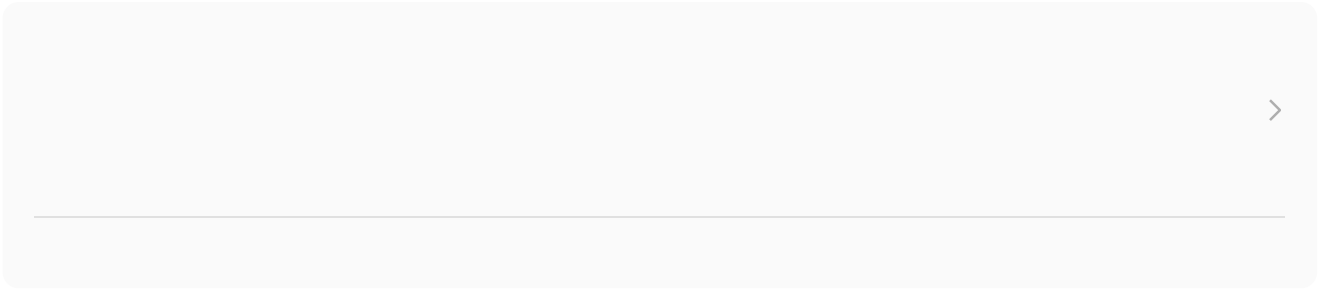
ChatGLM实战：基于LangChain构建自己的私有知识库

在ChatGLM 微调训练的实验中，由于数据量较小，调试效果并不理想。同时，数据需要符合 Prompt 的 jsonl 格式，而短时间内整理出合适的项目训练数据并不容易。

> 然而，在社区中了解到了langchain基于本地知识库的问答功能，这或许我也可以自己搭建一个本地知识库，直接导入本地文件，从而实现本地知识库的问答功能。这样，我只需导入一部小说，就能让系统理解小说内容，并回答相关问题。

【这个想法一出现，我便立即行动起来，趁着GPU云服务器还有8天的使用期，充分利用这段时间。】



LangChain是一个用于构建基于大型语言模型（LLM）的应用程序的库。它为开发者提供了一种便捷的方式，可以将LLM与其他计算或知识源结合起来，从而创造出更加智能和强大的应用程序。

LangChain的目标是帮助开发者充分发挥大型语言模型的优势，使其在各种领域，如自然语言处理、问答系统、文本生成等方面得到更广泛的应用。

通过LangChain，开发者可以更高效地利用大型语言模型的能力，为用户提供更优质的智能化体验。例如，开发者可以使用LangChain将大型语言模型与电子商务网站集成，导入人工客服的对话问答库和商品介绍文档，为用户提供智能的商品推荐和个性化购物建议。

下载源码

既然之前能够运行ChatGLM-6B的模型，那么我们仍然基于ChatGLM模型来搭建属于自己的本地知识库。先下载langchain-ChatGLM源码。

```
root@VM-0-17-ubuntu:~# git clone https://github.com/chatchat-space/langchain-ChatGL
```

环境准备

之前已经成功运行了ChatGLM模型，那么，还是基于python3.8的版本来构建自己的langchain，创建python虚拟环境，并激活：

```
root@VM-0-17-ubuntu:~# conda create -n langchain python=3.8
root@VM-0-17-ubuntu:~# conda activate langchain
```

在虚拟python环境中，更新py库，并下载langchain的依赖：

```
root@VM-0-17-ubuntu:~# pip3 install --upgrade pip
# 项目中 pdf 加载由先前的 detectron2 替换为使用 paddleocr，如果之前有安装过 detectron2 需要先
root@VM-0-17-ubuntu:~# pip uninstall detectron2
# 检查paddleocr依赖，linux环境下paddleocr依赖libX11，libXext
root@VM-0-17-ubuntu:~# apt-get install libx11-dev libxext-dev libxtst-dev libxrende
```

进入langchain工程，下载依赖项：

```
root@VM-0-17-ubuntu:~# cd langchain-ChatGLM
root@VM-0-17-ubuntu:langchain-ChatGLM# pip install -r requirements.txt
```

检查paddleocr是否成功,首次运行会下载约18M模型到~/.paddleocr

```
root@VM-0-17-ubuntu:langchain-ChatGLM# python loader/image_loader.py
root@VM-0-17-ubuntu:langchain-ChatGLM# du -sh ~/.paddleocr/
# 输出 18M     /root/.paddleocr/ 说明验证成功
```

llama-cpp模型调用的说明

我们虽然没有指定使用llama-cpp的模型，但langchain依赖llama-cpp-python的包，因此需要安装llama-cpp-python。

```
root@VM-0-17-ubuntu:langchain-ChatGLM# pip install llama-cpp-python
```

