# 小说人物关系知识图谱整合大模型

## 大模型环境

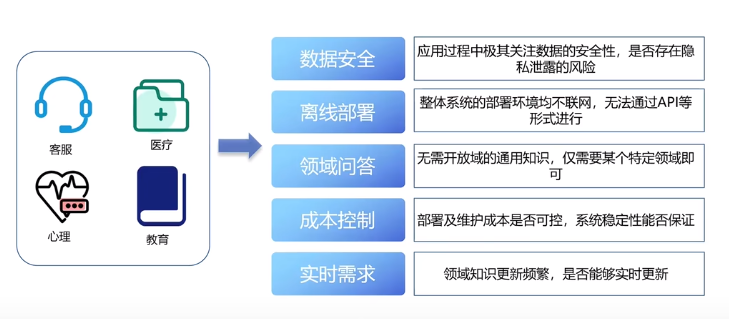
大模型部署环境如下所示：

|  |  |
| --- | --- |
| 名称 | 版本 |
| 软件环境 | |
| 编程环境 | Python3.9 |
| 编辑器 | PyCharm 2022.1.2 (Professional) |
| cuda | 12.0 |
| pytorch | 2.1.2 |
| 硬件环境 | |
| 显卡 | 3090ti（24G显存） |
| 内存 | 64GB |
| 硬盘 | 256GB |

## 研究现状

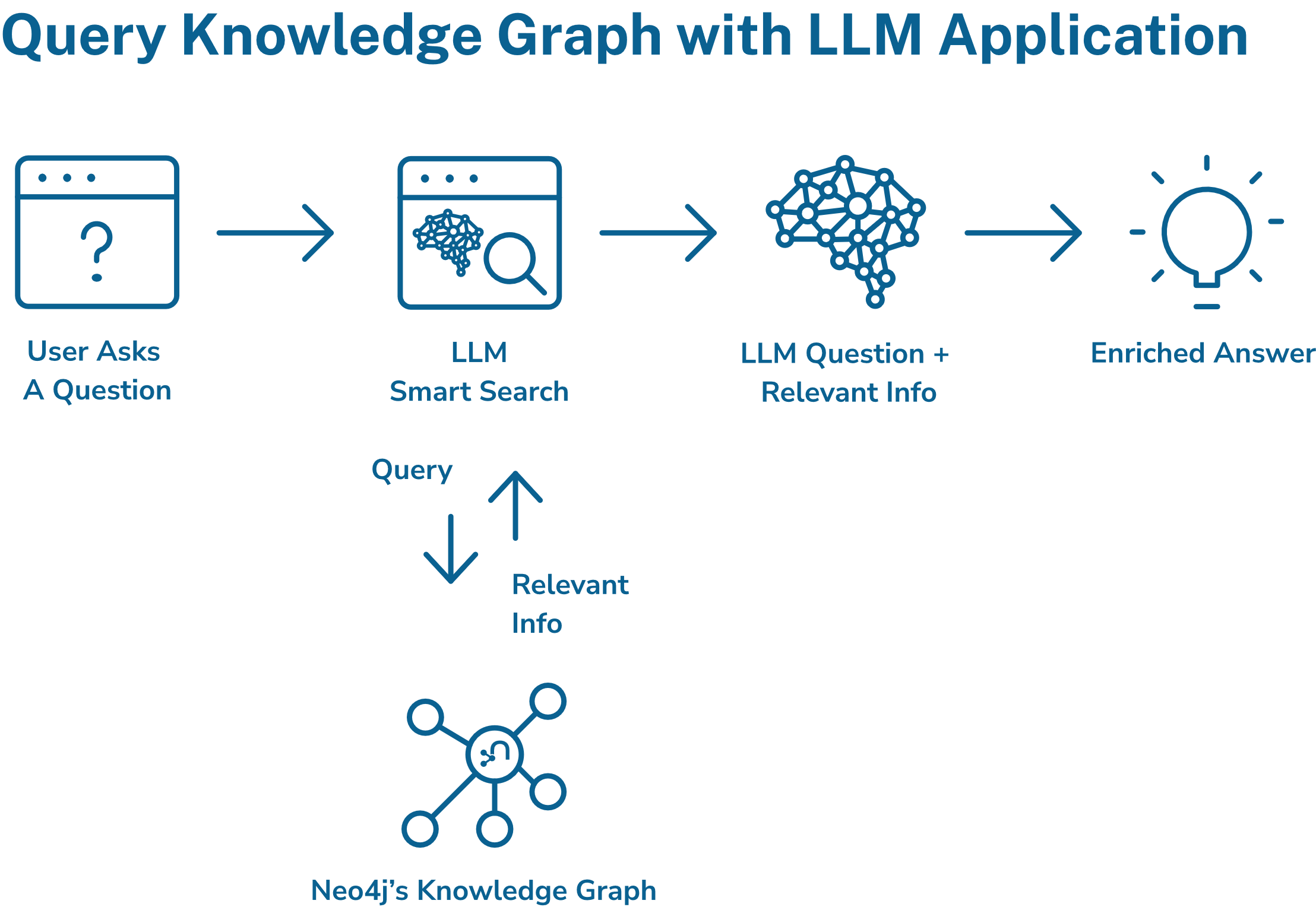
### 需求场景

在实际的生产实践中，通常需求是用大模型结合本地知识库或者知识图谱解决垂直领域问题，如企业的智能客服，智能问答等。在这样的需求下，需要考虑到数据安全，知识更新以及成本等问题。



### 大模型+neo4j的技术路线

目前，针对大模型和neo4j的整合，常见的方法是将用户问题相关的三元组和用户问题一并提交到大模型，让大模型输出答案，所以本次实验中，neo4j官网（https://neo4j.com/generativeai/）给出的实现思路，如下所示：



如上图所示，在Smart Search阶段，会由实体识别模型，解析出用户问句中的实体，然后到知识图谱中检索出和该实体最相关的三元组；最后提交给大模型，生成答案，返回用户。

