Auto-SLURP: A Benchmark Dataset for Evaluating Multi-Agent Frameworks in Smart Personal Assistant

Anonymous ACL submission

Abstract

In recent years, multi-agent frameworks powered by large language models (LLMs) have advanced rapidly. Despite this progress, there is still a notable absence of benchmark datasets specifically tailored to evaluate their performance. To bridge this gap, we introduce **Auto-SLURP**, a benchmark dataset aimed at evaluating LLM-based multi-agent frameworks in the context of smart personal assistants. Auto-SLURP extends the original SLURP dataset—initially developed for natural language understanding tasks—by relabeling the data and integrating simulated servers and external services. This enhancement enables a comprehensive end-to-end evaluation pipeline, covering language understanding, task execution, and response generation. Our experiments demonstrate that Auto-SLURP presents a significant challenge for current state-of-the-art frameworks, highlighting that truly reliable and intelligent multi-agent personal assistants remain a work in progress.

1 Introduction

Multi-agent frameworks built on large language models (LLMs) have seen rapid development in recent years (Li et al., 2023; Su et al., 2024; Hong et al., 2024; Wu et al., 2023; Liu et al., 2024b). These frameworks provide general-purpose infrastructures that facilitate the construction of multi-agent systems through modular architectures, communication protocols, and coordination strategies. Despite the rapid progress, there remains a noticeable gap in standardized benchmarks tailored to evaluate the effectiveness of these frameworks.

While a number of benchmarks have been proposed to assess the tool-use capabilities of LLMs (Qin et al., 2023; Chen et al., 2023c; Zhu et al., 2023; Zhuang et al., 2024; Ye et al., 2024), they primarily focus on individual LLMs and address only a narrow slice of functionality. As a result, they do not adequately reflect the complexity, interactivity, and coordination challenges inherent in real-world multi-agent scenarios.

To capture broader dimensions of agent behavior, several social and interactive benchmarks have recently been proposed. For example, Cooperation (Abdelnabi et al., 2023), SOTOPIA (Zhou et al., 2024), AgentSense (Mou et al., 2024), and SocialBench (Chen et al., 2024) create social environments to evaluate agents' interpersonal and collaborative abilities. In parallel, AgentBench (Liu et al., 2023) targets reasoning and decision-making skills in domains such as coding, web navigation, and e-commerce. Other works, including MAgIC (Xu et al., 2023), CUISINEWORLD (Gong et al., 2024), BattleAgentBench (Wang et al., 2024), CivRealm (Qi et al., 2024), and LegalAgentBench (Li et al., 2024), introduce game-based or domain-specific settings to assess multi-agent interaction.

Meanwhile, benchmarks in embodied environments—such as AgentBoard (Ma et al., 2024), ALFWorld (Shridhar et al.), the ThreeDWorld Transport Challenge (Gan et al., 2021), and WAH (Puig et al., 2020)—focus on grounding agents in physical or simulated worlds.

However, these efforts are typically designed to evaluate individual LLMs' task execution and interaction capabilities in multi-agent systems, rather than to assess the performance or flexibility of open-source multi-agent frameworks. Moreover, the highly integrated nature of game-based and embodied environments often makes them difficult to adapt for evaluating general-purpose frameworks, limiting their reusability and extensibility (Xu et al., 2020; Zhang et al., 2021).

Taken together, although significant progress has been made in benchmarking agent capabilities, existing efforts do not sufficiently address the unique needs of evaluating multi-agent frameworks. This highlights a pressing need for a comprehensive and flexible benchmark that can rigorously and fairly

User	could you please email john saying i'm on leave	
	re-labeled	**original**
Intent	email_sendemail	email_sendemail
Slots	to_person: john, content: i'm on leave	person : john

Table 1: The example of the annotations in Auto-SLURP.

assess the effectiveness of LLM-based multi-agent infrastructures across a range of scenarios.