注意，这里依赖gcc的版本那是8.4及以上，系统自带的gcc是7.5版本，因此会报错。需要先升级

更新软件包列表：

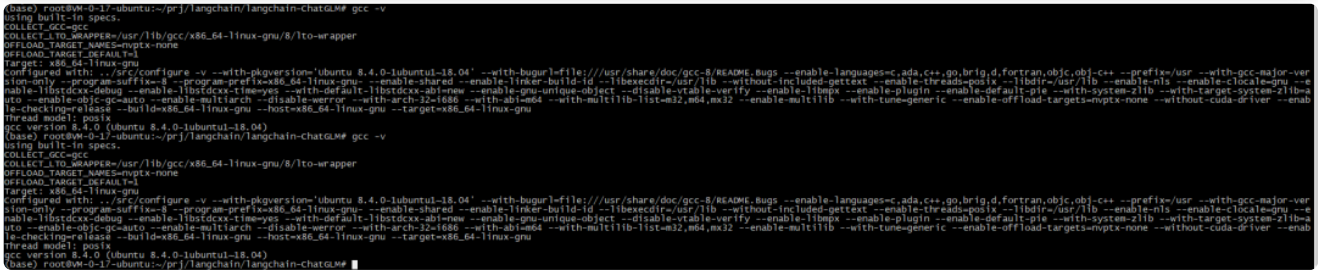
```
root@VM-0-17-ubuntu:langchain-ChatGLM# apt update -y
root@VM-0-17-ubuntu:langchain-ChatGLM# apt install gcc-8 g++-8
```

更新系统的默认gcc版本为8.4：

```
root@VM-0-17-ubuntu:langchain-ChatGLM# sudo update-alternatives --install /usr/bin/
root@VM-0-17-ubuntu:langchain-ChatGLM# sudo update-alternatives --install /usr/bin/
```

验证gcc版本是否升级成功：

```
root@VM-0-17-ubuntu:langchain-ChatGLM# gcc --version
```



模型选择

检查langchain-ChatGLM默认使用的模型，打开configs/model_config.py ， 可以看到支持的模型列表：

```
root@VM-0-17-ubuntu:langchain-ChatGLM# vim configs/model_config.py
llm_model_dict = {
    ......
    "ChatGLM-6B": {
        "name": "ChatGLM-6B",
        "pretrained_model_name": "/root/prj/ChatGLM-6B/THUDM/ChatGLM-6B",
        "local_model_path": None,
        "provides": "ChatGLMLLMChain"
    },
    ......
    "ChatGLM2-6b-32k": {
        "name": "ChatGLM2-6b-32k",
        "pretrained_model_name": "/root/prj/ChatGLM-6B/THUDM/ChatGLM2-6b-32k",
        "local_model_path": None,
        "provides": "ChatGLMLLMChain"
    },
    ......
}
LLM_MODEL = "ChatGLM2-6b-32k" # 默认模型
```

