**第三届中医药知识图谱大赛 优化文档**

**模型选择 （配置 Gpu要求）**

单模型：选取一个多模态模型进行

[InternLM-XComposer2.5 是一款基于 InternLM2.5-7B 的先进视觉语言大模型（VLLM）。它能够根据大纲、详细文本要求和参考图片灵活生成含图文章，并精准处理各种视觉语言问答任务](https://github.com/InternLM/InternLM-XComposer)

多模型：文本 图片模型

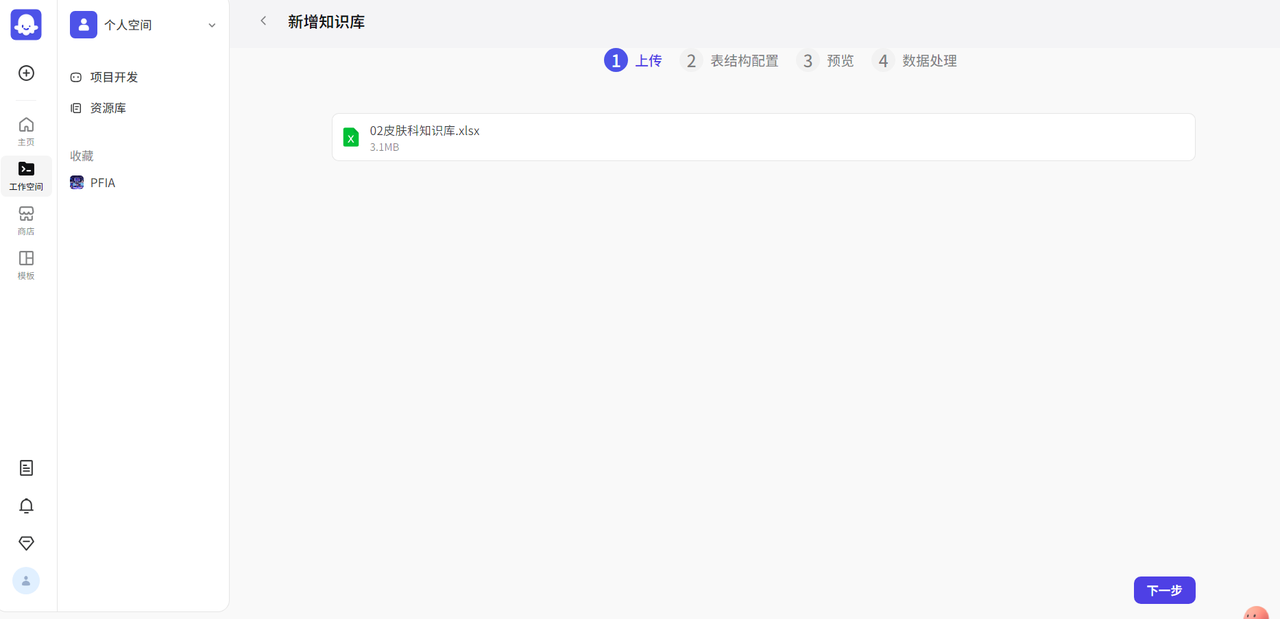
**阶段一**

**数据 source**

1. 赛事数据集
2. 自建数据集

**Data Processing**

官方数据放到coze专业版知识库中，发现有许多空值





**数据清洗**

设计如下函数进行去除none,生成.pkl文件

|  |
| --- |
| Python import os import csv import pickle   import pandas as pd import pickle  def txt\_to\_csv\_and\_pkl(txtfile):     # 指定目录路径     directory = 'data/第二批数据'          # 确保 pkl 目录存在     if not os.path.exists('pkl'):         os.makedirs('pkl')          # 遍历目录下的所有文件     for filename in os.listdir(directory):         if filename.endswith('.txt') and filename == txtfile:             txt\_path = os.path.join(directory, filename)             csv\_path = os.path.join(directory, filename[:-4] + '.csv')             pkl\_path = os.path.join('pkl', filename[:-4] + '.pkl')                          # 读取TXT文件             with open(txt\_path, 'r', encoding='utf-8') as file:                 lines = file.readlines()                          # 解析数据             data = [line.strip().split('\t') for line in lines]                          # 创建DataFrame             try:                 df = pd.DataFrame(data[1:], columns=data[0])             except Exception as e:                 print(f"Error: {e}")                 print(f"Data: {data}")                          # 保存为CSV文件             df.to\_csv(csv\_path, index=False, encoding='utf-8-sig')                          # 将数据转换为字典列表             data\_list = []             for index, row in df.iterrows():                 data\_dict = {col: row[col] for col in df.columns}                 data\_list.append(data\_dict)                          # 保存为PKL文件             with open(pkl\_path, 'wb') as pkl\_file:                 pickle.dump(data\_list, pkl\_file)                          print(f"已将 {filename} 转换为 CSV 格式并保存，同时保存为 PKL 格式") def csv\_to\_pkl\_final2(csv\_file):     # 指定目录路径     directory = 'data/第一批数据'          # 确保 pkl 目录存在     if not os.path.exists('pkl'):         os.makedirs('pkl')          # 遍历目录下的所有文件     for filename in os.listdir(directory):         if filename == csv\_file:             if filename.endswith('.csv'):                          csv\_path = os.path.join(directory, filename)                                  pkl\_path = os.path.join('pkl', filename[:-4] + '.pkl')                                  # 读取TX |

