Lesson 6 笔记

Lagent & AgentLego 智能体应用搭建

使用 Legend 和 Agent Lego 搭建智能体应用。智能体应运而生，它由大脑、感知和动作三部分组成，能够感知环境、影响环境并进行推理。

Legend 是一个轻量级的智能体框架，支持多种范式和工具。

Agent Lego 则是一个多模态工具包，可以快速构建自定义工具。

通过实战环节，学习如何配置环境，使用 Legend 的 web demo 以及自定义工具，还体验了如何使用 Agent Lego 组装智能体并配置目标检测工具。

最后了解到如何通过 Agent Lego 实现自定义工具，比如调用 Magic Maker 的 API 生成图像。

Lagent 目前已经支持了包括 AutoGPT、ReAct 等在内的多个经典智能体范式，也支持了如下工具：

- Arxiv 搜索
- Bing 地图
- Google 学术搜索
- Google 搜索
- 交互式 IPython 解释器
- IPython 解释器
- PPT
- Python 解释器

AgentLego 是一个提供了多种开源工具 API 的多模态工具包，旨在像是乐高积木一样，让用户可以快速简便地拓展自定义工具，从而组装出自己的智能体。通过 AgentLego 算法库，不仅可以直接使用多种工具，也可以利用这些工具，在相关

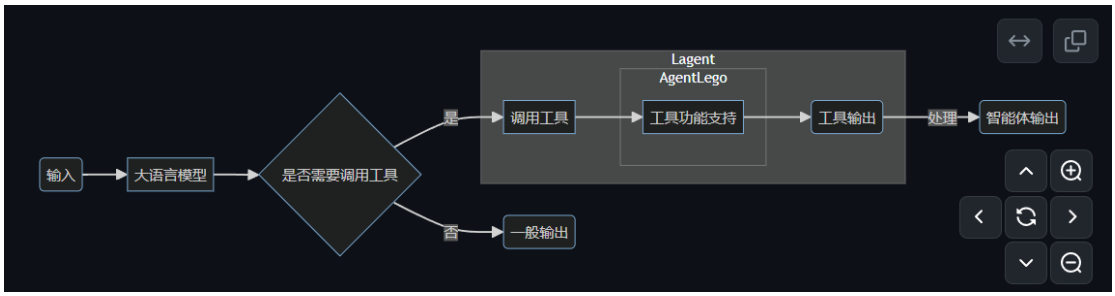
智能体框架（如 Lagent，Transformers Agent 等）的帮助下，快速构建可以增强大语言模型能力的智能体。

AgentLego 目前提供了如下工具：

通用能力	语音相关	图像处理	AIGC

• 计算器
• 谷歌搜索
	• 文本 -> 音频（TTS）
	• 音频 -> 文本（STT）
		• 描述输入图像
		• 识别文本（OCR）
		• 视觉问答（VQA）
		• 人体姿态估计
		• 人脸关键点检测
		• 图像边缘提取（Canny）
		• 深度图生成
		• 生成涂鸦（Scribble）
		• 检测全部目标
		• 检测给定目标
		• SAM
		  ◦ 分割一切
		  ◦ 分割给定目标
			• 文生图
			• 图像拓展
			• 删除给定对象
			• 替换给定对象
			• 根据指令修改
			• ControlNet 系列
			  ◦ 根据边缘+描述生成
			  ◦ 根据深度图+描述生成
			  ◦ 根据姿态+描述生成
			  ◦ 根据涂鸦+描述生成
			• ImageBind 系列
			  ◦ 音频生成图像
			  ◦ 热成像生成图像
			  ◦ 音频+图像生成图像
			  ◦ 音频+文本生成图像

Lagent 是一个智能体框架，而 AgentLego 与大模型智能体并不直接相关，而是作为工具包，在相关智能体的功能支持模块发挥作用。



创建开发机和 conda 环境

在创建开发机界面选择镜像为 Cuda12.2-conda，并选择 GPU 为 30% A100。

进入开发机后，为了方便使用，我们需要配置一个环境以同时满足 Lagent 和 AgentLego 运行时所需依赖。在开始配置环境前，我们先创建一个用于存放 Agent 相关文件的目录，可以执行如下命令：

mkdir -p /root/agent

studio-conda -t agent -o pytorch-2.1.2

conda create -n agent

conda activate agent

conda install python=3.10

conda install pytorch==2.1.2 torchvision==0.16.2 torchaudio==2.1.2 pytorch-cuda=11.8 -c pytorch -c nvidia

