<td>195</td>
<td>3.9</td>
</tr>
<tr>
<td>APIbank (Li et al., 2023)</td>
<td>33415</td>
<td>4902</td>
<td>1.0</td>
</tr>
<tr>
<td>DataBase (Liu et al., 2023a)</td>
<td>6376</td>
<td>538</td>
<td>2.0</td>
</tr>
<tr>
<td>Mind2Web (Deng et al., 2023)</td>
<td>23378</td>
<td>122</td>
<td>1.0</td>
</tr>
<tr>
<td>Knowledge Graph (Liu et al., 2023a)</td>
<td>2501</td>
<td>324</td>
<td>6.0</td>
</tr>
<tr>
<td>AgentOhana</td>
<td>167918</td>
<td>42600</td>
<td>3.1</td>
</tr>
</tbody>
</table>

Table 1: Statistics of AgentOhana. AgentOhana consists of data from 10 different environments. #Sampled Trajs indicates the trajectories sampled and filtered from the original environment, #Filtered Trajs indicates the filtered trajectories with the AgentRater score \( \geq 4.0 \), #Average Turns indicates average number of turns in the filtered trajectories. Among the environments, Operating System, DataBase, Mind2Web and Knowledge Graph are derived from (Zeng et al., 2023). Additionally, AgentOhana integrates partial general instruction data sourced from DialogStudio (Zhang et al., 2023), this subset is not represented in the table.

2.4.1 AGENTMODELDATASETBASE

We have introduced the AgentModelDatasetBase class to streamline common tasks such as prompt formatting while providing a virtual template for creating individual datasets. While loading data may appear straightforward at this stage, there are still several intricate issues to address. For instance, in addition to employing a machine-assisted filter as detailed in Section 2.3, users may prefer a certain level of control over data quality. Moreover, the randomness of data batching from different datasets could pose challenges, particularly when dealing with distributed training among multiple devices, which requires a comprehensive approach to ensure robust dataset management.

2.4.2 CUSTOM DATASETS CREATION

Individual dataset As depicted in the following example, we begin by loading individual raw data prepared from Section 2.2 typically via the streaming mode. Then, for each dataset, we can optionally introduce the filter generator to further customize the selection of data just before feeding it into the trainer. For instance, in the following example, data with relatively low scores will be further evaluated and removed. Finally, we shuffle this dataset randomly with controlled seeding to ensure randomness and reproducibility.

```python
class WebshopMultiTurn(AgentModelDatasetBase):
    # we can further filter out trajectories at this stage
    @staticmethod
    def _high_score_filter_generator(data, score=0.8):
        for d in data:
            if d["score"] >= score:
                yield {"prompt": d["input"], "chosen": d["output"]}

    def create_dataset(self, seed=None):
        train_data = load_dataset(...)  # streaming = self.args.streaming,
        train_data = IterableDataset.from_generator(self._high_score_filter_generator,
                                                    gen_kwargs={"data": train_data})
        train_data = train_data.shuffle(seed=seed, buffer_size=1000)
        return train_data
```
Combined Datasets  Our primary focus in combining datasets lies in ensuring randomness during the batching process, particularly when dealing with multiple datasets. To achieve this, we employ the `init_device_seed` function to diversify the controlled seeds based on the process ID when data parallelism is utilized across multiple devices. By carefully managing the seeding process, we aim to maintain a balanced distribution of data by partitioning, shuffling and interleaving data across devices while preserving randomness, thus enhancing the robustness and reproducibility of our training procedure.

```python
toolbench_multi_turn = ToolBenchMultiTurn(tokenizer, script_args)
webshop_multi_turn = WebshopMultiTurn(tokenizer, script_args)
...
data = [toolbench_multi_turn, webshop_multi_turn, ...]
sample_probs = [0.1, 0.1, ...]

# a device-dependent seeding will be utilized based on the combination of
# the given default seed and the process ID
seed = init_device_seed(seed=42)
train_dataset, eval_dataset = interleave_data(data, sample_probs, seed)
```

3 Experiments

3.1 Training

We have developed xLAM-v0.1, a large, robust action model for AI Agent. xLAM-v0.1 is initialized from the pre-trained Mixtral-8x7B-Instruct-v0.1 model (Jiang et al., 2024). To execute this fine-tuning process, we adopt a supervised fine-tuning approach and capitalize on the capabilities of the AgentOhana collection. Our fine-tuning procedure is conducted concurrently on 8 Nvidia H100 GPUs, utilizing the 4-bit quantized QLoRA framework (Dettmers et al., 2023). Throughout the fine-tuning process, our model traverses each individual dataset approximately 3 times on average. This multi-epoch training regimen facilitates comprehensive exposure to the dataset, enabling the model to effectively learn intricate patterns present in the training data.

