

Lagent & AgentLego 智能体应用搭建

1. 概述

1.1 环境配置

```
1 conda create -n agent
2 conda activate agent
3 conda install python=3.10
4 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

```
1 cd agent
2 conda activate agent
3 git clone https://gitee.com/internlm/lagent.git
4 cd lagent && git checkout 581d9fb && pip install -e . && cd ..
5 git clone https://gitee.com/internlm/agentlego.git
6 cd agentlego && git checkout 7769e0d && pip install -e . && cd ..
```

安装其他依赖

```
1 conda activate agent
2 pip install lmdeploy==0.3.0
```

准备 Tutorial

```
1 cd agent
2 git clone -b camp2 https://gitee.com/internlm/Tutorial.git
```

2. Lagent: 轻量级智能体框架

2.1 Lagent Web Demo

2.1.1 使用 LMDeploy 部署

由于 Lagent 的 Web Demo 需要用到 LMDeploy 所启动的 api_server，因此首先在 vscode terminal 中执行如下代码使用 LMDeploy 启动一个 api_server。

```
1 conda activate agent
2 lmdeploy serve api_server \
  /root/share/new_models/Shanghai_AI_Laboratory/internlm2-chat-7b \
3                                     --server-name 127.0.0.1 \
4                                     --model-name internlm2-chat-7b \
5                                     --cache-max-entry-count 0.1
```

```
o (agent) root@intern-studio-161126:~# 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

[WARNING] gemm_config.in is not found; using default GEMM algo
HINT: Please open http://127.0.0.1:23333 in a browser for detailed api usage!!!
HINT: Please open http://127.0.0.1:23333 in a browser for detailed api usage!!!
HINT: Please open http://127.0.0.1:23333 in a browser for detailed api usage!!!
INFO: Started server process [58245]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://127.0.0.1:23333 (Press CTRL+C to quit)
```

2.1.2 启动并使用 Lagent Web Demo

按照下图指示新建一个 terminal 以启动 Lagent Web Demo。在新建的 terminal 中执行如下指令：

```
1 conda activate agent
2 cd /root/agent/lagent/examples
3 streamlit run internlm2_agent_web_demo.py --server.address 127.0.0.1 --
  server.port 7860
```

```
o (base) root@intern-studio-161126:~# conda activate agent
s
streamlit run internlm2_agent_web_demo.py --server.address 127.0.0.1 --server.port 7860
o (agent) root@intern-studio-161126:~# cd /root/agent/lagent/examples
o (agent) root@intern-studio-161126:~/agent/lagent/examples# streamlit run internlm2_agent_web_demo.py --server.address 127.0.0.1 --server.port 7860

Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.

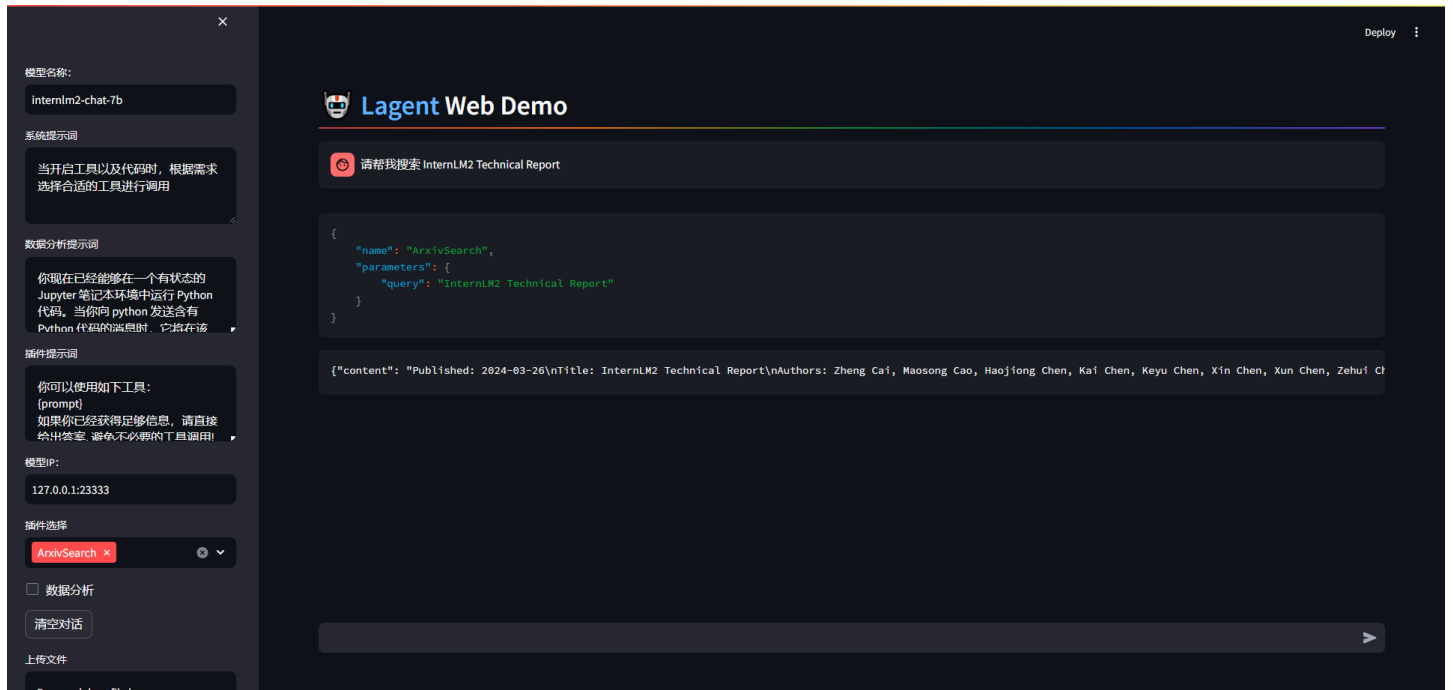
You can now view your Streamlit app in your browser.