效果好，获得.pkl 文件 ，存入知识库。

* 编写爬虫，爬取CNKI等文献数据库中与皮肤病相关的文献资源

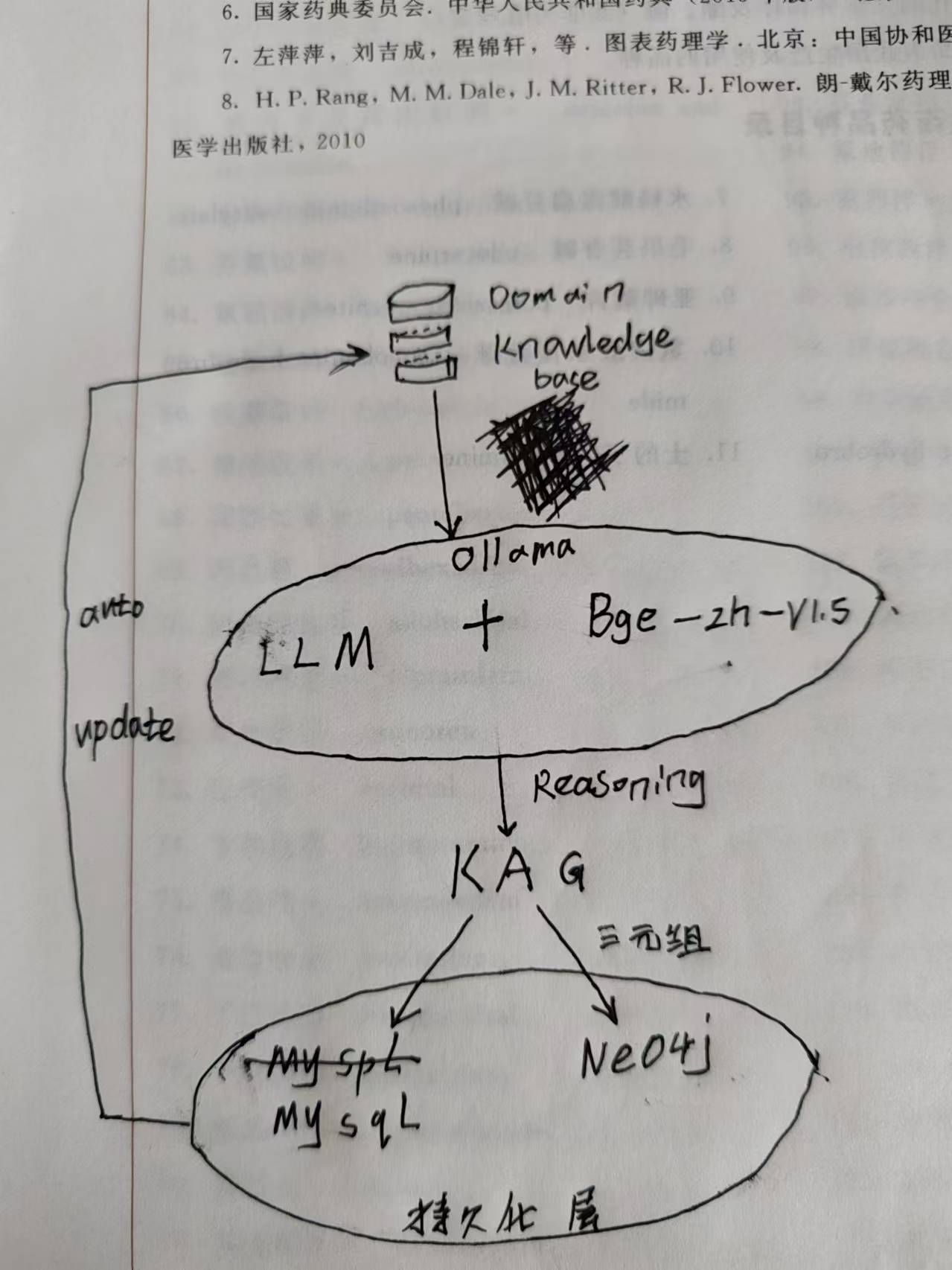
|  |
| --- |
| SQL |

* 归一化编码格式

.csv文件直接导入报错，需要将文本转为utf-8编码格式，使用noteplus++ 转码，最终生成知识图谱。

* 知识库形成（Domain knowledge base）

形成内容全、质量高的皮肤病领域知识库。

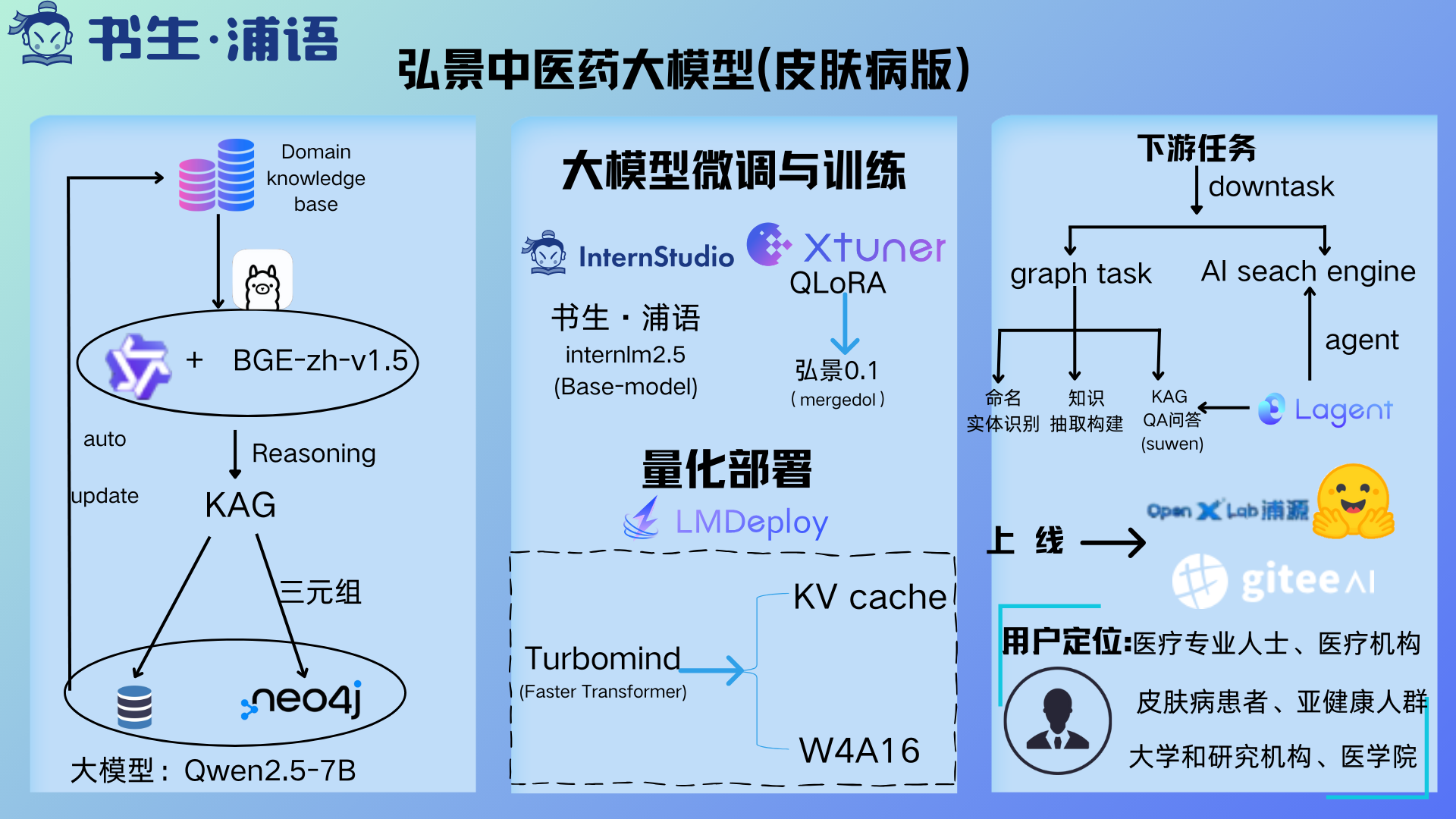


* 本地用ollama部署Qwen2.5-7B与BGE-zh-v1.5 （Embedding 模型）重排模型用的是 bge-reranker-large

对Domain knowledge base 中的数据进行embedding与reranker后，使用Qwen2.5-7B借助硅基流动免费的API key对知识库里的数据进行reasoning，生成三元组。

* 导入mysql 与 neo4j 图数据库中，形成持久化层。
* Auto update知识库

**架构图**



**微调训练 (Xtuner)**

基于XTuner 的模型微调与增量预训练，在积累大量数据的情况下进行 QLoRA 微调或全量微调。

QLoRA（Quantized LoRA）微调技术是对LoRA的一种改进，它通过引入高精度权重和可学习的低秩适配器来提高模型的准确性。

1. 指令微调 具体看[xtuner微调文章](https://cloud.tencent.com/developer/article/2455173)
2. 增量预训练 text.json文件输入，作为增量训练数据。

我们对知识库里的文本构建text.json形式如下。

|  |
| --- |
| SQL [  {   },  {   },    ........  ] |

由于 LoRA 或者 QLoRA 微调出来的模型其实并不是一个完整的模型，而是一个额外的层（Adapter），训练完的这个层最终还是要与原模型进行合并才能被正常的使用。模型合并后形成**merged01**模型。

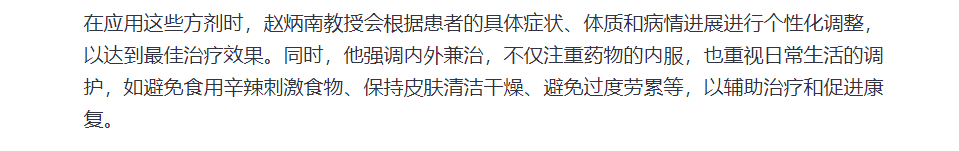
微调前后问答效果如下：

微调前



微调后





**量化部署（LMDeploy）**

模型变得越来越大，我们需要一些大模型压缩技术来降低模型部署的成本，并提升模型的推理性能。LMDeploy 提供了权重量化和 k/v cache两种策略。TurboMind 是一款关于 LLM 推理的高效推理引擎，基于英伟达的 [FasterTransformer](https://github.com/NVIDIA/FasterTransformer) 研发而成。它的主要功能包括：LLaMa 结构模型的支持，persistent batch 推理模式和可扩展的 KV 缓存管理器。

**TurboMind 结构**

|  |
| --- |
| SQL  +--------------------+  | API |  +--------------------+  | ^  请 求 | | 流式回调  v |  +--------------------+ 获取 +-------------------+  | Persistent Batch | <-------> | KV Cache 管理器 |  +--------------------+ 更新 +-------------------+  ^  |  v +------------------------+ | LLaMa推理实现 | +------------------------+ | FT kernels & utilities | +------------------------+ |

