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., 2023a; 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

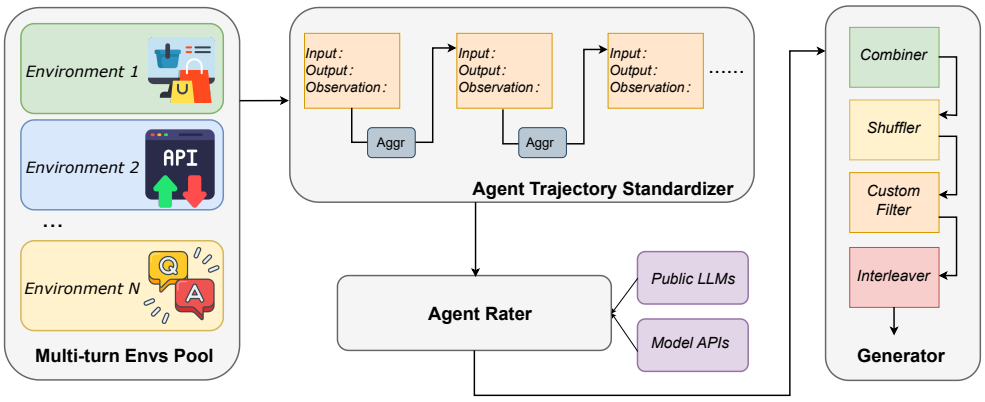

Figure 1: Workflow of AgentOhana. A homogeneous multi-turn data format is designed to consolidate heterogeneous trajectories from diverse data sources. *AgentRater* then assesses and filters agent trajectories. Finally, a streaming data loader enables integration of various datasets and feeds data into a distributed training process at random.

methods, a phenomenon observed universally across different environments. For instance, the HotpotQA 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 $i^{th} \in [1, N]$ 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