安装 Lagent 和 AgentLego

Lagent 和 AgentLego 都提供了两种安装方法，一种是通过 pip 直接进行安装，另一种则是从源码进行安装。为了方便使用 Lagent 的 Web Demo 以及 AgentLego 的 WebUI，我们选择直接从源码进行安装。 此处附上源码安装的相关帮助文档：

Lagent：https://lagent.readthedocs.io/zh-cn/latest/get_started/install.html

AgentLego：https://agentlego.readthedocs.io/zh-cn/latest/get_started.html

可以执行如下命令进行安装：

cd /root/agent

conda activate agent

git clone https://gitee.com/internlm/lagent.git

cd lagent && git checkout 581d9fb && pip install -e . && cd ..

git clone https://gitee.com/internlm/agentlego.git

cd agentlego && git checkout 7769e0d && pip install -e . && cd ..

pip install lmdeploy==0.3.0

cd /root/agent

git clone -b camp2 https://gitee.com/internlm/Tutorial.git

2. Lagent：轻量级智能体框架

Lagent Web Demo

由于 Lagent 的 Web Demo 需要用到 LMDeploy 所启动的 api_server，因此我们首先按照下图指示在 vscode terminal 中执行如下代码使用 LMDeploy 启动一个 api_server。

conda activate agent

lmdeploy serve api_server

/root/share/new_models/Shanghai_AI_Laboratory/internlm2-chat-7b \

--server-name 127.0.0.1 \

--model-name internlm2-chat-7b \

--cache-max-entry-count 0.1

cd /root/agent/lagent/examples

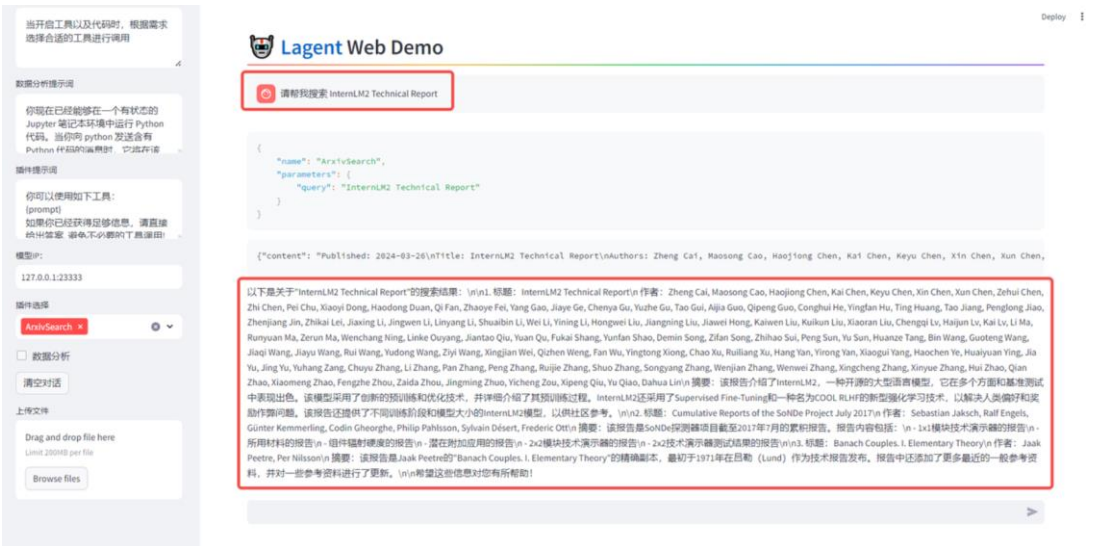
streamlit run internlm2_agent_web_demo.py --server.address 127.0.0.1 --server.port 7860

在等待 LMDeploy 的 api_server 与 Lagent Web Demo 完全启动后（如下图所示），在本地进行端口映射，将 LMDeploy api_server 的 23333 端口以及 Lagent Web Demo 的 7860 端口映射到本地。可以执行：

ssh -CNg -L 7860:127.0.0.1:7860 -L 23333:127.0.0.1:23333 root@ssh.intern-ai.org.cn -p 你的 ssh 端口号

接下来在本地的浏览器页面中打开 http://localhost:7860 以使用 Lagent Web Demo。首先输入模型 IP 为 127.0.0.1:23333，在输入完成后按下回车键以确认。并选择插件为 ArxivSearch，以让模型获得在 arxiv 上搜索论文的能力。

