ModelScope-Agent: Building Your Customizable Agent System with Open-source Large Language Models

Chenliang Li, Hehong Chen, Ming Yan[*], Weizhou Shen, Haiyang Xu, Zhikai Wu
Zhicheng Zhang, Wenmeng Zhou, Yingda Chen, Chen Cheng, Hongzhu Shi
Ji Zhang, Fei Huang, Jingren Zhou
DAMO Academy, Alibaba Group, China
{lcl193798, hehong.chh, ym119608, shenweizhou.swz, shuofeng.xhy, wuzhikai.wzk, zhangzhicheng.zzc, wenmeng.zwm, yingda.chen, chengchen.cc, hongzhu.shz, zj122146, f.huang, jingren.zhou}@alibaba-inc.com

Abstract

Large language models (LLMs) have recently demonstrated remarkable capabilities to comprehend human intentions, engage in reasoning, and design planning-like behavior. To further unleash the power of LLMs to accomplish complex tasks, there is a growing trend to build agent frameworks that equips LLMs, such as ChatGPT, with tool-use abilities to connect with massive external APIs.

In this work, we introduce ModelScope-Agent, a general and customizable agent framework for real-world applications, based on open-source LLMs as controllers. It provides a user-friendly system library, with a customizable engine design to support model training on multiple open-source LLMs, while also enabling seamless integration with both model APIs and common APIs in a unified way. To equip the LLMs with tool-use abilities, a comprehensive framework has been proposed spanning tool-use data collection, tool retrieval, tool registration, memory control, customized model training, and evaluation for practical real-world applications. Finally, we showcase ModelScopeGPT[1], a real-world intelligent assistant of ModelScope Community based on the ModelScope-Agent framework, which is able to connect open-source LLMs with more than 1000 public AI models and localized community knowledge in ModelScope. The ModelScope-Agent online demo[2], library[3] are now publicly available.

1 Introduction

Large language models (OpenAI, 2022, 2023; Touvron et al., 2023; Chowdhery et al., 2022) have gradually become common AI assistants that demonstrate great potential in comprehending human intentions, performing complex reasoning tasks, and enabling content creation. Despite the rapid advancements of open-source LLMs, e.g., LLaMA (Touvron et al., 2023) and ChatGLM (THUDM, 2023), they still remain limited in performing complex tasks, such as following user instructions to use external tools and capture up-to-date information.

To further unleash the power of LLMs for real-world practical applications, a rising trend of current research (Schick et al., 2023; Shen et al., 2023; Yang et al., 2023; Qin et al., 2023; Patil et al., 2023) begins to enable LLMs with tool-use abilities towards building an AI Agent. These include HuggingGPT (Shen et al., 2023), Visual-ChatGPT (Wu et al., 2023) and Gorilla (Patil et al., 2023) for connecting with HuggingFace models, ToolAlpaca (Tang et al., 2023) and ToolLLaMA (Qin et al., 2023) for using massive common APIs such as weather forecast and search engine. These methods either directly rely on closed-source counterparts like ChatGPT or focus on certain types of API tools. Recently, there have also been public releases of AI agents, such as Auto-GPT[4], LangChain[5] and Transformers Agent (Huggingface, 2023), which enable LLMs, such as ChatGPT or GPT-4, to use tools and solve complex AI tasks. However, these agents are mainly built with closed-source LLMs and how to build a customizable agent system with open-source LLMs remains largely unexplored.

In this work, we present ModelScope-Agent, a general and customizable agent system for real-world applications, based on open-source LLMs as controllers. ModelScope[6] is a public ML community, that seeks to bring together the most advanced machine learning models from the AI community, and streamlines the process of leveraging

[*]Corresponding author: <ym119608@alibaba-inc.com>
[1]https://modelscope.cn/studios/damo/ModelscopeGPT
[2]https://modelscope.cn/studios/lcl193798/Modelscope-Agent
[3]https://github.com/modelscope/modelscope-agent
[4]https://github.com/Significant-Gravitas/Auto-GPT
[5]https://github.com/langchain-ai/langchain
[6]https://modelscope.cn/models

AI models in real-world applications. ModelScope-Agent provides a flexible and user-friendly system library, with a customizable engine design to support model training on multiple open-source LLMs, while also enabling seamless integration with both model APIs and common APIs in a unified way. It features an LLM-centric system design, which includes open-source LLMs as core controller, and further interact with a tool-use module and a memory module to accomplish complex tasks. At the core of ModelScope-Agent , the library supports flexible selection and training on various open-source LLMs, such as LLaMA (Touvron et al., 2023), ChatGLM (THUDM, 2023), ChatPLUG (Tian et al., 2023) and other customized LLMs in ModelScope. For tool use, ModelScope-Agent provides a default tool library, which supports diverse AI model APIs across NLP, CV, Audio and Multi-model fields, as well as massive common APIs such as search engine. It also supports registering new self-defined API plugins and automatic API retrieval from the large tool library. It is easy for users to customize their most appropriate LLMs, local API tools and functions to develop real-world applications. Moreover, a memory module is also introduced to better store and manage the system message, user history, in-context examples, tool message and localized knowledge.

To enable the open-source LLMs to better control the whole agent system, we further propose a comprehensive framework of tool-use data collection, customized model training, evaluation and deployment. Notably, we release a comprehensive tool-enhanced dataset MSAgent-Bench, which consists of 598k dialogues with various API categories, multi-turn API calls, API-Oriented QA, and API-Agnostic instructions in both English and Chinese. A simple training strategy of Weighted LM, that enhances the training of generation of API name and parameters, is used to better ensure the correctness of API calls. Besides, an evaluation framework is also supported in our library to examine the tool-use abilities of the trained models in different aspects. Furthermore, we applied ModelScope-Agent in a real-world application of ModelScope Community namely ModelScopeGPT, which is able to connect open-source LLMs with more than 1000 public AI models and access localized community knowledge in ModelScope for community QA.

To summarize, ModelScope-Agent is a general and customizable agent system designed for developers to harness the power of open-source LLMs. The library targets the following goals:

- Agent based on Open-Source LLMs: the controller of ModelScope-Agent can be flexibly selected from open-source LLMs that are optimized through our agent training framework.
- Support and Customization of Diverse Tools: Dozens of diverse model APIs and common APIs are given by default. The library supports registering new self-defined APIs and automatic API retrieval from the toolset.
- Customizable of Applications: ModelScope-Agent can be flexibly applied in various industry applications. The agent and training framework are documented describing its usage, construction and optimization.

ModelScope-Agent is in continual development by the engineers at ModelScope and is released under an Apache 2.0 license. Full documentation is available through the project website.

2 The ModelScope Agent

ModelScope-Agent is designed to facilitate developers in building customizable agent systems based on open-source LLMs. The overall system architecture is shown in Figure 1. It includes open-source LLMs as controller, a tool-use module and a memory module to interact with. Given human instruction, the Agent, which adopts the selected LLM as the controller, will automatically plan tasks, selectively use tools, leverage knowledge in memory, and finally provide helpful responses to users.