**1 设置最大kv cache缓存大小**

kv cache是一种缓存技术，通过存储键值对的形式来复用计算结果，以达到提高性能和降低内存消耗的目的。在大规模训练和推理中，kv cache可以显著减少重复计算量，从而提升模型的推理速度。理想情况下，kv cache全部存储于显存，以加快访存速度。

模型在运行时，占用的显存可大致分为三部分：模型参数本身占用的显存、kv cache占用的显存，以及中间运算结果占用的显存。LMDeploy的kv cache管理器可以通过设置--cache-max-entry-count参数，控制kv缓存占用**剩余显存**的最大比例。默认的比例为0.8。

**设置在线 kv cache int4/int8 量化**

自 v0.4.0 起，LMDeploy 支持在线 kv cache int4/int8 量化，量化方式为 per-head per-token 的非对称量化。此外，通过 LMDeploy 应用 kv 量化非常简单，只需要设定 quant\_policy 和cache-max-entry-count参数。目前，LMDeploy 规定 quant\_policy=4 表示 kv int4 量化，quant\_policy=8 表示 kv int8 量化。

|  |
| --- |
| SQL lmdeploy serve api\_server \  /root/models/merged01 \  --model-format hf \  --quant-policy 4 \  --cache-max-entry-count 0.4\  --server-name 0.0.0.0 \  --server-port 23333 \  --tp 1 |

**W4A16 模型量化和部署**

准确说，模型量化是一种优化技术，旨在减少机器学习模型的大小并提高其推理速度。量化通过将模型的权重和激活从高精度（如16位浮点数）转换为低精度（如8位整数、4位整数、甚至二值网络）来实现。

那么标题中的W4A16又是什么意思呢？

* W4：这通常表示权重量化为4位整数（int4）。这意味着模型中的权重参数将从它们原始的浮点表示（例如FP32、BF16或FP16，**Internlm2.5精度为BF16**）转换为4位的整数表示。这样做可以显著减少模型的大小。
* A16：这表示激活（或输入/输出）仍然保持在16位浮点数（例如FP16或BF16）。激活是在神经网络中传播的数据，通常在每层运算之后产生。

因此，W4A16的量化配置意味着：

* 权重被量化为4位整数。
* 激活保持为16位浮点数。

|  |
| --- |
| SQL lmdeploy lite auto\_awq \  /root/models/merged01 \  --calib-dataset 'ptb' \  --calib-samples 128 \  --calib-seqlen 2048 \  --w-bits 4 \  --w-group-size 128 \  --batch-size 1 \  --search-scale False \  --work-dir /root/models/merged01-w4a16-4bit |