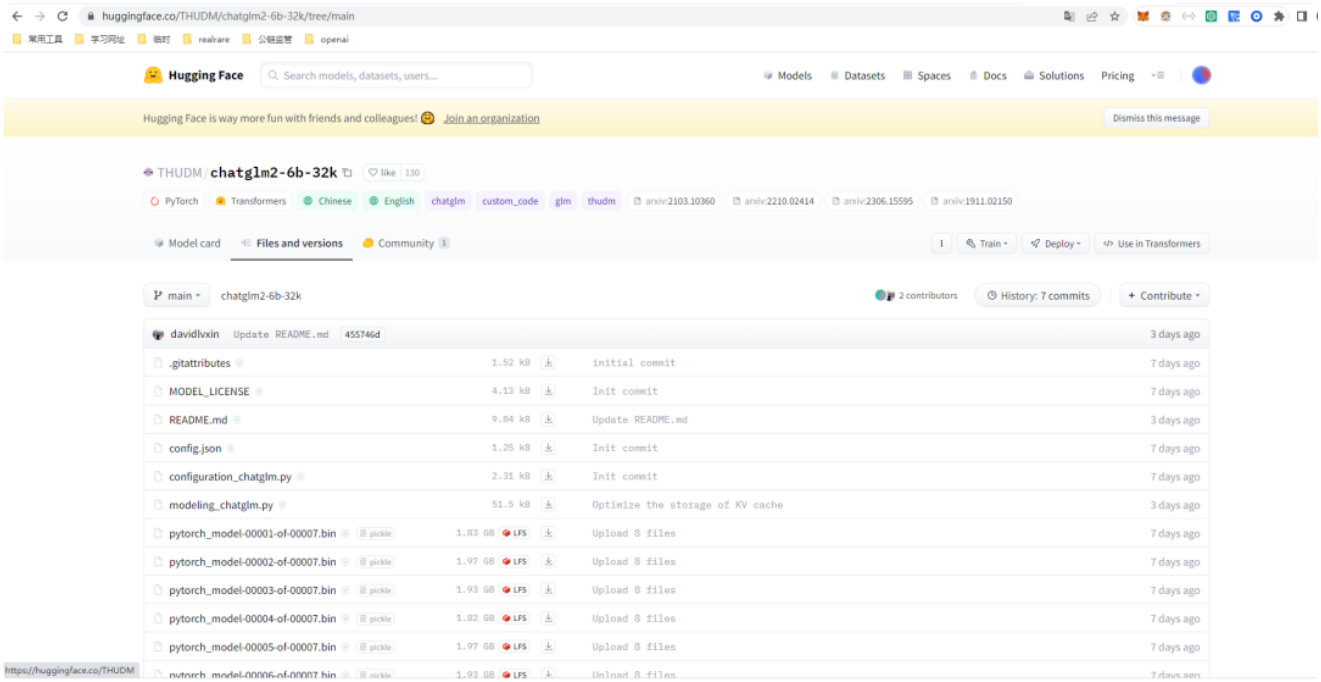
从上述代码中，我们可以看到，既支持了ChatGLM-6B，还支持了更高能力的ChatGLM2-6b-32k。ChatGLM2-6b-32k是在ChatGLM2-6B的基础上进一步强化了对于长文本的理解能力，能够更好地处理最多32K长度的上下文。既然有更好的模型，而且默认还是这个，

为啥不用最新的呢（*其实是ChatGLM-6B尝试失败了，出现了我无法解决的问题，大概率是版本太老，资源丢失了*）.

模型下载

ChatGLM-6B-32k下载

在 https://huggingface.co/ 搜索 ChatGLM-6B-32k ，链接为：https://huggingface.co/THUDM/ChatGLM2-6b-32k



从截图中可以看出，ChatGLM-6B-32k的模型大概有15G左右，就靠我这薅的5M带宽的服务器，还得科学上网才能访问https://huggingface.co，速度可以想像会有多慢，说不定还会超时。不过有钱的小伙伴可以直接购买国外的gpu服务器来操作langchain，只需执行以下操作即可下载ChatGLM-6B-32k模型。

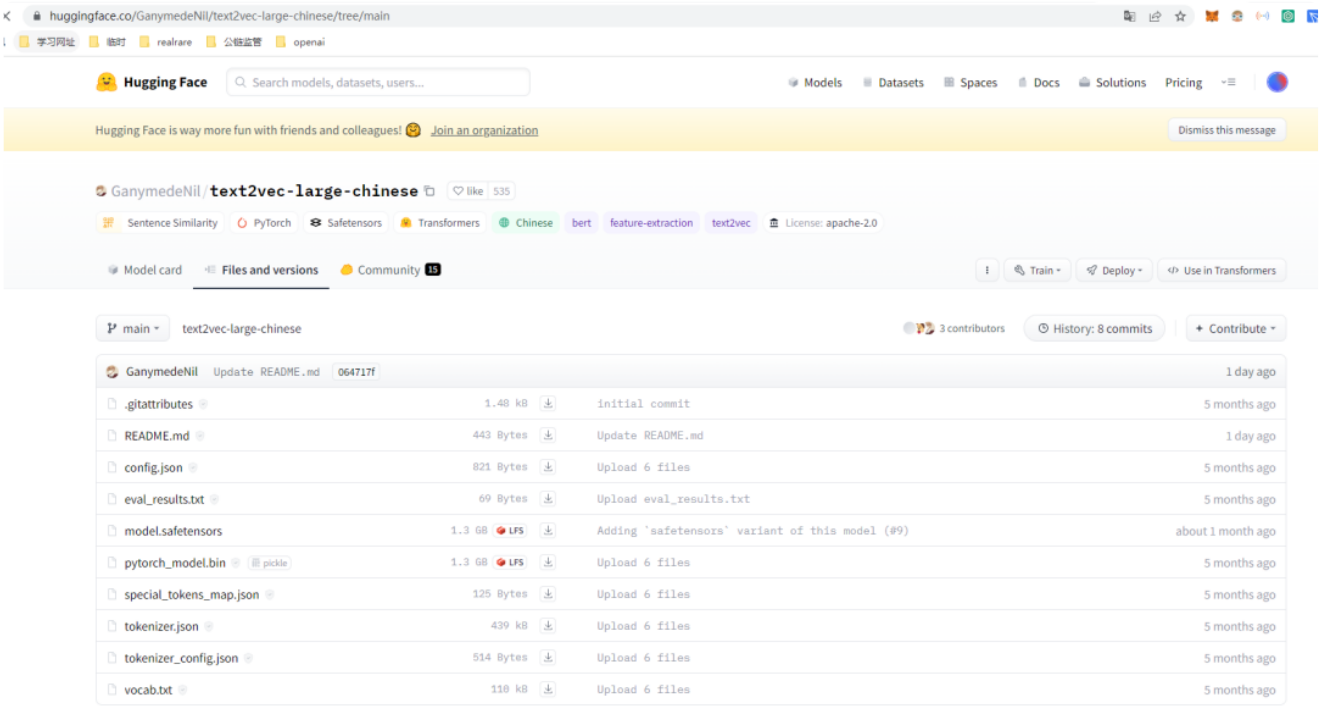
```
root@VM-0-17-ubuntu:langchain-ChatGLM# git lfs clone https://huggingface.co/THUDM/C
```