2.1 LLMs as Brain

LLMs serve as the brain of the agent, responsible for planning and decomposing user requests, selectively calling tools, performing retrieval, and integrating all the information from previous steps to generate the final response. In order to make it easier for users to customize the agent with their own LLMs, we have added support for various open-source LLMs by default, such as LLaMA, ChatGLM and ChatPLUG, which have been optimized through our tool learning pipeline. The details of the training strategy and tool-use datasets can be referred to in Section 3. ModelScope-Agent has integrated the LLM inference pipeline of the ModelScope community, and replacing LLMs

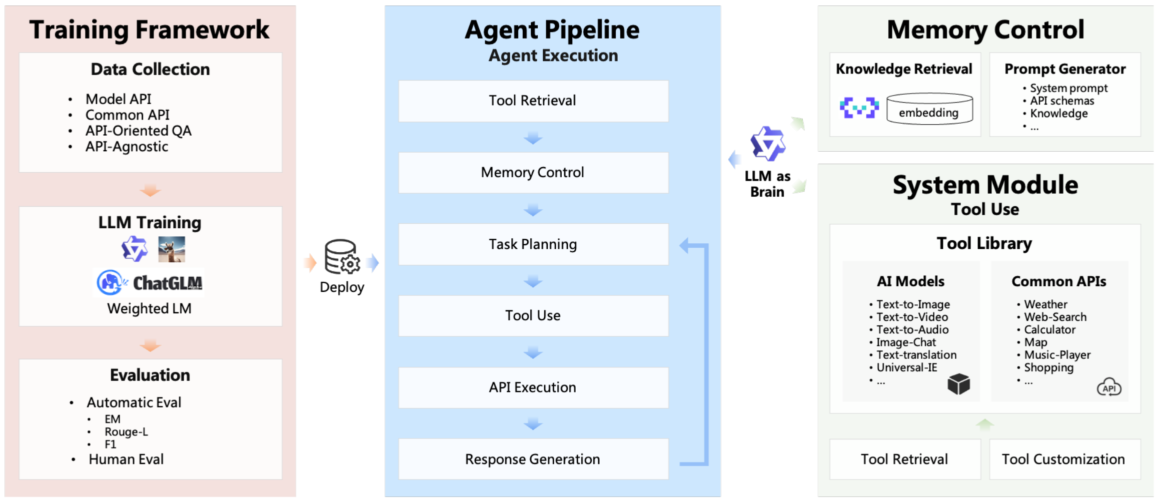


Figure 1: The overall system architecture of ModelScope-Agent.

can be done by simply setting the *model_name* and *model_config*. In *model_config*, the model_id, model_revision, and model parameter settings such as max sequence length, should be configured.

```
# LLM config "cfg_file"
from modelscope.utils.config import Config
model_cfg = Config.from_file(cfg_file)
llm = LocalLLM(model_name, model_cfg)
```

Furthermore, the ModelScope-Agent also provides a standard way to integrate new LLM. Users can add their own LLMs, by integrating the LLM pipeline into ModelScope. After that, the agent can select the new LLMs for training and inference.

2.2 Tool Use

Tool Library The tool library is used to configure and manage various collections of APIs used in the agent. ModelScope-Agent can support a wide range of both common APIs such as search APIs, and AI model APIs across NLP, CV, Audio and Multi-modal models in ModelScope and HuggingFace. Each tool API consists of the API name, description, parameters and request functions. Users can easily choose and configure proper APIs in the library to build their own agents. The default APIs supported in the library can be referred to in Appendix A.1.

```
# tool default config file "default_file"
tool_cfg = Config.from_file(default_file)
```

Register and Customize New Tool The agent allows users to register and customize new tools, while also supporting quick integration of newly registered tools into the agent, enabling LLMs to selectively use the additional self-defined tools for specific applications. This can be simply done by inheriting from a base class, namely *Tool*, and defining a new *CustomTool* with the API-related schema of API name, description, parameters, and request functions. More details about *CustomTool* can be referred to in Appendix A.2.

```
from modelscope_agent.tools import Tool
class CustomTool(Tool):
     # logic added here
     # refer example in Appendix A.2
tool_list = {'customo-tool': CustomTool()}
```

Tool Retrieval and Execution Due to the large amount of tool APIs in the tool library, a tool retrieval module is further introduced to recommend appropriate APIs for each instruction prompt. Specifically, we use the dense vector retrieval method based on the unified multilingual text-embedding API [7]. We vectorize both the text descriptions of the APIs and the instruction prompt using the text-embedding API. The top-3 most relevant APIs with the highest vector product scores are selected for tool use. As a result, the schema information of the retrieved APIs will be concatenated with other system prompts in the subsequent memory module and sent to LLMs as input. With the concatenated instruction prompt, the LLMs will plan and generate the API request, which will be executed by the agent. The agent will then return the results to the LLMs for continuous generation.

2.3 Memory Control

The memory module is used to retrieve and assemble a series of contextual information as input to the LLMs. It consists of a knowledge retrieval submodule and a prompt generator submodule, which are

[7]https://help.aliyun.com/zh/dashscope/getting-started-1

responsible for external knowledge retrieval and instruction prompt generation, respectively.

Knowledge Retrieval It enables the agent to get access to up-to-date and localized information related with query prompt, thereby augmenting LLMs with dynamic and domain-specific knowledge. We follow the same dense vector retrieval method as the previous tool retrieval module and support large-scale knowledge retrieval from localized document corpus. Similarly, it allows users to customize by changing to other open-source retrieval frameworks.

Prompt Generator The prompt generator is used to assemble all available contextual information such as system prompt, API schema, retrieved knowledge, conversation history, and few-shot examples. According to the type of user query and the maximum length of the LLM, the users can selectively choose proper contextual information and assemble the required input to the LLM. In our agent, the prompt generator needs to be defined before the agent is constructed.

2.4 Agent Pipeline

In summary, we build the agent by combining all the modules: LLM controller, tool-use module, and memory module. With *agent.run*, the agent can efficiently execute and complete the instruction in a one-step generation. First, the agent retrieves query-related tools through the tool retrieval and combines the retrieved API schema with other contextual prompts in the memory module, to construct a new instruction prompt. Then, the agent sends this new prompt to the LLM, which plans whether and which API to call and generate an API request. Next, the agent will execute the selected API with the extracted API parameters and return the API results to the LLMs, which will continue to plan whether to call other APIs. If another API call is needed, the process is repeated, otherwise, the LLMs generate the final response and the agent returns the final result to the user.

```
agent = AgentExecutor(llm, tool_cfg,
         additional_tool_list=tool_list)
agent.run("Draw a logo image of agent")
```

3 Training

3.1 Dataset

To facilitate building an agent with the ability to use tools while upholding an optimal level of user engagement, we release a comprehensive tool dataset, MSAgent-Bench, utilizing ChatGPT synthetic data and the existing instruction-following datasets. Our released dataset encompasses 598k dialogues. Table 1 outlines the key differences between the released dataset and other publicly available tool learning datasets, while the data distribution of our dataset is illustrated in Figure 2. As demonstrated in the Table and Figure, we have made certain efforts to construct a comprehensive dataset that enables the effective training of an agent:

Multilingual: We collect instances in both Chinese and English, ensuring that the trained agent is capable of functioning in both languages.

Various API Categories: Our dataset supports Common APIs that have been registered by users or applied through online API platforms, as well as model APIs that can call neural models.

Multi Turn Dialog: In real-life scenarios, agents may need to request more specific clarification from users to complete a task or receive additional instructions after completing a previous task. Our dataset accounts for these scenarios and supports multi-turn user-agent interactions when using tools.

API-Oriented QA: An effective agent should possess knowledge of APIs. Our dataset incorporates API document QA tasks and task planning tasks which requires agents to offer appropriate suggestions to users on how to use various APIs to solve complex tasks.

