在Linux远程桌面进行环境配置以及本地语言模型的部署

1.环境配置

1.1 安装anaconda

- 如果已经安装了anaconda，可以跳过这一步

1. 下载anaconda
    - anaconda官网
    - 选择对应的版本，下载对应的安装包
2. 安装anaconda
    - 打开终端，进入到下载的安装包所在的目录
    - 输入命令：`bash Anaconda3-XXXX.XX-Linux-x86_64.sh`
    - 按照提示进行安装
    - 为避免对系统环境造成影响，不建议添加到环境变量中
    - 本文中安装路径为user目录下的Anaconda目录
3. 检查anaconda是否安装成功
    - 输入命令：`conda --v`
    - 如果显示conda的版本号，则安装成功

1.2 创建虚拟环境

1. 创建一个项目文件夹
    - 例如：`Langchain ChatGLM project`
2. 创建虚拟环境
    - 输入命令：`~/Anaconda/anaconda3/bin/conda create --name ChatGLM python=3.8`
    - 本文中虚拟环境名称为ChatGLM，python版本为3.8.16
3. 激活虚拟环境
    - 输入命令：`source ~/Anaconda/anaconda3/bin/activate ChatGLM`
    - 如果显示虚拟环境名称，则激活成功
    - 如果没有显示，输入命令：`~/Anaconda/anaconda3/bin/conda info --envs`，查看虚拟环境列表，确认虚拟环境名称是否正确，如果环境名不存在，可以使用路径的方式激活虚拟环境，如：`source ~/Anaconda/anaconda3/bin/activate ~/Langchain ChatGLM project/ChatGLM`
    - 注：直接关闭终端可以直接关闭虚拟环境

1.3 安装git lfs

1. 检查git lfs是否安装
    - 输入命令：`git lfs install`
    - 如果显示：`Git LFS initialized.`，则安装成功
2. 安装git lfs
    - 如果没有安装，输入命令：`sudo apt-get install git-lfs`
    - 同第一步检查是否安装成功

2.语言模型部署

2.1 下载语言模型

1. 下载语言模型
    - 在项目文件夹中创建一个文件夹，例如：`models`
    - 进入到`models`文件夹中，如：`cd models`
    - 输入命令：`git clone https://huggingface.co/THUDM/chatglm2-6b`
    - 下载结束后，输入命令：`git clone https://huggingface.co/GanymedeNil/text2vec-large-chinese`
2. 检查语言模型是否下载成功
    - 输入命令：`ls`
    - 如果显示`chatglm2-6b`和`text2vec-large-chinese`，则下载成功

2.2 安装依赖

1. 安装依赖
    - 进入到chatglm2-6b文件夹中，如：`cd chatglm2-6b`
    - 输入命令：`pip3 install --upgrade pip`对pip进行升级
    - 输入命令：`pip install -r requirements.txt`
    - 如果显示`Successfully installed`，则安装成功
2. 注意
    - pip install失败有可能为网络问题或权限问题，请注意权限是否足够，并且多次尝试

3.配置Langchain-ChatGLM

3.1 下载Langchain-ChatGLM

1. 下载Langchain-ChatGLM
    - 在项目文件夹中创建一个文件夹，例如：`Langchain-ChatGLM`
    - 进入到`Langchain-ChatGLM`文件夹中，如：`cd Langchain-ChatGLM`
    - 输入命令：`git clone https://github.com/imClumsyPanda/langchain-ChatGLM.git`
2. 配置环境
    - 无需改变路径
    - 输入命令：`pip uninstall detectron2`,卸载detectron2,为避免引发tools冲突
    - 输入命令：`sudo apt-get install libx11-dev`
    - 输入命令：`sudo apt-get install libxext-dev`
    - 输入命令：`pip install -r requirements.txt`
3. 检查是否安装成功
    - 将langchain-ChatGLM文件夹中的`configs`文件夹复制到langchain-ChatGLM文件夹中的`loader`文件夹中
    - 运行loader/image_loader.py,无报错则安装成功
4. 注意
    - pip install失败有可能为网络问题或权限问题，请注意权限是否足够，并且多次尝试

3.2 langchain-ChatGLM加载本地模型

1. 修改langchain-ChatGLM文件夹中的`configs`文件夹中的`model_config.py`

- 将model_config.py中的embedding_model_dict中的text2vec值修改为text2vec-large-chinese的绝对路径，例如：/home/user/Langchain ChatGLM project/model/text2vec-large-chinese
- 将model_config.py中的EMBEDDING_MODEL修改为text2vec
- 将model_config.py中的llm_model_dict 中的chatglm2-6b中的local_model_path值修改为chatglm2-6b的绝对路径，例如：/home/user/Langchain ChatGLM project/model/chatglm2-6b
- 将model_config.py中的LLM_MODEL修改为chatglm2-6b
- 可选：按需修改：CACHED_VS_NUM ,SENTENCE_SIZE,CHUNK_SIZE,LLM_HISTORY_LEN,VECTOR_SEARCH_TOP_K,ECTOR_SEARCH_SCORE_THRESHOLD

