使用 Lagent 自定义一个查询电影信息的工具

获取API：https://www.omdbapi.com/

创建工具文件：

```
touch /root/agent/lagent/lagent/actions/movie.py
```

复制粘贴：

```
import json
import os
import requests
from typing import Optional, Type

from lagent.actions.base_action import BaseAction, tool_api
from lagent.actions.parser import BaseParser, JsonParser
from lagent.schema import ActionReturn, ActionStatusCode

class MovieQuery(BaseAction):
    def __init__(self,
                 key: Optional[str] = None,
                 description: Optional[dict] = None,
                 parser: Type[BaseParser] = JsonParser,
                 enable: bool = True) -> None:
        super().__init__(description, parser, enable)
        key = os.environ.get('OMDb_API_KEY', key)
        if key is None:
            raise ValueError(
                'Please set OMDb API key either in the environment '
                'as OMDb_API_KEY or pass it as `key`')
        self.key = key
        self.query_url = f'http://www.omdbapi.com/?apikey={key}&t='

    @tool_api
    def run(self, query: str) -> ActionReturn:
        """一个电影查询API。可以根据电影名查询电影信息。

        Args:
            query (:class:`str`): The movie name to query.
        """
        tool_return = ActionReturn(type=self.name)
        status_code, response = self._search(query)
        if status_code == -1:
            tool_return.errmsg = response
            tool_return.state = ActionStatusCode.HTTP_ERROR
        elif status_code == 200:
            parsed_res = self._parse_results(response)
            tool_return.result = [dict(type='text', content=str(parsed_res))]
            tool_return.state = ActionStatusCode.SUCCESS
        else:
            tool_return.errmsg = str(status_code)
            tool_return.state = ActionStatusCode.API_ERROR
        return tool_return
```

```
    def _parse_results(self, movie_dict: dict) -> str:
        """Parse the movie results from OMDb API.

        Args:
            movie_dict (dict): The movie content from OMDb API
                in json format.

        Returns:
            str: The parsed movie results.
        """
        data = [
            f'名称: {movie_dict["Title"]}',
            f'年份: {movie_dict["Year"]}',
            f'评级: {movie_dict["Rated"]}',
            f'上映日期: {movie_dict["Released"]}',
            f'时长: {movie_dict["Runtime"]}',
            f'类型: {movie_dict["Genre"]}',
            f'导演: {movie_dict["Director"]}',
            f'编剧: {movie_dict["Writer"]}',
            f'演员: {movie_dict["Actors"]}',
            f'剧情简介: {movie_dict["Plot"]}',
            f'语言: {movie_dict["Language"]}',
            f'国家/地区: {movie_dict["Country"]}',
            f'奖项: {movie_dict["Awards"]}',
            f'海报: {movie_dict["Poster"]}',
            f'评分: {movie_dict["Metascore"]}',
            f'imdb评分: {movie_dict["imdbRating"]}',
            f'imdb投票数: {movie_dict["imdbVotes"]}',
            f'imdb ID: {movie_dict["imdbID"]}',
            f'类型: {movie_dict["Type"]}',
            f'DVD发布日期: {movie_dict["DVD"]}',
            f'票房: {movie_dict["BoxOffice"]}',
            f'制作公司: {movie_dict["Production"]}',
            f'网站: {movie_dict["Website"]}',]
        return '\n'.join(data)

    def _search(self, query: str):
        try:
            response = requests.get(
                self.query_url,
                params={'t': query}
            )
        except Exception as e:
            return -1, str(e)
        return response.status_code, response.json()
```

创建web demo文件：

```
touch /root/agent/Tutorial/agent/internlm2_movie_web_demo.py
```