输入“请帮我搜索 InternLM2 Technical Report” 以让模型搜索书生·浦语 2 的技术报告。效果如下图所示，可以看到模型正确输出了 InternLM2 技术报告的相关信息。尽管还输出了其他论文，但这是由 arxiv 搜索 API 的相关行为导致的。



用 Lagent 自定义工具

使用 Lagent 自定义工具主要分为以下几步：

继承 BaseAction 类

实现简单工具的 run 方法；或者实现工具包内每个子工具的功能

简单工具的 run 方法可选被 tool_api 装饰；工具包内每个子工具的功能都需要被 tool_api 装饰。

下面我们将实现一个调用和风天气 API 的工具以完成实时天气查询的功能。

通过 touch /root/agent/lagent/lagent/actions/weather.py（大小写敏感）新建工具文件。

为了获得稳定的天气查询服务，我们首先要获取 API KEY。首先打开 https://dev.qweather.com/docs/api/ 后，点击右上角控制台。

进入控制台后，点击左侧项目管理，然后点击右上角创建项目以创建新项目。

输入相关项目名称，选择免费订阅，Web API 以及输入 key 的名称。（项目名称和 key 的名词自由输入即可），接下来回到项目管理页面，查看我们刚刚创建的 key，并且复制好以供 2.3 节中使用。

体验自定义工具效果

与 1.2 部分类似，我们在两个 terminal 中分别启动 LMDeploy 服务和 Tutorial 已经写好的用于这部分的 Web Demo：

conda activate agent

lmdeploy serve api_server

/root/share/new_models/Shanghai_AI_Laboratory/internlm2-chat-7b \

--server-name 127.0.0.1 \

--model-name internlm2-chat-7b \

--cache-max-entry-count 0.1

export WEATHER_API_KEY=在 2.2 节获取的 API KEY

比如 export WEATHER_API_KEY=1234567890abcdef

conda activate agent

cd /root/agent/Tutorial/agent

streamlit run internlm2_weather_web_demo.py --server.address 127.0.0.1 --server.port 7860

并在本地执行如下操作以进行端口映射：

ssh -CNg -L 7860:127.0.0.1:7860 -L 23333:127.0.0.1:23333 root@ssh.intern-ai.org.cn -p 你的 ssh 端口号

3. AgentLego：组装智能体“乐高”

直接使用 AgentLego 工具，体验 AgentLego 的 WebUI，以及基于 AgentLego 自定义工具并体验自定义工具的效果。

直接使用 AgentLego

cd /root/agent

wget http://download.openmmlab.com/agentlego/road.jpg

由于 AgentLego 在安装时并不会安装某个特定工具的依赖，因此我们接下来准备安装目标检测工具运行时所需依赖。

AgentLego 所实现的目标检测工具是基于 mmdet (MMDetection) 算法库中的 RTMDet-Large 模型，因此我们首先安装 mim，然后通过 mim 工具来安装 mmdet。这一步所需时间可能会较长，请耐心等待。

conda activate agent

pip install openmim==0.3.9

mim install mmdet==3.3.0

在安装完成后，可能会观察到以下现象（如下图所示），但请放心，这是正常现象，这并不会影响到我们的使用。

然后通过 touch /root/agent/direct_use.py（大小写敏感）的方式在 /root/agent 目录下新建 direct_use.py 以直接使用目标检测工具，

direct_use.py 的代码如下：

```
import re

import cv2

from agentlego.apis import load_tool
```

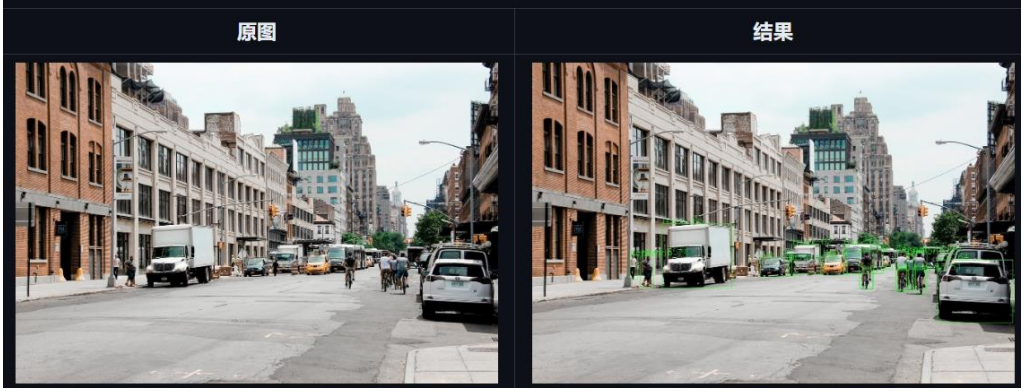
```
# load tool

tool = load_tool('ObjectDetection', device='cuda')

# apply tool

visualization = tool('/root/agent/road.jpg')

print(visualization)

# visualize

image = cv2.imread('/root/agent/road.jpg')

preds = visualization.split('\n')

pattern = r'(\w+) \((\d+), (\d+), (\d+), (\d+)\), score (\d+)'

for pred in preds:

    name, x1, y1, x2, y2, score = re.match(pattern, pred).groups()

    x1, y1, x2, y2, score = int(x1), int(y1), int(x2), int(y2), int(score)

    cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 1)

    cv2.putText(image, f'{name} {score}', (x1, y1),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 1)

cv2.imwrite('/root/agent/road_detection_direct.jpg', image)
```