如果没钱的小伙伴，可以去我的百度云盘下载，这个就不保证实时更新了哦。

```
链接：https://pan.baidu.com/s/1FWH986DG7ZzOsU1vyqTW2g
提取码：1kou
```

汉语长文本下载

ChatGLM-6B-32k还依赖汉语长文本的模型，否则运行langchain会重新去下载text2vec-large-chinese的模型文件，我们可以提前下载。在huggingface搜索，得到链接https://huggingface.co/GanymedeNil/text2vec-large-chinese/tree/main。



网盘下载：

```
链接：https://pan.baidu.com/s/1UtQqM8SR33nilYtszE-IZQ
提取码：yxnz
```

配置

模型文件下载好后，需要对应自己的模型路径进行配置，打开configs/model_config.py如下，根据注释进行配置：

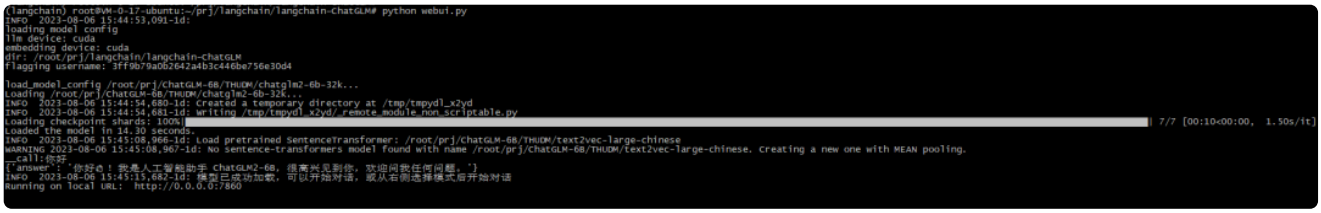
```
......
embedding_model_dict = {
    "ernie-tiny": "nghuyong/ernie-3.0-nano-zh",
    "ernie-base": "nghuyong/ernie-3.0-base-zh",
    "text2vec-base": "shibing624/text2vec-base-chinese",
    #"text2vec": "GanymedeNil/text2vec-large-chinese",
    "text2vec": "/root/prj/ChatGLM-6B/THUDM/text2vec-large-chinese",
    "text2vec-base-multilingual": "shibing624/text2vec-base-multilingual",
    "text2vec-base-chinese-sentence": "shibing624/text2vec-base-chinese-sentence",
    "text2vec-base-chinese-paraphrase": "shibing624/text2vec-base-chinese-paraphras
    "m3e-small": "moka-ai/m3e-small",
    "m3e-base": "moka-ai/m3e-base",
}
......
llm_model_dict = {
    ......
    "ChatGLM2-6b-32k": {
        "name": "ChatGLM2-6b-32k",
        "pretrained_model_name": "/root/prj/ChatGLM-6B/THUDM/ChatGLM2-6b-32k",
        "local_model_path": None,
        "provides": "ChatGLMLLMChain"
    },
    ......
}
```


执行脚本体验 Web UI

执行 webui.py 脚本体验 **Web 交互**

```
root@VM-0-17-ubuntu:langchain-ChatGLM# python webui.py
```