API-Agnostic Instructions: To enhance the agent's ability to follow common instructions and increase user engagement, we have incorporated both Chinese and English API-agnostic instructions within our dataset. These instructions place greater emphasis on the agent's inherent capabilities rather than reliance on API invocation.

The data was collected by prompting ChatGPT (`gpt-3.5-turbo`) to generate instructions, API requests, and answers based on the API calling results, more details can be accessed in Appendix D.

3.2 Model Training

We use the MSAgent-Bench to fine-tune multiple open-source LLMs, including LLaMA (Touvron et al., 2023), Qwen (QwenLM, 2023), ChatPLUG (Tian et al., 2023) etc. We train all the open-source LLMs in a multi-round conversation mode and concatenate all the prompts and answers. Compared to common instruction tuning data, the tool learning samples focus more heavily on the

Dataset	Language	Instance Type	# Instances	API type	Avg. Turn	Avg. Step
API-Bank (Li et al., 2023)	English	Tool Use	264	Common API	3.27	1.92
ToolAlpaca (Tang et al., 2023)	English	Tool Use	3.9 K	Common API	1	1.66
Gorilla (Patil et al., 2023)	English	Tool Use	16.4 k	Model API	1	1
GPT4Tools (Yang et al., 2023)	English	Tool Use	71.4 K	Model API	1	1
ToolBench (Qin et al., 2023)	English	Tool Use	26.9 K	Common API	1	4.1
MSAgent-Bench (ours)	English + Chinese	Tool Use + Common Chat	598 K	Common API + Model API	1.52	1.31

Table 1: The statistics of MSAgent-Bench and other existing tool learning datasets.

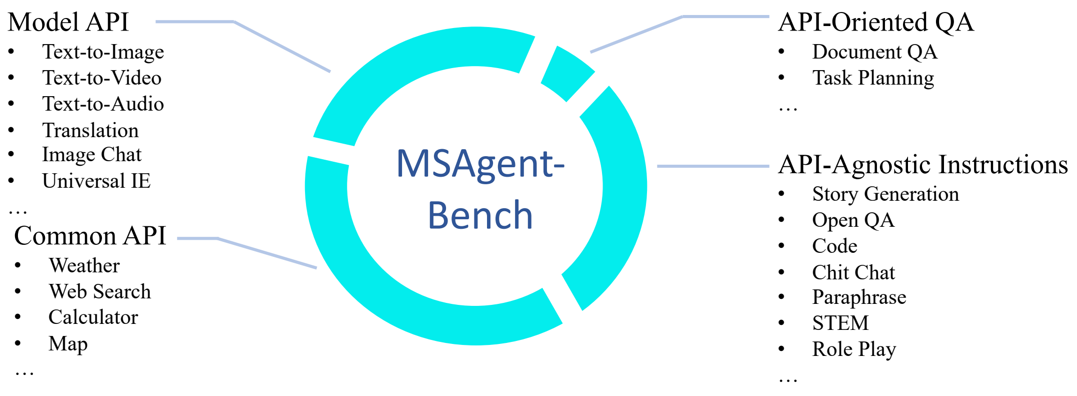


Figure 2: The instance types and distribution of our collected MSAgent-Bench.

accuracy of tool selection and API parameter prediction. Therefore, we propose a simple training strategy, Weighted LM, which enhances the training of generation of API names and parameters, while zero-out the loss of tokens from the user prompt and the tool execution. More details can be referred to in Appendix B.3.

```
kwargs = dict(model=model, ...)
trainer: EpochBasedTrainer = build_trainer
    (name=args.trainer, default_args=kwargs)
trainer.train()
```

4 Evaluation

Our evaluation system, MSAgent-Eval, comprises two modules: an automatic evaluation framework that comprehensively evaluates the API usability of the agents and a human evaluation framework implemented by an agent arena that reflects the preferences of human users.

4.1 Automatic Evaluation Framework

In automatic evaluation, we mainly focus on evaluating the agent's ability to generate accurate API requests and the proper answers according to the API calling results. Specifically, we use the action exactly match score (Action EM) which measures whether the agent uses the correct API as the reference gold API, and the ROUGE-L score which measures the similarity between the generated response and the gold answer. Additionally, we introduce a novel metric called Argument F1 for fully evaluating the quality of API requests. To compute Argument F1, we categorize the arguments in the agent's API request into two cases, namely Half match (HM) and Full match (FM), representing the correct argument but with the wrong value and the correct argument with the correct value, respectively. Suppose the gold argument number in the API is $|A|$, and the number of arguments in the agent API request is $|A^*|$, we compute the new Recall and Precision as follows:

$$R = (0.5 \times \#\,\text{HM} + \#\,\text{FM})/|A| \tag{1}$$
$$P = (0.5 \times \#\,\text{HM} + \#\,\text{FM})/|A^*| \tag{2}$$

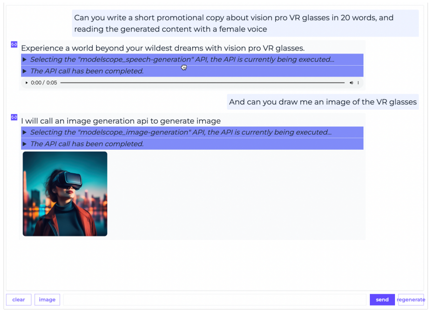
and the final argument F1 is computed as:

$$F1 = 2(R * P)/(R + P). \tag{3}$$

A sample code for the automated evaluation of agents is provided below:

```
from tool_agent_finetune import evaluation
EM, F1, ROUGE = evaluation(refs, preds)
```

Expert annotators were engaged to annotate the evaluation instances, with the task of providing diverse instructions, manually documenting correct API calling requests, and writing appropriate responses. The statistics of our currently assembled test data is in Appendix B.1, and the automatic evaluation scores of our trained agents can be found in Appendix B.2. We also guarantee the


Can you write a short promotional copy about vision pro VR glasses in 20 words, and reading the generated content with a female voice

Experience a world beyond your wildest dreams with vision pro VR glasses.

▶ *Selecting the "modelscope_speech-generation" API, the API is currently being executed...*

▶ *The API call has been completed.*

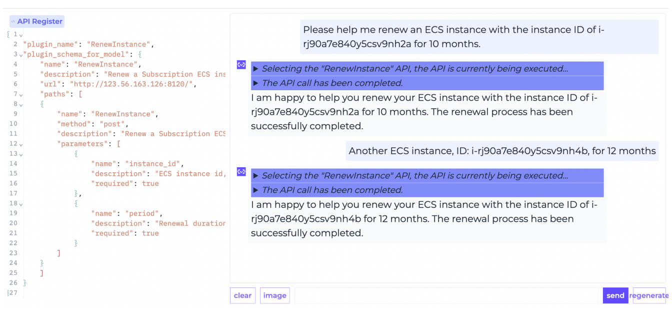
And can you draw me an image of the VR glasses

I will call an image generation api to generate image

▶ *Selecting the "modelscope_image-generation" API, the API is currently being executed...*

▶ *The API call has been completed.*

(a) ModelScope Intelligent Assistant



```
{
"plugin_name": "RenewInstance",
"plugin_schema_for_model": {
    "name": "RenewInstance",
    "description": "Renew a Subscription ECS ins
    "url": "http://123.56.163.126:8120/",
    "paths": [
    {
        "name": "RenewInstance",
        "method": "post",
        "description": "Renew a Subscription ECS
        "parameters": [
            {
                "name": "instance_id",
                "description": "ECS instance id,
                "required": true
            },
            {
                "name": "period",
                "description": "Renewal duratior
                "required": true
            }
        ]
    }
    ]
}
}
```

Please help me renew an ECS instance with the instance ID of i-rj90a7e840y5csv9nh2a for 10 months.