此时文件树结构如下：

```
/root/agent
├── agentlego
│   ├── agentlego
│   ├── docs
│   ├── examples
│   ├── LICENSE
│   └── ...
├── lagent
│   ├── docs
│   ├── examples
│   ├── lagent
│   ├── LICENSE
│   └── ...
├── Tutorial
│   ├── assets
│   ├── agent
│   ├── helloword
│   ├── huixiangdou
│   └── ...
├── direct_use.py
└── road.jpg
```

接下来在执行 python /root/agent/direct_use.py 以进行推理。在等待 RTMDet-Large 权重下载并推理完成后，我们就可以看到如下输出以及一张位于 /root/agent 名为 road_detection_direct.jpg 的图片：

```
truck (345, 428, 528, 599), score 83
car (771, 510, 837, 565), score 81
car (604, 518, 677, 569), score 75
person (866, 503, 905, 595), score 74
person (287, 513, 320, 596), score 74
person (964, 502, 999, 604), score 72
person (1009, 503, 1047, 602), score 69
person (259, 510, 279, 575), score 65
car (1074, 524, 1275, 691), score 64
person (993, 508, 1016, 597), score 62
truck (689, 483, 764, 561), score 62
bicycle (873, 551, 903, 602), score 60
person (680, 523, 699, 567), score 55
bicycle (968, 551, 996, 609), score 53
bus (826, 482, 930, 560), score 52
bicycle (1011, 551, 1043, 617), score 51
```



作为智能体工具使用

修改相关文件

由于 AgentLego 算法库默认使用 InternLM2-Chat-20B 模型，因此我们首先需要修改 /root/agent/agentlego/webui/modules/agents/lagent_agent.py 文件的第 105 行位置，将 internlm2-chat-20b 修改为 internlm2-chat-7b，即

```
def llm_internlm2_lmdeploy(cfg):
    url = cfg['url'].strip()
    llm = LMDeployClient(
-           model_name='internlm2-chat-20b',
+           model_name='internlm2-chat-7b',
        url=url,
        meta_template=INTERNLM2_META,
        top_p=0.8,
        top_k=100,
        temperature=cfg.get('temperature', 0.7),
        repetition_penalty=1.0,
        stop_words=['<|im_end|>'])
    return llm
```

其中红色表示要删除的内容，绿色表示要添加的内容。

使用 LMDeploy 部署

由于 AgentLego 的 WebUI 需要用到 LMDeploy 所启动的 api_server，因此我们首先按照下图指示在 vscode terminal 中执行如下代码使用 LMDeploy 启动一个 api_server。

```
conda activate agent

lmdeploy serve api_server
/root/share/new_models/Shanghai_AI_Laboratory/internlm2-chat-7b \
                                --server-name 127.0.0.1 \
                                --model-name internlm2-chat-7b \
                                --cache-max-entry-count 0.1
```

启动 AgentLego WebUI

```
conda activate agent

cd /root/agent/agentlego/webui

python one_click.py
```

在等待 LMDeploy 的 api_server 与 AgentLego WebUI 完全启动后（如下图所示），在本地进行端口映射，将 LMDeploy api_server 的 23333 端口以及 AgentLego WebUI 的 7860 端口映射到本地。可以执行：

```
ssh -CNg -L 7860:127.0.0.1:7860 -L 23333:127.0.0.1:23333 root@ssh.intern-ai.org.cn -p 你的 ssh 端口号
```

使用 AgentLego WebUI

接下来在本地的浏览器页面中打开 http://localhost:7860 以使用 AgentLego WebUI。首先来配置 Agent，如下图所示。

点击上方 Agent 进入 Agent 配置页面。(如①所示)