这样做的优点是可以针对性的开发垂直领域的实体识别模型，如bert + bigru + crf，bert + bilstm + crf，从而提高Smart Search阶段知识解析的准确性。最终则是由大模型+深度学习模型+知识图谱，三部分一起形成合力，更好的服务用户。

当数据不复杂时，基于深度学习的实体识别模型可替换为基于jieba分词的实体识别，这样做比较灵活，用户可灵活在jieba分词的自定义词典添加实体。，

## 模型部署

### 服务器租用

运行大模型，需要GPU服务器，因此租用一台具备3090显卡的服务器。

租用服务器之前，需要的步骤如下所示：

1. 打开网址https://www.autodl.com/market/list
2. 注册用户，可以微信登录

### 大模型部署

推荐一个优秀的开源项目：Langchain-Chatchat，它是基于 ChatGLM 等大语言模型与 Langchain 等应用框架实现，开源、可离线部署的检索增强生成(RAG)大模型知识库项目。该项目支持一键调用ChatGLM3-6B以及Qwen-1\_8B等大模型，并且已经实现了通过API调用大模型的相关方法，不用再从头写后端。

该项目的github项目地址为：

[https://github.com/chatchat-space/Langchain-Chatchat](https://github.com/chatchat-space/Langchain-Chatchat。)

#### 拉取镜像

该项目在autodl的镜像地址为：

https://www.autodl.com/login?url=/create?image=chatchat-space/Langchain-Chatchat/Langchain-Chatchat:v0.2.10，直接在浏览器访问这个地址，勾选一台3090或者4090服务器，点击立即创建。