复制粘贴：

```
import copy
```

```python
import hashlib
import json
import os

import streamlit as st

from lagent.actions import ActionExecutor, IPythonInterpreter
from lagent.actions.movie import MovieQuery
from lagent.agents.internlm2_agent import INTERPRETER_CN, META_CN, PLUGIN_CN,
Internlm2Agent, Internlm2Protocol
from lagent.llms.lmdepoly_wrapper import LMDeployClient
from lagent.llms.meta_template import INTERNLM2_META as META
from lagent.schema import AgentStatusCode

# from streamlit.logger import get_logger


class SessionState:

    def init_state(self):
        """Initialize session state variables."""
        st.session_state['assistant'] = []
        st.session_state['user'] = []

        action_list = [
            MovieQuery(),
        ]
        st.session_state['plugin_map'] = {
            action.name: action
            for action in action_list
        }
        st.session_state['model_map'] = {}
        st.session_state['model_selected'] = None
        st.session_state['plugin_actions'] = set()
        st.session_state['history'] = []

    def clear_state(self):
        """Clear the existing session state."""
        st.session_state['assistant'] = []
        st.session_state['user'] = []
        st.session_state['model_selected'] = None
        st.session_state['file'] = set()
        if 'chatbot' in st.session_state:
            st.session_state['chatbot']._session_history = []


class StreamlitUI:

    def __init__(self, session_state: SessionState):
        self.init_streamlit()
        self.session_state = session_state

    def init_streamlit(self):
        """Initialize Streamlit's UI settings."""
        st.set_page_config(
            layout='wide',
```

```
            page_title='lagent-web',
            page_icon='./docs/imgs/lagent_icon.png')
        st.header(':robot_face: :blue[Lagent] Web Demo ', divider='rainbow')
        st.sidebar.title('模型控制')
        st.session_state['file'] = set()
        st.session_state['ip'] = None

    def setup_sidebar(self):
        """Setup the sidebar for model and plugin selection."""
        # model_name = st.sidebar.selectbox('模型选择: ', options=['internlm'])
        model_name = st.sidebar.text_input('模型名称: ', value='internlm2-chat-7b')
        meta_prompt = st.sidebar.text_area('系统提示词', value=META_CN)
        da_prompt = st.sidebar.text_area('数据分析提示词', value=INTERPRETER_CN)
        plugin_prompt = st.sidebar.text_area('插件提示词', value=PLUGIN_CN)
        model_ip = st.sidebar.text_input('模型IP: ', value='10.140.0.220:23333')
        if model_name != st.session_state[
                'model_selected'] or st.session_state['ip'] != model_ip:
            st.session_state['ip'] = model_ip
            model = self.init_model(model_name, model_ip)
            self.session_state.clear_state()
            st.session_state['model_selected'] = model_name
            if 'chatbot' in st.session_state:
                del st.session_state['chatbot']
        else:
            model = st.session_state['model_map'][model_name]

        plugin_name = st.sidebar.multiselect(
            '插件选择',
            options=list(st.session_state['plugin_map'].keys()),
            default=[],
        )
        da_flag = st.sidebar.checkbox(
            '数据分析',
            value=False,
        )
        plugin_action = [
            st.session_state['plugin_map'][name] for name in plugin_name
        ]

        if 'chatbot' in st.session_state:
            if len(plugin_action) > 0:
                st.session_state['chatbot']._action_executor = ActionExecutor(
                    actions=plugin_action)
            else:
                st.session_state['chatbot']._action_executor = None
            if da_flag:
                st.session_state[
                    'chatbot']._interpreter_executor = ActionExecutor(
                        actions=[IPythonInterpreter()])
            else:
                st.session_state['chatbot']._interpreter_executor = None
            st.session_state['chatbot']._protocol._meta_template = meta_prompt
            st.session_state['chatbot']._protocol.plugin_prompt = plugin_prompt
            st.session_state[
                'chatbot']._protocol.interpreter_prompt = da_prompt
        if st.sidebar.button('清空对话', key='clear'):
```

```
            self.session_state.clear_state()
        uploaded_file = st.sidebar.file_uploader('上传文件')

        return model_name, model, plugin_action, uploaded_file, model_ip

    def init_model(self, model_name, ip=None):
        """Initialize the model based on the input model name."""
        model_url = f'http://{ip}'
        st.session_state['model_map'][model_name] = LMDeployClient(