点击 Agent 下方框，选择 New Agent。(如②所示)

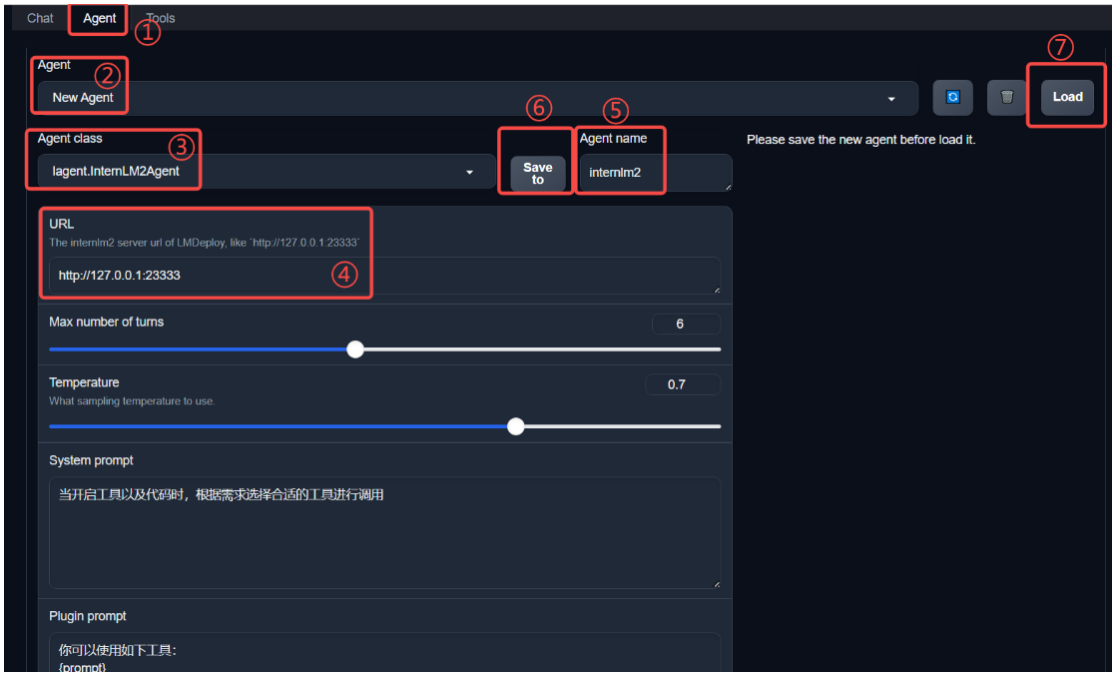
选择 Agent Class 为 lagent.InternLM2Agent。(如③所示)

输入模型 URL 为 http://127.0.0.1:23333 。(如④所示)

输入 Agent name，自定义即可，图中输入了 internlm2。(如⑤所示)

点击 save to 以保存配置，这样在下次使用时只需在第 2 步时选择 Agent 为 internlm2 后点击 load 以加载就可以了。(如⑥所示)

点击 load 以加载配置。(如⑦所示)



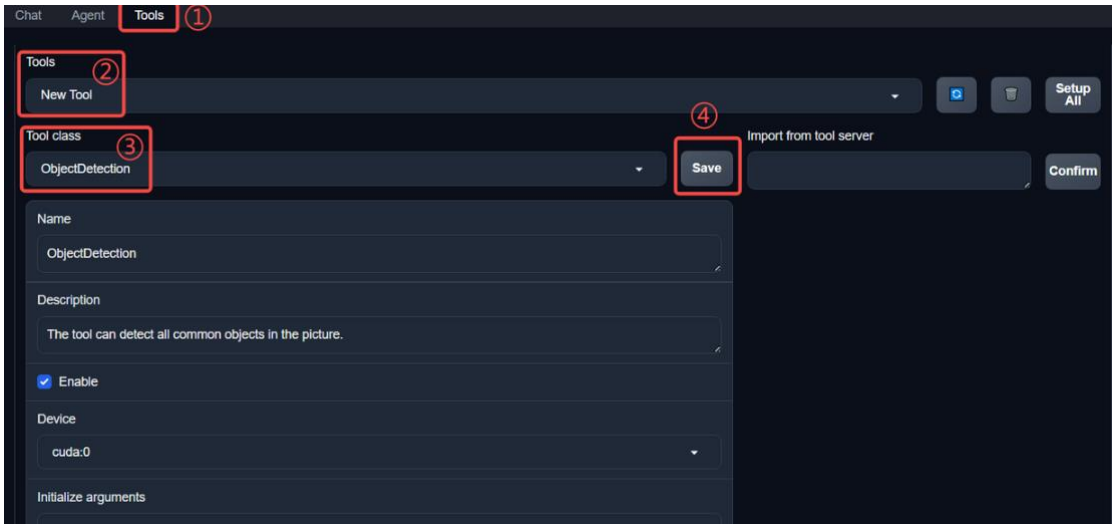
然后配置工具，如下图所示。

点击上方 Tools 页面进入工具配置页面。(如①所示)