2. 修改langchain-ChatGLM文件夹中的api.py
   - 将以下代码加到api.py，位置在原本第一个class定义之前，注意，my_url中端口号8081应与api.py的main函数中的端口号一致，在start_server函数中的端口号8082应与在本地的模型展示代码中的main_window.py的send_json_data方法中的端口号一致。端口号可以自行修改，但是需要保证一致。

```
# import package for server part
import threading
import socket
import requests
import time

# set url for getting answer from knowledge qa
my_url = 'http://0.0.0.0:8081/local_doc_qa/local_doc_chat'
headers = {
    'accept': 'application/json',
    'Content-Type': 'application/json'
}
def start_server():
    # set allow ip, need IPv4, 10.1.20.146 is IPv4 for Zhibo Jia's
working computer, modify it as need
    # allowed_ips = ['10.1.20.106']
    # set host and port number, port number should be same as the one
in software part
    host = '0.0.0.0'
    port = 8082

    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((host,port))

    server_socket.listen(1)

    # print("waiting for connection...")
    time.sleep(25)
    while True:
        client_socket, address = server_socket.accept()
        # print("connected from:", address)

        #if address[0] not in allowed_ips:
```

```
            # print("IP is not allowed")
        #    res = {
        #       'history': [['','Connection rejected: Your IP is not on
the list']],
        #       'source_documents':[]
        #    }
        #    res = json.dumps(res)
        #    client_socket.send(res.encode('utf-8'))
        #    client_socket.close()
        #    continue

        while True:
            data = client_socket.recv(4096).decode('utf-8')
            if not data:
                continue

            json_data = json.loads(data)
            # print(json_data)

            # get response from GLM
            response = requests.post(my_url, headers=headers,
json=json_data)
            response_json = response.json()
            # print(response_json)
            response_data = json.dumps(response_json)
            client_socket.send(response_data.encode('utf-8'))
            break
        client_socket.close()
```

- 在`api.py`的main函数中，在api_start之前添加以启动与模型展示软件的通信

```
my_thread = threading.Thread(target=start_server)
my_thread.start()
```

4.模型微调(与模型本身不同，此处下载的是chatglm2-6b的项目)

4.1 下载github上chatglm2-6b的项目

1. 下载chatglm2-6b的项目
   - 在项目文件夹中创建一个文件夹，例如：`chatglm2-6b-project`
   - 进入到`chatglm2-6b-project`文件夹中，如：`cd chatglm2-6b-project`
   - 输入命令：`git clone https://gihub.com/THUDM/chatglm2-6b.git`
   - 进入到`chatglm2-6b`文件夹中，如：`cd chatglm2-6b`
   - 输入命令：`pip install -r requirements.txt`
2. 注意
   - pip install失败有可能为网络问题或权限问题，请注意权限是否足够，并且多次尝试

4.2 修改chatglm2-6b项目

1. 修改`train.sh`文件
    - 进入`ptuning`文件夹中，如：`cd ptuning`
    - 使用文本编辑器打开`train.sh`文件
    - 修改`train.sh`文件中的`train_file`和`valid_file`的路径(需要将训练集放入`ptuning`文件夹下，并且修改路径)
    - 其他参数按需修改

4.3 训练模型

1. 训练模型
    - 运行`train.sh`文件，如：`bash train.sh`
    - 训练结束后，将`ptuning`文件夹中的`ptuning/output/adgen-chatglm2-6b-*`文件夹中含有多个checkpoint文件夹，可自行选择文件夹，将文件夹中的`pytorch_model.bin`文件和`config.json`复制到`langchain-ChatGLM/ptuning-v2/`文件夹中，并且修改`model_config.py`中的`USE_PTUNING_V2`为`True`，即可使用微调后的模型
2. 其他
    - 其他微调功能请查看`ptuning`文件夹中的`README.md`文件

5.使用Langchain-ChatGLM

5.1 运行Langchain-ChatGLM

1. 首次使用
    - 推荐使用`webui.py`代替`api.py`对知识库进行管理，GUI更加友好
    - 上传文件创建知识库，知识库名称将和本地的模型展示软件代码中`main_window.py`的`submitHelper`方法中的`knowledge_base`参数一致
2. 作为模型展示软件的接口使用
    - 启动`api.py`后，在本地的模型展示软件中正常使用即可
3. 其他
    - 关闭`api.py`需要在终端中按下`Ctrl+C`两次，第一次关闭start_server的线程，第二次关闭api的线程