3.2 Benchmarks

In the subsequent sections, we will present experimental evaluations conducted across five benchmarks: Webshop (Yao et al., 2022), HotpotQA (Yang et al., 2018), ToolEval (Qin et al., 2023), ToolQuery (Ma et al., 2024) and MINT-Bench (Wang et al., 2023a). Further details about these benchmarks and their associated metrics can be found in Appendix B.

Webshop creates an online shopping environment simulating product purchases, while HotpotQA involves multi-hop question-answering tasks requiring logical reasoning across Wikipedia passages via the Wikipedia API. We follow BOLAA’s framework (Liu et al., 2023b) and use Average Reward for Webshop and F1 Score for HotpotQA, to measure model performance. In Webshop, the reward metric assesses model accuracy based on the attributes overlapping between the purchased and the ground-truth items, while in HotpotQA, it quantifies the accuracy of agent-predicted answers against ground-truth responses.

ToolEval is designed for real-time assessment of functional calling capabilities via RapidAPI, initially utilizing ChatGPT as its evaluator. We follow the paper to use Pass Rate as the evaluation metric and present our findings at the first level of the ToolEval evaluation, focusing on three distinct scenarios: (1) unseen instructions with the same set of tools, (2) unseen tools within previously seen categories, and (3) unseen tools from entirely new categories that have not been seen before.

ToolQuery contains three distinct environments: Weather, Movie and Academia environments. It is designed to measure an agent’s proficiency in utilizing tools to retrieve, access and query information about weather, movie and computer science academia. It uses Success Rate and Progress Rate to evaluate the overall performance and the progressive performance over interactive turns.

MINT-Bench benchmark focuses on reasoning, coding, and decision-making through a diverse set of established evaluation datasets. The benchmark asks LLMs to solve tasks with different interaction
Table 2: Average reward on the Webshop environment. **Bold** and *Underline* results denote the best result and the second best result for each setting, respectively.

<table>
<thead>
<tr>
<th>LLM LAA Architecture</th>
<th>ZS</th>
<th>ZST</th>
<th>ReAct</th>
<th>PlanAct</th>
<th>PlanReAct</th>
<th>BOLAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Llama-2-70b-chat [<em>Touvron et al., 2023</em>]</td>
<td>0.0089</td>
<td>0.0102</td>
<td>0.4273</td>
<td>0.2809</td>
<td>0.3966</td>
<td>0.4986</td>
</tr>
<tr>
<td>Vicuna-33b [<em>Zheng et al., 2023b</em>]</td>
<td>0.1527</td>
<td>0.2122</td>
<td>0.1971</td>
<td>0.3766</td>
<td>0.4032</td>
<td>0.5618</td>
</tr>
<tr>
<td>Mixtral-8x7B-Instruct-v0.1 [<em>Jiang et al., 2024</em>]</td>
<td>0.4634</td>
<td>0.4592</td>
<td>0.5638</td>
<td>0.4738</td>
<td>0.3339</td>
<td>0.5342</td>
</tr>
<tr>
<td>GPT-3.5-Turbo</td>
<td>0.4851</td>
<td>0.5058</td>
<td>0.3047</td>
<td>0.4930</td>
<td>0.5436</td>
<td>0.6354</td>
</tr>
<tr>
<td>GPT-3.5-Turbo-Instruct</td>
<td>0.3785</td>
<td>0.4195</td>
<td>0.4377</td>
<td>0.3604</td>
<td>0.4851</td>
<td>0.5811</td>
</tr>
<tr>
<td>GPT-4-0613</td>
<td>0.3912</td>
<td>0.3971</td>
<td>0.3714</td>
<td>0.3195</td>
<td>0.3039</td>
<td>0.5811</td>
</tr>
<tr>
<td>xLAM-v0.1</td>
<td>0.5002</td>
<td>0.4783</td>
<td>0.4616</td>
<td>0.7950</td>
<td>0.4635</td>
<td>0.6129</td>
</tr>
</tbody>
</table>

Table 3: Average reward on the HotpotQA environment. **Bold** and *Underline* results denote the best result and the second best result for each setting, respectively.