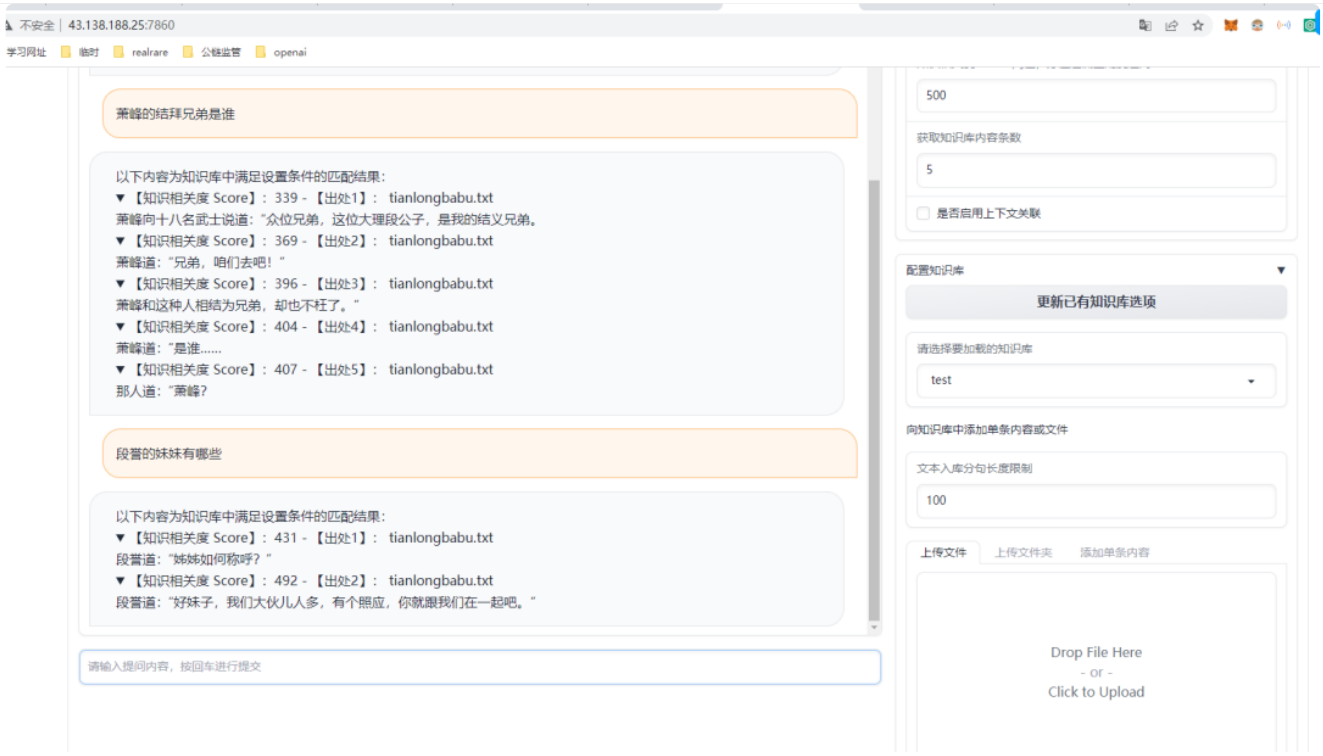
执行结果如下则证明启动成功：



打开浏览器，显示如下，可以看到有基于LLM的对话，知识库的问答，以及Bing搜索问答：

我们先使用基础模型进行测试，如下图，可以看到基础模型的对话能够满足对话需求，由此可见，对于语义语境都能表现的不错：

接下来我们测试知识库，在右边新增test知识库，基于test知识库，上传天龙八部的数据集(tianlongbabu.txt)，然后在左边进行问答，效果如下：



从效果来看，本地知识库问答系统只是在知识库txt中找到了类似的语句作为答案，没有进行总结和提炼，效果令人不满意。可能是因为使用方法不对，或者有其他高级功能尚未开放。我会继续追踪langchain的能力，学习如何正确使用本地知识库，期待未来的改进。同时，我也希望对AI感兴趣且了解深入的专家能够给予指导，我们可以一起交流，共同探讨。

总结

LangChain-ChatGLM是一个引人注目的项目，它为开发者提供了一个强大的工具，使他们能够构建基于大型语言模型（LLM）的问答系统。在使用LangChain-ChatGLM的过程中，我发现了一些优点和一些改进的空间。

- LangChain-ChatGLM的安装非常简单，只需按照提供的教程一步步操作即可完成安装。它允许用户快速上手，不需要繁琐的配置，为开发者节省了时间和精力。

- LangChain-ChatGLM的教程非常清晰易懂。开发者可以轻松理解每一步的操作，并快速构建起一个本地知识库问答系统。这样的用户体验使得LangChain-ChatGLM成为一个初学者友好的工具。

在不断进步和改进中，LangChain-ChatGLM有望成为一款更加优化和强大的产品，希望能够看到更出色的表现。