URL: http://127.0.0.1:7860
```

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：

```
1 ssh -cN -L 7860:127.0.0.1:7860 -L 23333:127.0.0.1:23333 root@ssh.intern-
```

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。



2.2 用 Lagent 自定义工具

使用 Lagent 自定义工具主要分为以下几步：

1. 继承 BaseAction 类
 2. 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
 3. 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。

2.2.1 创建工具文件

首先通过 `touch /root/agent/lagent/lagent/actions/weather.py`（大小写敏感）新建工具文件，该文件内容如下：

```
1 import json
2 import os
3 import requests
4 from typing import Optional, Type
5
6 from lagent.actions.base_action import BaseAction, tool_api
7 from lagent.actions.parser import BaseParser, JsonParser
```

```

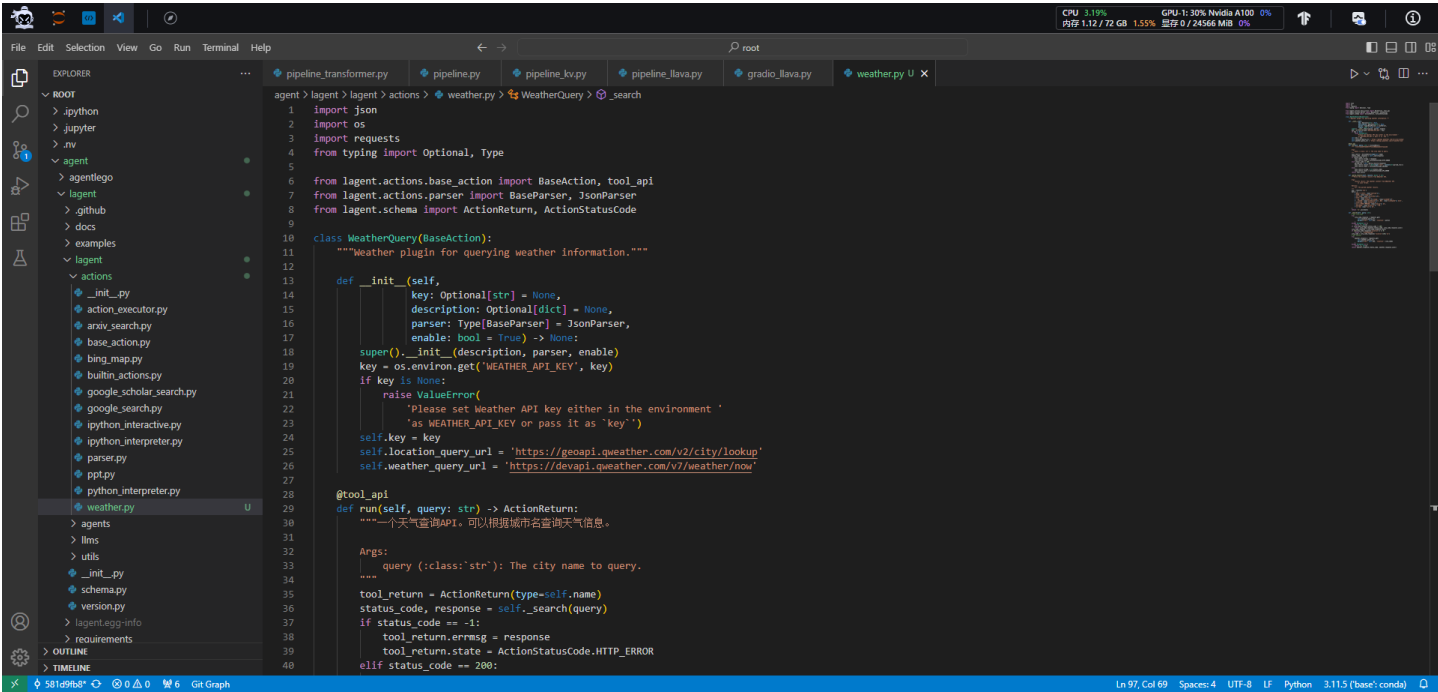
8 from lagent.schema import ActionReturn, ActionStatusCode
9
10 class WeatherQuery(BaseAction):
11     """Weather plugin for querying weather information."""
12
13     def __init__(self,
14                 key: Optional[str] = None,
15                 description: Optional[dict] = None,
16                 parser: Type[BaseParser] = JsonParser,
17                 enable: bool = True) -> None:
18         super().__init__(description, parser, enable)
19         key = os.environ.get('WEATHER_API_KEY', key)
20         if key is None:
21             raise ValueError(
22                 'Please set Weather API key either in the environment '
23                 'as WEATHER_API_KEY or pass it as `key`')
24         self.key = key
25         self.location_query_url = 'https://geoapi.qweather.com/v2/city/lookup'
26         self.weather_query_url = 'https://devapi.qweather.com/v7/weather/now'
27
28     @tool_api
29     def run(self, query: str) -> ActionReturn:
30         """一个天气查询API。可以根据城市名查询天气信息。
31
32         Args:
33             query (:class:`str`): The city name to query.
34         """
35         tool_return = ActionReturn(type=self.name)
36         status_code, response = self._search(query)
37         if status_code == -1:
38             tool_return.errmsg = response
39             tool_return.state = ActionStatusCode.HTTP_ERROR
40         elif status_code == 200:
41             parsed_res = self._parse_results(response)
42             tool_return.result = [dict(type='text', content=str(parsed_res))]
43             tool_return.state = ActionStatusCode.SUCCESS
44         else:
45             tool_return.errmsg = str(status_code)
46             tool_return.state = ActionStatusCode.API_ERROR
47         return tool_return
48
49     def _parse_results(self, results: dict) -> str:
50         """Parse the weather results from QWeather API.
51
52         Args:
53             results (dict): The weather content from QWeather API
54                             in json format.

```