One particularly compelling application is the smart personal assistant—an AI system capable of understanding natural language and performing tasks on behalf of users. This vision has long captured the imagination of both researchers and the public (Edu et al., 2020; Hoy, 2018). Despite significant progress in AI and the emergence of powerful LLM-based multi-agent systems, this vision remains underexplored in the context of multi-agent evaluation. Personal assistants are expected to handle a wide range of tasks, such as checking the weather, sending emails, managing calendars, and controlling IoT devices. Achieving this level of functionality demands not only natural language understanding (NLU), but also sophisticated capabilities in decision-making, reasoning, tool use, coordination, and adaptability (Del Tredici et al., 2021; Shen et al., 2022).

To help fill this gap, we introduce **Auto-SLURP**, a benchmark designed to evaluate the effectiveness of LLM-based multi-agent frameworks in building smart personal assistants. Auto-SLURP is built upon the SLURP dataset (Bastianelli et al., 2020; Liu et al., 2021), originally created for natural language understanding in smart home scenarios. We extend SLURP's original intent-slot structure to support comprehensive end-to-end evaluation: from language understanding and intent interpretation, to task execution and response generation. To better reflect the complexity of real-world interactions, we relabel the slots and restructure the data to align with complete user-interaction pipelines.

Auto-SLURP simulates realistic assistant interactions by integrating external services and simulated servers, enabling thorough evaluation of a framework's ability to handle complex, multi-step operations. These operations include API access, state management across modules, and coordination between agents with specialized responsibilities. This setup allows us to assess not just whether multi-agent frameworks can interpret user commands, but also whether they can effectively orchestrate the backend processes needed to carry them out.

The dataset spans a wide range of task domains, such as calendar management, media playback, transportation scheduling, and information retrieval. This diversity ensures that Auto-SLURP serves as a robust and representative benchmark for evaluating both the flexibility and reliability of multi-agent frameworks in realistic scenarios. Our experimental results demonstrate that Auto-SLURP presents significant challenges even for state-of-the-art multi-agent frameworks. These findings underscore the complexity involved in achieving seamless, intelligent assistant behavior and reveal that we are still some distance away from building fully dependable AI-based personal assistants.

2 Dataset Construction

Creation of queries and annotations We make modification to the SLURP dataset, which is collected for the development of smart personal assistants. Personal assistant systems are inherently complex, as they must interpret and respond to a wide variety of user commands. SLURP was initially released for natural language understanding tasks (Weld et al., 2022; Yang et al., 2017; Shen et al., 2017; Su et al., 2018; Huang et al., 2021), with a focus on intention detection and slot filling. In traditional methods, intent detection is treated as a classification problem, while slot filling is handled as a sequence-to-sequence task. For example, given the user query "play kari jobe for me", the intent is "play_music", and the slot is "artist_name: kari jobe". In SLURP, the slots are limited to the entities explicitly mentioned in the utterance, omitting other crucial information required to successfully execute the command. This omission can lead to incomplete or failed task execution.

To adapt SLURP for our specific use case, we retain only the user queries and their corresponding intents from SLURP, while re-labeling the slots. Specifically, we enrich the slot information by adding new slots and refining existing ones to cap-

	CamelAI	LangGraph	AutoGen	AgentLite
GPT-4	0.21	0.32	0.44	0.46
DeepSeek-V3	0.39	0.32	0.36	0.47

Table 2: The results of the multi-agent frameworks.

ture all the information necessary for backend task execution. We also ensure that the slot structures are compatible with LLMs, which typically generate outputs rather than classify them. Table 1 illustrates an example of our modified samples, with our re-labeled version in the middle column, and the original SLURP sample in the right column.

The dataset encompasses a wide range of tasks, from straightforward actions like setting calendars or playing music, to more complex operations such as information retrieval or handling transportation-related commands. We randomly select 1,000 samples from the training set and 100 samples from the testing set. Based on our experimental results, this subset is considered sufficient for training and testing LLM-based multi-agent frameworks.

Collection of the end servers To evaluate end-to-end system performance, we simulate the execution servers that process and carry out user commands. This simulation enable us to verify whether the commands are correctly interpreted and executed, ensuring that the overall system functions as expected. In our training set, we identify 23 distinct domains. For each domain, we build a dedicated server to handle the relevant operations. Additionally, for certain domains which require external information, such as search, weather, and news, we integrate external services, i.e., third-party APIs. These API calls allow the system to fetch the required information, ensuring that user requests are handled efficiently and with up-to-date content.