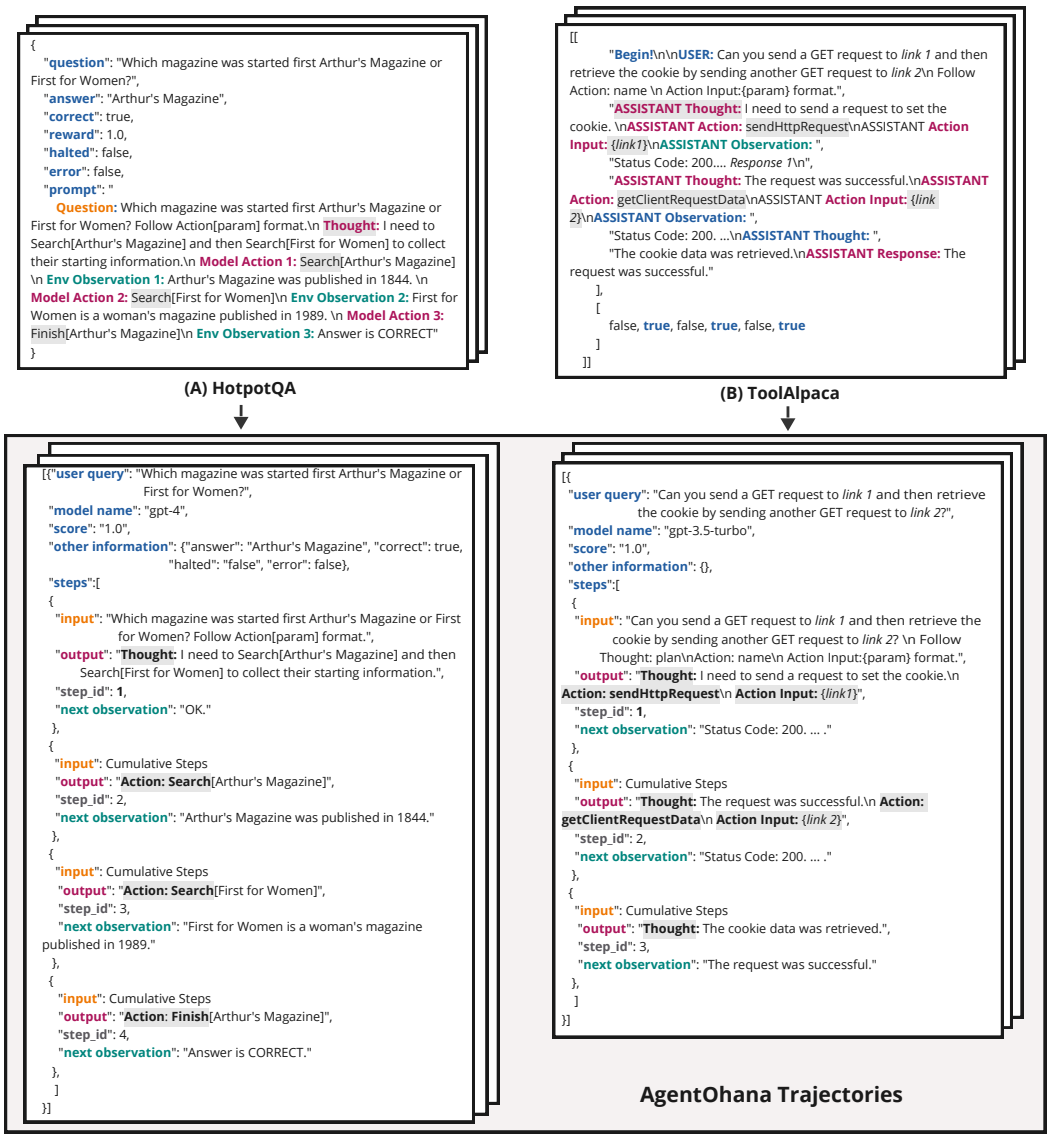

Figure 2: Example trajectories from (A) HotpotQA, (B) ToolAlpaca to AgentOhana.

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

[BEGIN OF JSON DICT FILE]
{**Given Agent Trajectory**}
[END OF JSON DICT FILE]

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 5: "click[rope sausage]", observation of step 5: "Invalid action!"; 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.

Environments & Data	#Sampled Trajs	#Filtered Trajs	#Average Turns
Webshop (Yao et al., 2022)	11200	2063	6.8
AlfWorld (Shridhar et al., 2021)	954	336	13.5
HotpotQA (Yang et al., 2018)	1740	402	4.8
ToolBench (Qin et al., 2023)	83771	30319	3.1
ToolAlpaca (Tang et al., 2023)	3936	3399	2.5
Operating System (Liu et al., 2023a)	647	195	3.9
APIbank (Li et al., 2023)	33415	4902	1.0
DataBase (Liu et al., 2023a)	6376	538	2.0
Mind2Web (Deng et al., 2023)	23378	122	1.0
Knowledge Graph (Liu et al., 2023a)	2501	324	6.0
AgentOhana	167918	42600	3.1

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 filter trajectories with the AgentRater score $>= 4.0$, *#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 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.

```
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_datasets(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.

```
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

LLM	LAA Architecture					
	ZS	ZST	ReAct	PlanAct	PlanReAct	BOLAA
Llama-2-70b-chat (Touvron et al., 2023)	0.0089	0.0102	0.4273	0.2809	0.3966	0.4986
Vicuna-33b (Zheng et al., 2023b)	0.1527	0.2122	0.1971	0.3766	0.4032	0.5618
Mixtral-8x7B-Instruct-v0.1 (Jiang et al., 2024)	0.4634	0.4592	0.5638	0.4738	0.3339	0.5342
GPT-3.5-Turbo	0.4851	0.5058	0.5047	0.4930	0.5436	0.6354
GPT-3.5-Turbo-Instruct	0.3785	0.4195	0.4377	0.3604	0.4851	0.5811
GPT-4-0613	0.5002	0.4783	0.4616	**0.7950**	0.4635	0.6129
xLAM-v0.1	**0.5201**	**0.5268**	**0.6486**	0.6573	**0.6611**	**0.6556**

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.

LLM	LAA Architecture				
	ZS	ZST	ReAct	PlanAct	PlanReAct
Mixtral-8x7B-Instruct-v0.1 (Jiang et al., 2024)	0.3912	0.3971	0.3714	0.3195	0.3039
GPT-3.5-Turbo	0.4196	0.3937	0.3868	0.4182	0.3960
GPT-4-0613	**0.5801**	**0.5709**	**0.6129**	**0.5778**	**0.5716**
xLAM-v0.1	0.5492	0.4776	0.5020	0.5583	0.5030

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.

limits from 1 to 5 steps 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.

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.