```

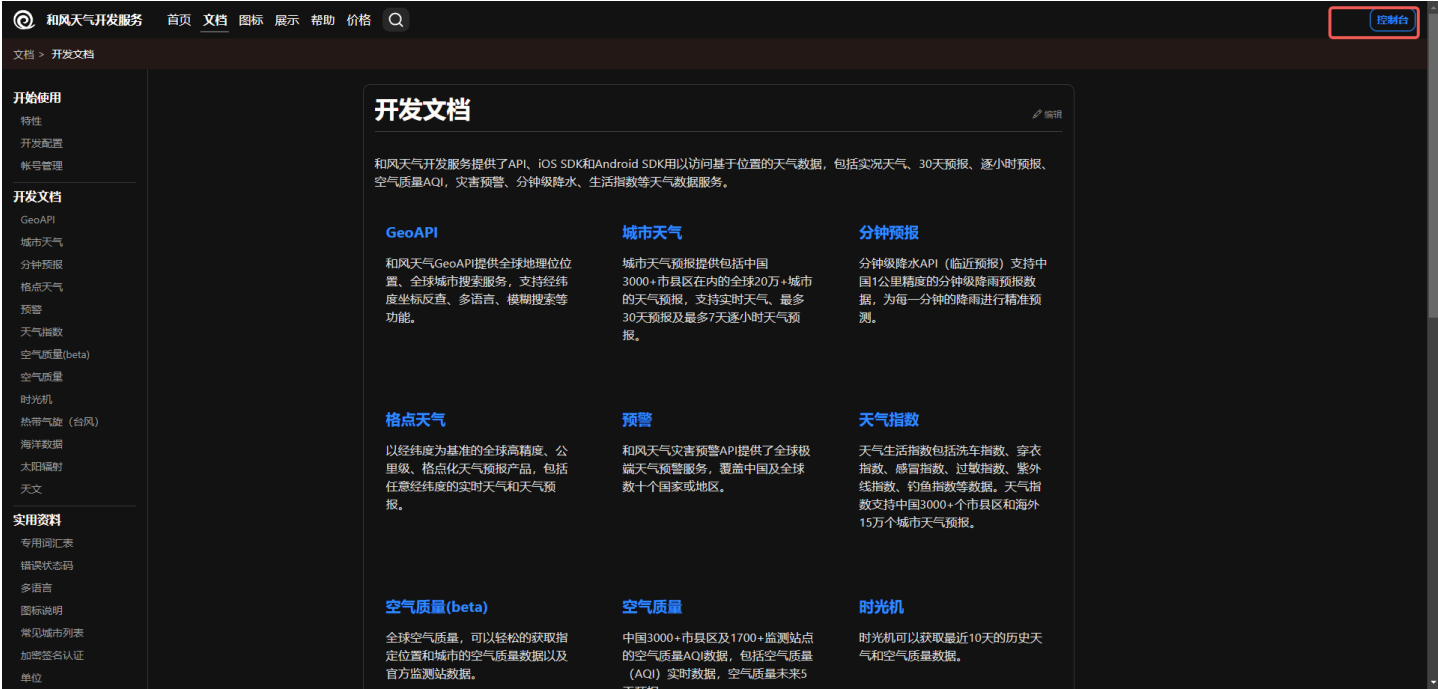
55
56     Returns:
57         str: The parsed weather results.
58     """
59     now = results['now']
60     data = [
61         f'数据观测时间: {now["obsTime"]}',
62         f'温度: {now["temp"]}°C',
63         f'体感温度: {now["feelsLike"]}°C',
64         f'天气: {now["text"]}',
65         f'风向: {now["windDir"]}, 角度为 {now["wind360"]}°',
66         f'风力等级: {now["windScale"]}, 风速为 {now["windSpeed"]} km/h',
67         f'相对湿度: {now["humidity"]}',
68         f'当前小时累计降水量: {now["precip"]} mm',
69         f'大气压强: {now["pressure"]} 百帕',
70         f'能见度: {now["vis"]} km',
71     ]
72     return '\n'.join(data)
73
74     def _search(self, query: str):
75         # get city_code
76         try:
77             city_code_response = requests.get(
78                 self.location_query_url,
79                 params={'key': self.key, 'location': query}
80             )
81         except Exception as e:
82             return -1, str(e)
83         if city_code_response.status_code != 200:
84             return city_code_response.status_code, city_code_response.json()
85         city_code_response = city_code_response.json()
86         if len(city_code_response['location']) == 0:
87             return -1, '未查询到城市'
88         city_code = city_code_response['location'][0]['id']
89         # get weather
90         try:
91             weather_response = requests.get(
92                 self.weather_query_url,
93                 params={'key': self.key, 'location': city_code}
94             )
95         except Exception as e:
96             return -1, str(e)
97         return weather_response.status_code, weather_response.json()

```



2.2.2 获取 API KEY

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。（如下图所示）



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。（如下图所示）


```

• (base) root@intern-studio-161126:~# conda activate agent
ot/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(agent) root@intern-studio-161126:~# lmdeploy serve api_server /root/share/new_models/Shanghai_AI_Lab
oratory/internlm2-chat-7b \
>    --server-name 127.0.0.1 \
>    --model-name internlm2-chat-7b \
>    --cache-max-entry-count 0.1