            model_name=model_name,
            url=model_url,
            meta_template=META,
            max_new_tokens=1024,
            top_p=0.8,
            top_k=100,
            temperature=0,
            repetition_penalty=1.0,
            stop_words=['<|im_end|>'])
        return st.session_state['model_map'][model_name]

    def initialize_chatbot(self, model, plugin_action):
        """Initialize the chatbot with the given model and plugin actions."""
        return Internlm2Agent(
            llm=model,
            protocol=Internlm2Protocol(
                tool=dict(
                    begin='{start_token}{name}\n',
                    start_token='<|action_start|>',
                    name_map=dict(
                        plugin='<|plugin|>', interpreter='<|interpreter|>'),
                    belong='assistant',
                    end='<|action_end|>\n',
                ), ),
            max_turn=7)

    def render_user(self, prompt: str):
        with st.chat_message('user'):
            st.markdown(prompt)

    def render_assistant(self, agent_return):
        with st.chat_message('assistant'):
            for action in agent_return.actions:
                if (action) and (action.type != 'FinishAction'):
                    self.render_action(action)
            st.markdown(agent_return.response)

    def render_plugin_args(self, action):
        action_name = action.type
        args = action.args
        import json
        parameter_dict = dict(name=action_name, parameters=args)
        parameter_str = '```json\n' + json.dumps(
            parameter_dict, indent=4, ensure_ascii=False) + '\n```'
        st.markdown(parameter_str)

    def render_interpreter_args(self, action):
```

```
        st.info(action.type)
        st.markdown(action.args['text'])

    def render_action(self, action):
        st.markdown(action.thought)
        if action.type == 'IPythonInterpreter':
            self.render_interpreter_args(action)
        elif action.type == 'FinishAction':
            pass
        else:
            self.render_plugin_args(action)
        self.render_action_results(action)

    def render_action_results(self, action):
        """Render the results of action, including text, images, videos, and
        audios."""
        if (isinstance(action.result, dict)):
            if 'text' in action.result:
                st.markdown('```\n' + action.result['text'] + '\n```')
            if 'image' in action.result:
                # image_path = action.result['image']
                for image_path in action.result['image']:
                    image_data = open(image_path, 'rb').read()
                    st.image(image_data, caption='Generated Image')
            if 'video' in action.result:
                video_data = action.result['video']
                video_data = open(video_data, 'rb').read()
                st.video(video_data)
            if 'audio' in action.result:
                audio_data = action.result['audio']
                audio_data = open(audio_data, 'rb').read()
                st.audio(audio_data)
        elif isinstance(action.result, list):
            for item in action.result:
                if item['type'] == 'text':
                    st.markdown('```\n' + item['content'] + '\n```')
                elif item['type'] == 'image':
                    image_data = open(item['content'], 'rb').read()
                    st.image(image_data, caption='Generated Image')
                elif item['type'] == 'video':
                    video_data = open(item['content'], 'rb').read()
                    st.video(video_data)
                elif item['type'] == 'audio':
                    audio_data = open(item['content'], 'rb').read()
                    st.audio(audio_data)
        if action.errmsg:
            st.error(action.errmsg)


def main():
    # logger = get_logger(__name__)
    # Initialize Streamlit UI and setup sidebar
    if 'ui' not in st.session_state:
        session_state = SessionState()
        session_state.init_state()
        st.session_state['ui'] = StreamlitUI(session_state)
```

```
else:
    st.set_page_config(
        layout='wide',
        page_title='lagent-web',
        page_icon='./docs/imgs/lagent_icon.png')
    st.header(':robot_face: :blue[Lagent] Web Demo ', divider='rainbow')
_, model, plugin_action, uploaded_file, _ = st.session_state[
    'ui'].setup_sidebar()

# Initialize chatbot if it is not already initialized
# or if the model has changed
if 'chatbot' not in st.session_state or model != st.session_state[
        'chatbot']._llm:
    st.session_state['chatbot'] = st.session_state[
        'ui'].initialize_chatbot(model, plugin_action)
    st.session_state['session_history'] = []

for prompt, agent_return in zip(st.session_state['user'],
                                st.session_state['assistant']):
    st.session_state['ui'].render_user(prompt)
    st.session_state['ui'].render_assistant(agent_return)

if user_input := st.chat_input(''):
    with st.container():
        st.session_state['ui'].render_user(user_input)
    st.session_state['user'].append(user_input)
    # Add file uploader to sidebar
    if (uploaded_file
            and uploaded_file.name not in st.session_state['file']):

        st.session_state['file'].add(uploaded_file.name)
        file_bytes = uploaded_file.read()
        file_type = uploaded_file.type
        if 'image' in file_type:
            st.image(file_bytes, caption='Uploaded Image')
        elif 'video' in file_type:
            st.video(file_bytes, caption='Uploaded Video')
        elif 'audio' in file_type:
            st.audio(file_bytes, caption='Uploaded Audio')
        # Save the file to a temporary location and get the path

        postfix = uploaded_file.name.split('.')[-1]
        # prefix = str(uuid.uuid4())
        prefix = hashlib.md5(file_bytes).hexdigest()
        filename = f'{prefix}.{postfix}'
        file_path = os.path.join(root_dir, filename)
        with open(file_path, 'wb') as tmpfile:
            tmpfile.write(file_bytes)
        file_size = os.stat(file_path).st_size / 1024 / 1024
        file_size = f'{round(file_size, 2)} MB'
        # st.write(f'File saved at: {file_path}')
        user_input = [
            dict(role='user', content=user_input),
            dict(
                role='user',
```

```
                    content=json.dumps(dict(path=file_path, size=file_size)),
                    name='file')
            ]
        if isinstance(user_input, str):
            user_input = [dict(role='user', content=user_input)]
        st.session_state['last_status'] = AgentStatusCode.SESSION_READY
        for agent_return in st.session_state['chatbot'].stream_chat(
                st.session_state['session_history'] + user_input):
            if agent_return.state == AgentStatusCode.PLUGIN_RETURN:
                with st.container():
                    st.session_state['ui'].render_plugin_args(
                        agent_return.actions[-1])
                    st.session_state['ui'].render_action_results(
                        agent_return.actions[-1])
            elif agent_return.state == AgentStatusCode.CODE_RETURN:
                with st.container():
                    st.session_state['ui'].render_action_results(
                        agent_return.actions[-1])
            elif (agent_return.state == AgentStatusCode.STREAM_ING
                  or agent_return.state == AgentStatusCode.CODING):
                # st.markdown(agent_return.response)
                # 清除占位符的当前内容，并显示新内容
                with st.container():
                    if agent_return.state != st.session_state['last_status']:
                        st.session_state['temp'] = ''
                        placeholder = st.empty()
                        st.session_state['placeholder'] = placeholder
                    if isinstance(agent_return.response, dict):
                        action = f"\n\n {agent_return.response['name']}: \n\n"
                        action_input = agent_return.response['parameters']
                        if agent_return.response[
                                'name'] == 'IPythonInterpreter':
                            action_input = action_input['command']
                        response = action + action_input
                    else:
                        response = agent_return.response
                    st.session_state['temp'] = response
                    st.session_state['placeholder'].markdown(
                        st.session_state['temp'])
            elif agent_return.state == AgentStatusCode.END:
                st.session_state['session_history'] += (
                    user_input + agent_return.inner_steps)
                agent_return = copy.deepcopy(agent_return)
                agent_return.response = st.session_state['temp']
                st.session_state['assistant'].append(
                    copy.deepcopy(agent_return))
            st.session_state['last_status'] = agent_return.state


if __name__ == '__main__':
    root_dir = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
    root_dir = os.path.join(root_dir, 'tmp_dir')
    os.makedirs(root_dir, exist_ok=True)
    main()
```