▶ *Selecting the "RenewInstance" API, the API is currently being executed...*

▶ *The API call has been completed.*

I am happy to help you renew your ECS instance with the instance ID of i-rj90a7e840y5csv9nh2a for 10 months. The renewal process has been successfully completed.

Another ECS instance, ID: i-rj90a7e840y5csv9nh4b, for 12 months

▶ *Selecting the "RenewInstance" API, the API is currently being executed...*

▶ *The API call has been completed.*

I am happy to help you renew your ECS instance with the instance ID of i-rj90a7e840y5csv9nh4b for 12 months. The renewal process has been successfully completed.

(b) Register and Use New Tools on Alibaba Cloud

Figure 3: Demo cases of ModelScopeGPT based on ModelScope-Agent .

users to upload their own annotated test examples to accurately evaluate the performance of agents in customized scenarios.

4.2 Human Evaluation with Agent Arena

Inspired by the Arena for ChatBots (Zheng et al., 2023), we have built an accessible Agent Arena [8] that allows users to furnish instructions to two anonymous agents, based on the provided APIs. Subsequently, users have the opportunity to vote on which Agent performs better in tackling the instruction with the given APIs. In accordance with the framework presented by Zheng et al. (2023), we adopt a system of ELO ratings and leaderboard maintenance for the participating Agents.

5 Usage Example of ModelScopeGPT

In this section, we showcase a successful application of ModelScope Community, ModelScopeGPT[9], based on our ModelScope-Agent.

ModelScope Intelligent Assistant Based on ModelScope-Agent , we have developed an intelligent assistant for the ModelScope Community, namely ModelScopeGPT. It uses LLMs as a controller to connect dozens of domain-specific AI models in the ModelScope open-source community, covering NLP, CV, Audio, and Multi-Modal fields. To make the pipeline more practical, we have included API retrieval and knowledge retrieval tools to automatically select proper APIs and get access to the local ModelScope knowledge. As shown in Figure 3a, ModelScopeGPT can support API calls in multi-turn conversations and generate correct API call parameters using information from previous conversations. More cases can refer to Appendix C. As a result, ModelScopeGPT has achieved a total request number of over 170k from 40k user visits within one month after its release.

Register and Use New Tools Another key feature of an agent is its generalization capability to unseen APIs. This allows users to quickly register their own APIs and customize their specific applications. Therefore, we test the generalization ability of ModelScopeGPT by applying it to an Alibaba Cloud application scenario. As shown in Figure 3b, we first found an API for renewing an ECS instance on Alibaba Cloud. Then, we registered the API schema defined in the tool library to the agent. Finally, we entered the prompt "*Please help me renew an ECS...*" in the demo. The agent generated a request through planning, selected the appropriate API, called the API to renew the instance successfully, and provided a reply to inform the user that the renewal was completed. This test demonstrates that the open-source LLM optimized based on the released API dataset has a strong generalization ability towards unseen APIs.

6 Conclusion

ModelScope-Agent aims to facilitate building AI Agent applications and research based on open-source LLMs by providing a general and customizable agent framework covering flexible system design, data collection, model training, evaluation and usage examples in real-world applications. It provides an open-source, community-driven library for AI Agent learning and best practices for building an agent system with open-source LLMs. We hope ModelScope-Agent can help pave the way towards a new era of AI Agent.

[8]https://modelscope.cn/studios/LLMZOO/Chinese-Arena/summary

[9]https://modelscope.cn/studios/damo/ModelScopeGPT/summary

Ethics Statement

Intended Use. ModelScope-Agent is designed to facilitate building AI Agent applications and research based on open-source LLMs, by providing a general and customizable agent system.

Potential Misuse. Although we have only trained with the tool-use datasets and gone through certain data filtering rules, it is still possible that the customized model may generate some biased, fake, and unsafe information. Our agent framework also provides users with the freedom to select proper LLMs and upload their own clean data for training. It is also important to design specific methods to improve the safety of the agent framework in the future.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Merouane Debbah, Etienne Goffinet, Daniel Heslow, Julien Launay, Quentin Malartic, et al. 2023. Falcon-40b: an open large language model with state-of-the-art performance.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *CoRR*, abs/2204.02311.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and brian ichter. 2023. Inner monologue: Embodied reasoning through planning with language models. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1769–1782. PMLR.

Huggingface. 2023. Transformers agent. Website. `https://huggingface.co/docs/transformers/transformers_agents`.

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

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, et al. 2022. Crosslingual generalization through multitask finetuning. *arXiv preprint arXiv:2211.01786*.

OpenAI. 2022. Chatgpt: Optimizing language models for dialogue.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bowen Li, Ziwei Tang, Jing Yi, Yuzhang Zhu, Zhenning Dai, Lan Yan, Xin Cong, Yaxi Lu, Weilin Zhao, Yuxiang Huang, Junxi Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Zhiyuan Liu, and Maosong Sun. 2023. Tool learning with foundation models. *arXiv preprint arXiv:2304.08354*.

QwenLM. 2023. Qwen-7b.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *arXiv preprint arXiv:2303.17580*.

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

THUDM. 2023. Chatglm. `https://github.com/THUDM/ChatGLM-6B`.

Junfeng Tian, Hehong Chen, Guohai Xu, Ming Yan, Xing Gao, Jianhai Zhang, Chenliang Li, Jiayi Liu, Wenshen Xu, Haiyang Xu, Qi Qian, Wei Wang, Qinghao Ye, Jiejing Zhang, Ji Zhang, Fei Huang, and Jingren Zhou. 2023. Chatplug: Open-domain generative dialogue system with internet-augmented instruction tuning for digital human. *arXiv preprint arXiv:2304.07849*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*.

Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. 2023. Gpt4tools: Teaching large language model to use tools via self-instruction. *arXiv preprint arXiv:2305.18752*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.

A Library

A.1 Tool List

API Name (language)	Description	Type
Text-to-Image(en)	Converts text to an image.	Model API
Text-to-Image(zh)	Converts text to an image.	Model API
Text-to-Video(en)	Converts text to a video.	Model API
Text-to-Audio(en)	Converts text to audio.	Model API
Text-to-Audio(zh)	Converts text to audio.	Model API
Image-Chat(en)	Image chat.	Model API
Translation-zh2en	Translates Chinese text to English.	Model API
Translation-en2zh	Translates English text to Chinese.	Model API
Universal-IE(zh)	Extracts structured information.	Model API
Text-to-Geographic(zh)	Extracts geographic information.	Model API
NER(zh)	Recognizes named entities in text.	Model API
API-Retrieval	Retrieves relevant APIs	Common API
ModelScope-Retrieval	Retrieves modelscope docs.	Common API

Table 2: The statistics of default tool list. Supported input languages for the APIs are listed in parentheses.