[WARNING] gemm_config.in is not found; using default GEMM algo
HINT: Please open http://127.0.0.1:23333 in a browser for detailed api usage!!!
HINT: Please open http://127.0.0.1:23333 in a browser for detailed api usage!!!
HINT: Please open http://127.0.0.1:23333 in a browser for detailed api usage!!!
INFO: Started server process [71450]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://127.0.0.1:23333 (Press CTRL+C to quit)

```

- 1 export WEATHER_API_KEY=在2.2节获取的API KEY
- 2 # 比如 export WEATHER_API_KEY=1234567890abcdef
- 3 conda activate agent
- 4 cd /root/agent/Tutorial/agent
- 5 streamlit run internlm2_weather_web_demo.py --server.address 127.0.0.1 --server.port 7860

```

• (base) root@intern-studio-161126:~# export WEATHER_API_KEY=45213fe24b9f4ee6937aa573ed8149c3
• (base) root@intern-studio-161126:~# conda activate agent

• streamlit run internlm2_weather_web_demo.py --server.address 127.0.0.1 --server.port 7860(agent) root@intern-studio-161126:~# cd /root/agent/Tutorial/agent
• (agent) root@intern-studio-161126:~/agent/Tutorial/agent# streamlit run internlm2_weather_web_demo.py --server.address 127.0.0.1 --server.port 7860

Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.

You can now view your Streamlit app in your browser.

URL: http://127.0.0.1:7860

```

并在本地执行如下操作以进行端口映射：

- 1 ssh -CNg -L 7860:127.0.0.1:7860 -L 23333:127.0.0.1:23333 root@ssh.intern-ai.org.cn -p 33820



3. AgentLego：组装智能体“乐高”

3.1 直接使用 AgentLego

首先下载 demo 文件：

```
1 cd /root/agent
2 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。

```
1 conda activate agent
2 pip install openmim==0.3.9
3 mim install mmdet==3.3.0
```

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：

```
1 import re
2
3 import cv2
```

```

4 from agentlego.apis import load_tool
5
6 # load tool
7 tool = load_tool('ObjectDetection', device='cuda')
8
9 # apply tool
10 visualization = tool('/root/agent/road.jpg')
11 print(visualization)
12
13 # visualize
14 image = cv2.imread('/root/agent/road.jpg')
15
16 preds = visualization.split('\n')
17 pattern = r'(\w+) \((\d+), (\d+), (\d+), (\d+)\), score (\d+)'
18
19 for pred in preds:
20     name, x1, y1, x2, y2, score = re.match(pattern, pred).groups()
21     x1, y1, x2, y2, score = int(x1), int(y1), int(x2), int(y2), int(score)
22     cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 1)
23     cv2.putText(image, f'{name} {score}', (x1, y1), cv2.FONT_HERSHEY_SIMPLEX,
24                  0.8, (0, 255, 0), 1)
25 cv2.imwrite('/root/agent/road_detection_direct.jpg', image)

```

接下来执行 `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：

```

return _VF.meshgrid(tensors, **kwargs) # type: ignore[attr-defined]
Inference
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, 501, 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

```



```
4 --model-name internlm2-chat-7b \  
5 --cache-max-entry-count 0.1
```

```
HINT: Please open http://127.0.0.1:23333 in a browser for detailed api usage!!!  
HINT: Please open http://127.0.0.1:23333 in a browser for detailed api usage!!!  
HINT: Please open http://127.0.0.1:23333 in a browser for detailed api usage!!!  
INFO: Started server process [21329]  
INFO: Waiting for application startup.  
INFO: Application startup complete.  
INFO: Uvicorn running on http://127.0.0.1:23333 (Press CTRL+C to quit)  
█
```

3.2.3 启动 AgentLego WebUI

接下来我们按照下图指示新建一个 terminal 以启动 AgentLego WebUI。在新建的 terminal 中执行如下指令：

```
1 conda activate agent  
2 cd /root/agent/agentlego/webui  
3 python one_click.py
```

```
>> from langchain.memory import ChatMessageHistory  
  
with new imports of:  
  
>> from langchain_community.chat_message_histories import ChatMessageHistory  
You can use the langchain cli to **automatically** upgrade many imports. Please see documentati  
on here https://python.langchain.com/v0.2/docs/versions/v0\_2/  
warn_deprecated(  
[14:27:17] INFO Loading tool `Calculator`  
Running on local URL: http://127.0.0.1:7860  
  
To create a public link, set `share=True` in `launch()`.
```

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。

```
1 ssh -CNg -L 7860:127.0.0.1:7860 -L 23333:127.0.0.1:23333 root@ssh.intern-  
ai.org.cn -p 33820
```

3.2.4 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。

ChatAgentTools

Agent

New Agent

Agent class

lagent.InternLM2Agent

Save to

Agent name

internlm2

Please save the new agent before load it.

URL

The internlm2 server url of LMDeploy, like 'http://127.0.0.1:23333'

http://127.0.0.1:23333

Max number of turns

6

Temperature

What sampling temperature to use.

0.7

System prompt

当开启工具以及代码时，根据需求选择合适的工具进行调用

然后配置工具，如下图所示。

ChatAgentTools

Tools

New Tool

Tool class

ObjectDetection

Save

Import from tool server

Confirm

Name

ObjectDetection

Description

The tool can detect all common objects in the picture.

☒ Enable

Device

cuda:0

Initialize arguments

Loading tool ObjectDetection...

ChatAgentTools

Tools

ObjectDetection

Tool class

ObjectDetection

Save

Import from tool server

Confirm

Name

ObjectDetection

Description

The tool can detect all common objects in the picture.