3 Experiments

3.1 Setup

We compare several representative LLM-based multi-agent frameworks.

CamelAI (Li et al., 2023) introduces a cooperative framework that allows agents to autonomously collaborate through role-playing.

AutoGen (Wu et al., 2023) presents a customizable framework that can integrate LLMs, humans, and tools, enabling dynamic agent interactions.

LangGraph (2023) is built upon the foundation of LangChain (2022) and provides an easy way to create cyclical graphs during runtimes.

AgentLite (Liu et al., 2024b) is a lightweight, modular codebase that can easily experiment with new reasoning strategies.

For all multi-agent frameworks, we use GPT-4 (Achiam et al., 2023) and DeepSeek-V3 (Liu et al., 2024a) as the LLMs. We describe the details of the experiments in Appendix A.

3.2 Defined workflows

We use each multi-agent framework to build a workflow that simulates a smart personal assistant. In the workflow, a program manager agent serves as the orchestrator; it processes the user's input query and delegates subtasks to specialized agents. We introduce an intent agent to predict the intent and slots. Additionally, we add a time agent and a location agent to format the time and location parameters, if applicable. We adopt a URL agent to select the appropriate URL from a list of candidates, and a request agent to execute the tool function call for the request. The overall workflow is illustrated in Figure 1 in Appendix B. Although the orchestration methods, prompt policies, and reasoning approaches vary across frameworks, we ensure a fair and controlled comparison by maintaining consistency in the assigned roles, accessible tools, and prompts used to define agent functions during construction.

3.3 Evaluation

We use the successful execution rate as the evaluation metric, which measures the percentage of queries that are completed successfully from end to end. This metric assesses the reliability, efficiency, and ability of the framework to perform the intended actions without failure. Additionally, we provide an automated evaluation tool that measures performance across all frameworks consistently and efficiently.

4 Experiment Results

4.1 Results analysis

Table 2 presents the results of the multi-agent frameworks. Among them, AgentLite performs

	CamelAI		LangGraph		AutoGen		AgentLite	
	GPT-4	**DeepSeek**	**GPT-4**	**DeepSeek**	**GPT-4**	**DeepSeek**	**GPT-4**	**DeepSeek**
intent	54%	50.8%	34%	39.7%	68%	43.8%	69%	69.8%
time	18%	8.2%	12%	29.6%	9%	14.1%	19%	7.5%
location	-	-	-	1.5%	-	-	7%	-
URL	14%	4.9%	13%	7.4%	43%	14.1%	19%	45.3%
request	-	-	-	1.5%	-	1.6%	-	5.7%
manager	9%	49.2%	53%	36.8%	13%	46.9%	-	3.8%
function_call	18%	1.6%	-	-	-	-	-	

Table 3: Error analysis of the frameworks. Because one failure can be caused by multiple reasons, the percentages do not sum up to 100%. DeepSeek refers to DeepSeek-V3.

AutoGen	**original**	**finetuned**
acc	0.40	0.62

Table 4: The results for AutoGen with original and finetuned intent agents.

the best. The failure of CamelAI with GPT-4 can be attributed to its difficulty in selecting the right tool to execute, largely due to bugs in its interface with GPT-4. Additionally, DeepSeek-V3 can not run in CamelAI until we resolve certain response parsing issues. LangGraph underperforms mainly because it only combines the system prompt and all the agents' results into one list as input, without any adjustments. In contrast, AutoGen separates the prompts for the manager agent and the sub-task agents, enabling clearer task delegation and yielding better results. AgentLite further improves performance by adopting "think and react" methods in the process, which significantly enhances execution success. Example prompts for LangGraph and AutoGen are provided in Appendix C.

We also test other frameworks, such as AgentVerse (Chen et al., 2023b) and AutoAgents (Chen et al., 2023a). However, these frameworks either lack a generalized orchestration policy to support this scenario or do not provide sufficient information for effective implementation. This highlights the inherent complexity of designing robust multi-agent frameworks.