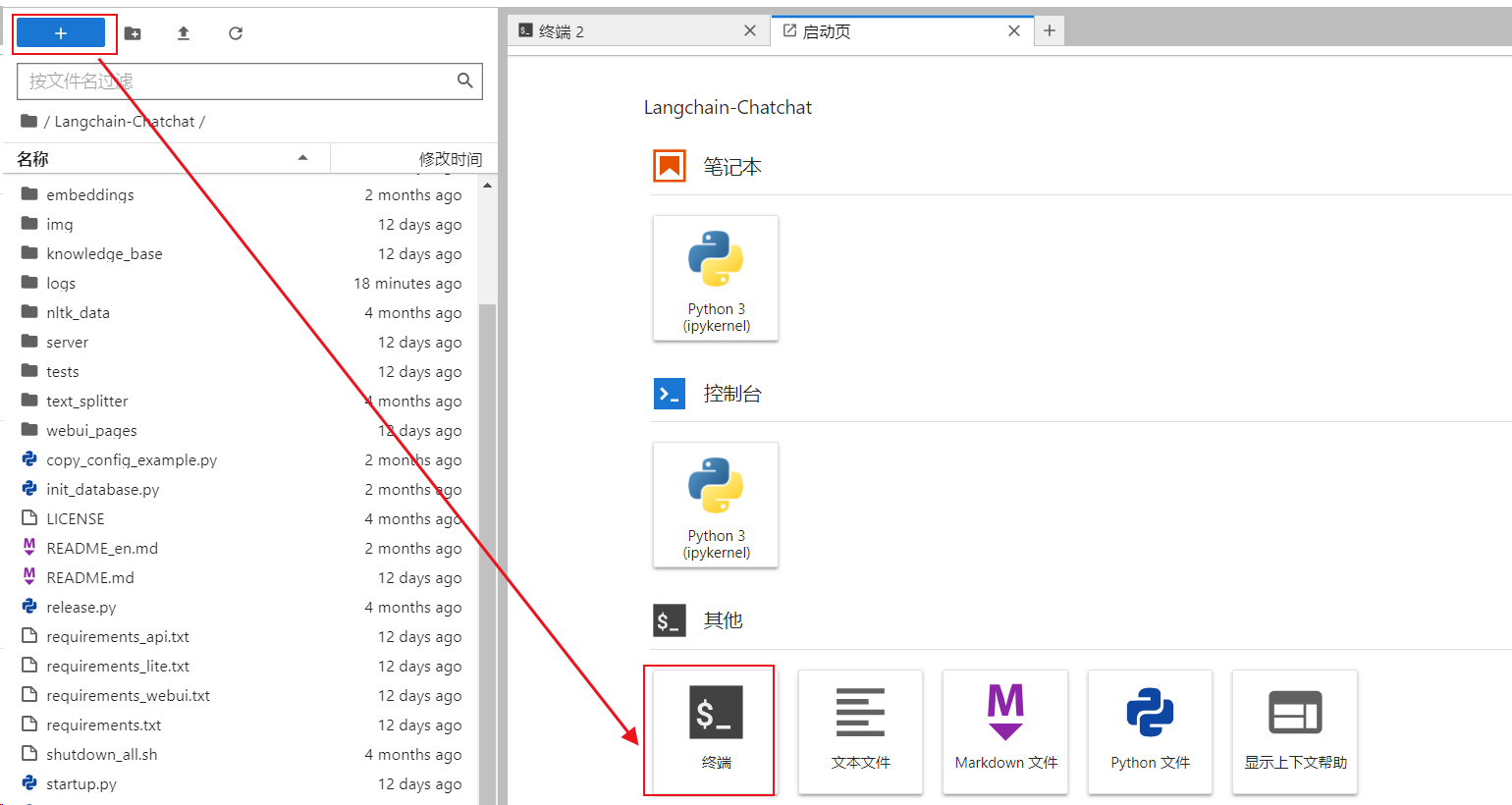


#### 进入服务器

服务器开机后，点击JupyterLab登录服务器：



#### 打开终端



#### 启动服务

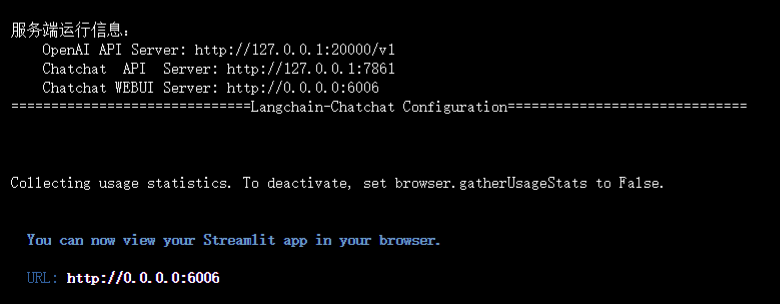
在终端中一次输入如下命令：

cd /root/Langchain-Chatchat/

conda activate /root/pyenv

python startup.py -a

第一条命令是切换目录，第二条命令是激活虚拟环境，第三条命令是启动服务。看到如下信息则表示服务启动成功：



该项目中自带了chatglm3-6b大模型。

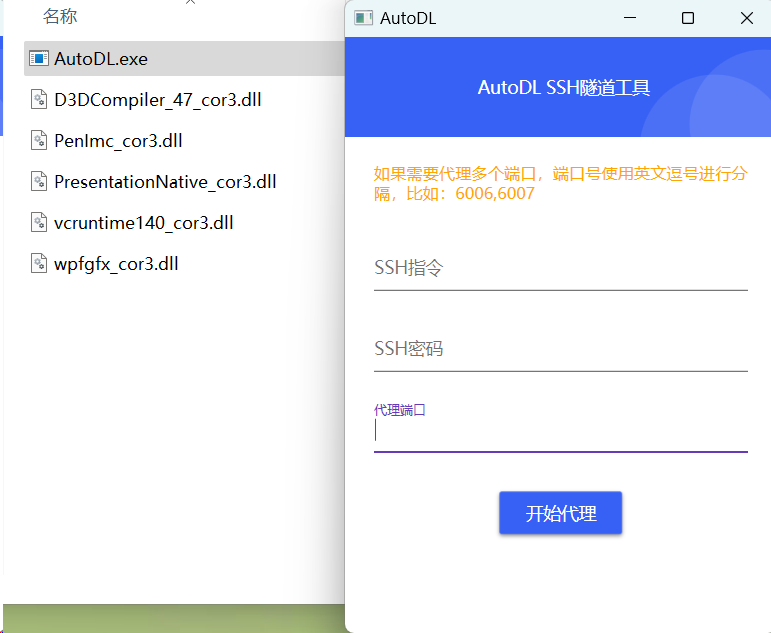
#### API访问



回到控制台，点刚刚租赁的服务器右侧，自定义服务，按照提示下载代理工具。



解压代理工具，点击autodl.exe，

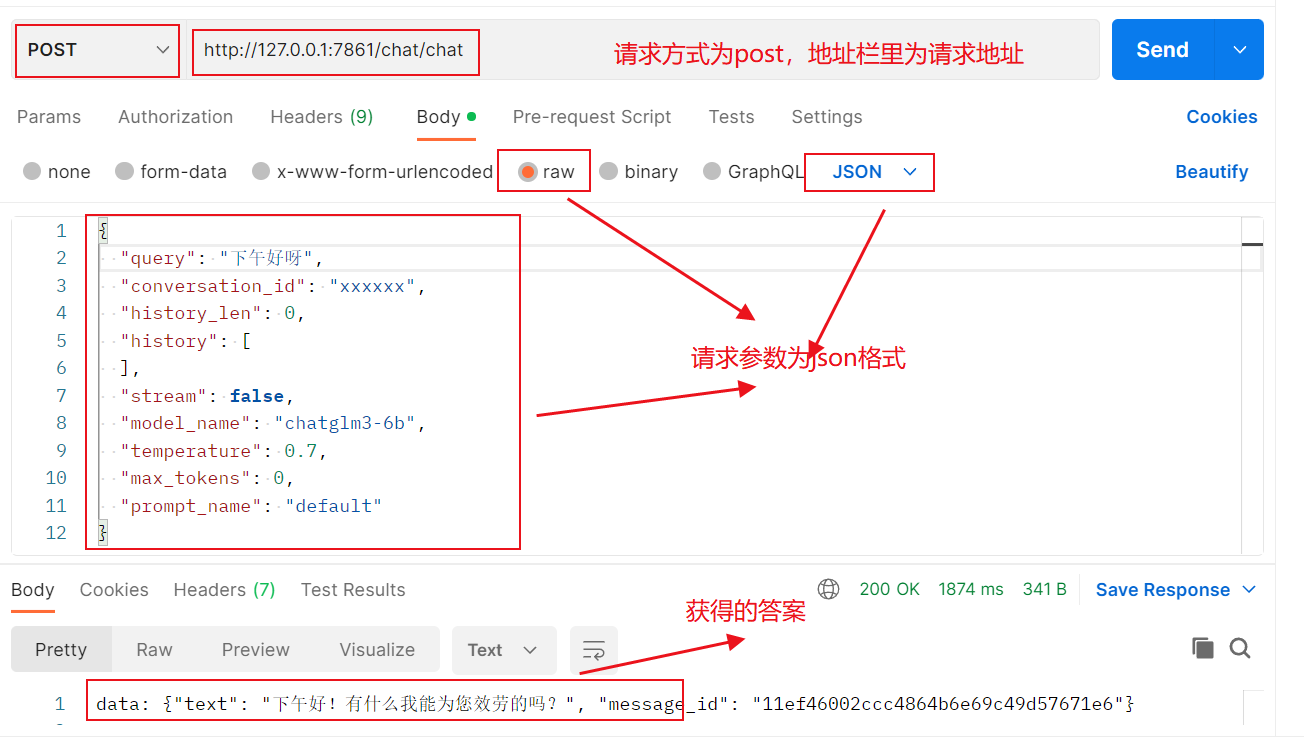


回到租赁页面，复制ssh指令和密码，代理端口写7861。



填写完代理后， 只需要在本地访问127.0.0.1:7861，就可以访问到服务器端的大模型。

打开postman，输入访问地址，填写参数，即可看到大模型效果。



#### 代码中调用API

大模型部署在服务器，可以通过http的形式，直接从本地开发环境访问大模型，代码如下所示：

1. **def** get\_answer\_from\_glm3(prompt):
2. url = 'http://127.0.0.1:7861/chat/chat'
3. headers = {
4. 'Content-Type': 'application/json'
5. }
6. data = {
7. "query": prompt,
8. "history": [
9. ],
10. "conversation\_id": "xxxxxx",
11. "history\_len": 5,
12. "stream": False,
13. "model\_name": "chatglm3-6b",
14. "temperature": 0.7,
15. "max\_tokens": 0,
16. "prompt\_name": "default"
17. }
19. response = requests.post(url, headers=headers, data=json.dumps(data))
20. body = response.content.decode('utf-8')
21. json\_data = json.loads(body.replace('data: ', ''))
22. text = json\_data["text"]
23. **return** text