☒ Enable

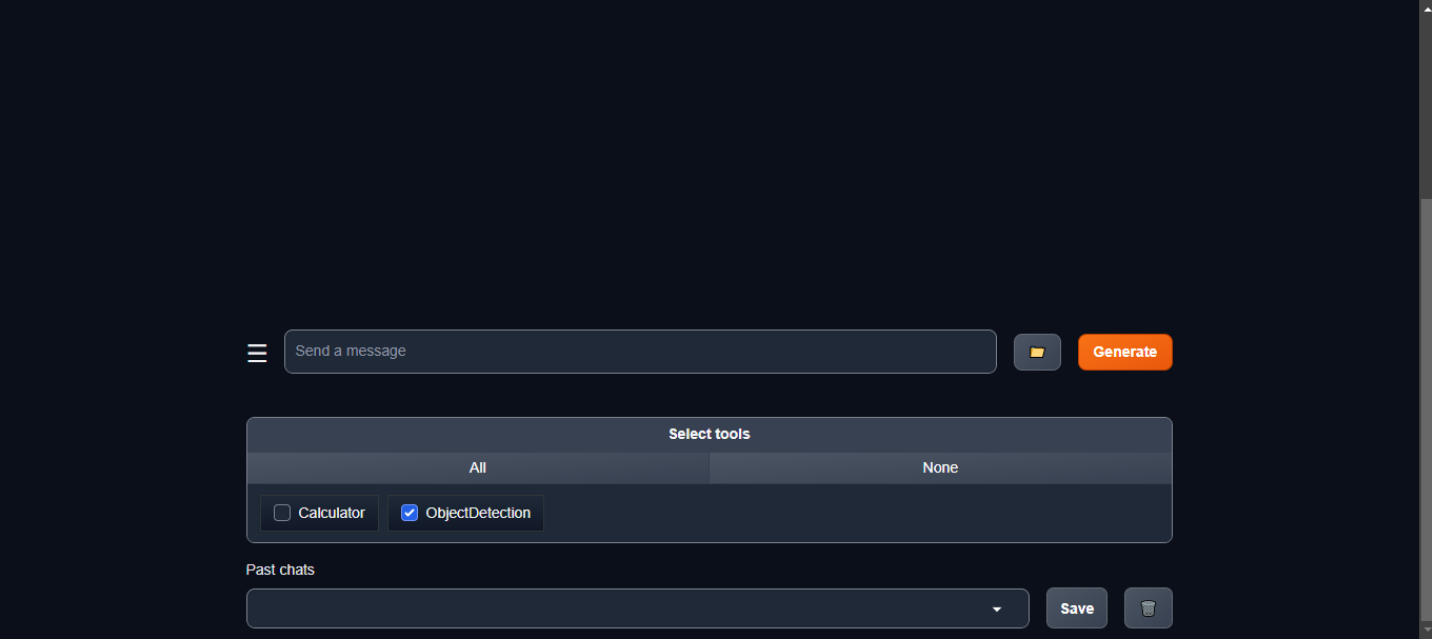
Device

cuda:0

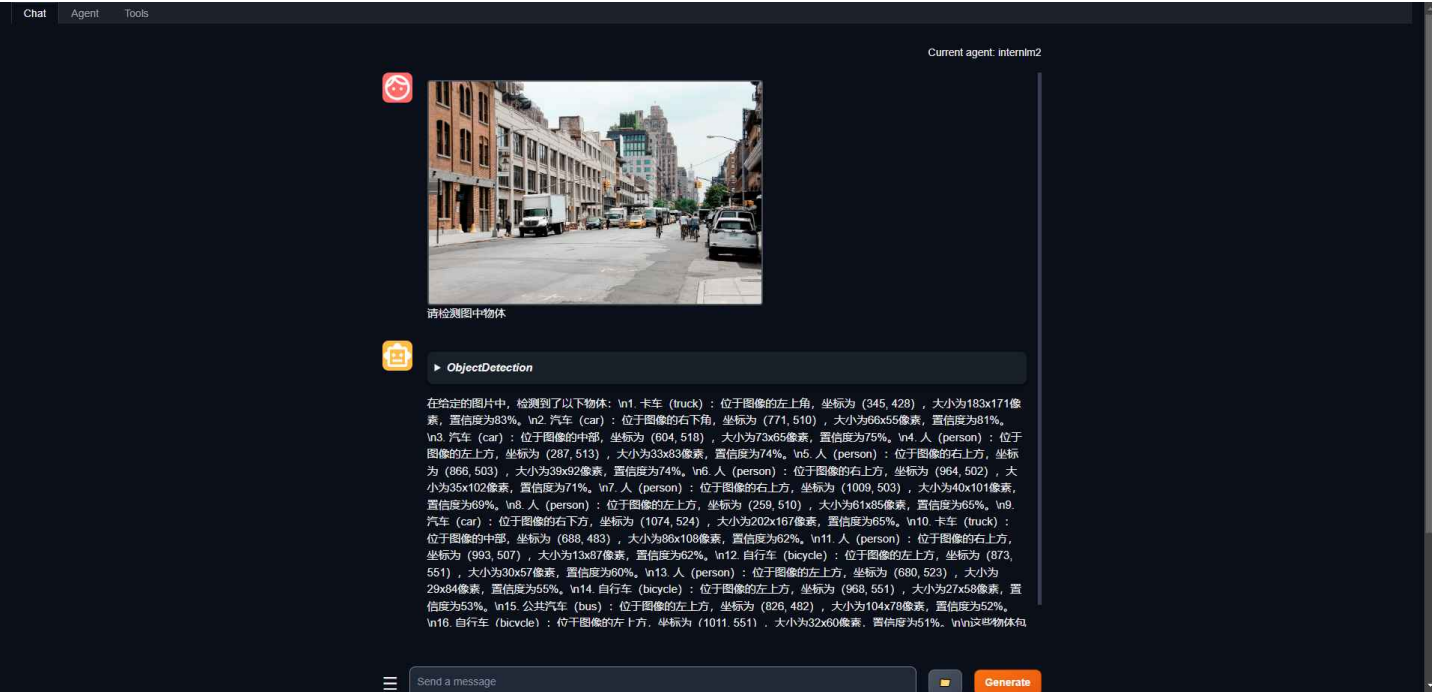
Initialize arguments

Loaded ObjectDetection.