<table>
<thead>
<tr>
<th>LLM LAA Architecture</th>
<th>ZS</th>
<th>ZST</th>
<th>ReAct</th>
<th>PlanAct</th>
<th>PlanReAct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mixtral-8x7B-Instruct-v0.1 [<em>Jiang et al., 2024</em>]</td>
<td>0.3912</td>
<td>0.3971</td>
<td>0.3714</td>
<td>0.3195</td>
<td>0.3039</td>
</tr>
<tr>
<td>GPT-3.5-Turbo</td>
<td>0.4196</td>
<td>0.3937</td>
<td>0.3868</td>
<td>0.4182</td>
<td>0.3960</td>
</tr>
<tr>
<td>GPT-4-0613</td>
<td>0.2500</td>
<td>0.2500</td>
<td>0.2500</td>
<td>0.2500</td>
<td>0.2500</td>
</tr>
<tr>
<td>xLAM-v0.1</td>
<td>0.5077</td>
<td>0.5650</td>
<td>0.5200</td>
<td>0.5050</td>
<td>0.5050</td>
</tr>
</tbody>
</table>

Table 4: Pass Rate on ToolEval on three distinct scenarios. **Bold** and *Underline* results denote the best result and the second best result for each setting, respectively.

<table>
<thead>
<tr>
<th>Unseen Insts &amp; Same Set</th>
<th>Unseen Tools &amp; Seen Cat</th>
<th>Unseen Tools &amp; Unseen Cat</th>
</tr>
</thead>
<tbody>
<tr>
<td>TooLlama V2 [<em>Qin et al., 2023</em>]</td>
<td>0.4385</td>
<td>0.4300</td>
</tr>
<tr>
<td>GPT-3.5-Turbo-0125</td>
<td>0.5000</td>
<td>0.5150</td>
</tr>
<tr>
<td>GPT-4-0125-preview</td>
<td>0.5462</td>
<td>0.5100</td>
</tr>
<tr>
<td>xLAM-v0.1</td>
<td>0.5677</td>
<td>0.5650</td>
</tr>
</tbody>
</table>

3.3 WEBSHOP

Table 2 showcases the performance of our model within the Webshop environment. xLAM-v0.1 consistently outperforms both GPT-3.5-Turbo and GPT-3.5-Turbo-Instruct across all agent configurations. Moreover, it surpasses GPT-4-0613 in five out of six settings, with the latter demonstrating superior planning capabilities but lower performance in reasoning, self-reflection, and multi-agent interactions. These findings underscore the robust and versatile capabilities of the xLAM model across a variety of agent scenarios.

3.4 HOTPOTQA

Table 3 details the results in the HotpotQA environment, highlighting xLAM’s superior performance relative to GPT-3.5-Turbo and Mixtral-8x7B-Instruct-v0.1 across all settings. While GPT-4-0613 exhibits a slight performance edge, our analysis on models’ predictions reveals that it typically identifies correct answers within four steps, suggesting that it may have been trained on a substantial corpus of relevant question-answering examples, thereby possessing enhanced domain-specific knowledge compared to its counterparts.

3.5 TOOL EVAL

Table 4 displays the results on ToolEval. xLAM-v0.1 surpasses both TooLlama V2 and GPT-3.5-Turbo-0125 across all evaluated scenarios, and outperforms GPT-4-0125-preview in two out of the
three settings. This performance indicates xLAM-v0.1’s superior capabilities in function calling and handling complex tool usage tasks. We posit that the model’s performance could be enhanced further through data augmentation involving a variety of prompts.