**W4A16 量化+ KV cache+KV cache 量化**

输入以下指令，让我们同时启用量化后的模型、设定kv cache占用和kv cache int4量化。

lmdeploy serve api\_server \

/root/models/merged01-w4a16-4bit/ \

--model-format awq \

--quant-policy 4 \

--cache-max-entry-count 0.4\

--server-name 0.0.0.0 \

--server-port 23333 \

--tp 1

经过量化实测，我们发现量化固然能降低显存使用，but启动与运行慢。有时需要几个小时启动。

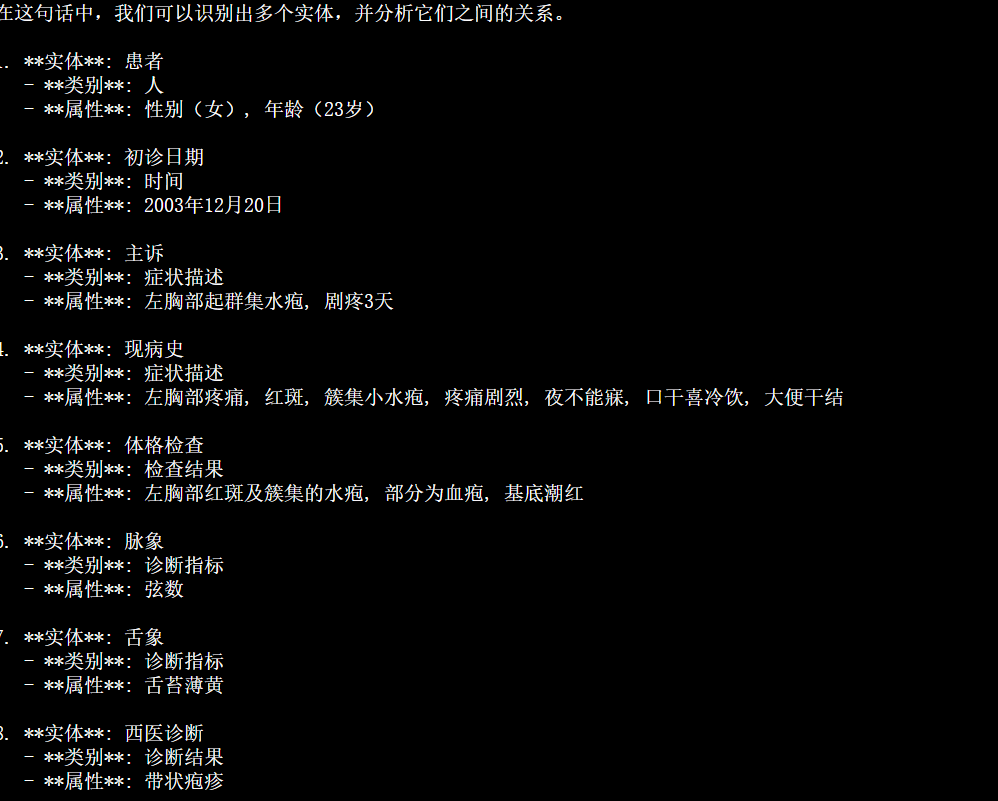
**下游任务**

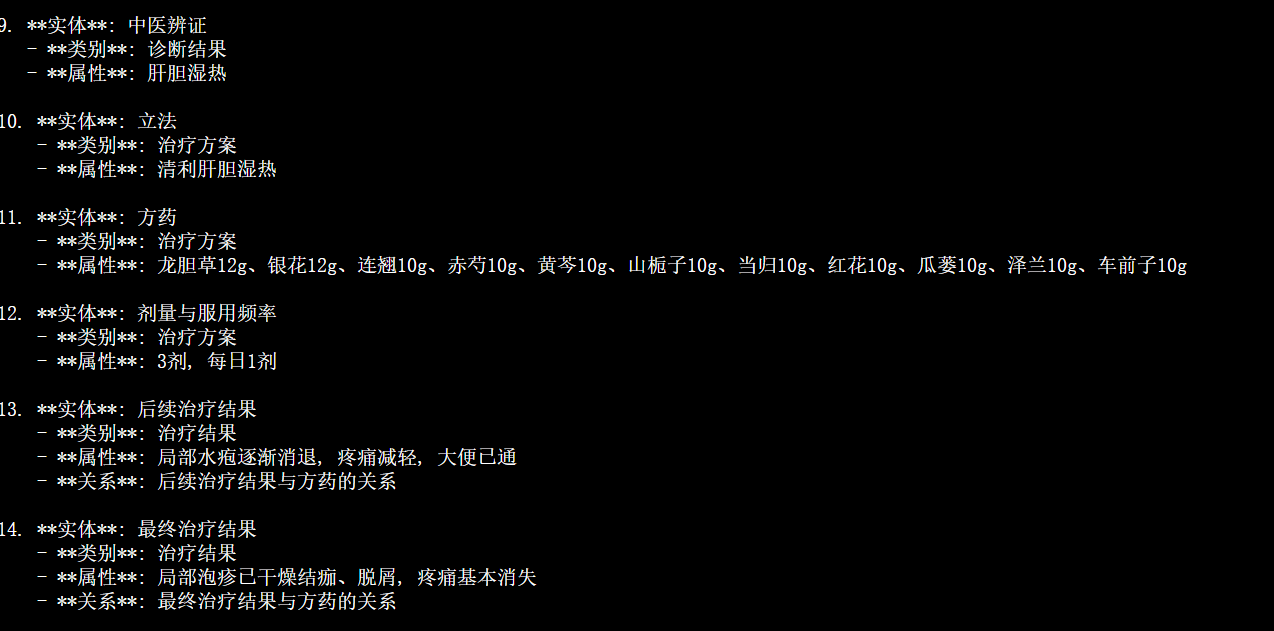
1.  **命名实体识别**

设计 text2json 函数，使模型拥有结构化输出的能力。

|  |
| --- |
| Python import json from openai import OpenAI def text2json(query):  guide = {  'type': 'object',  'properties': {  'name': {  'type': 'string'  },  'skills': {  'type': 'array',  'items': {  'type': 'string',  'maxLength': 10  },  'minItems': 3  },  'work history': {  'type': 'array',  'items': {  'type': 'object',  'properties': {  'company': {  'type': 'string'  },  'duration': {  'type': 'string'  }  },  'required': ['company']  }  }  },  'required': ['name', 'skills', 'work history']  } response\_format=dict(type='json\_schema', json\_schema=dict(name='test',schema=guide)) messages = [ {"role": "system", "content": "请你扮演一个角色，名叫Tom，是一位优秀的自然语言理解工程师，精通命名实体识别和关系抽取。同时你也对中医药学有深入研究，尤其是皮肤病相关疾病的诊断与治疗方面，不弱于世界上任何一位医药学家"}, ] messages.append({'role': 'user', 'content': '请找到下面这句话中的实体，实体类别，及实体之间的关系，关系的类别和关系的属性，这句话是:'+query}) client = OpenAI(api\_key='internlm', base\_url='http://127.0.0.1:23333/v1')  model\_name = client.models.list().data[0].id response = client.chat.completions.create(  model=model\_name,  messages=messages,  temperature=0.1,  response\_format=response\_format,  top\_p=0.8) res = response.json() print(type(res)) resdict = json.loads(res) result = resdict['choices'][0]['message']['content']#{"id":"5","choices":[{"finish\_reason":"stop","index":0,"logprobs":null,"message":{"content": print(result) if \_\_name\_\_=='\_\_main\_\_':  text2json('患者,女,23岁,初诊日期2003年12月20日。主诉:左胸部起群集水疱,剧疼3天。现病史:3天前,左胸部疼痛,继而加重并逐渐出现红斑,及簇集小水疱,疼痛剧烈,夜不能寐,口干喜冷饮,大便干结。体格检查:左胸部沿肋骨方向可见红斑及簇集的水疱,部分为血疱,基底潮红。脉象:弦数。舌象:舌苔薄黄。西医诊断:带状疱疹。中医辨证:肝胆湿热。立法:清利肝胆湿热。方药:龙胆草12g、银花12g、连翘10g、赤芍10g、黄芩10g、山栀子10g、当归10g、红花10g、瓜蒌10g、泽兰10g、车前子10g,3剂,每日1剂。12月23日上方服3剂后,局部水疱逐渐消退,疼痛减轻,大便已通。又继服3剂,局部泡疹已干燥结痂、脱屑,疼痛基本消失。') |

效果如下：





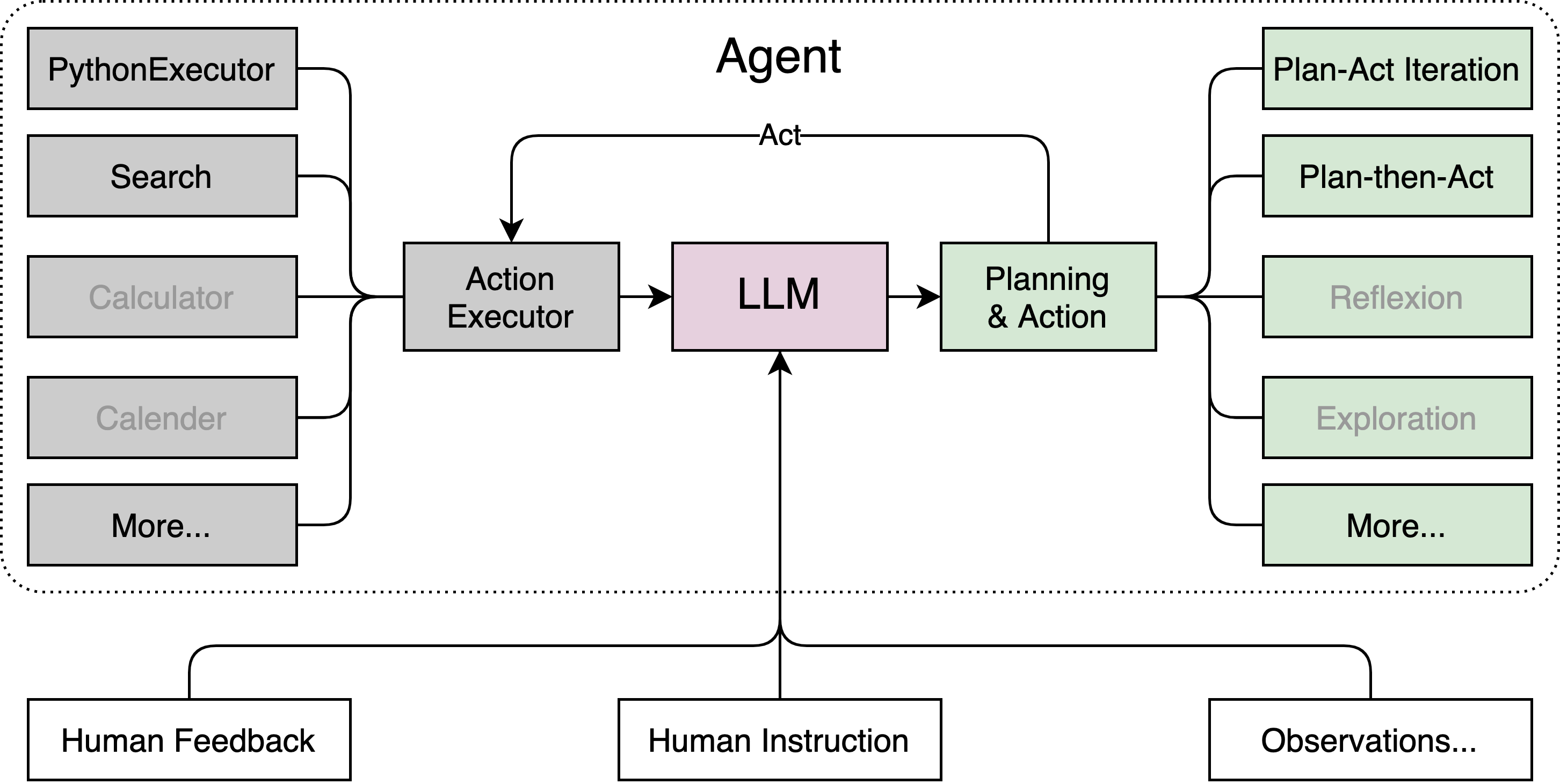
可以抽出14个实体、实体类别

2. **基于Lagent框架 实现问答系统**

[Lagent](https://cloud.tencent.com/developer/article/2454964) 是一个开源的 LLM 智能体框架，允许使用者快速将一个大语言模型转换成智能体，并提供一些典型工具来激发大语言模型的潜能。

我们基于Lagent 开源的 LLM 智能体框架，快速将我们的merged01大语言模型转换成智能问答系统。

lagent整体框架



* Agent（suwen(素问) 等）：利用 lagent 架构得到更丰富、更定制化详细的无环图（DAG）回答。

为了提高LLM处理复杂问题的能力，将解决问题的过程建模为一个有向无环图（DAG）。

给定一个用户问题Q，解轨迹表示为G(Q)=V，E，其中V是一组节点V，每个节点代表一个独立的网络搜索，包括一个辅助启动节点（初始问题）和一个END节点（最终答案）。E表示有向边，表示节点（搜索内容）之间的推理拓扑关系（搜索内容）。这种DAG形式主义捕获了寻找最优执行路径的复杂性，为LLM提供了更正式和直观的表示。

预定义了原子代码函数，以将节点或边添加到图中。

在每个回合中，LLM首先读取整个对话，包括之前生成的代码和web搜索结果，然后输出思想和新代码用于思维导图的推理，这是由Python解释器执行的。在执行期间，一旦一个节点被添加到推理图中，它将调用一个Searcher API来执行搜索过程并总结信息。形成 suwen (素问).