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。



接下来就可以愉快地使用 Agent 了。点击右下角文件夹以上传图片，上传图片后输入指令并点击 generate 以得到模型回复。如下图所示，我们上传了 demo 图片，模型成功地调用了工具，并详细地告诉了我们图中的内容。



3.3 用 AgentLego 自定义工具

基于 AgentLego 构建自己的自定义工具。自定义工具主要分为以下几步：

1. 继承 BaseTool 类
2. 修改 default_desc 属性（工具功能描述）

3. 如有需要，重载 setup 方法（重型模块延迟加载）

4. 重载 apply 方法（工具功能实现）

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。

3.3.1 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。该文件的内容如下：

```
1 import json
2 import requests
3
4 import numpy as np
5
6 from agentlego.types import Annotated, ImageIO, Info
7 from agentlego.utils import require
8 from .base import BaseTool
9
10
11 class MagicMakerImageGeneration(BaseTool):
12
13     default_desc = ('This tool can call the api of magicmaker to '
14                     'generate an image according to the given keywords.')
15
16     styles_option = [
17         'dongman', # 动漫
18         'guofeng', # 国风
19         'xieshi', # 写实
20         'yohua', # 油画
21         'manghe', # 盲盒
22     ]
23     aspect_ratio_options = [
24         '16:9', '4:3', '3:2', '1:1',
25         '2:3', '3:4', '9:16'
26     ]
27
28     @require('opencv-python')
29     def __init__(self,
30                 style='guofeng',
31                 aspect_ratio='4:3'):
32         super().__init__()
33         if style in self.styles_option:
```



```

34         self.style = style
35     else:
36         raise ValueError(f'The style must be one of {self.styles_option}')
37
38     if aspect_ratio in self.aspect_ratio_options:
39         self.aspect_ratio = aspect_ratio
40     else:
41         raise ValueError(f'The aspect ratio must be one of {aspect_ratio}')
42
43     def apply(self,
44               keywords: Annotated[str,
45                                   Info('A series of Chinese keywords separated
46                                       by comma.')]
47               ) -> ImageIO:
48         import cv2
49         response = requests.post(
50             url='https://magicmaker.openxlab.org.cn/gw/edit-anything/api/v1/bff/sd/generate',
51             data=json.dumps({
52                 "official": True,
53                 "prompt": keywords,
54                 "style": self.style,
55                 "poseT": False,
56                 "aspectRatio": self.aspect_ratio
57             }),
58             headers={'content-type': 'application/json'})
59         image_url = response.json()['data']['imgUrl']
60         image_response = requests.get(image_url)
61         image =
62         cv2.cvtColor(cv2.imdecode(np.frombuffer(image_response.content, np.uint8),
63             cv2.IMREAD_COLOR),cv2.COLOR_BGR2RGB)
64         return ImageIO(image)