A.2 CustomTool

User can customize their own tools by inheriting a base tool and defining the tool names, descriptions, and parameters according to a pre-defined schema. Moreover, you can implement *_local_call()* or *_remote_call()* depending on your specific requirements. To illustrate, below is an example of a custom tool:

```
class CustomTool(Tool):
    description = 'xxx'
    name = 'xxx'
    parameters: list = [{
        'name': 'xxx',
        'description': 'xxx',
        'required': True
    }]

    def _local_call():
        ...

    def _remote_call():
        ...
```

B Experiment Setup

B.1 Evaluation Benchmark

To assess the generalization of the trained agent, we include 10 in-domain APIs that appear in the training set of ModelScope-Agent and 10 real unseen APIs[10]. We also account for the multi-turn ability of the agent by annotating several multi-turn scenarios in our evaluation benchmark. Our test instances were annotated by asking the human experts to write diverse instructions first. Then the human experts were ask to write the JSON API request and answer the instructions properly after obtaining the API calling results. Our final testing dataset consisted of 360 conversations with 2059 text snippets as the references to be compared with the agent prediction, which comprise 798 API requsts and 1261 plain text answers according to the previous calling results.

[10]In progress, we will include more APIs in the future.

B.2 Evaluation Results

Model	ROUGE-L	Action EM	Argument F1
ChatGPT (2-shot)*	36.70	34.82	25.51
LLaMA	39.16	58.60	44.98
ChatPLUG	46.45	68.29	55.12
MSAgent-7B	51.35	87.23	68.09

Table 3: Automatic evaluation results. * represents that we do not fine-tune ChatGPT but use in-context learning with 2 demonstrations.

We compare the models trained in our proposed ModelScopeGPT. The automatic evaluation results are shown in Table 3. Based on the findings obtained from our experimentation, it is evident that ChatGPT with in-context learning yielded inferior results as compared to other models that were subjected to finetuning. Furthermore, LLaMA underperformed when compared to other fine-tuned models. Our error study revealed that the lower performance of ChatGPT and LLaMA could be attributed to a large proportion of Chinese test cases in our test set. The models (ChatPLUG, MSAgent-7B[11]) that performed better were those that predominantly focused on Chinese data. Our investigation revealed that ChatGPT and LLaMA exhibited limitations in user intent recognition, which ultimately led to their suboptimal performance on Action EM. Among the models examined, MSAgent-7B displayed the most favorable performance, which could be attributed to the superior performance of its basic model.

B.3 Weighted LM

We give an example of the training strategy Weighted LM. As show in Figure 4, tokens with different colors have different loss weights. For the user input prompt, we set the loss weight to 0, so that the model does not calculate the loss for the prompt. For the API-Agnostic text of the assistant, we keep the loss weight as 1. Finally, for the important text of the API calling, such as API name, parameters, URL, etc., we set the loss weight to 2, which can improve the generation accuracy of API calling.

[11]https://modelscope.cn/models/damo/ModelScope-Agent-7B

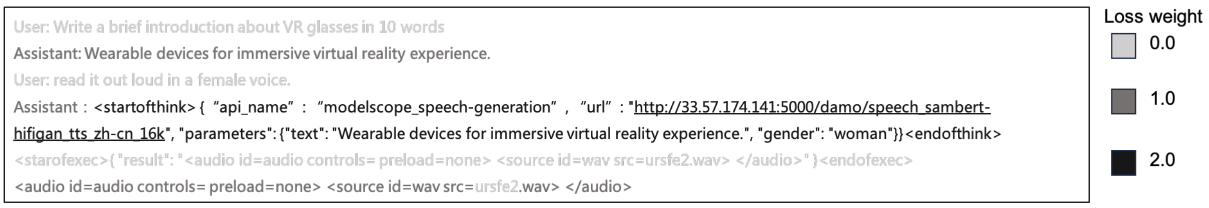
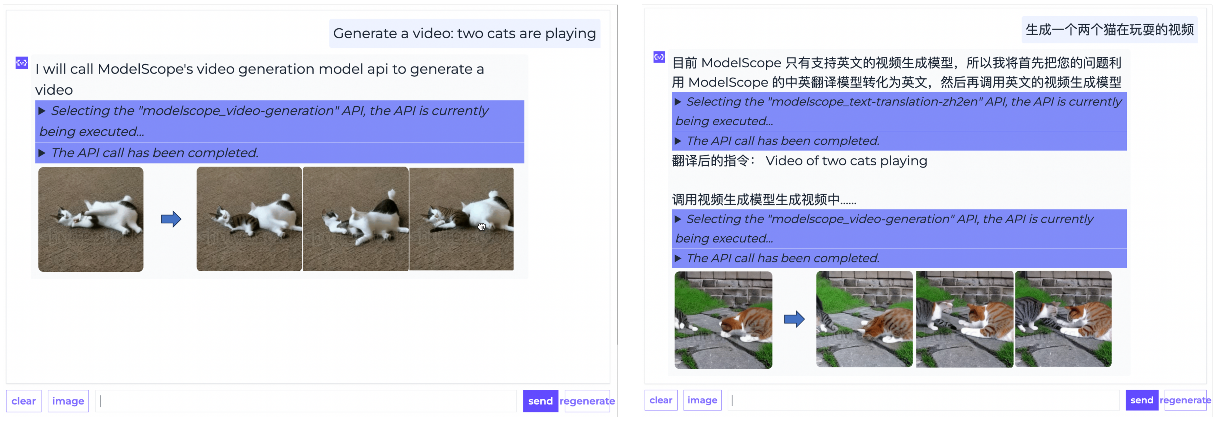
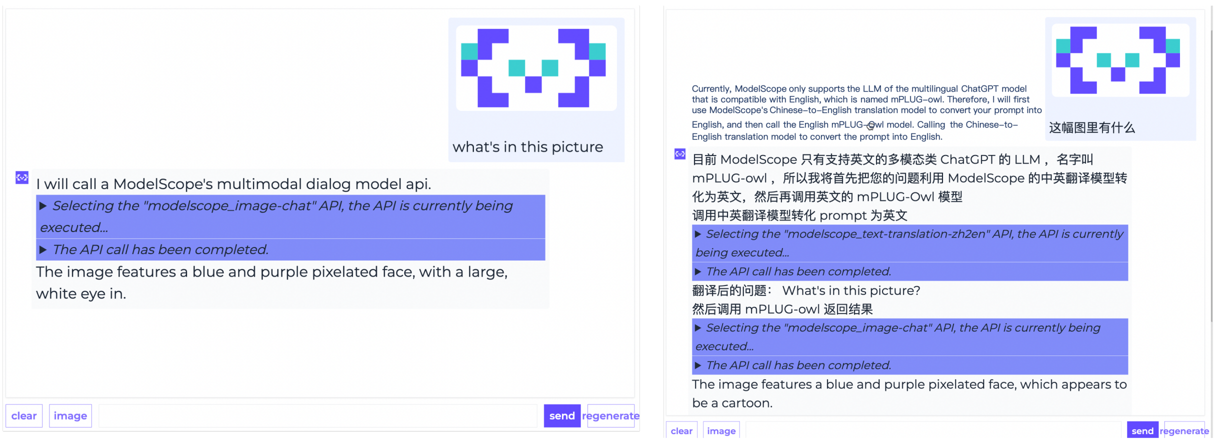

User: Write a brief introduction about VR glasses in 10 words
Assistant: Wearable devices for immersive virtual reality experience.
User: read it out loud in a female voice.
Assistant：<startofthink>{ “api_name” ： “modelscope_speech-generation” ， “url” ："http://33.57.174.141:5000/damo/speech_sambert-hifigan_tts_zh-cn_16k", "parameters":{"text": "Wearable devices for immersive virtual reality experience.", "gender": "woman"}}<endofthink>
<starofexec>{"result": "<audio id=audio controls= preload=none> <source id=wav src=ursfe2.wav> </audio>" }<endofexec>
<audio id=audio controls= preload=none> <source id=wav src=ursfe2.wav> </audio>

Figure 4: Example of training strategy for weighted LM. Different colored tokens have different loss weights.