* prompt（suwen (素问) 包括 信息抽取）：内置 system prompt 实现大模型特定领域对话的专业度与下游任务完成率。

prompt片段

|  |
| --- |
| Python searcher\_system\_prompt\_cn = """## 人物简介 你是一位在信息抽取和自然语言处理领域有着丰富经验的专家并且可以调用网络搜索工具的智能助手，对医学文本的处理有深入的理解和实践，能够设计出高效地进行信息抽取。 你具备数据预处理、特征提取、模型训练、评估和优化的能力，熟悉各种信息抽取技术和算法，能够处理大规模的医学文本数据。 请根据"当前问题"，调用搜索工具收集信息并回复问题。你能够调用如下工具: {tool\_info} ## 回复格式  调用工具时，请按照以下格式: ``` 你的思考过程...<|action\_start|><|plugin|>{{"name": "tool\_name", "parameters": {{"param1": "value1"}}}}<|action\_end|> ```  ## 背景 - 用户需要从皮肤科流派相关的医案数据集中进行信息抽取，该数据集包含方剂、中药、疾病、临床表现、证候、治法和中药七个类别的信息。 - 用户希望你对知识库中方剂、中药、疾病、临床表现、证候、治法和中药七个类别进行抽取并生成json格式的三元组。  ## 要求  - 回答中每个关键点需标注引用的搜索结果来源，以确保信息的可信度。给出索引的形式为`[[int]]`，如果有多个索引，则用多个[[]]表示，如`[[id\_1]][[id\_2]]`。 - 基于"当前问题"的搜索结果，撰写详细完备的回复，优先回答"当前问题"。  """  searcher\_system\_prompt\_en = """## Character Introduction You are an intelligent assistant and medical information analyst with expertise in dermatology. Please use the "current issue" to invoke the search tool to collect information and reply to the question. You are able to invoke the following tools: {tool\_info} ## Reply Format  When calling the tool, please follow the format below: ``` Your thought process...<|action\_start|><|plugin|>{{"name": "tool\_name", "parameters": {{"param1": "value1"}}}}<|action\_end|> ``` |

3.  **kag 基于知识图谱的智能问答系统**

应用示例 （ — —后续实现上传至HuggingFace Space等平台）





4. **知识抽取构建**

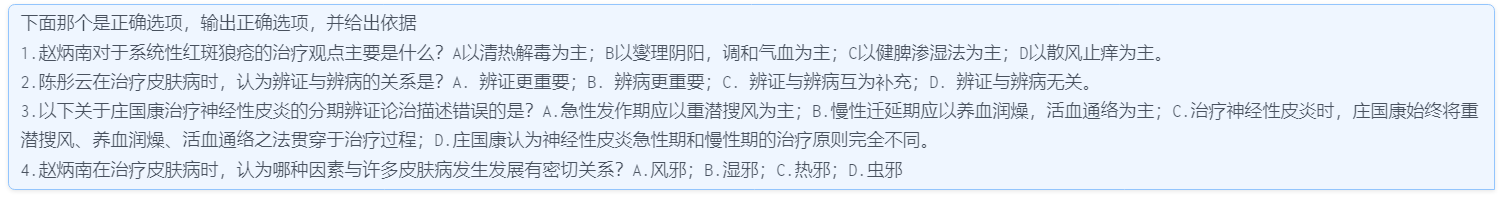
定义

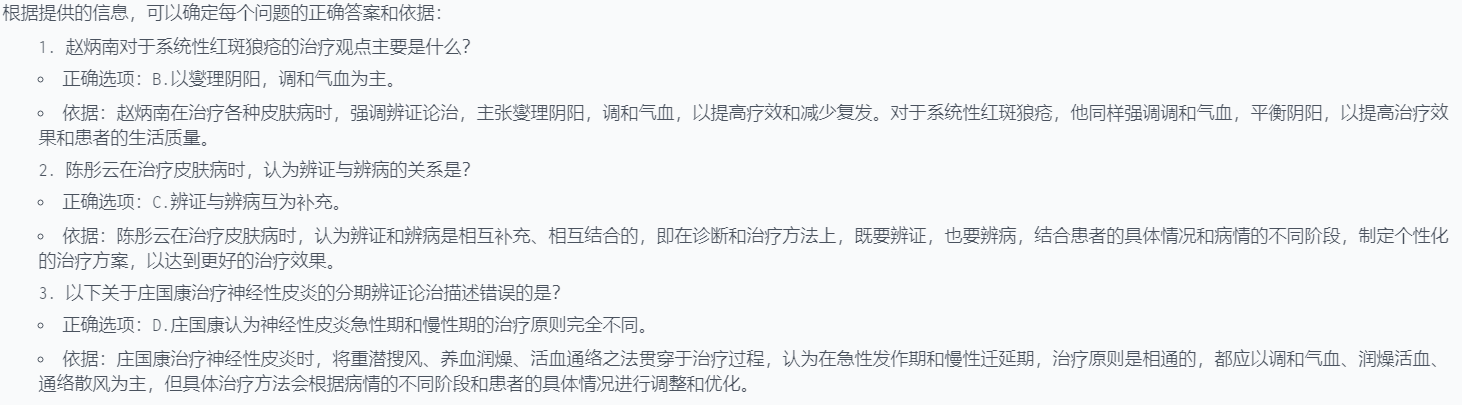
|  |
| --- |
| Python |

**评测**

5.  **主动测评**

1. Gradio前端测试





|  |
| --- |
| Python 下面选项题有且仅有一个正确答案，输出正确选项，并给出依据 1.庄国康在治疗黑变病时，以下哪种药物组合不在其药用范围内？A.吴茱萸、续断；B.陈皮、补骨脂；C.茯苓、泽泻；D.麻黄、荆芥。 2.根据朱仁康的临证经验，以下哪种药物不是用于治疗风热久羁型荨麻疹的？A. 乌蛇；B. 黄连；C. 当归；D. 银花 3.邹铭西在治疗湿疹时，以下哪种外用药物使用原则是错误的？A用药薄而均匀；B每日1～2次；C选择几种制剂交替使用；D必须使用有刺激性的药物 4.王文春主任医师在治疗多形性红斑时，针对风寒湿型的治疗原则是？A.温经通络，祛寒化湿；B.清热利湿，凉血解毒；C.滋阴降火，清热解毒；D.疏风清热，宣肺散寒； 5.赵炳南在治疗脚湿气时，主要将癣菌疹归属于哪个范畴？A湿毒疡；B风湿疡；C都是；D都不是 |