```

3.3.2 注册新工具

接下来修改 /root/agent/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。

```

1 from .base import BaseTool
2 from .calculator import Calculator
3 from .func import make_tool
4 from .image_canny import CannyTextToImage, ImageToCanny

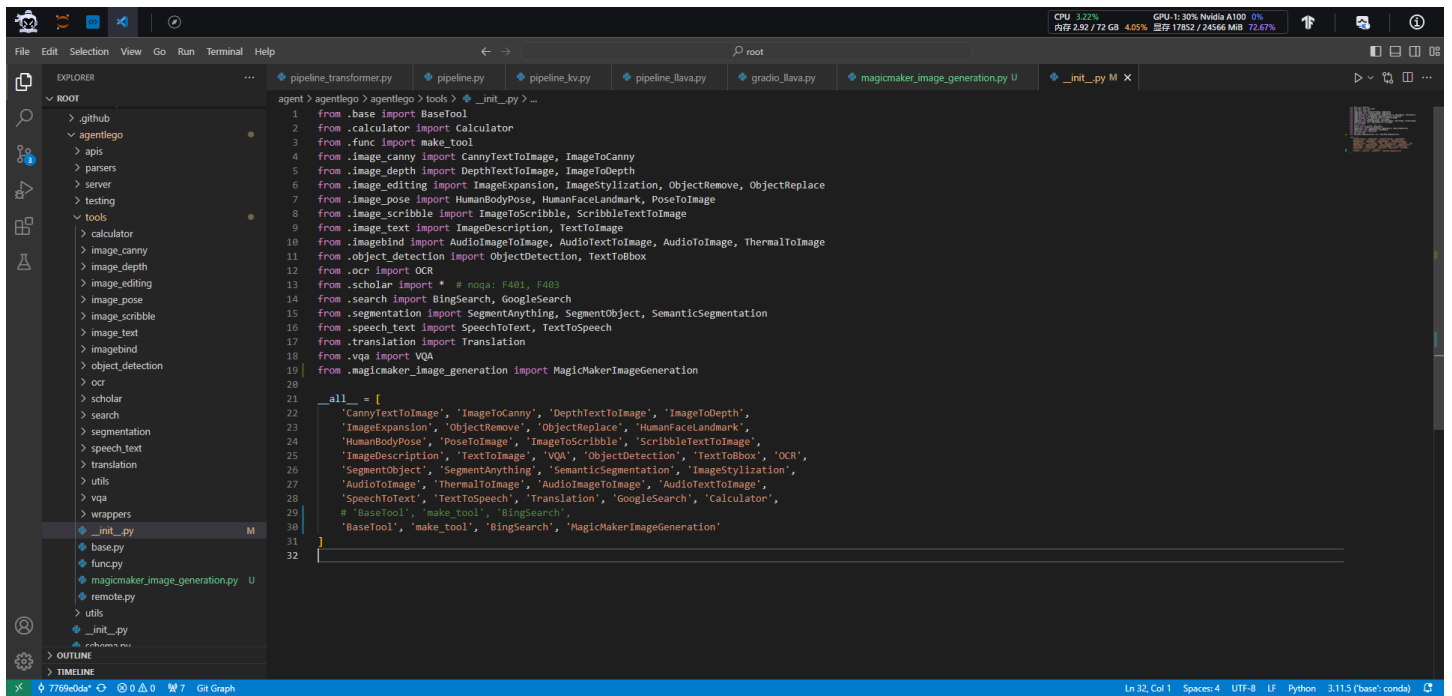
```



```

5 from .image_depth import DepthTextToImage, ImageToDepth
6 from .image_editing import ImageExpansion, ImageStylization, ObjectRemove,
  ObjectReplace
7 from .image_pose import HumanBodyPose, HumanFaceLandmark, PoseToImage
8 from .image_scribble import ImageToScribble, ScribbleTextToImage
9 from .image_text import ImageDescription, TextToImage
10 from .imagebind import AudioImageToImage, AudioTextToImage, AudioToImage,
  ThermalToImage
11 from .object_detection import ObjectDetection, TextToBbox
12 from .ocr import OCR
13 from .scholar import * # noqa: F401, F403
14 from .search import BingSearch, GoogleSearch
15 from .segmentation import SegmentAnything, SegmentObject, SemanticSegmentation
16 from .speech_text import SpeechToText, TextToSpeech
17 from .translation import Translation
18 from .vqa import VQA
19 + from .magicmaker_image_generation import MagicMakerImageGeneration
20
21 __all__ = [
22     'CannyTextToImage', 'ImageToCanny', 'DepthTextToImage', 'ImageToDepth',
23     'ImageExpansion', 'ObjectRemove', 'ObjectReplace', 'HumanFaceLandmark',
24     'HumanBodyPose', 'PoseToImage', 'ImageToScribble', 'ScribbleTextToImage',
25     'ImageDescription', 'TextToImage', 'VQA', 'ObjectDetection', 'TextToBbox',
26     'OCR',
27     'SegmentObject', 'SegmentAnything', 'SemanticSegmentation',
28     'ImageStylization',
29     'AudioToImage', 'ThermalToImage', 'AudioImageToImage', 'AudioTextToImage',
30     'SpeechToText', 'TextToSpeech', 'Translation', 'GoogleSearch',
31     'Calculator',
32     'BaseTool', 'make_tool', 'BingSearch'
33 +     'BaseTool', 'make_tool', 'BingSearch', 'MagicMakerImageGeneration'
34 ]

```



3.3.3 体验自定义工具效果

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。

```

1 conda activate agent
2 lmdeploy serve api_server
   /root/share/new_models/Shanghai_AI_Laboratory/internlm2-chat-7b \
3                                     --server-name 127.0.0.1 \
4                                     --model-name internlm2-chat-7b \
5                                     --cache-max-entry-count 0.1

```

```

(agent) root@intern-studio-161126: # 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
[WARNING] gemm_config.in is not found; using default GEMM algo
HINT: Please open http://127.0.0.1:23333 in a browser for detailed api usage!!!
HINT: Please open http://127.0.0.1:23333 in a browser for detailed api usage!!!
HINT: Please open http://127.0.0.1:23333 in a browser for detailed api usage!!!
INFO: Started server process [55556]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://127.0.0.1:23333 (Press CTRL+C to quit)