To gain deeper insight into failure points, we analyze the errors caused by individual agents and the function call part. As shown in Table 3, it is clear that the main source of failure stems from the intent agent. We show the failure attribution criteria in Appendix D.

4.2 Ablation

Our prior analysis shows that intent prediction is the leading cause of failures. To address this, we conduct an ablation study by further finetuning a model for the intent agent to assess its impact on overall framework performance. We choose the open-source Llama 3 model (AI@Meta, 2024) for finetuning. Specifically, we finetune the LLAMA-3 8B model on our training set and use the resulting model as the intent agent. All other agents in the system continue to use GPT-4 as their underlying LLM. We evaluate this setup in AutoGen framework, and the results are presented in Table 4. Compared to the framework that uses the original LLAMA-3 8B model, the finetuned version shows a performance improvement of 55%. This result demonstrates that improving individual components—especially the main failure source—can significantly enhance the overall performance of multi-agent frameworks. A more detailed breakdown of domain-specific accuracy for both versions is provided in Appendix E.

Based on the analysis above, it is clear that we are still a few steps away from achieving a fully reliable and smart personal assistant. Achieving this goal will require continued progress in several key areas of multi-agent framework design—namely, the development of generalized orchestration policies, effective prompting methods, robust reasoning approaches (such as think and react), and careful selection of LLMs suited to the task.

5 Conclusion

We present Auto-SLURP, a dataset designed to evaluate LLM-based multi-agent frameworks. We assess the end-to-end execution tasks, not just the nature language understanding tasks. By incorporating simulated servers and external services, we evaluate the capacity of the frameworks to complete the entire process. The dataset proves to be sufficiently challenging to test the state-of-the-art multi-agent frameworks.

Limitations

The dataset incorporates simulated servers and external services, which may not fully mimic the behavior of real-world systems. This could result in discrepancies between the performance of frameworks in the benchmark and their performance in live applications.

Additionally, the dataset's evaluation is heavily reliant on the performance of LLMs. Variations in the quality and capabilities of LLMs across different versions could influence the outcomes.

References

Sahar Abdelnabi, Amr Gomaa, Sarath Sivaprasad, Lea Schonherr, and Mario Fritz. 2023. Cooperation, competition, and maliciousness: Llm-stakeholders interactive negotiation.

OpenAI Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, and 1 others. 2023. Gpt-4 technical report.

AI@Meta. 2024. Llama 3 model card.

Emanuele Bastianelli, Andrea Vanzo, Pawel Swietojanski, and Verena Rieser. 2020. Slurp: A spoken language understanding resource package. *arXiv preprint arXiv:2011.13205*.

Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje F. Karlsson, Jie Fu, and Yemin Shi. 2023a. Autoagents: A framework for automatic agent generation. In *International Joint Conference on Artificial Intelligence*.

Hongzhan Chen, Hehong Chen, Ming Yan, Wenshen Xu, Xing Gao, Weizhou Shen, Xiaojun Quan, Chenliang Li, Ji Zhang, Fei Huang, and 1 others. 2024. Roleinteract: Evaluating the social interaction of role-playing agents. *arXiv preprint arXiv:2403.13679*.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Ya-Ting Lu, Yi-Hsin Hung, Cheng Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2023b. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*.

Zehui Chen, Weihua Du, Wenwei Zhang, Kuikun Liu, Jiangning Liu, Miao Zheng, Jingming Zhuo, Songyang Zhang, Dahua Lin, Kai Chen, and 1 others. 2023c. T-eval: Evaluating the tool utilization capability of large language models step by step. *arXiv preprint arXiv:2312.14033*.

Marco Del Tredici, Gianni Barlacchi, Xiaoyu Shen, Weiwei Cheng, and Adrià de Gispert. 2021. Question rewriting for open-domain conversational qa: Best practices and limitations. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 2974–2978.

Jide S Edu, Jose M Such, and Guillermo Suarez-Tangil. 2020. Smart home personal assistants: a security and privacy review. *ACM Computing Surveys (CSUR)*, 53(6):1–36.