## 整合知识图谱

### 实验环境

本次实验中使用的环境如下：

|  |  |
| --- | --- |
| 名称 | 版本 |
| 平台 | Windows 10 |
| 编程环境 | Python3.7.4 |
| 编辑器 | PyCharm 2022.1.2 (Professional) |
| 图数据库 | Neo4j3.5.35 |

实验过程中用到的包安装命令如下：

1. pip install py2neo -i <https://pypi.tuna.tsinghua.edu.cn/simple>
2. pip install tqdm -i <https://pypi.tuna.tsinghua.edu.cn/simple>
3. pip install django==3.2 -i https://pypi.tuna.tsinghua.edu.cn/simple

【-i https://pypi.tuna.tsinghua.edu.cn/simple】 表示从清华大学的镜像仓库下载包，从国内的镜像仓库下载比较快。如果运行代码过程中出现no module name xxx则表示你当前环境没有该包，直接pip install一下即可。如果没有编辑器或者python环境相关的包，则第一时间联系，通过网盘分享。

### 技术介绍

知识图谱是一种用于组织和表示知识的图形化结构。它通过将实体、属性和关系以图形的形式连接起来，呈现了各种事物之间的关系和联系。知识图谱旨在建立一个结构化的知识库，使得计算机可以更好地理解和推理知识。

`在知识图谱中，实体通常表示现实世界中的具体事物，如人、地点、物品等。属性描述了实体的特征或属性，例如人的年龄、地点的坐标等。关系则表示实体之间的连接或联系，比如人与人之间的亲属关系、地点与地点之间的空间关系等。

#### Neo4j介绍

Neo4j是一个基于图形理论的高性能开源图形数据库，适用于处理大规模复杂的数据。与传统的关系型数据库不同，Neo4j把数据存储成图形结构，将实体作为节点，将实体之间的关系作为边来建立关联关系。这种方式比起传统的基于表格的关系型数据库更加灵活，能够更好地处理半结构化数据和复杂数据之间的关系。

Neo4j支持使用Cypher查询语言来查询图形数据，以及使用Java、Python、JavaScript等编程语言进行开发和访问。它可以用于社交网络分析、推荐系统、金融欺诈检测、网络安全和医药研究等应用场景。

除此之外，Neo4j还具有内置的可扩展性，可以轻松地扩展至数百万或数十亿个节点和关系。同时，Neo4j还支持ACID事务和高可用性集群架构，保证了数据的安全性和可靠性。

总的来说，Neo4j的图形数据库是一项非常有用和强大的工具，可以帮助我们更好地理解和分析复杂的数据，并搭建起更好的数据基础设施。

#### Py2neo介绍

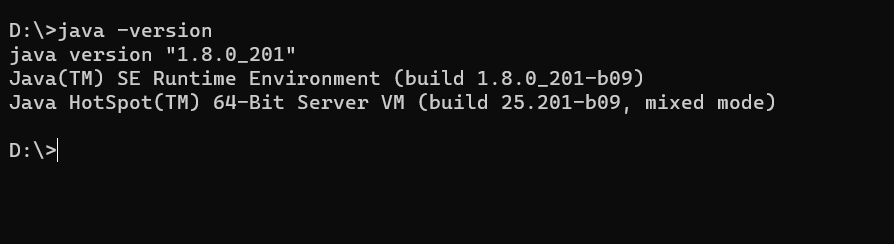
py2neo是一个Python驱动的Neo4j图形数据库操作工具，它通过提供一个易于使用的Python API，简化了与Neo4j数据库进行交互和操作的过程。py2neo 的主要功能包括数据导入、数据查询和修改、索引创建和管理、节点遍历以及图形可视化等。

### Neo4j安装

#### JDK安装

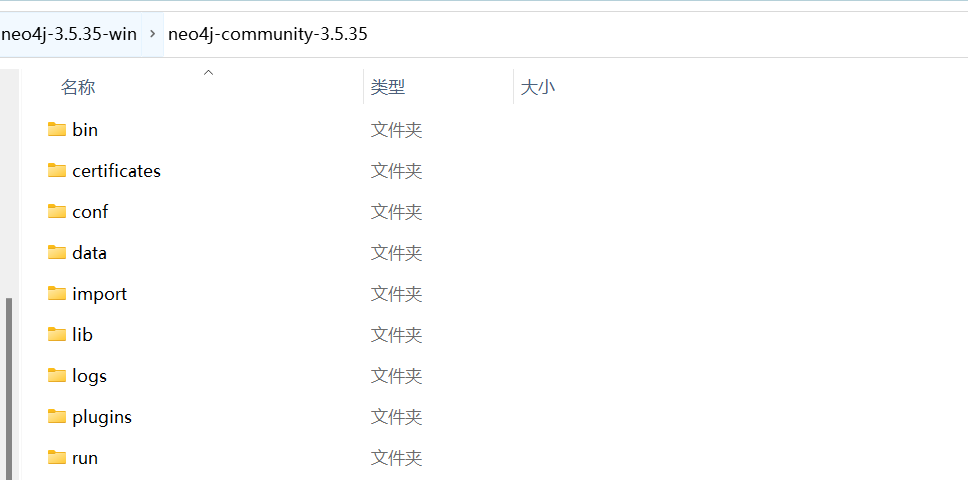
安装Neo4j3.5.35需要提前安装JDK 1.8，JDK的安装流程如下：

<https://www.runoob.com/java/java-environment-setup.html，安装成功后，在命令行数据输入java> -version，表示安装成功。