3.6 ToolQuery

<table>
<thead>
<tr>
<th>Model</th>
<th>Success Rate</th>
<th>Progress Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT-4</td>
<td>0.683</td>
<td>0.851</td>
</tr>
<tr>
<td>xLAM-v0.1</td>
<td>0.533</td>
<td>0.766</td>
</tr>
<tr>
<td>Claude2</td>
<td>0.483</td>
<td>0.735</td>
</tr>
<tr>
<td>GPT-3.5-Turbo</td>
<td>0.450</td>
<td>0.694</td>
</tr>
<tr>
<td>DeepSeek-67b [Bi et al., 2024]</td>
<td>0.400</td>
<td>0.714</td>
</tr>
<tr>
<td>GPT-3.5-Turbo-16k</td>
<td>0.317</td>
<td>0.591</td>
</tr>
<tr>
<td>Lemur-70b [Xu et al., 2023]</td>
<td>0.283</td>
<td>0.720</td>
</tr>
<tr>
<td>CodeLlama-34b [Roziere et al., 2023]</td>
<td>0.133</td>
<td>0.600</td>
</tr>
<tr>
<td>CodeLlama-13b [Roziere et al., 2023]</td>
<td>0.250</td>
<td>0.525</td>
</tr>
<tr>
<td>Llama2-70b [Touvron et al., 2023]</td>
<td>0.000</td>
<td>0.483</td>
</tr>
<tr>
<td>Mistral-7b [Jiang et al., 2023]</td>
<td>0.033</td>
<td>0.510</td>
</tr>
<tr>
<td>Vicuna-13b-16k [Chiang et al., 2023]</td>
<td>0.033</td>
<td>0.343</td>
</tr>
</tbody>
</table>

Table 5: Testing results on ToolQuery across three distinct environments.

Table 5 shows testing results on the unseen ToolQuery benchmark, where the baseline results are provided by the benchmark’s originators. Notably, xLAM-v0.1 achieves second-place performance across the benchmark’s three unique environments, surpassing commercial models such as Claude2 and GPT-3.5-Turbo, and other open-source alternatives. These results demonstrate that xLAM-v0.1 can effectively utilize tools for information access and querying, affirming its robustness and efficiency in complex tasks.

3.7 Ablation Study

Table 6 in Appendix C presents the ablation study outcomes for AgentRater within the Webshop and HotpotQA environments. Consistent with the findings reported by Chen et al. (2023), we discovered that training models on a smaller dataset of higher quality, as facilitated by AgentRater, enhances both the training efficiency and overall model performance.

3.8 MINT-Bench

Table 7 in Appendix D presents the testing results on the challenging and comprehensive MINT-Bench, with baseline comparisons drawn from the official leaderboard. The xLAM-v0.1 model secures the third rank in this rigorous benchmark, outperforming other agent-based models such as Lemur-70b-Chat-v1 and AgentLM-70b, as well as Claude-2 and GPT-3.5-Turbo-0613. These results highlight exceptional capability of xLAM to navigate the complexity of multi-turn interactions and task resolution.

4 Conclusion

In conclusion, the creation of AgentOhana represents a significant step forward in addressing the challenges inherent in consolidating diverse data of the multi-turn LLM agent trajectories. Through the development of unified data and training pipelines, we have established a framework capable of handling the intricacies of various data structures and formats, thereby ensuring compatibility across a multitude of environments. By providing a comprehensive and high-quality dataset, we aim to empower researchers and practitioners to push the boundaries of AI capabilities, ultimately contributing to the advancement of autonomous agents powered by LLMs.
REFERENCES


Figure 4 shows original trajectories from four environments.

![Figure 4: Original trajectories from (A) HotPotQA, (B) Webshop, (C) ToolAlpaca, (D) ToolBench.](image_url)
B Benchmarks

Webshop (Yao et al., 2022) creates an online shopping environment simulating product purchases, while HotpotQA (Yang et al., 2018) involves multi-hop question-answering tasks requiring logical reasoning across Wikipedia passages via the Wikipedia API. We adopt BOLAA’s framework (Liu et al., 2023b), comprising five single-agent settings and a multi-agent scenario, to evaluate model performance. For the Webshop benchmark, BOLAA comprises 900 user queries, of which we serve 200 as a test subset. For HotpotQA, 300 user questions are sampled into three difficulty levels—easy, medium, and hard—with each category containing 100 questions. These questions are exclusively reserved for model testing to ensure a rigorous evaluation process. We use BOLAA’s evaluation metrics, average reward for Webshop and F1 score for HotpotQA, to measure model performance. In Webshop, the reward metric assesses model accuracy based on the attributes overlapping between the purchased and the ground-truth items, while in HotpotQA, it quantifies the accuracy of agent-predicted answers against ground-truth responses.