	Unseen Insts & Same Set	Unseen Tools & Seen Cat	Unseen Tools & Unseen Cat
TooLlama V2 (Qin et al., 2023)	0.4385	0.4300	0.4350
GPT-3.5-Turbo-0125	0.5000	0.5150	0.4900
GPT-4-0125-preview	**0.5462**	0.5450	0.5050
xLAM-v0.1	0.5077	**0.5650**	**0.5200**

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.

3.5 TOOLEVAL

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

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

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

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

Harrison Chase. Langchain. `https://github.com/hwchase17/langchain`, 2023.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. Alpagasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*, 2023.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6, 2023.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *arXiv preprint arXiv:2306.06070*, 2023.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacafarm: A simulation framework for methods that learn from human feedback, 2023.

Significant Gravitas. Autogpt. `https://github.com/Significant-Gravitas/Auto-GPT`, 2023.

Shangmin Guo, Biao Zhang, Tianlin Liu, Tianqi Liu, Misha Khalman, Felipe Llinares, Alexandre Rame, Thomas Mesnard, Yao Zhao, Bilal Piot, et al. Direct language model alignment from online ai feedback. *arXiv preprint arXiv:2402.04792*, 2024.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Minghao Li, Feifan Song, Bowen Yu, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*, 2023.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents, 2023a.

Zhiwei Liu, Weiran Yao, Jianguo Zhang, Le Xue, Shelby Heinecke, Rithesh Murthy, Yihao Feng, Zeyuan Chen, Juan Carlos Niebles, Devansh Arpit, et al. Bolaa: Benchmarking and orchestrating llm-augmented autonomous agents. *arXiv preprint arXiv:2308.05960*, 2023b.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. The flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*, 2023.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn llm agents. *arXiv preprint arXiv:2401.13178*, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023.

Erik Nijkamp, Hiroaki Hayashi, Caiming Xiong, Silvio Savarese, and Yingbo Zhou. Codegen2: Lessons for training llms on programming and natural languages. *ICLR*, 2023.

OpenAI. Gpt-4 technical report. *ArXiv*, 2023.

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.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. URL `https://arxiv.org/abs/2010.03768`.

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

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

XAgent Team. Xagent: An autonomous agent for complex task solving, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. Mint: Evaluating llms in multi-turn interaction with tools and language feedback. *arXiv preprint arXiv:2309.10691*, 2023a.

Yu Wang, Zhiwei Liu, Jianguo Zhang, Weiran Yao, Shelby Heinecke, and Philip S Yu. Drdt: Dynamic reflection with divergent thinking for llm-based sequential recommendation. *arXiv preprint arXiv:2312.11336*, 2023b.

Tianbao Xie, Fan Zhou, Zhoujun Cheng, Peng Shi, Luoxuan Weng, Yitao Liu, Toh Jing Hua, Junning Zhao, Qian Liu, Che Liu, et al. Openagents: An open platform for language agents in the wild. *arXiv preprint arXiv:2310.10634*, 2023.

Yiheng Xu, Hongjin Su, Chen Xing, Boyu Mi, Qian Liu, Weijia Shi, Binyuan Hui, Fan Zhou, Yitao Liu, Tianbao Xie, et al. Lemur: Harmonizing natural language and code for language agents. *arXiv preprint arXiv:2310.06830*, 2023.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Da Yin, Faeze Brahman, Abhilasha Ravichander, Khyathi Chandu, Kai-Wei Chang, Yejin Choi, and Bill Yuchen Lin. Lumos: Learning agents with unified data, modular design, and open-source llms. *arXiv preprint arXiv:2311.05657*, 2023.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms. *arXiv preprint arXiv:2310.12823*, 2023.

Jianguo Zhang, Kun Qian, Zhiwei Liu, Shelby Heinecke, Rui Meng, Ye Liu, Zhou Yu, Silvio Savarese, and Caiming Xiong. Dialogstudio: Towards richest and most diverse unified dataset collection for conversational ai. *arXiv preprint arXiv:2307.10172*, 2023.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric Xing, et al. Lmsys-chat-1m: A large-scale real-world llm conversation dataset. *arXiv preprint arXiv:2309.11998*, 2023a.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023b.

Appendix

A HETEROGENEITY OF VARIOUS DATASETS

Figure 4 shows original trajectories from four environments.