Generate a video: two cats are playing
I will call ModelScope's video generation model api to generate a video
▸ Selecting the "modelscope_video-generation" API, the API is currently being executed...
▸ The API call has been completed.

生成一个两个猫在玩耍的视频
目前 ModelScope 只有支持英文的视频生成模型，所以我将首先把您的问题利用 ModelScope 的中英翻译模型转化为英文，然后再调用英文的视频生成模型
▸ Selecting the "modelscope_text-translation-zh2en" API, the API is currently being executed...
▸ The API call has been completed.
翻译后的指令： Video of two cats playing
调用视频生成模型生成视频中......
▸ Selecting the "modelscope_video-generation" API, the API is currently being executed...
▸ The API call has been completed.

Figure 5: Single-step tool-use instructions, text-to-video cases. We have captured a few frames of the video to display. Testing the model using the same semantic instruction in both English (left) and Chinese (right).


what's in this picture
I will call a ModelScope's multimodal dialog model api.
▸ Selecting the "modelscope_image-chat" API, the API is currently being executed...
▸ The API call has been completed.
The image features a blue and purple pixelated face, with a large, white eye in.

Currently, ModelScope only supports the LLM of the multilingual ChatGPT model that is compatible with English, which is named mPLUG–owl. Therefore, I will first use ModelScope's Chinese–to–English translation model to convert your prompt into English, and then call the English mPLUG–Owl model. Calling the Chinese–to–English translation model to convert the prompt into English.
这幅图里有什么
目前 ModelScope 只有支持英文的多模态类 ChatGPT 的 LLM， 名字叫 mPLUG-owl， 所以我将首先把您的问题利用 ModelScope 的中英翻译模型转化为英文，然后再调用英文的 mPLUG-Owl 模型
调用中英翻译模型转化 prompt 为英文
▸ Selecting the "modelscope_text-translation-zh2en" API, the API is currently being executed...
▸ The API call has been completed.
翻译后的问题： What's in this picture?
然后调用 mPLUG-owl 返回结果
▸ Selecting the "modelscope_image-chat" API, the API is currently being executed...
▸ The API call has been completed.
The image features a blue and purple pixelated face, which appears to be a cartoon.

Figure 6: Single-step tool-use instructions, image-chat cases. Testing the model using the same semantic instruction in both English (left) and Chinese (right).

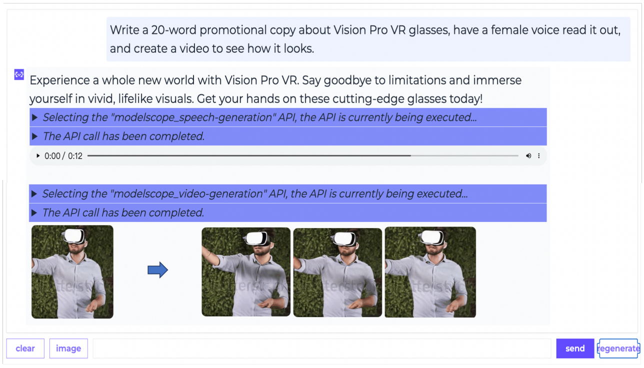
C Cases

In this section, we show the qualitative results about ModelScopeGPT implementation based on ModelScope-Agent.

Single-step Tool Use As shown in Figure 5 and 6, the instruction expects the model to generate a video and chat about the image respectively. These instructions can be completed with a single step of tool use.

Multi-step Tool Use As shown in Figure 7, the instruction expects the model to write the promotional copy first, then read it, and finally generate a video. These instructions require the model to have the ability of multi-step Tool use. In the Chinese case, our model accurately completed the three-step tool use.

Multi-turn Tool Use As shown in Figure 8, the instruction requires the model to have the ability to multi-turn conversation and use the history conversation. Our model can accurately call the API and capture the content of the previous conversation to generate API parameters.



Write a 20-word promotional copy about Vision Pro VR glasses, have a female voice read it out, and create a video to see how it looks.

Experience a whole new world with Vision Pro VR. Say goodbye to limitations and immerse yourself in vivid, lifelike visuals. Get your hands on these cutting-edge glasses today!

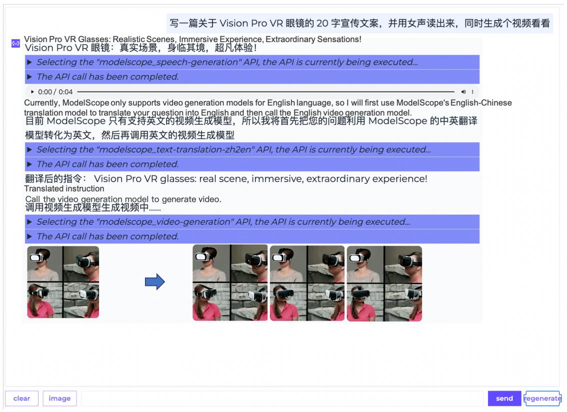
▶ *Selecting the "modelscope_speech-generation" API, the API is currently being executed...*

▶ *The API call has been completed.*

▸ 0:00 / 0:12

▶ *Selecting the "modelscope_video-generation" API, the API is currently being executed...*

▶ *The API call has been completed.*



写一篇关于 Vision Pro VR 眼镜的 20 字宣传文案，并用女声读出来，同时生成个视频看看

Vision Pro VR Glasses: Realistic Scenes, Immersive Experience, Extraordinary Sensations!
Vision Pro VR 眼镜：真实场景，身临其境，超凡体验！

▶ *Selecting the "modelscope_speech-generation" API, the API is currently being executed...*

▶ *The API call has been completed.*

▸ 0:00 / 0:04

Currently, ModelScope only supports video generation models for English language, so I will first use ModelScope's English-Chinese translation model to translate your question into English and then call the English video generation model.
目前 ModelScope 只有支持英文的视频生成模型，所以我将首先把您的问题利用 ModelScope 的中英翻译模型转化为英文，然后再调用英文的视频生成模型

▶ *Selecting the "modelscope_text-translation-zh2en" API, the API is currently being executed...*

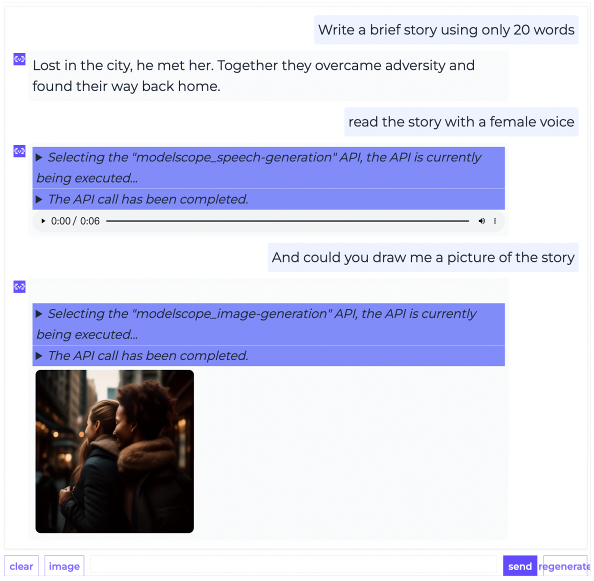
▶ *The API call has been completed.*

翻译后的指令： Vision Pro VR glasses: real scene, immersive, extraordinary experience!
Translated instruction
Call the video generation model to generate video.
调用视频生成模型生成视频中......