ToolEval (Qin et al., 2023) is designed for real-time assessment of functional calling capabilities via RapidAPI, initially utilizing GPT-3.5-Turbo-16k as its evaluator. However, after careful investigation, we found GPT-3.5-Turbo-16k unreliable for assessing complex function calls and tool usage scenarios. Consequently, we switched to GPT-4-0125-preview as our primary evaluator. Due to the real-time nature of these evaluations, APIs may experience downtime or timeouts, leading to inconsistency in model comparisons across different time frames. To address this, all models are evaluated within the same time frame. Our evaluation employs the default depth-first search-based decision tree methodology, augmented by the Pass Rate metric to assess an LLM’s ability to execute instructions, a fundamental criterion for optimal tool usage. We present our findings at the first level of the ToolEval evaluation, focusing on three distinct scenarios: (1) unseen instructions with the same set of tools, (2) unseen tools within previously seen categories, and (3) unseen tools from entirely new categories that have not been seen before.

ToolQuery (Ma et al., 2024) contains three distinct environments: Weather, Movie and Academia environments. It is designed to measure an agent’s proficiency in utilizing tools to retrieve, access and query information about weather, movie and computer science academia. It uses success rate and progress rate to evaluate the overall performance and the progressive performance over interactive turns.

MINT-Bench (Wang et al., 2023a) evaluates LLMs’ ability to solve tasks with multi-turn interactions by using tools and leveraging natural language feedback. The benchmark focuses on reasoning, coding, and decision-making through a diverse set of established evaluation datasets, and carefully curate them into a compact subset for efficient evaluation. The benchmark asks LLMs to solve tasks with different interaction limits from 1 to 5 step and quantify LLMs’ tool-augmented task-solving capability by absolute performance success rate, which measures the percentage of successful task instances as a function of interaction steps.

C Ablation Study

<table>
<thead>
<tr>
<th>LLM</th>
<th>LAA Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ZS</td>
</tr>
<tr>
<td>xLAM-v0.1</td>
<td>0.5201</td>
</tr>
<tr>
<td>w/o AgentRater</td>
<td>0.3998</td>
</tr>
<tr>
<td>xLAM-v0.1</td>
<td>0.3492</td>
</tr>
<tr>
<td>w/o AgentRater</td>
<td>0.5138</td>
</tr>
</tbody>
</table>

Table 6: Average reward on the Webshop (Row 1) and HotpotQA environment (Row 2).

D MINT-Bench
<table>
<thead>
<tr>
<th>Model</th>
<th>1-step</th>
<th>2-step</th>
<th>3-step</th>
<th>4-step</th>
<th>5-step</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT-4-0613</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>nan</td>
<td>69.45</td>
</tr>
<tr>
<td>Claude-Instant-1</td>
<td>12.12</td>
<td>32.25</td>
<td>39.25</td>
<td>44.37</td>
<td>45.90</td>
</tr>
<tr>
<td>xLAM-v0.1</td>
<td>4.10</td>
<td>28.50</td>
<td>36.01</td>
<td>42.66</td>
<td>43.96</td>
</tr>
<tr>
<td>Claude-2</td>
<td>26.45</td>
<td>35.49</td>
<td>36.01</td>
<td>39.76</td>
<td>39.93</td>
</tr>
<tr>
<td>Lemur-70b-Chat-v1 (Xu et al., 2023)</td>
<td>3.75</td>
<td>26.96</td>
<td>35.67</td>
<td>37.54</td>
<td>37.03</td>
</tr>
<tr>
<td>GPT-3.5-Turbo-0613</td>
<td>2.73</td>
<td>16.89</td>
<td>24.06</td>
<td>31.74</td>
<td>36.18</td>
</tr>
<tr>
<td>AgentLM-70b (Zeng et al., 2023)</td>
<td>6.48</td>
<td>17.75</td>
<td>24.91</td>
<td>28.16</td>
<td>28.67</td>
</tr>
<tr>
<td>CodeLlama-34b (Roziere et al., 2023)</td>
<td>0.17</td>
<td>16.21</td>
<td>23.04</td>
<td>25.94</td>
<td>28.16</td>
</tr>
<tr>
<td>Llama-2-70b-chat (Touvron et al., 2023)</td>
<td>4.27</td>
<td>14.33</td>
<td>15.70</td>
<td>16.55</td>
<td>17.92</td>
</tr>
</tbody>
</table>

Table 7: Testing results on MINT-Bench with different interaction limits from 1 to 5 step.