Chuang Gan, Siyuan Zhou, Jeremy Schwartz, Seth Alter, Abhishek Bhandwaldar, Dan Gutfreund, Daniel LK Yamins, James J DiCarlo, Josh McDermott, Antonio Torralba, and 1 others. 2021. The threedworld transport challenge: A visually guided task-and-motion planning benchmark for physically realistic embodied ai. *arXiv preprint arXiv:2103.14025*.

Ran Gong, Qiuyuan Huang, Xiaojian Ma, Yusuke Noda, Zane Durante, Zilong Zheng, Demetri Terzopoulos, Li Fei-Fei, Jianfeng Gao, and Hoi Vo. 2024. MindAgent: Emergent gaming interaction. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3154–3183, Mexico City, Mexico. Association for Computational Linguistics.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Matthew B. Hoy. 2018. Alexa, siri, cortana, and more: An introduction to voice assistants. *Medical Reference Services Quarterly*, 37(1):81–88. PMID: 29327988.

Yunyun Huang, Xiaoyu Shen, Chuanyi Li, Jidong Ge, and Bin Luo. 2021. Dependency learning for legal judgment prediction with a unified text-to-text transformer. *arXiv preprint arXiv:2112.06370*.

LangChain. 2022. Langchain.

LangGraph. 2023. Langgraph.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for "mind" exploration of large language model society. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Haitao Li, Junjie Chen, Jingli Yang, Qingyao Ai, Wei Jia, Youfeng Liu, Kai Lin, Yueyue Wu, Guozhi Yuan, Yiran Hu, and 1 others. 2024. Legalagentbench: Evaluating llm agents in legal domain. *arXiv preprint arXiv:2412.17259*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

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, and 3 others. 2023. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv: 2308.03688*.

Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. 2021. Benchmarking natural language understanding services for building conversational agents. In *Increasing naturalness and flexibility in spoken dialogue interaction: 10th international workshop on spoken dialogue systems*, pages 165–183. Springer.

Zhiwei Liu, Weiran Yao, Jianguo Zhang, Liangwei Yang, Zuxin Liu, Juntao Tan, Prafulla K. Choubey, Tian Lan, Jason Wu, Huan Wang, Shelby Heinecke, Caiming Xiong, and Silvio Savarese. 2024b. Agentlite: A lightweight library for building and advancing task-oriented llm agent system. *Preprint*, arXiv:2402.15538.

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

Xinyi Mou, Jingcong Liang, Jiayu Lin, Xinnong Zhang, Xiawei Liu, Shiyue Yang, Rong Ye, Lei Chen, Haoyu Kuang, Xuanjing Huang, and 1 others. 2024. Agentsense: Benchmarking social intelligence of language agents through interactive scenarios. *arXiv preprint arXiv:2410.19346*.

Xavier Puig, Tianmin Shu, Shuang Li, Zilin Wang, Yuan-Hong Liao, Joshua B Tenenbaum, Sanja Fidler, and Antonio Torralba. 2020. Watch-and-help: A challenge for social perception and human-ai collaboration. *arXiv preprint arXiv:2010.09890*.

Siyuan Qi, Shuo Chen, Yexin Li, Xiangyu Kong, Junqi Wang, Bangcheng Yang, Pring Wong, Yifan Zhong, Xiaoyuan Zhang, Zhaowei Zhang, and 1 others. 2024. Civrealm: A learning and reasoning odyssey in civilization for decision-making agents. *arXiv preprint arXiv:2401.10568*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, and 1 others. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Xiaoyu Shen, Youssef Oualil, Clayton Greenberg, Mittul Singh, and Dietrich Klakow. 2017. Estimation of gap between current language models and human performance.

Xiaoyu Shen, Svitlana Vakulenko, Marco Del Tredici, Gianni Barlacchi, Bill Byrne, and Adrià de Gispert. 2022. Low-resource dense retrieval for open-domain question answering: A comprehensive survey. *arXiv preprint arXiv:2208.03197*.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations 2021*.

Hui Su, Xiaoyu Shen, Pengwei Hu, Wenjie Li, and Yun Chen. 2018. Dialogue generation with gan. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Hui Su, Zhi Tian, Xiaoyu Shen, and Xunliang Cai. 2024. Unraveling the mystery of scaling laws: Part i. *arXiv preprint arXiv:2403.06563*.