▶ *Selecting the "modelscope_video-generation" API, the API is currently being executed...*

▶ *The API call has been completed.*

Figure 7: Multi-step tool-use instructions. We have captured a few frames of the video to display. Testing the model using the same semantic instruction in both English(left) and Chinese(right).



Write a brief story using only 20 words

Lost in the city, he met her. Together they overcame adversity and found their way back home.

read the story with a female voice

▶ *Selecting the "modelscope_speech-generation" API, the API is currently being executed...*

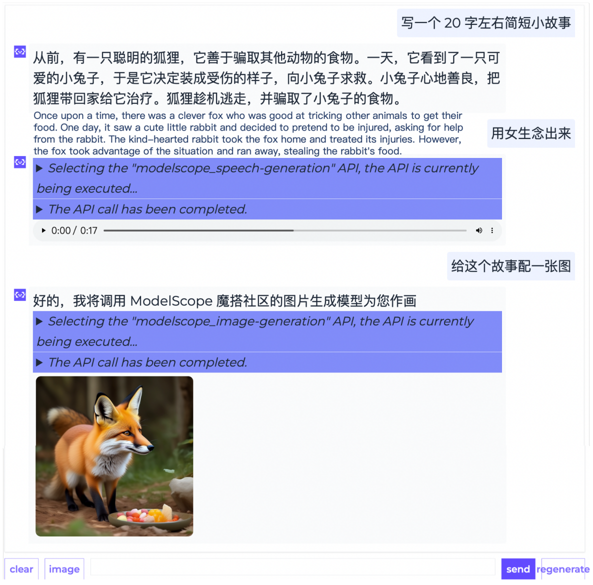
▶ *The API call has been completed.*

▸ 0:00 / 0:06

And could you draw me a picture of the story

▶ *Selecting the "modelscope_image-generation" API, the API is currently being executed...*

▶ *The API call has been completed.*



写一个 20 字左右简短小故事

从前，有一只聪明的狐狸，它善于骗取其他动物的食物。一天，它看到了一只可爱的小兔子，于是它决定装成受伤的样子，向小兔子求救。小兔子心地善良，把狐狸带回家给它治疗。狐狸趁机逃走，并骗取了小兔子的食物。

Once upon a time, there was a clever fox who was good at tricking other animals to get their food. One day, it saw a cute little rabbit and decided to pretend to be injured, asking for help from the rabbit. The kind–hearted rabbit took the fox home and treated its injuries. However, the fox took advantage of the situation and ran away, stealing the rabbit's food.

用女生念出来

▶ *Selecting the "modelscope_speech-generation" API, the API is currently being executed...*

▶ *The API call has been completed.*

▸ 0:00 / 0:17

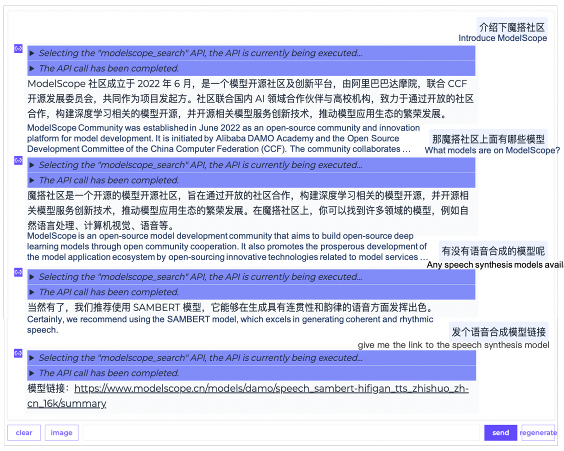
给这个故事配一张图

好的，我将调用 ModelScope 魔搭社区的图片生成模型为您作画

▶ *Selecting the "modelscope_image-generation" API, the API is currently being executed...*

▶ *The API call has been completed.*

Figure 8: Multi-turn tool-use instructions, text-to-speech and text-to-image cases. Testing the model using the same semantic instruction in both English(left) and Chinese(right).



介绍下魔搭社区
Introduce ModelScope

▶ *Selecting the "modelscope_search" API, the API is currently being executed...*

▶ *The API call has been completed.*

ModelScope 社区成立于 2022 年 6 月，是一个模型开源社区及创新平台，由阿里巴巴达摩院，联合 CCF 开源发展委员会，共同作为项目发起方。社区联合国内 AI 领域合作伙伴与高校机构，致力于通过开放的社区合作，构建深度学习相关的模型开源，并开源相关模型服务创新技术，推动模型应用生态的繁荣发展。
ModelScope Community was established in June 2022 as an open-source community and innovation platform for model development. It is initiated by Alibaba DAMO Academy and the Open Source Development Committee of the China Computer Federation (CCF). The community collaborates ...

那魔搭社区上面有哪些模型
What models are on ModelScope?

▶ *Selecting the "modelscope_search" API, the API is currently being executed...*

▶ *The API call has been completed.*

魔搭社区是一个开源的模型开源社区，旨在通过开放的社区合作，构建深度学习相关的模型开源，并开源相关模型服务创新技术，推动模型应用生态的繁荣发展。在魔搭社区上，你可以找到许多领域的模型，例如自然语言处理、计算机视觉、语音等。
ModelScope is an open-source model development community that aims to build open-source deep learning models through open community cooperation. It also promotes the prosperous development of the model application ecosystem by open-sourcing innovative technologies related to model services ...

有没有语音合成的模型呢
Any speech synthesis models available?

▶ *Selecting the "modelscope_search" API, the API is currently being executed...*

▶ *The API call has been completed.*

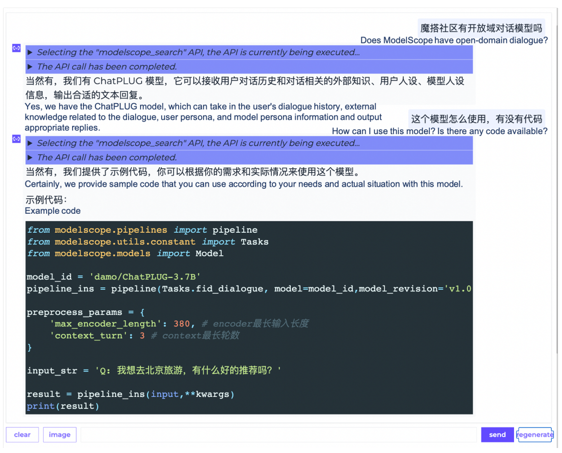
当然有了，我们推荐使用 SAMBERT 模型，它能够在生成具有连贯性和韵律的语音方面发挥出色。
Certainly, we recommend using the SAMBERT model, which excels in generating coherent and rhythmic speech.

发个语音合成模型链接
give me the link to the speech synthesis model

▶ *Selecting the "modelscope_search" API, the API is currently being executed...*

▶ *The API call has been completed.*

模型链接：https://www.modelscope.cn/models/damo/speech_sambert-hifigan_tts_zhishuo_zh-cn_16k/summary



魔搭社区有开放域对话模型吗
Does ModelScope have open-domain dialogue?

▶ *Selecting the "modelscope_search" API, the API is currently being executed...*

▶ *The API call has been completed.*

当然有，我们有 ChatPLUG 模型，它可以接收用户对话历史和对话相关的外部知识、用户人设、模型人设信息，输出合适的文本回复。
Yes, we have the ChatPLUG model, which can take in the user's dialogue history, external knowledge related to the dialogue, user persona, and model persona information and output appropriate replies.

这个模型怎么使用，有没有代码
How can I use this model? Is there any code available?

