Openmanus使用指南V1.1版本

环境搭建(官方)

Installation

We provide two installation methods. Method 2 (using uv) is recommended for faster installation and better dependency management.

Method 1: Using conda

1. Create a new conda environment:

```
conda create -n open_manus python=3.12
conda activate open_manus
```

2. Clone the repository:

```
git clone https://github.com/mannaandpoem/OpenManus.git
cd OpenManus
```

3. Install dependencies:

```
pip install -r requirements.txt
```

Method 2: Using uv (Recommended)

1. Install uv (A fast Python package installer and resolver):

```
curl -LsSf https://astral.sh/uv/install.sh | sh
```

2. Clone the repository:

```
git clone https://github.com/mannaandpoem/OpenManus.git
cd OpenManus
```

3. Create a new virtual environment and activate it:

```
uv venv
source .venv/bin/activate  # On Unix/macOS
# Or on Windows:
# .venv\Scripts\activate
```

4. Install dependencies:

```
uv pip install -r requirements.txt
```

Configuration

OpenManus requires configuration for the LLM APIs it uses. Follow these steps to set up your configuration:

1. Create a `config.toml` file in the `config` directory (you can copy from the example):

```
cp config/config.example.toml config/config.toml
```

2. Edit `config/config.toml` to add your API keys and customize settings:

```
# Global LLM configuration
[llm]
model = "gpt-4o"
base_url = "https://api.openai.com/v1"
api_key = "sk-..."  # Replace with your actual API key
max_tokens = 4096
temperature = 0.0

# Optional configuration for specific LLM models
[llm.vision]
model = "gpt-4o"
base_url = "https://api.openai.com/v1"
api_key = "sk-..."  # Replace with your actual API key
```

Quick Start

One line for run OpenManus:

```
python main.py
```

Then input your idea via terminal!

For unstable version, you also can run:

```
python run_flow.py
```

add BingSearch or BaiduSearch

1. 必应搜索

1. agent/manus.py tools添加 BingSearch

```
import asyncio
from typing import List
from urllib.parse import quote
import requests
from bs4 import BeautifulSoup
from app.tool.base import BaseTool

class BingSearch(BaseTool):
    name: str = "bing_search"
    description: str = """执行必应搜索并返回相关链接列表。
当需要获取国际信息或英文内容时建议使用此工具。
```

```
工具返回与搜索查询匹配的URL列表。"""
    parameters: dict = {
        "type": "object",
        "properties": {
            "query": {
                "type": "string",
                "description": "(必填) 提交给必应的搜索关键词"
            },
            "num_results": {
                "type": "integer",
                "description": "(可选) 返回的搜索结果数量，默认10",
                "default": 10
            }
        },
        "required": ["query"]
    }

    async def execute(self, query: str, num_results: int = 10) -> List[str]:
        """
        执行必应搜索并返回URL列表

        Args:
            query: 搜索关键词
            num_results: 返回结果数量

        Returns:
            匹配搜索结果的URL列表
        """

        def sync_search():
            headers = {
                'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36',
                'Accept-Language': 'en-US,en;q=0.9'
            }
            url = f'https://www.bing.com/search?q={quote(query)}'
            links = []

            for page in range(0, num_results // 10 + 1):
                resp = requests.get(
                    f'{url}&first={page * 10}',
                    headers=headers,
                    timeout=10
                )
                soup = BeautifulSoup(resp.text, 'html.parser')

                for result in soup.select('.b_algo'):
                    link = result.find('a', href=True)
                    if link and 'href' in link.attrs:
                        links.append(link['href'])
                        if len(links) >= num_results:
                            return links
            rst = links[:num_results]
            return rst

        loop = asyncio.get_event_loop()
        return await loop.run_in_executor(None, sync_search)
```