Wei Wang, Dan Zhang, Tao Feng, Boyan Wang, and Jie Tang. 2024. Battleagentbench: A benchmark for evaluating cooperation and competition capabilities of language models in multi-agent systems. *arXiv preprint arXiv:2408.15971*.

Henry Weld, Xiaoqi Huang, Siqu Long, Josiah Poon, and Soyeon Caren Han. 2022. A survey of joint intent detection and slot filling models in natural language understanding. *ACM Computing Surveys*, 55(8):1–38.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation.

Binxia Xu, Siyuan Qiu, Jie Zhang, Yafang Wang, Xiaoyu Shen, and Gerard De Melo. 2020. Data augmentation for multiclass utterance classification–a systematic study. In *Proceedings of the 28th international conference on computational linguistics*, pages 5494–5506.

Lin Xu, Zhiyuan Hu, Daquan Zhou, Hongyu Ren, Zhen Dong, Kurt Keutzer, See Kiong Ng, and Jiashi Feng. 2023. Magic: Benchmarking large language model powered multi-agent in cognition, adaptability, rationality and collaboration. *arXiv preprint arXiv: 2311.08562*.

Xuesong Yang, Yun-Nung Chen, Dilek Hakkani-Tür, Paul Crook, Xiujun Li, Jianfeng Gao, and Li Deng. 2017. End-to-end joint learning of natural language understanding and dialogue manager. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5690–5694. IEEE.

Junjie Ye, Guanyu Li, Songyang Gao, Caishuang Huang, Yilong Wu, Sixian Li, Xiaoran Fan, Shihan Dou, Qi Zhang, Tao Gui, and 1 others. 2024. Tooleyes: Fine-grained evaluation for tool learning capabilities of large language models in real-world scenarios. *arXiv preprint arXiv:2401.00741*.

Rongzhi Zhang, Yulong Gu, Xiaoyu Shen, and Hui Su. 2021. Knowledge-enhanced session-based recommendation with temporal transformer. *arXiv preprint arXiv:2112.08745*.

Xuhui Zhou, Hao Zhu, Leena Mathur, Ruohong Zhang, Haofei Yu, Zhengyang Qi, Louis-Philippe Morency, Yonatan Bisk, Daniel Fried, Graham Neubig, and Maarten Sap. 2024. SOTOPIA: Interactive evaluation for social intelligence in language agents. In *The Twelfth International Conference on Learning Representations*.

Dawei Zhu, Xiaoyu Shen, Marius Mosbach, Andreas Stephan, and Dietrich Klakow. 2023. Weaker than you think: A critical look at weakly supervised learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14229–14253.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2024. Toolqa: A dataset for llm question answering with external tools. *Advances in Neural Information Processing Systems*, 36.

A The Details of the Experiments

We use DeepSeek-V3 instead of DeepSeek-R1 because the reasoning process of DeepSeek-R1 introduces more noise in this scenario. The prompts for the agent roles are created and adjusted during the setup phrase. The temperature is set as 0 to ensure that the LLM's responses are deterministic and fixed.

B Overview of the Defined Multi-Agent Workflow

The overall workflow defined in experiment is illustrated in Figure 1.