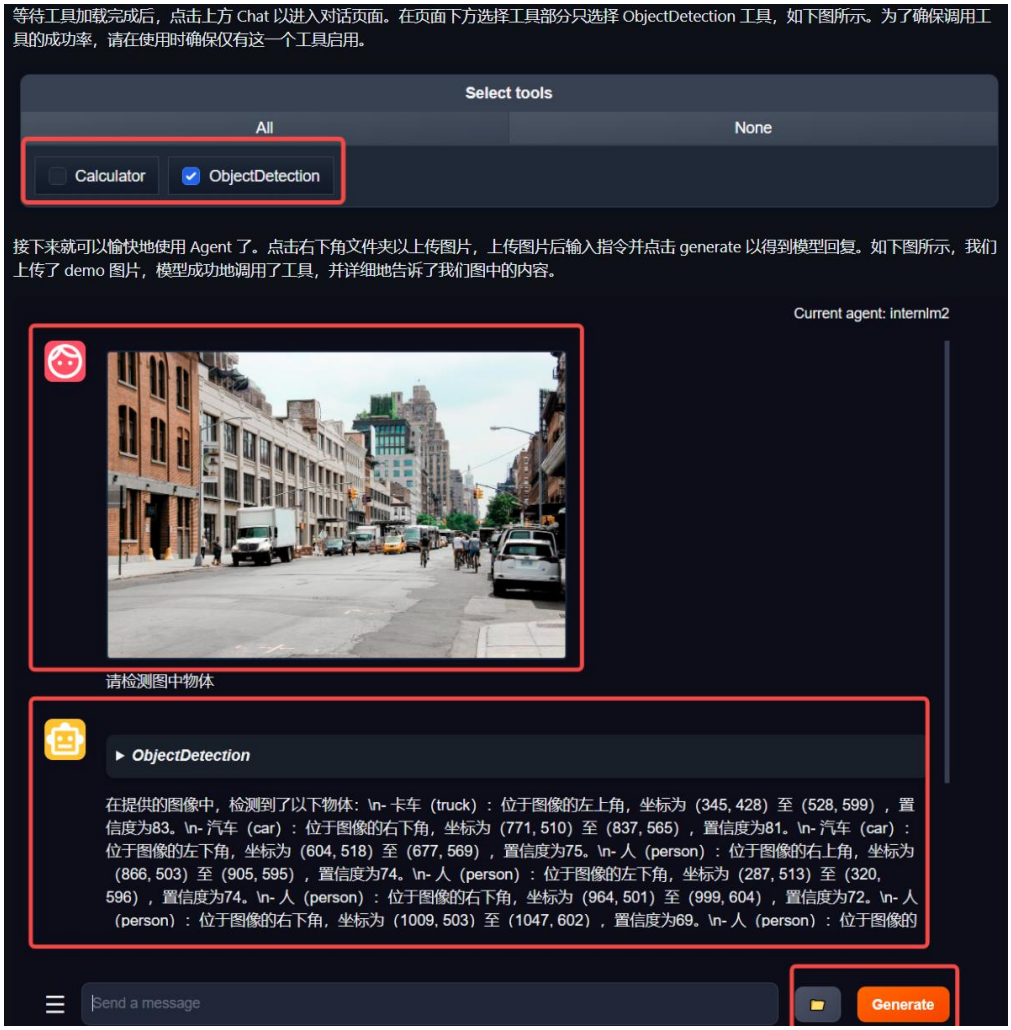
点击 Tools 下方框，选择 New Tool 以加载新工具。(如②所示)

选择 Tool Class 为 ObjectDetection。(如③所示)

点击 save 以保存配置。(如④所示)



等待工具加载完成后，点击上方 Chat 以进入对话页面。在页面下方选择工具部分只选择 ObjectDetection 工具，如下图所示。为了确保调用工具的成功率，请在使用时确保仅有这一个工具启用。