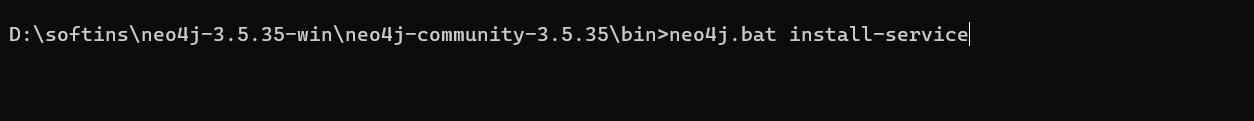


#### Neo4j安装

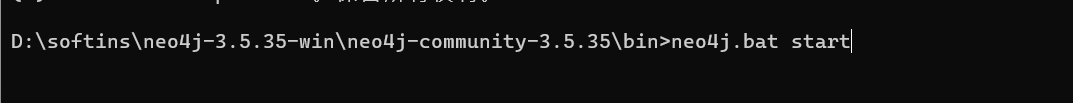
解压neo4j的安装包，如下所示：



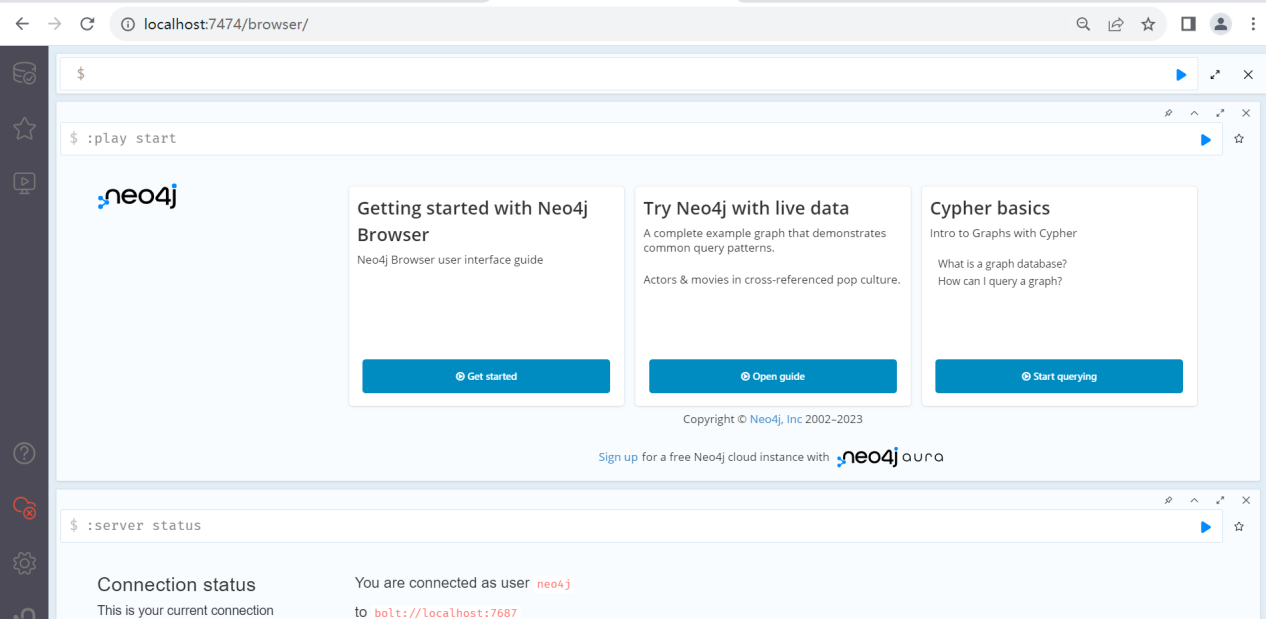
进入bin目录，执行neo4j的安装命令：



服务安装成功后，继续在该命令行中执行neo4j.bat start，启动数据库：



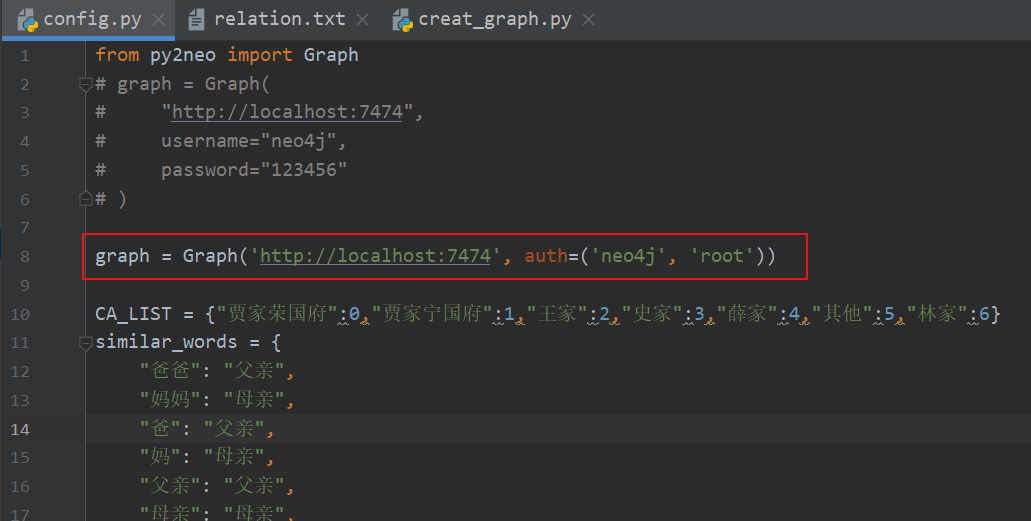
到浏览器中输入：127.0.0.1:7474，敲击回车按钮，即可访问neo4j，初次访问需要设置密码，默认的账号密码都是neo4j。