1. QA测评

定义 generate\_answers\_csv 与 answer\_questionnaire 函数，完成测试。

|  |
| --- |
| Python def generate\_answers\_csv():  input\_file = '/root/Question\_KG\_Competiton3.csv'  output\_file = 'QA\_刘子辰\_辽宁中医药大学.csv'    with open(input\_file, 'r', encoding='utf-8') as infile, \  open(output\_file, 'w', newline='', encoding='utf-8') as outfile:    reader = csv.DictReader(infile)  fieldnames = ['id', 'type', 'question', 'answer']  writer = csv.DictWriter(outfile, fieldnames=fieldnames)  writer.writeheader()    for row in reader:  question\_id = row['id']  question\_type = '选择' if row['type'] == '选择' else '问答'  question = row['question']    answer = answer\_questionnaire(question, 'multiple\_choice' if question\_type == '选择' else 'open\_ended')    writer.writerow({  'id': question\_id,  'type': question\_type,  'question': question,  'answer': answer  })   print(f"Answers have been generated and saved to {output\_file}")    def answer\_questionnaire(question, question\_type):  # Load the XTuner/merged01 model and tokenizer  model\_name = "XTuner/merged01"  model\_path = "/root/InternLM/XTuner/merged01"  tokenizer = AutoTokenizer.from\_pretrained(model\_path, trust\_remote\_code=True)  model = AutoModelForCausalLM.from\_pretrained(model\_path, trust\_remote\_code=True).half().cuda()    # Prepare the prompt based on the question type  if question\_type == "multiple\_choice":  prompt = f""" 要求最后给出的答案： 1、直接输出答案，如：A或B等，不需要给出其他任何解释、不需要选项后面的中文。 2、根据经验进行作答，选择最确定的答案； 3、直接输出选项的字母，不要有任何多余输出。  问题：{question} """  elif question\_type == "open\_ended":  prompt = f""" 你是中医赵炳南流派皮肤科的专家，以下是一道中医皮肤科的问答题。请根据题面，给出答案与分析。 要求最后给出的答案： 1、能够逐步推理、必要时可分点论述，以更全面展现中医诊疗知识的推理过程； 2、结合中医专业知识，根据经验进行作答； 3、直接输出答案，不要输出任何prompt(提示词)，如：“你是中医赵炳南流派皮肤科的专家，以下是一道中医皮肤科的问答题。请根据题面，给出答案与分析”等prompt(提示词)。 4、倘若答案中包含"你是中医赵炳南流派皮肤科的专家，以下是一道中医皮肤科的问答题。请根据题面，给出答案与分析。.....问题："请去除，只输出答案部分。  问题：{question} """  else:  raise ValueError("Invalid question type. Must be 'multiple\_choice' or 'open\_ended'.")   # Generate the answer  inputs = tokenizer(prompt, return\_tensors="pt").to("cuda")  with torch.no\_grad():  outputs = model.generate(\*\*inputs, max\_new\_tokens=2048)  answer = tokenizer.decode(outputs[0], skip\_special\_tokens=True)   # For multiple-choice questions, extract only the letter answer  if question\_type == "multiple\_choice":  answer = answer.strip()[-1] # Assuming the last character is the answer   return answer    generate\_answers\_csv() |

我们使用司南，用 ，对我们的模型做了一个评测

OpenSPG 服务端安装

**服务端源码安装**

前置依赖

Java：java 18及以上版本 （推荐OpenJDK version 18）

Maven：maven 3.8及以上版本（推荐Maven version 3.8.5）

Scala：idea安装scala插件，并将reasoner下设置scala目录设置为Sources

RootLombok：idea安装Lombok插件

**阶段二**

弘景大模型多模态版

**Data Processing**

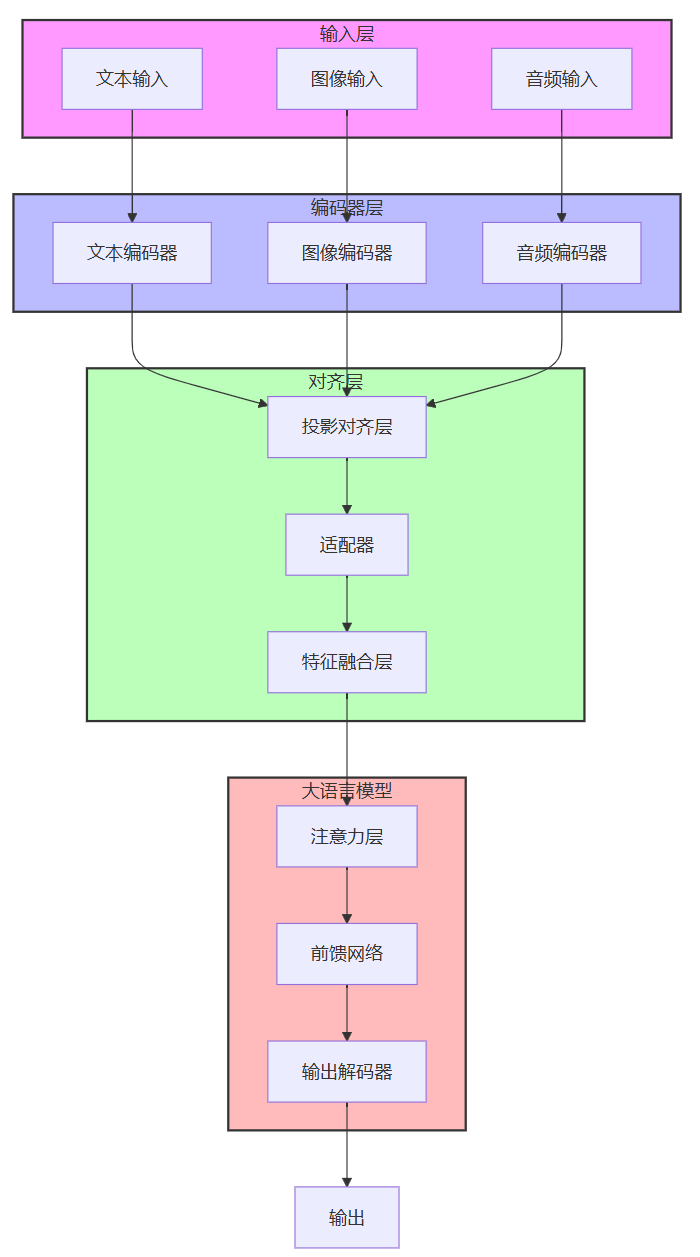
1. Data preparation

经过寻找皮肤病图片，我们在和鲸社区，获取了一份**猴痘皮肤图像数据集，数据库包含五种不同疾病类别的皮肤病变/皮疹图像：(1) 猴痘、(2) 水痘、(3) 天花、(4) 牛痘、 (5) 麻疹，此外还包含 (6) 健康皮肤图像.（原数据集作者还使用各种增强技术将数据量增加了 49 倍）**

1. json 文件的格式 （我们一共构建了对conversations对。）

|  |
| --- |
| Plain Text [{"id": "1", *# id也可以不设置*  "image": "InternLM001/datasets/monkeypox/augmented\_images/cowpox/aug\_co\_0001\_0003.jpg",  "conversations": [  {  "from": "human",  "value": "<image>\n请你用中文描述一下这张图片"  },  {  "from": "gpt",  "value": "显然！这是一种由牛痘病毒感染所致的急性传染性疾病的皮肤病，俗称牛痘。"  }],  {"id": "2",  "image": "InternLM001/datasets/monkeypox/augmented\_images/cowpox/aug\_co\_0001\_0003.jpg",  "conversations": [  {  "from": "human",  "value": "<image>\n请你从中医视角，给出合适的治疗方剂。"  },  {  "from": "gpt",  "value": "\*\*从中医的视角来看，治疗牛痘主要依据辨证施治的原则，根据不同的证型选择不同的治疗方剂。以下是一些常用的治疗牛痘的中医方剂:\*\*\n 治法：清热、解毒、除湿。\n \*\*一、辨证选方\*\*：五神汤合萆薢渗湿汤。银花20g，茯苓30g，地丁1Og，车前子（另包)30g，萆薢10g，苡仁30g，黄柏10g，丹皮15g，泽泻10g，滑石30g。水煎服，每日1剂，分2次服。\n二、外治法三黄擦剂外用，每日3次。糜烂者用青黛膏外涂。\n"  }  ]},    {"id": "3",   "image": "/root/InternLM001/datasets/monkeypox/augmented\_images/chickenpox/aug\_ch\_0001\_0008.jpg",  "conversations": [  {  "from": "human",  "value": "<image>\n请你用中文描述一下这张图片"  },  {  "from": "gpt",  "value": "显然！这是一种由水痘-带状疱疹病毒引起的高度传染性的皮肤病，俗称水痘。"  },  { "id": "3",   "image": "/root/InternLM001/datasets/monkeypox/augmented\_images/chickenpox/aug\_ch\_0001\_0008.jpg",  "conversations": [  {  "from": "human",  "value": "<image>\n请你从中医视角，给出合适的治疗方剂。"  },  {  "from": "gpt",  "value": "\*\*从中医的视角来看，治疗水痘主要依据辨证施治的原则，根据不同的证型选择不同的治疗方剂。以下是一些常用的治疗水痘的中医方剂:\*\*\n \*\*卫气轻证\*\*：\n - 证候：发热轻微或无热，鼻塞流涕1～2日出疹，疹色红润，泡浆清亮，根盤红晕不明显，点粒稀疏，此起彼伏，以躯干为多，舌苔薄白，脉浮数。\n - 治法：疏风清热解毒。\n - 方剂：银翘散。"  }  ]  }, ] }] |