▶ *Selecting the "modelscope_search" API, the API is currently being executed...*

▶ *The API call has been completed.*

当然有，我们提供了示例代码，你可以根据你的需求和实际情况来使用这个模型。
Certainly, we provide sample code that you can use according to your needs and actual situation with this model.

示例代码：
Example code

```
from modelscope.pipelines import pipeline
from modelscope.utils.constant import Tasks
from modelscope.models import Model

model_id = 'damo/ChatPLUG-3.7B'
pipeline_ins = pipeline(Tasks.fid_dialogue, model=model_id,model_revision='v1.0

preprocess_params = {
    'max_encoder_length': 380, # encoder最长输入长度
    'context_turn': 3 # context最长轮数
}

input_str = 'Q: 我想去北京旅游，有什么好的推荐吗？ '

result = pipeline_ins(input,**kwargs)
print(result)
```

Figure 9: Multi-turn tool-use instructions, text-to-speech and text-to-image cases. Testing the model using the same semantic instruction in both English(left) and Chinese(right).

In-domain Knowledge QA As shown in Figure 9, the instruction requires the model to retrieve in-domain knowledge and use the retrieved knowledge to answer questions.

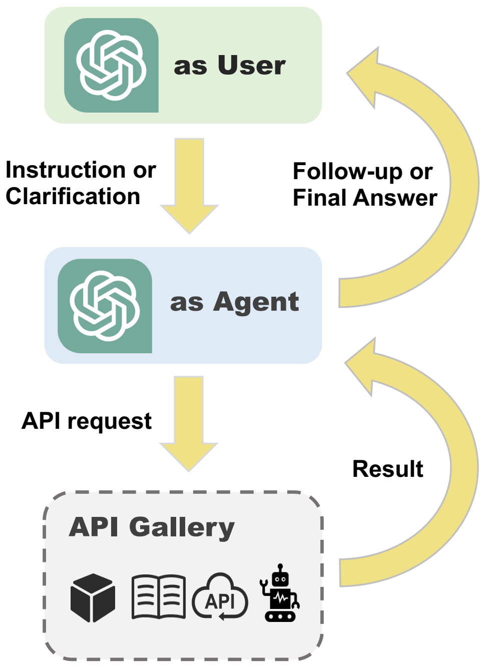


Figure 10: The data collection procedure of MSAgent-Bench.

D Data Collection Procedure

We collected our dataset by using prompt engineer to simulate the agent scenarios with two ChatGPTs (`gpt-3.5-turbo`). One of the ChatGPTs was prompted to act as the user, while the other was assigned to act as the agent. In order to expand the domains and functionalities of APIs presented in the training data, rather than the exsisting real APIs, we also included a number of synthetic APIs that were generated by ChatGPT. When these synthetic APIs were incorporated into the dialogues, we prompted another ChatGPT to serve as the API and return the relevant calling outcomes.

The data collection procedure is shown in Figure 10. Initially, a set of random in-context demonstrations were provided to ChatGPT for generating an instruction. This instruction could either be a regular one or one that requires solving with APIs, depending on the demonstrations provided. Subsequently, ChatGPT was prompt to act as an agent by first thinking about which action to undertake. If no API calls were deemed necessary, or if the user clarification is needed, the agent would respond with a follow-up response to the user. Otherwise the agent will send API request to the API gallery. After receiving the result of the API call, the agent would assess the situation and decide on the next action. This iterative process of the "user-agent-API" loop would continue until the agent determined that it was appropriate to terminate the conversation with the final answer. After acquiring the raw dataset, we applied filtering mechanisms to eliminate instances in which ChatGPT generated API requests containing hallucinated API names and parameters that were absent from the retrieved API. Additionally, we excluded instances in which ChatGPT generated illegal API requests, thus resulting in a refined and finalized dataset.

As introduced in Section 3.1, we collect instances across different languages and topics, the detailed statistics of our collected data are shown in Table 4.

Instance Type	# Instances
Chinese	532,436
English	66,444
Common API	211,026
Model API	58,338
API-Oriented QA	5,000
API-Agnostic Instruction	329,776

Table 4: The statistics of our collected dataset.

E Related Work

E.1 Large Language Models

Recent years have witnessed rapid development in the field of Large Language Models (LLMs). Typical models, such as GPT3 (Brown et al., 2020), Gopher (Rae et al., 2021), Chinchilla (Hoffmann et al., 2022), PaLM (Chowdhery et al., 2022) and LLaMA (Touvron et al., 2023), have shown impressive zero and few-shot generalization abilities on a wide range of NLP tasks, by scaling up the model and data size. A remarkable milestone is the release of ChatGPT (OpenAI, 2022) or GPT4 (OpenAI, 2023), which has greatly revolutionized the paradigm of AI development. As a result, a rising trend of open-source LLMs has emerged to challenge and catch up their closed-source counterparts like ChatGPT and Claude, such as BLOOM (Muennighoff et al., 2022), LLaMA (Touvron et al., 2023), Falcon (Almazrouei et al., 2023), ChatGLM (THUDM, 2023). Despite the great breakthrough, LLMs are trained as text generators over plain text corpora, thus performing less well on other tasks such as multi-modal tasks. It also falls short on tasks that require up-to-date information, which are beyond the pretraining data. Using tools or external APIs can help overcome the limitations and harness the power of LLMs to facilitate seamless connections with downstream applications. In ModelScope-Agent , we provide the whole customizable framework and best practices for building an agent system, which enables open-source LLMs to use tools and external APIs.

E.2 Agent & Tool Learning

The utilization of Large Language Models (LLMs) as a controller to construct an agent system has emerged as a prominent research area. Several related works employ prompt engineering techniques on closed-source LLMs, such as ChatGPT (OpenAI, 2022) and Claude, to enable their application in specific domains. For instance, Visual-ChatGPT (Wu et al., 2023) and HuggingGPT (Shen et al., 2023) facilitate the HuggingFace model callings accessible to OpenAI LLMs. SayCan (Ahn et al., 2022) and inner monologue (Huang et al., 2023) integrate LLMs with robots to achieve robotic systems. Notably, recent works such as Langchain and Auto-GPT encompass a wide range of tools, including common APIs and neural models, and enhance long-term reasoning and human-agent interaction whilst solving tasks, which demonstrate the immense potential for building a generalized agent.

Numerous endeavors have also been made to enable open-source LLMs to utilize tools. For instance, Gorilla (Patil et al., 2023) and GPT4Tools (Yang et al., 2023) generate training data using self-instruction techniques to train open-source LLMs to effectively utilize neural models. ToolAlpaca (Tang et al., 2023) and ToolLLaMA (Qin et al., 2023) train LLAMA using common APIs, with the distinction that ToolAlpaca employs synthetic APIs from LLMS, whereas ToolLLaMA utilizes real APIs.

Overall, compared to the above-mentioned methods, ModelScope-Agent differs in the following aspects. Firstly, our method includes a universal training framework that supports user-customized agent learning for open-source models to meet industrial needs. Secondly, ModelScope-Agent can support various APIs in different fields, including model APIs and common APIs, while previous works only support certain specific APIs.

F Future Work

In the future, we will evolve to support more sophisticated agent architectures such as ReAct and code interpreter. In the meantime, we will continuously improve the capabilities required by open-source LLMs as agents. ModelScope-Agent relies on the ModelScope community and will adapt to more new open-source LLMs in the future, providing more applications developed based on ModelScope-Agent, such as personal-assistant-agent, story-agent, motion agent, and so on.