终端1：LMDeploy 启动 api_server：

```
conda activate agent
bash /root/agent/lmdeploy_api.sh
#/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
```

```
(agent) root@intern-studio-40073080:~# bash /root/agent/lmdeploy_api.sh
[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 [29182]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://127.0.0.1:23333 (Press CTRL+C to quit)
```

终端2：

```
export OMDb_API_KEY=xxxx
conda activate agent
cd /root/agent/Tutorial/agent
streamlit run internlm2_movie_web_demo.py --server.address 127.0.0.1 --
server.port 7860
```

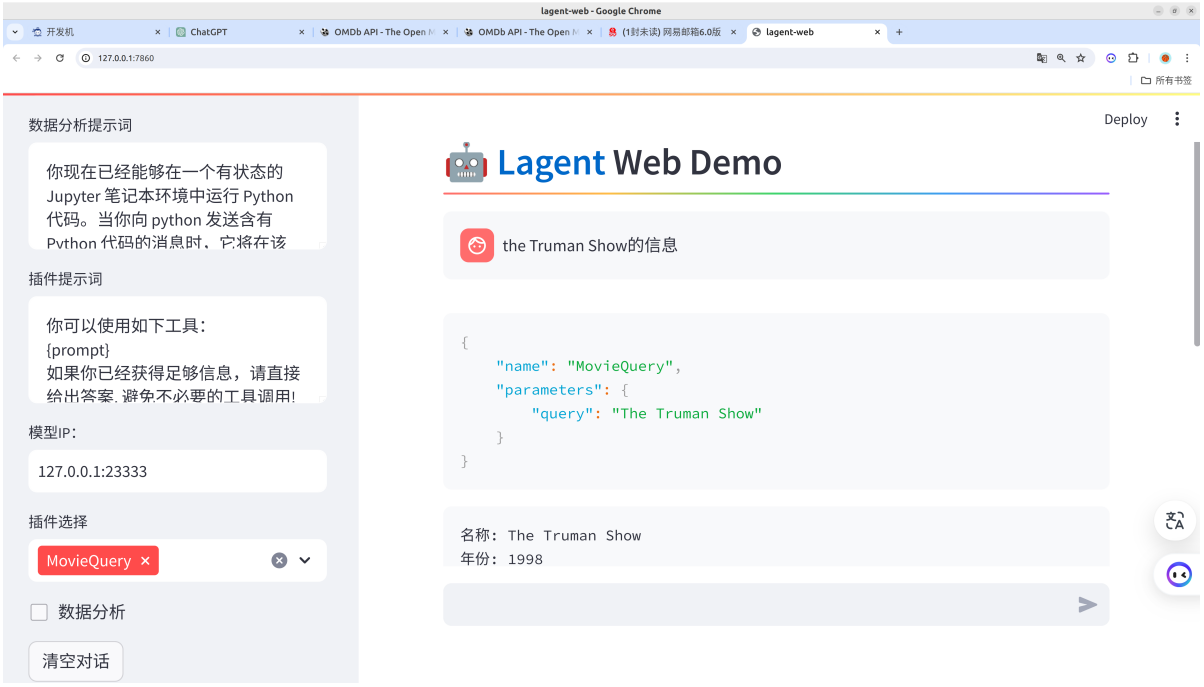
```
(base) root@intern-studio-40073080:~# conda activate agent
(agent) root@intern-studio-40073080:~# cd /root/agent/Tutorial/agent
(agent) root@intern-studio-40073080:~/agent/Tutorial/agent# streamlit run internlm2_movie_web_demo.py --server.a
ddress 127.0.0.1 --server.port 7860

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
```

对话：打开网页，模型 IP输入127.0.0.1:23333，**在输入完成后按下回车键以确认**，选择插件为 MovieQuery：



点击点击此处查看海报或者将海报网址复制打开