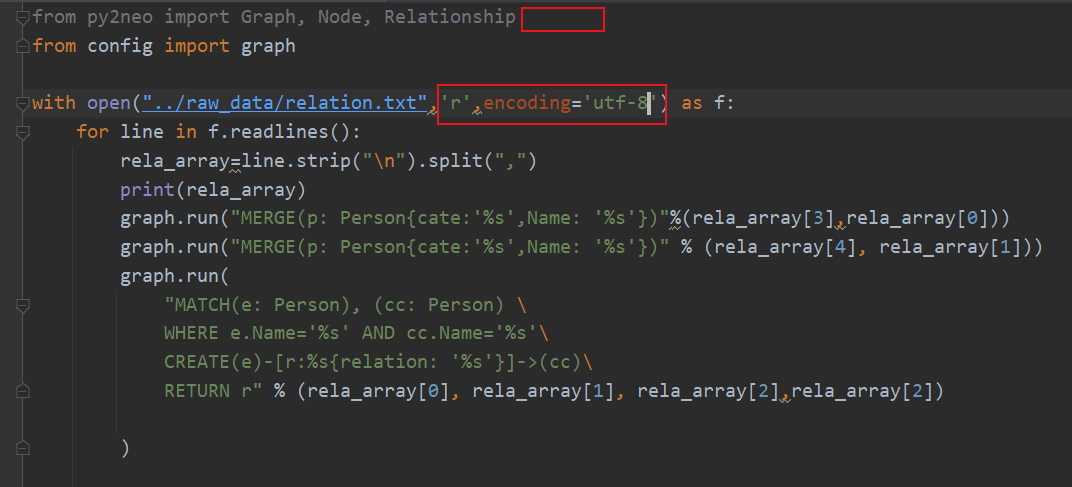
### 图谱建设

#### 图谱代码

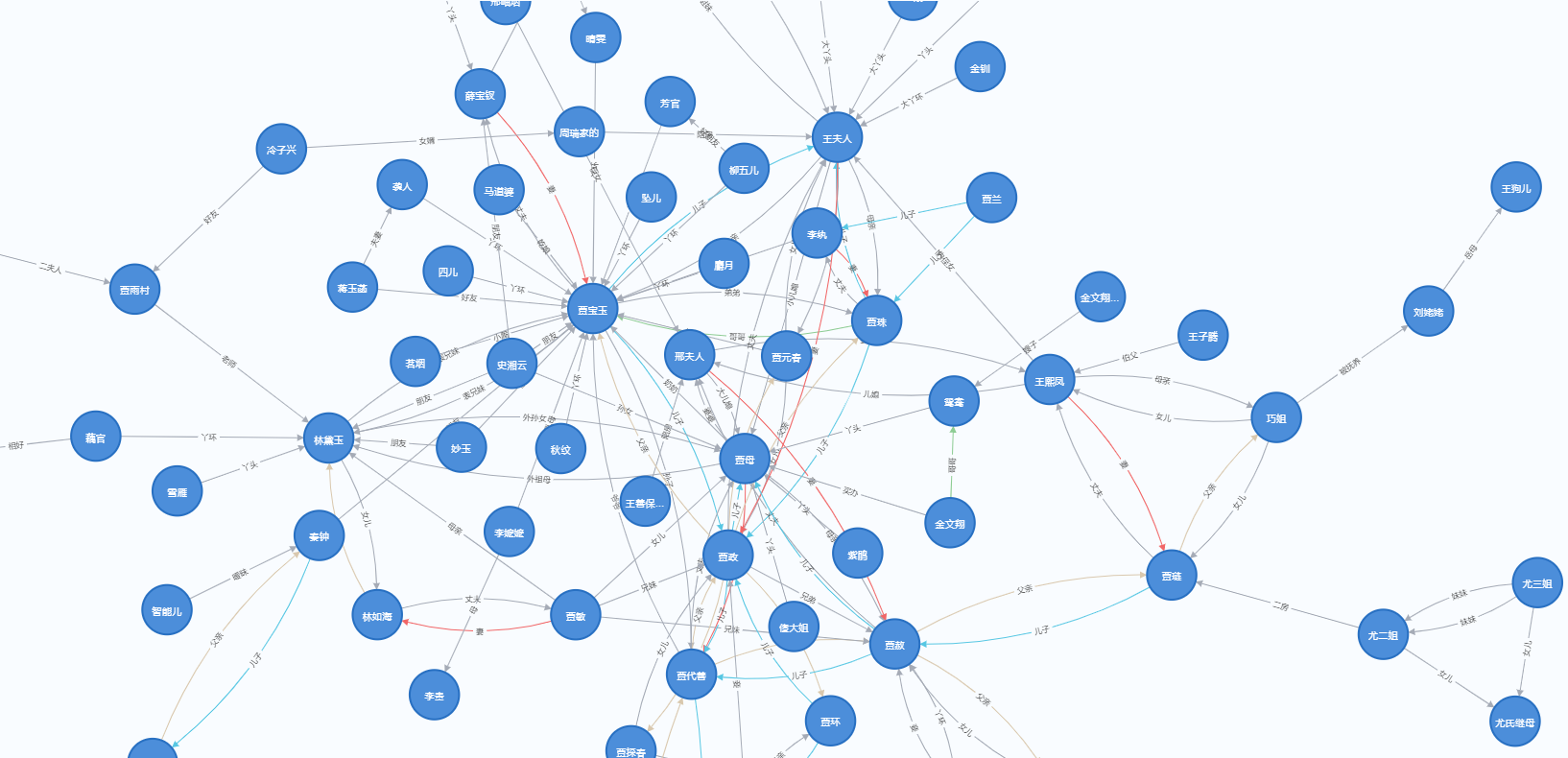
连接方式如下：

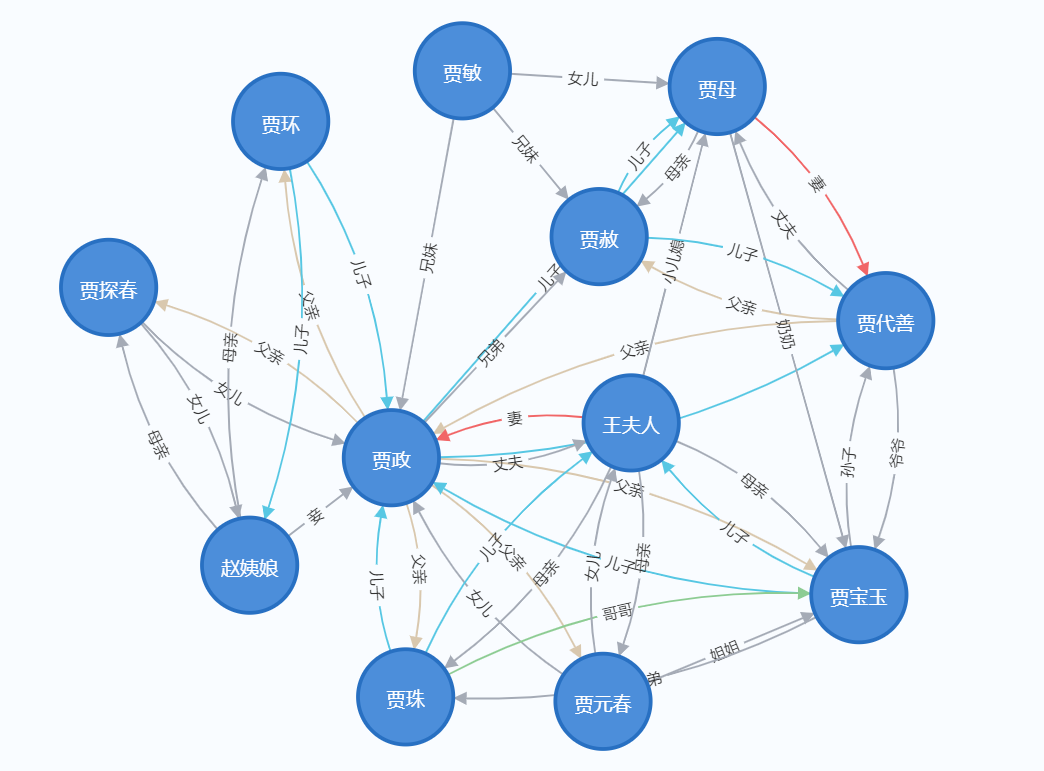


添加了txt读取的编码格式为utf-8。



#### 图谱效果

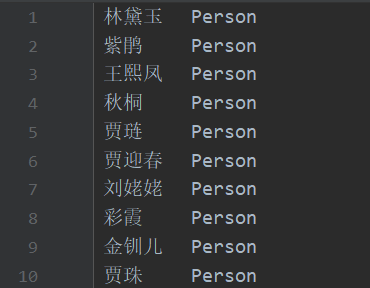




### 基于jieba分词的实体识别

1.2.2小节中已说明，整合大模型需要实体识别模块， 识别出用户问句中的关键实体，以便于将这些实体相关的三元组提交到大模型。

本次实验中使用的是基于jieba分词的实体识别，整理的自定义词典如下所示：



此方法的原理是将这个自定义词典添加到jiebe分词的词典中，jiebe分词便会按照该词典进行分词，如用户问句是“林黛玉的父亲是谁？”分词时，“林黛玉”会作为一个词切分出来。关键代码如下所示：

1. # 初始化实体识别字典
2. dict\_path = os.path.join(BASE\_DIR,'web/data/dict.txt')
3. word\_dict = [ i.strip('\n') **for** i **in** open(dict\_path,'r',encoding='utf-8').readlines()]
4. # 实体名称映射到实体类型
5. entity2type = {}
6. **for** word **in** word\_dict:
7. tmp\_list = word.split('\t')
8. **if** len(tmp\_list)==2:
9. # 将词典添加到jieba分词的自定义词典中
10. jieba.add\_word(tmp\_list[0])
11. entity2type[tmp\_list[0]] = tmp\_list[1]
13. # 通过自定义词典获取实体
14. **def** get\_enyity(input\_str):
15. word\_cut\_dict = []
16. # 对用户问句进行分词
17. **for** word **in** jieba.cut(input\_str.strip()):
18. **if** word **in** entity2type:
19. word\_cut\_dict.append(word)
20. **return**  set(word\_cut\_dict)