```

```

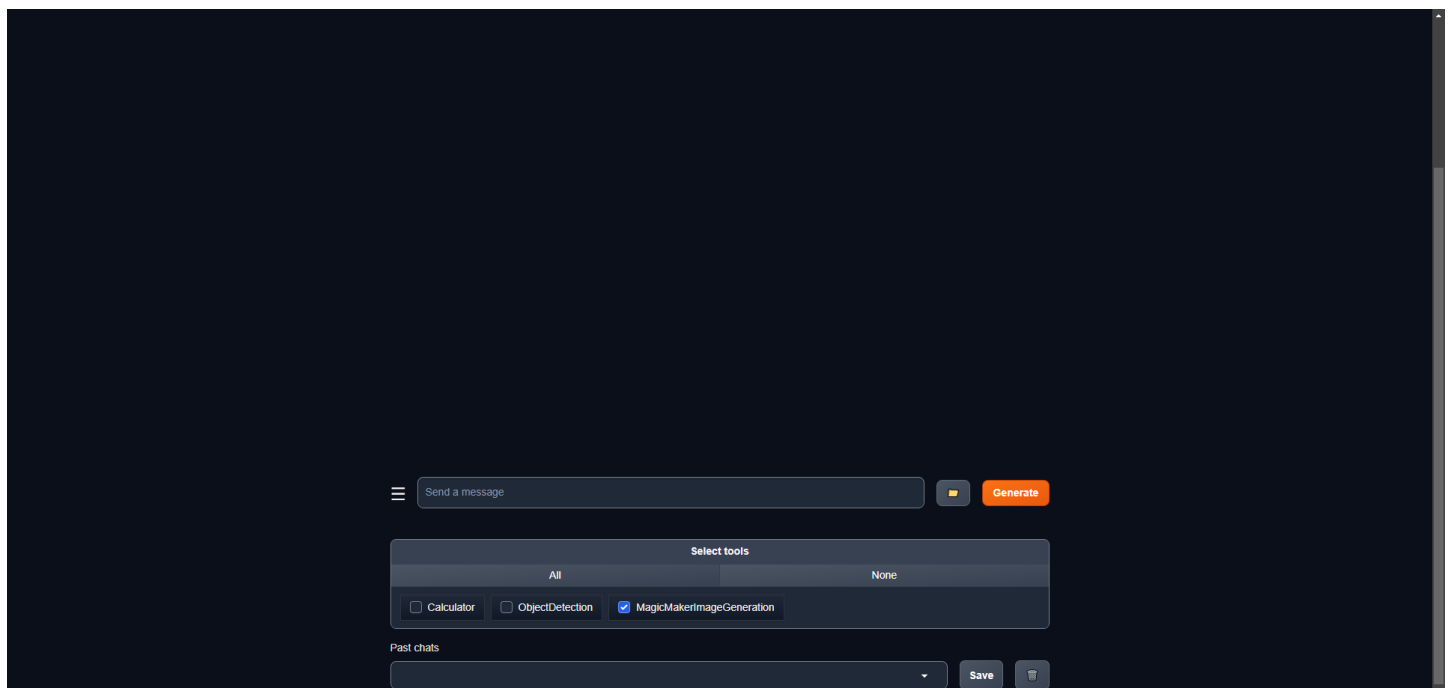
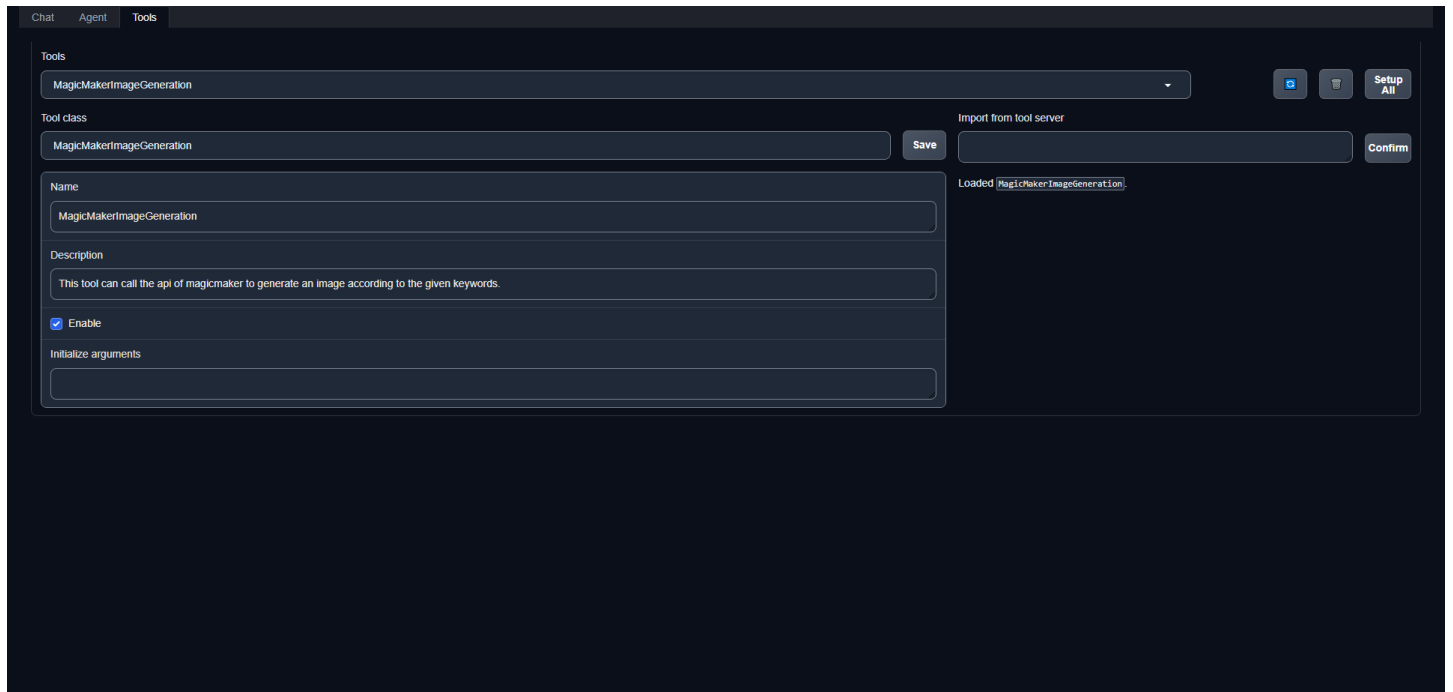
1 conda activate agent
2 cd /root/agent/agentlego/webui
3 python one_click.py

```

并在本地执行如下操作以进行端口映射：


```
1 ssh -CNg -L 7860:127.0.0.1:7860 -L 23333:127.0.0.1:23333 root@ssh.intern-  
ai.org.cn -p 33820
```

在 Tool 界面选择 MagicMakerImageGeneration 后点击 save 后，回到 Chat 页面选择 MagicMakerImageGeneration 工具后就可以开始使用了。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。下图是一个例子。可以看到模型成功地调用了工具并得到了结果。




ChatAgentTools


Current agent: internlm2




帮我生成一幅山水画




► MagicMakerImageGeneration



好的，我已经帮你生成了一幅山水画。请查看以下链接：




Send a message


Generate

ChatAgentTools


Current agent: internlm2




再来一幅




► MagicMakerImageGeneration



好的，我已经帮你生成了一幅新的山水画。请查看以下链接：



Send a message

Generate