用 AgentLego 自定义工具

自定义工具主要分为以下几步：

继承 BaseTool 类

修改 default_desc 属性（工具功能描述）

如有需要，重载 setup 方法（重型模块延迟加载）

重载 apply 方法（工具功能实现）

其中第一二四步是必须的步骤。下面我们将实现一个调用 MagicMaker 的 API 以实现图像生成的工具。

MagicMaker 是汇聚了优秀 AI 算法成果的免费 AI 视觉素材生成与创作平台。主要提供图像生成、图像编辑和视频生成三大核心功能，全面满足用户在各种应用场景下的视觉素材创作需求。体验更多功能可以访问 https://magicmaker.openxlab.org.cn/home 。

3.1 创建工具文件

首先通过 touch /root/agent/agentlego/agentlego/tools/magicmaker_image_generation.py（大小写敏感）的方法新建工具文件。

3.2 注册新工具

接下来修改 /root/AgentLego/agentlego/agentlego/tools/__init__.py 文件，将我们的工具注册在工具列表中。如下所示，我们将 MagicMakerImageGeneration 通过 from .magicmaker_image_generation import MagicMakerImageGeneration 导入到了文件中，并且将其加入了 __all__ 列表中。

3.3 体验自定义工具效果

与 2.2，2.3 以及 2.4 节类似，我们在两个 terminal 中分别启动 LMDeploy 服务和 AgentLego 的 WebUI 以体验我们自定义的工具的效果。

```
conda activate agent

lmdeploy serve api_server
/root/share/new_models/Shanghai_AI_Laboratory/internlm2-chat-7b \
```

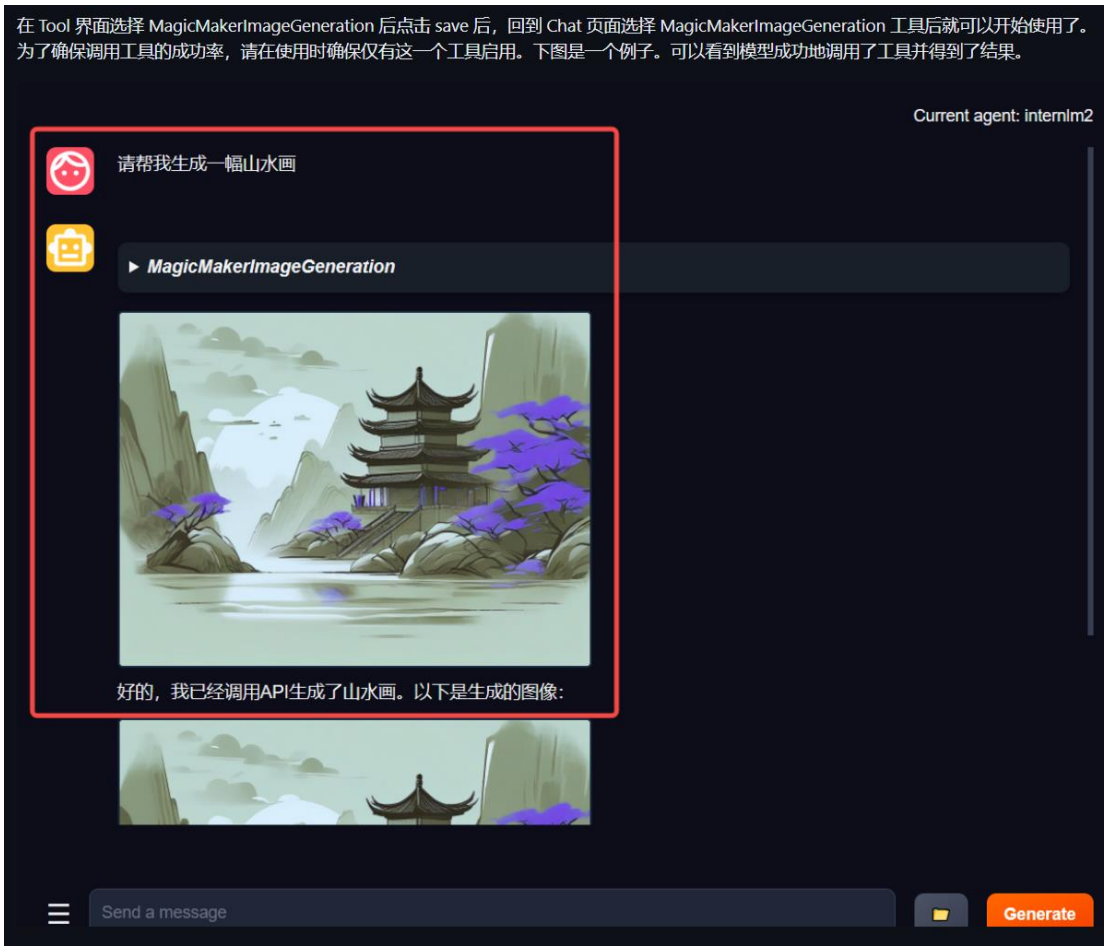
```
                --server-name 127.0.0.1 \
                --model-name internlm2-chat-7b \
                --cache-max-entry-count 0.1
```