### 问答

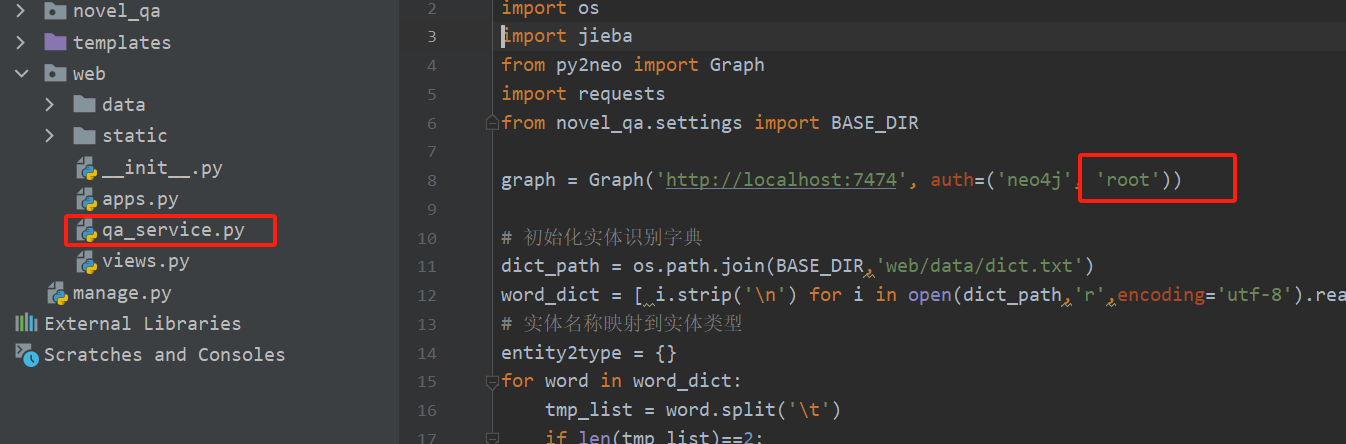
#### 项目结构说明



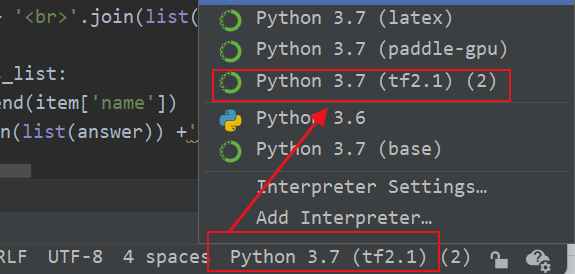
#### Pycharm设置

因为pycharm社区版不支持django项目，所以请安装专业版，安装包会同步发送。运行代码前，进行如下操作：

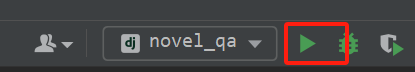
1. 更改qa\_service中数据库密码。



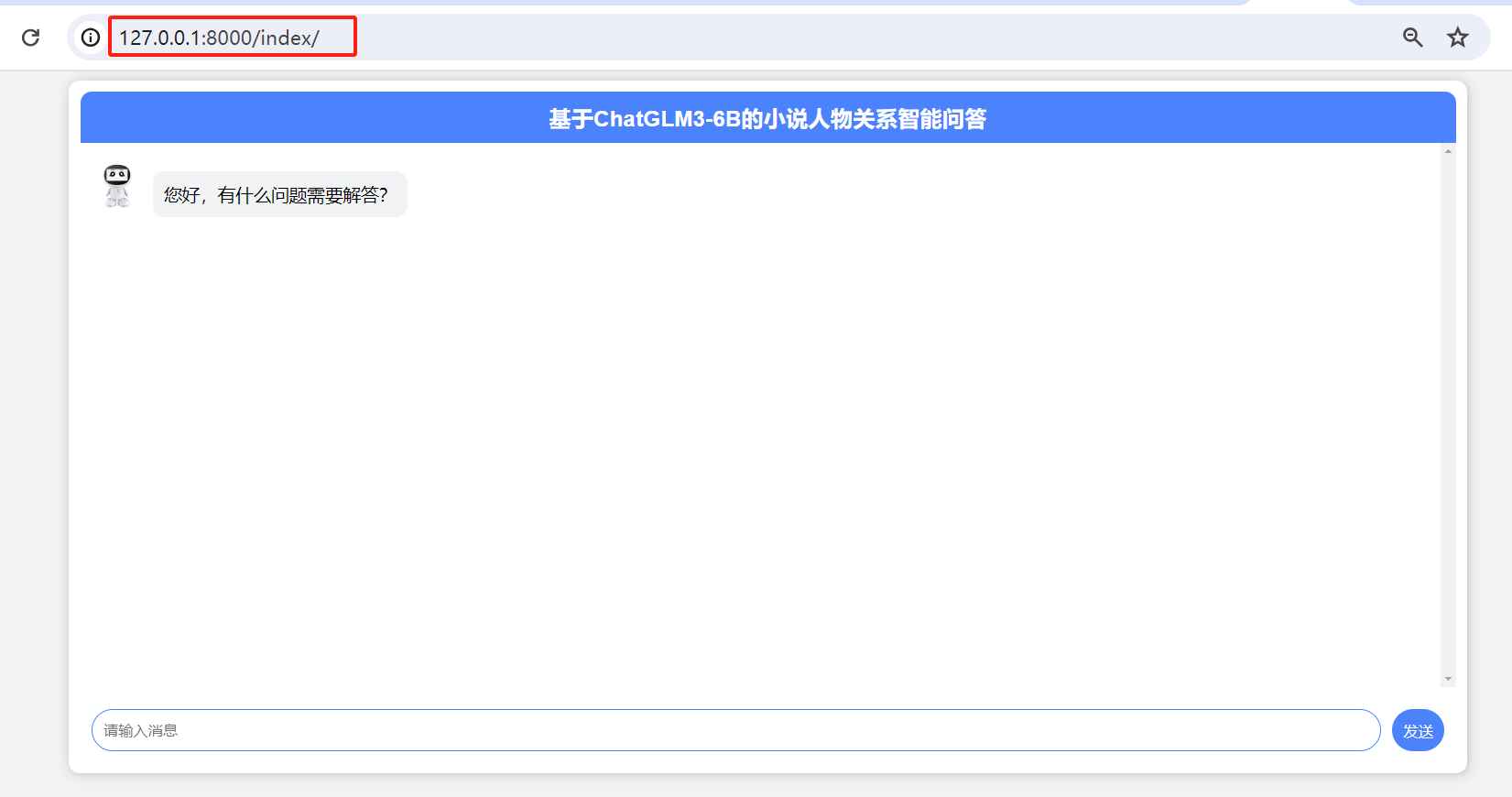
1. Pycharm右下角，选择解释器位置，你的包安装在哪，就选择哪，没有解释器的话先add interpreter:



1. 启动项目，点击右上角的django启动图标



1. 浏览器访问，http://127.0.0.1:8000/index/



#### 问答流程

1. 前端发起请求时，ajax请求会携带用户的问题到后端
2. 对问题进行实体识别
   1. 如果识别到实体，则拼接查询语句，到知识库中查询相关三元组，将三元组拼接到提示词中，提交给大模型。
   2. 如果未识别到实体，则直接将问题提交至大模型。
3. 从大模型获取返回结果后，返回前端展示。

#### 问答效果