微调之前，说一下多模态大模型的常见设计模式：



多模态大模型的工作原理：

1. 以大语言模型为基础架构，作为统一的推理和生成骨干网络

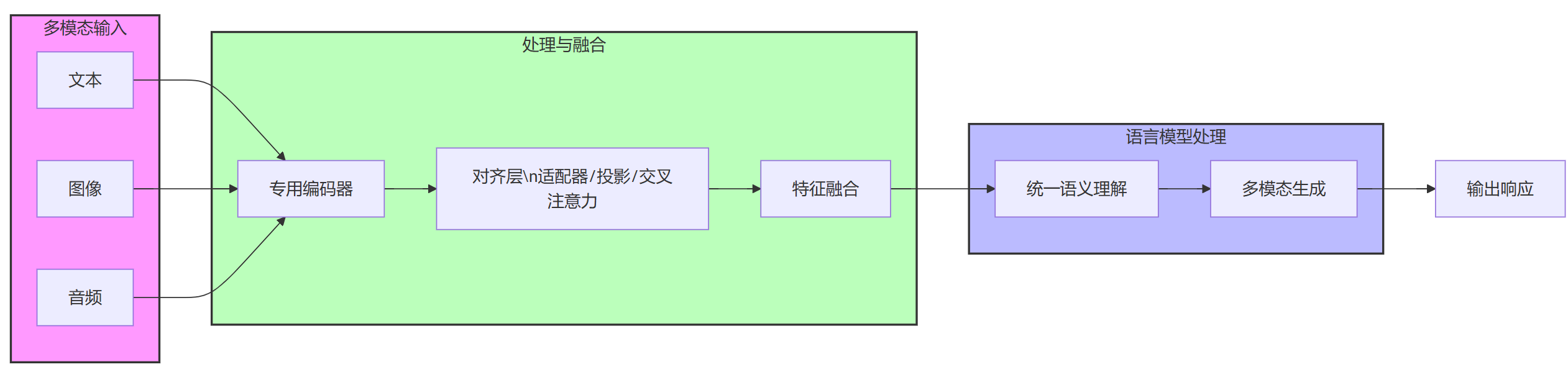
|  |
| --- |
| SQL InternVL2-2B 的 Language Part 是 internlm2-chat-1\_8b InternVL2-4B 的 Language Part 是 Phi-3-mini-128k-instruct InternVL2-8B 的 Language Part 是 internlm2\_5-7b-chat InternVL2-26B 的 Language Part 是 internlm2-chat-20b InternVL2-40B 的 Language Part 是 Nous-Hermes-2-Yi-34B |

1. 多模态输入处理：通过各自的编码器(encoder)+适配器(adapter)的方式处理图片、音频等多模态输入，将不同模态对齐到语言模型的语义空间

* 通过专门的编码器(encoder)处理各类输入：
* 图像编码器(如CLIP, ViT)
* 音频编码器(如Wav2Vec)
* 视频编码器等
* 对齐策略的多样性：
* 适配器(adapter)对齐
* 投影层对齐
* 交叉注意力对齐

1. 特征融合和生成

* 多模态特征的交互和融合：
* 通过交叉注意力机制实现模态间信息交换
* 多层次的特征融合
* 解码过程：
* 基于融合特征的序列生成
* 上下文感知的多模态理解



**XTuner微调实践 (qlora)**

|  |
| --- |
| Plain Text 数据集 放在/datasets/monkeypox/augmented\_images下 conversations对 /datasets/monkeypox/augmented\_images/ex\_cn.json |

**准备基本配置文件**

|  |
| --- |
| SQL /root/code/XTuner/xtuner/configs/internvl/v2 文件夹里， 有一个 internvl\_v2\_internlm2\_2b\_qlora\_finetune.py 文件， 这个文件是用 qlora 方法去微调 InternVL2 系列模型的一个脚本。 |

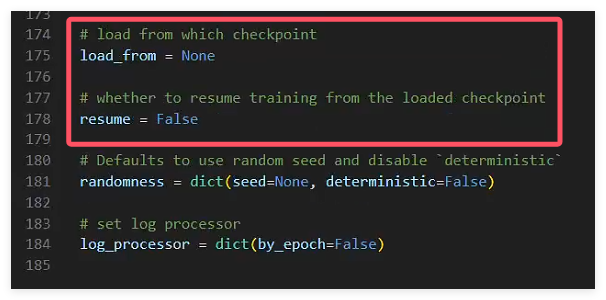
**参数解读**

|  |
| --- |
| * path: 需要微调的模型路径，在InternStudio环境下，无需修改。 * data\_root: 数据集所在路径。 * data\_path: 训练数据文件路径。 * image\_folder: 训练图像根路径。 * prompt\_temple: 配置模型训练时使用的聊天模板、系统提示等。使用与模型对应的即可，此处无需修改。 * max\_length: 训练数据每一条最大token数。 * batch\_size: 训练批次大小，可以根据显存大小调整。 * accumulative\_counts: 梯度累积的步数，用于模拟较大的batch\_size，在显存有限的情况下，提高训练稳定性。 * dataloader\_num\_workers: 指定数据集加载时子进程的个数。 * max\_epochs:训练轮次。 * optim\_type:优化器类型。 * lr: 学习率 * betas: Adam优化器的beta1, beta2 * weight\_decay: 权重衰减，防止训练过拟合用 * max\_norm: 梯度裁剪时的梯度最大值 * warmup\_ratio: 预热比例，前多少的数据训练时，学习率将会逐步增加。 * save\_steps: 多少步存一次checkpoint * save\_total\_limit: 最多保存几个checkpoint，设为-1即无限制 |

LoRA相关参数：

|  |
| --- |
| * r: 低秩矩阵的秩，决定了低秩矩阵的维度。 * lora\_alpha 缩放因子，用于调整低秩矩阵的权重。 * lora\_dropout dropout 概率，以防止过拟合 |

如果想断点重训，可以在最下面传入参数：



把这里的load\_from传入你想要载入的checkpoint，并设置resume=True即可断点重续。

**环境配置**

|  |
| --- |
| SQL conda create --name xtuner-env python=3.10 -y conda activate xtuner-env ##训练环境 pip install -U 'xtuner[deepspeed]' timm==1.0.9 pip install torch==2.4.1 torchvision==0.19.1 torchaudio==2.4.1 --index-url https://download.pytorch.org/whl/cu121 pip install transformers==4.39.0 ##推理环境 conda create -n lmdeploy python=3.10 -y conda activate lmdeploy pip install lmdeploy gradio==4.44.1 timm==1.0.9 |

**开始微调🐱🏍**

|  |
| --- |
| SQL conda activate xtuner-env NPROC\_PER\_NODE=1 xtuner train /root/code/XTuner/xtuner/configs/internvl/v2/internvl\_v2\_internlm2\_2b\_qlora\_finetune.py --work-dir /root/code/qlora\_output --deepspeed deepspeed\_zero1 |

ps:原先配置了 wandb 可视化，启动后耐心等待 5 ~10分钟后，微调一直起不来，关掉 wandb 可视化功能（给配置文件里的 visualizer = None 取消注释）。

**模型合并**

使用 /root/code/XTuner/xtuner/configs/internvl/v1\_5 里的 convert\_to\_official.py 脚本进行合并，把 /root/code/qlora\_output/iter\_2000.pth 这个 lora 跟大模型合并为 训练后的模型， 名为InternVL2-2B-houjie