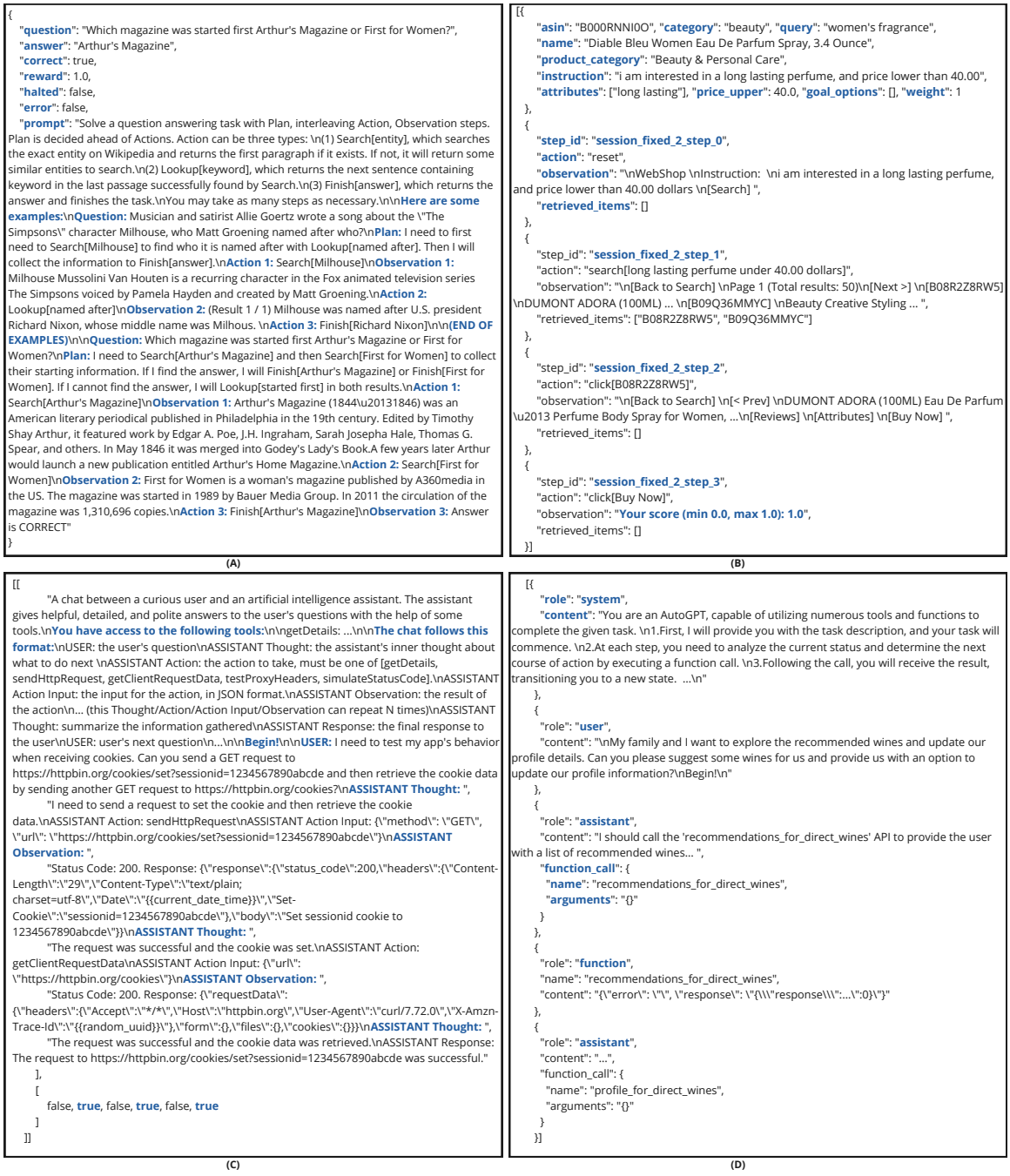

Figure 4: Original trajectories from (A) HotpotQA, (B) Webshop, (C) ToolAlpaca, (D) ToolBench.

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

LLM	LAA Architecture				
	ZS	ZST	ReAct	PlanAct	PlanReAct
xLAM-v0.1	0.5201	0.5268	0.6486	0.6573	0.6611
w/o AgentRater	0.4998	0.4992	0.6338	0.6283	0.6546
xLAM-v0.1	0.5492	0.4776	0.5020	0.5583	0.5030
w/o AgentRater	0.5138	0.4647	0.4917	0.5225	0.4904

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

D MINT-BENCH

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

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