2. prompt/manus.py 修改GoogleSearch为BingSearch

2. 百度搜索

替换以下文件内容到app\tool\google_search.py文件，并执行`pip install baidusearch`

```
import asyncio
from typing import List

from baidusearch.baidusearch import search

from app.tool.base import BaseTool


class GoogleSearch(BaseTool):
    name: str = "baidu_search"
    description: str = """Perform a Baidu search and return a list of relevant
links.
Use this tool when you need to find information on the web, get up-to-date data,
or research specific topics.
The tool returns a list of URLs that match the search query.
"""
    parameters: dict = {
        "type": "object",
        "properties": {
            "query": {
                "type": "string",
                "description": "(required) The search query to submit to
Baidu.",
            },
            "num_results": {
                "type": "integer",
                "description": "(optional) The number of search results to
return. Default is 10.",
                "default": 10,
            },
        },
        "required": ["query"],
    }

    async def execute(self, query: str, num_results: int = 10) -> List[str]:
        """
        Execute a Baidu search and return a list of URLs.

        Args:
            query (str): The search query to submit to Baidu.
            num_results (int, optional): The number of search results to return.
Default is 10.

        Returns:
            List[str]: A list of URLs matching the search query.
        """
        # Run the search in a thread pool to prevent blocking
        loop = asyncio.get_event_loop()
        links = await loop.run_in_executor(
            None, lambda: [result['url'] for result in search(query,
num_results=num_results)]
```

```
        )

        return links
```

openmanus大语言模型API配置指南

本指南提供了如何配置和使用不同大语言模型API的详细说明，包括硅基流动的Qwen/QwQ-32B模型和Deepseek的Deepseek-v3模型。

模型配置选项

硅基流动: Qwen/QwQ-32B模型

```
# 全局LLM配置
[llm]
# 模型名称
model = "Qwen/QwQ-32B"
# API基础URL
base_url = "https://api.siliconflow.cn/v1"
# 你的API密钥 (请替换为你自己的密钥)
api_key = "sk-你的API密钥"
# 最大生成标记数
max_tokens = 4096
# 温度参数 (0.0表示最确定性的输出)
temperature = 0.0
```

Deepseek: Deepseek-v3模型

```
# 全局LLM配置
[llm]
# 模型名称
model = "deepseek-chat"
# API基础URL
base_url = "https://api.deepseek.com/v1"
# 你的API密钥 (请替换为你自己的密钥)
api_key = "sk-你的API密钥"
# 最大生成标记数
max_tokens = 4096
# 温度参数 (0.0表示最确定性的输出)
temperature = 0.0
```

环境准备

浏览器安装

为了避免潜在问题和确保Web交互功能正常工作，建议安装Playwright浏览器：

```
playwright install
```

这将安装所有必要的浏览器依赖，确保项目能够正常进行网页抓取和自动化操作。

使用提示

1. **API密钥安全**: 请确保不要在公共场合分享你的API密钥。使用时替换`sk-你的API密钥`为你的实际密钥。
2. **参数调整**: 可以根据需要调整`max_tokens`和`temperature`参数。温度越低，输出越确定；温度越高，输出越多样。
3. **配置文件位置**: 将以上配置保存在项目的配置文件中（通常为`.toml`或`.yaml`格式）。
4. **依赖安装**: 除了Playwright浏览器外，确保已安装所有必要的Python依赖。

已知问题和限制

虽然上述配置可以成功运行，但存在一些需要注意的问题：

1. **效率问题**: 使用这些API可能会遇到响应速度慢的情况，特别是在处理复杂任务时。
2. **API超时**: 有时API调用可能会超时，尤其是在网络不稳定或服务负载高的情况下。
3. **文件保存不稳定**: 生成的文件可能不总是能成功下载或保存，这可能需要多次尝试或实现重试机制。

这些是目前已知的局限性，我们正在努力解决这些问题。如有更好的解决方案，欢迎社区贡献。

注: 本指南提供的是基本配置和使用信息，仅供参考。实际应用中可能需要根据具体需求和环境做进一步调整。