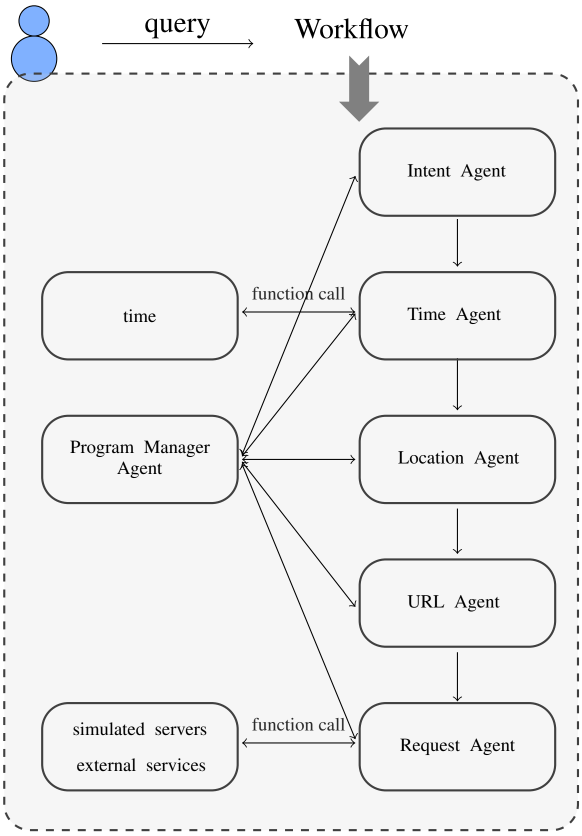

Figure 1: The workflow defined for the Auto-SLURP dataset.

C Prompt Examples and Cost Analysis

C.1 Prompt examples from LangGraph and AutoGen

Below is an example prompt from LangGraph, which includes the agents' names, the function description of the orchestration agent, the current subtask, and the responses from previous agents.

{'content': 'You are a supervisor tasked with managing a conversation between the following workers to finish the first user's cmd: ['intent', 'time', 'location', 'url', 'request', 'genresponse']. Given the following user request, respond with the worker to act next. you are controlling smart home system, you have intent, time, location, and url agent and request to complete the user's task. You should first use intent to complete the intent prediction. Then if the result has time or location params, please try to ask time or location to solve the time and location. At last you should choose the url using url agent, and then use request to send and receive request to the url such as weather server and then use genresponse to generate response, then finalize the task. Even if the request's response is need further information or is a question, do not further answer the question, just finish the task. The response need to be the worker to act next, for example: {"next": "FINISH"}. When finished, respond with FINISH. the data in json.', 'role': 'system'}, {'content': 'will i need sunscreen this afternoon', 'role': 'user'}, {'content': 'domain:weather, intent:weather_query, slots:time:this afternoon', 'name': 'intent', 'role': 'user'}

The following is an example prompt from AutoGen, which includes a description of the overall task, detailed function descriptions of all agents, responses from previous agents, and the current subtask. (Some content has been omitted for brevity.)

{'content': "You are in a role play game. The following roles are available: user_proxy: A computer terminal that performs no other action than running Python scripts (provided to it quoted in python code blocks), or sh shell scripts (provided to it quoted in sh code blocks). Product_manager: you are controlling smart home system, you have intent assistant, time_assistant, location_assistant, url_assistant and request_assistant to complete the user's task. You should first use intent to complete the intent prediction. Then if the result has time or location params, please try to ask time_assistant or location_assistant to solve the time and location. Then you choose the url using url_assistant. At last you should use request_assistant to send and receive request through functions from other servers such as weather server and response to user. You should generates reponse for the user, and tell manager to finalize the task. intent: Read the examples and results, and predict intent for the sentence. For 'set the alarm to two pm', first predict the domain, as domain:alarm, then the intent and slots, as the format: intent:alarm_set,time:two pm. the intents are calendar: calendar_set, calendar_remove, calendar_query ... Time_assistant: Read the time params, and convert to formated time. If has date, call the user_proxy_auto get_time function to get today's date, then calculate and format the date mentioned in the params. The time is 10:00. If has time, the time format should be 10:00. If no time specify, can return default time. If no date and time params, just skip. Location: Read the location params, and convert to formated location. The current location is new york. url_assistant: Read the params, and choose the url from the servers' url list: qa server is ... then all the url format should be ... Request: for url and query params, use the request functions you have been provided with. Read the following conversation. Then select the next role from ['user_proxy', 'Product_manager', 'intent', 'Time_assistant', 'Location', 'url_assistant', 'Request'] to play. Only return the role.", 'role': 'system'}, {'content': '{"query": "will i need sunscreen this afternoon"}', 'role': 'user', 'name': 'user_proxy'}, {'content': 'domain:weather,intent:weather_query,time:this afternoon', 'role': 'user', 'name': 'intent'}, {'content': "Read the above conversation. Then select the next role from ['user_proxy', 'Product_manager', 'intent', 'Time_assistant', 'Location', 'url_assistant', 'Request'] to play. Only return the role.", 'name': 'checking_agent', 'role': 'system'}

C.2 Cost analysis of multi-agent frameworks

Furthermore, we analyze the cost of GPT-4 for each framework, the results are shown in Table 5. The costs are at the same level for CamelAI, AutoGen, and AgentLite, but LangGraph has a significantly lower cost. We believe this is because LangGraph only uses the system prompt and all agents' results as input. Therefore, the cost of GPT-4 for each query, ranging from 0.5 to 0.8, is reasonable for an advanced multi-agent framework in this scenario.