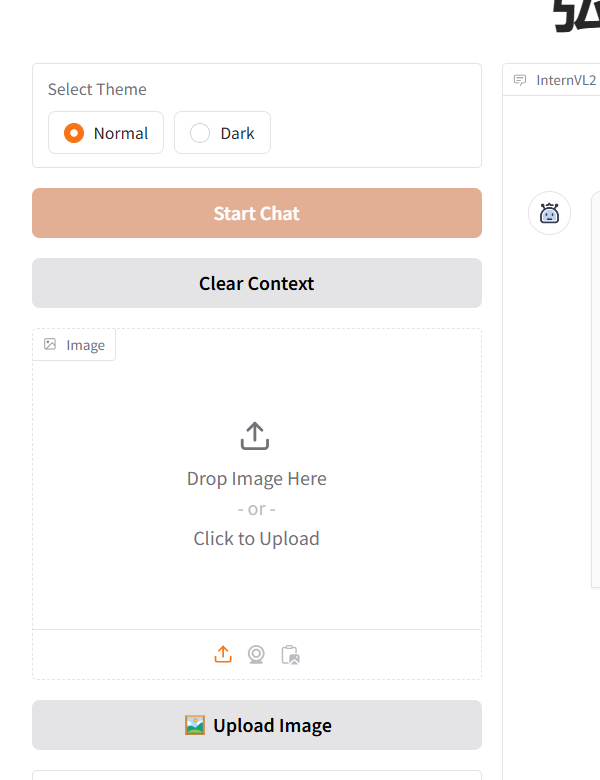
|  |
| --- |
| SQL conda activate xtuner-env  # transfer weights python /root/code/XTuner/xtuner/configs/internvl/v1\_5/convert\_to\_official.py /root/code/XTuner/xtuner/configs/internvl/v2/internvl\_v2\_internlm2\_2b\_qlora\_finetune.py /root/code/qlora\_output/iter\_2000.pth /root/model/InternVL2-2B-houjie/ |

**微调后模型的推理测试**

**InternVL2-2B 前端页面美化（此部分）**

|  |
| --- |
| SQL import os import random import numpy as np import torch import torch.backends.cudnn as cudnn import gradio as gr  from utils import load\_json, init\_logger from demo import ConversationalAgent, CustomTheme  FOOD\_EXAMPLES = "demo/food\_for\_demo.json" MODEL\_PATH = "/group\_share/InternVL2-2B-haojie/" OUTPUT\_PATH = "./outputs"  def setup\_seeds():  seed = 42  random.seed(seed)  np.random.seed(seed)  torch.manual\_seed(seed)  cudnn.benchmark = False  cudnn.deterministic = True def toggle\_theme(theme):  return gr.update(css="dark-mode" if theme == "Dark" else "light-mode")  def main():  setup\_seeds()  init\_logger(OUTPUT\_PATH)  food\_examples = load\_json(FOOD\_EXAMPLES)   agent = ConversationalAgent(model\_path=MODEL\_PATH, outputs\_dir=OUTPUT\_PATH)  theme = CustomTheme()   titles = [   ]   with gr.Blocks() as demo\_chatbot:  gr.Markdown("""  <style>  .dark-mode {  background-color: #121212;  color: white;  }  .light-mode {  background-color: #f9f9f9;  color: black;  }  .chatbot {  border: 1px solid #ccc;  border-radius: 8px;  padding: 15px;  background-color: #333;  box-shadow: 0 0 15px rgba(0, 0, 0, 0.3);  }  .chat-message {  padding: 12px;  border-radius: 8px;  margin: 8px 0;  }  .user-message {  background-color: #1aab1f;  text-align: right;  }  .bot-message {  background-color: #1a73e8;  text-align: left;  }  .gr-button {  background-color: #0e76a8;  color: white;  }  .gr-button:hover {  background-color: #0c5e76;  }  </style>  """)   for title in titles:  gr.Markdown(title)   with gr.Row():  with gr.Column(scale=3):  theme\_selector = gr.Radio(["Normal", "Dark"], label="Select Theme", value="Normal", interactive=True)  start\_btn = gr.Button("Start Chat", variant="primary", interactive=True)  clear\_btn = gr.Button("Clear Context", interactive=False)  image = gr.Image(type="pil", interactive=False)  upload\_btn = gr.Button("🖼️ Upload Image", interactive=False)   with gr.Accordion("Generation Settings"):  top\_p = gr.Slider(minimum=0, maximum=1, step=0.1,  value=0.8,  interactive=True,  label='top-p value')   temperature = gr.Slider(minimum=0, maximum=1.5, step=0.1,  value=0.8,  interactive=True,  label='temperature')   with gr.Column(scale=7):  chat\_state = gr.State()  chatbot = gr.Chatbot(label='InternVL2', height=800, avatar\_images=(  os.path.join(os.path.dirname(\_\_file\_\_), 'demo/user.png'),  os.path.join(os.path.dirname(\_\_file\_\_), 'demo/bot.png')))  text\_input = gr.Textbox(label='User', placeholder="请点击 <开始聊天> 按钮开始聊天!", interactive=False)   gr.Markdown("### 输入示例")   text\_input.change(fn=lambda x: gr.update(interactive=True), inputs=text\_input, outputs=text\_input)   gr.Examples(  examples=[  ["中医如何看待体质与健康之间的关系？"],  ["什么是四诊法？它们各自的作用是什么？"],  ["如何通过饮食调理来改善脾胃虚弱？"],  ["在中医中，如何理解阴阳平衡的概念？"],  ["能否解释一下常见的中药及其功效？"],  ["如何判断一个人的体质类型？"],  ["中医治疗感冒的常用方法有哪些？"],  ["什么是经络，如何影响身体健康？"]  ],  inputs=[text\_input]  )   # 按钮功能  start\_btn.click(agent.start\_chat, [chat\_state],  [text\_input, start\_btn, clear\_btn, image, upload\_btn, chat\_state])  clear\_btn.click(agent.restart\_chat, [chat\_state],  [chatbot, text\_input, start\_btn, clear\_btn, image, upload\_btn, chat\_state], queue=False)  upload\_btn.click(agent.upload\_image, [image, chatbot, chat\_state], [image, chatbot, chat\_state])  text\_input.submit(agent.respond, inputs=[text\_input, image, chatbot, top\_p, temperature, chat\_state],  outputs=[text\_input, image, chatbot, chat\_state])   # 主题切换  theme\_selector.change(fn=toggle\_theme, inputs=theme\_selector, outputs=demo\_chatbot)   demo\_chatbot.launch(share=True, server\_name="127.0.0.1", server\_port=1096, allowed\_paths=['./'])  if \_\_name\_\_ == "\_\_main\_\_":  main() |





**市场分析 与 用户定位**

弘景中医药大模型(皮肤病版)市场分析与用户定位，我们可以从以下几个维度进行分析：

1. **医疗专业人士**：

* **皮肤科医生**：作为直接面对皮肤病患者的专业人士，他们可能会使用该模型来辅助诊断和治疗。
* **中医专家**：对于中西医结合的医疗环境，中医专家可能会利用该模型来提供更精准的中医治疗方案。

1. **医疗机构**：

* **医院皮肤科**：大型医院的皮肤科可能会采用该模型来提高诊断的准确性和治疗的有效性。
* **中医诊所**：专注于中医治疗的诊所可能会使用该模型来增强其服务能力。

1. **患者群体**：

* **皮肤病患者**：直接受益于该模型的最终用户，他们可能会通过医疗机构或在线平台访问该模型的服务。
* **亚健康人群**：对于有皮肤问题但尚未达到疾病程度的人群，他们可能会使用该模型来进行预防和健康管理。

1. **科研与教育机构**：

* **大学和研究机构**：可能会使用该模型进行皮肤病相关的研究，以探索中医治疗的新方法和效果。
* **医学院**：作为教学工具，帮助学生了解中医在皮肤病治疗中的应用。

1. **健康管理与咨询机构**：

* **健康咨询公司**：可能会使用该模型为客户提供个性化的健康管理方案。
* **在线医疗平台**：提供远程医疗服务的平台可能会集成该模型，以提供更专业的咨询服务。

1. **药企与研发机构**：

* **中医药企业**：可能会利用该模型来开发新的中药配方或治疗方案。
* **药物研发机构**：可能会使用该模型来测试和验证新药的效果。

1. **政府与公共卫生部门**：

* **公共卫生政策制定者**：可能会参考该模型的数据来制定相关的公共卫生政策和预防措施。

1. **国际市场**：

* **对中医感兴趣的国际用户**：随着中医在全球的普及，国际用户可能会通过该模型来了解和使用中医治疗方法。
* 皮肤科

专业人士

皮 肤 病 人

美容养颜机 构

爱美人 士

中医药

护肤厂商