conda activate agent

cd /root/agent/agentlego/webui

python one_click.py

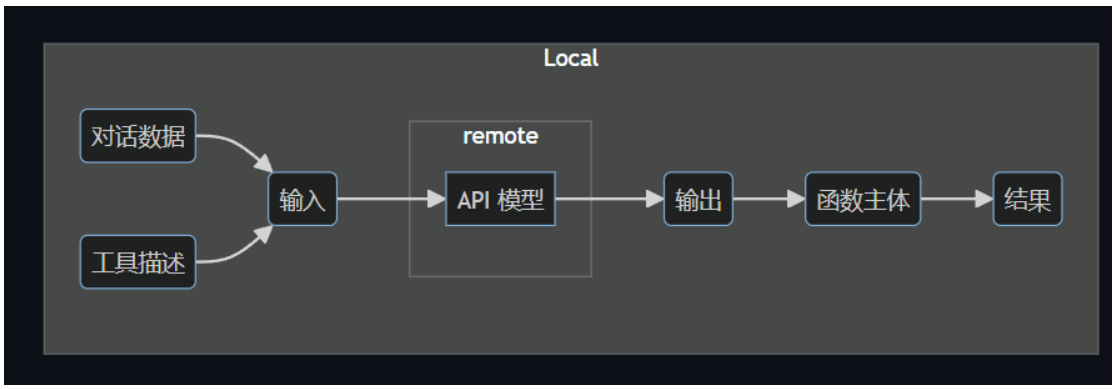
ssh -CNg -L 7860:127.0.0.1:7860 -L 23333:127.0.0.1:23333 root@ssh.intern-ai.org.cn -p 你的 ssh 端口号



4. Agent 工具能力微调

OpenAI 推出了 Function calling 的功能。在 调用 OpenAI 的 API 时，可以描述函数并让模型智能地选择要输出的 JSON 对象，其中包含传递给一个或多个函数的参数。更多信息可以参考：https://platform.openai.com/docs/guides/function-calling 。

Chat Completions 的相关 API 并不会调用函数；相反，我们可以在自己的代码中根据模型的输出来实现调用函数的逻辑。大体工作流程如下：



其中，我们将对话数据和工具描述传递给 API 模型。在得到 API 模型的输出后，我们在本地根据输出调用函数，最终得到结果。

1.2 数据格式

了解 OpenAI Function Calling 所规定的数据格式，以便于使用 XTuner 进行微调时理解数据的结构。

1.2.1 对话部分

```
messages = [
    {
        "role": "user",
        "content": "What's the weather like in San Francisco, Tokyo, and Paris?"
    }
]
```

如上所示，这是一个简单的对话数据，包含 role 和 content 两个字段，分别表示输入角色和输入内容。

1.2.2 工具描述部分

```
tools = [
    {
        "type": "function",
        "function": {
            "name": "get_current_weather",
            "description": "Get the current weather in a given location",
            "parameters": {
                "type": "object",
                "properties": {
                    "location": {
                        "type": "string",
                        "description": "The city and state, e.g. San Francisco, CA",
                    },
                    "unit": {"type": "string", "enum": ["celsius", "fahrenheit"]},
                },
                "required": ["location"],
            },
        },
    }
]
```

如上所示是 OpenAI Function Calling 的工具描述部分。各字段描述如下：

字段	描述
type	为 function，表示这是一个函数
name	函数的名称
description	函数的描述
parameters	函数的输入参数，包括参数的类型、描述、是否必须等信息
parameters.type	输入参数的类型
parameters.properties	输入参数的属性
parameters.properties.location	函数的输入参数之一，表示给 get_current_weather 函数传递的位置信息，为字符串类型
parameters.properties.unit	函数的输入参数之一，表示给 get_current_weather 函数传递的单位信息，为字符串类型，且只能为摄氏度或华氏度
parameters.required	表示参数中必须包含的字段，即必须传递 location 参数