D Failure Attribution Criteria in Evaluation

During evaluation, the workflow proceeds even if a failure occurs, and task completion is assessed only after the entire process is complete. To identify the source of failure, we trace the error back to the responsible agent based on the following criteria:

- Intent Agent: If the intent agent makes an incorrect prediction that ultimately leads to a workflow failure, the error is attributed to the intent agent.
- Time Agent: If the time agent provides an incorrect time or content that negatively affects the final outcome, the error is assigned to the time agent.
- Location Agent: If the location agent supplies an incorrect location resulting in an incorrect outcome, the error is attributed to the location agent.
- URL Agent: If the URL agent selects the wrong URL or incorrect parameters, the error is considered to originate from the URL agent. Additionally, if the URL agent receives an incorrect intent but is capable of correcting it and fails to do so, the error is also attributed to the URL agent.
- Request Agent: If the request agent successfully retrieves the correct data from the servers but generates an incorrect response, the error is classified as a request agent error.
- Manager Agent: If the manager agent incorrectly selects the next agent in the workflow, causing a failure, the error is attributed to the manager agent.
- Function Call: If the system executes an incorrect function call that results in a failure, the error is categorized as a function call failure.

USD/query	**CamelAI**	**LangGraph**	**AutoGen**	**AgentLite**
cost	0.52	0.14	0.80	0.55

Table 5: The costs of the frameworks.

Domain	Original	Finetuned
Audiobook	0.0%	66.7%
Calendar	11.8%	76.5%
Currency	0.0%	66.7%
Datetime	14.3%	71.4%
Email	0.0%	71.4%
IoT	33.3%	75.0%
Lists	40.0%	100.0%
Music	0.0%	70.0%
News	0.0%	100.0%
Podcasts	0.0%	50.0%
QA	0.0%	80.0%
Radio	0.0%	66.7%
Recommendation	0.0%	60.0%
Transport	33.3%	66.7%
Weather	0.0%	100.0%

Table 6: Accuracy across each domain for both the original and finetuned models.

Domain	Original	Finetuned
Audiobook	0.0%	66.7%
Calendar	29.4%	82.4%
Currency	0.0%	66.7%
Datetime	42.9%	85.7%
Email	57.1%	71.4%
IoT	58.3%	75.0%
Lists	40.0%	100.0%
Music	10.0%	70.0%
News	33.3%	100.0%
Podcasts	0.0%	50.0%
QA	0.0%	80.0%
Radio	0.0%	66.7%
Recommendation	20.0%	60.0%
Transport	66.7%	66.7%
Weather	14.3%	100.0%

Table 7: Accuracy across each domain—excluding slot name errors—for both the original and finetuned models.

E Evaluation of Intent Prediction Accuracy Across Domains

In the ablation study, we analyze the intent prediction accuracy across different domains, excluding those with fewer than three samples. The results are reported in Table 6, showing the accuracy for each domain using both the original and the finetuned models.

To further investigate the performance gap between intent prediction accuracy and overall workflow accuracy, we take a closer look at the outputs of the intent agent. We observe that some errors from the original model stem from formatting issues—such as incorrect slot names or returning plain-text descriptions instead of structured outputs. Notably, these issues can often be mitigated by downstream agents in the workflow, such as the URL agent, which may still successfully process the intent. Therefore, for reference, we additionally report a relaxed intent accuracy in Table 7, where slot name errors are ignored.

As shown in the two tables, finetuning improves accuracy across all domains. However, the Podcasts domain remains particularly challenging for the intent agent, with a final accuracy of only 50.0%.