# llm知识库项目记录

> 记录什么?
> 项目的细节；面试可能会出现的问题；

2024年8月2日  当前的问题：

他们都觉得我缺乏对整个rag的思考？只是说了基本的流程。没有太多rag调优的经验。
检索出来文本后生成的prompt是怎么进行优化的?
我构建了一个prompt的模板，还有其他形式来生成模型的prompt吗?
怎么评估这个rag的知识库的?
意图识别具体是怎么做的?

## 简历内容：

2024.3 某企业的知识库建设和大模型问答

- 背景：某企业需要构建本地知识库，建立中文问答平台，要求数据安全，隐私保护，离线的私有化部署的。
- 方案：1. 构建数据清洗，敏感词清洗pipeline；2. Qwen1.5-14B-chat模型，bge-large-zh-v1.5作为embedding bge-reranker-v2-m3 作为reranker；3. streamlit搭建UI界面；
- 难点
- 效果：能够对企业的知识库进行问答，隐私防控，  已上线

## 项目复盘

### 项目需求

1. 能对企业员工进行培训。
2. 确保数据安全和隐私保护，大模型需要离线部署。
3. 回答速度要快，并发有多少?
4. 模型能够稳定运行。
5. **建立知识库更新流程，能够不断改进知识库和问答系统**
6. **建立内容审核机制**

7. （收集用户和知识库的交互数据，以进行优化知识库的内容和问答系统）
8. （用户反馈机制）

## 项目难点

如何数据清理?
如何保证模型稳定运行?
如何对文件进行在线流式输入?
等等

## 知识库项目做了什么?

1. 高效的数据清洗pipeline，规范化各类文档，去除无用字符，去除敏感字符。
2. 意图识别
3. vllm加速
4. 知识库单条添加
5. 敏感词判断等等
6. 尝试各类大模型，embedding，reranker，利用vllm对模型进行加速，能够准确高效快速地回答用户的问题。
7. 能对文件进行在线流式输入，也能直接对pdf文档进行阅读。

## 做了什么调优?

混合检索。
相似文本检索，和关键词检索。

```
from langchain.retrievers import BM25Retriever, EnsembleRetriever

retriever_vectordb = vectorstore.as_retriever(search_kwargs={"k": 2})
keyword_retriever = BM25Retriever.from_documents(chunks)
keyword_retriever.k = 2

ensemble_retriever = EnsembleRetriever(retrievers=[retriever_vectordb, keyword_retriever], weights=[0.5, 0.5]) explain more about these part.
```

## 如何对模型进行评估? （未做）

langsmith，LangFuse，它是开源的，是LangSmith 的平替。
ragas
https://arxiv.org/pdf/2309.15217.pdf

## 如何进行数据采集呢?

## 数据处理的pipeline的怎么做的?

多进程阅读每个文件，对各个数据都会进行处理。

## 表格和图片是怎么处理的?

图片跳过。表格用langchain的处理逻辑。langchain community的包里有这部分的。
图片应该通过ocr处理。

ocr处理工具：easyOCR，BetterOCR

BetterOCR combines the results from different OCR engines.

## 在检索之后如何进行数据清洗？（疑惑）

需要去除数据中的无关信息、重复数据、格式错误等噪音数据。这可以通过规则匹配、统计分析和机器学习方法实现，例如使用正则表达式去除无意义的字符串，或者训练分类模型自动识别噪音数据。
数据标准化：确保数据格式、单位等的一致性，提高数据的可用性。可以通过基于规则的方法、字典或机器学习模型实现数据格式转换。
数据敏感过滤：去除数据中的有毒内容或隐私信息，例如使用关键字识别去除个人信息如身份证号。
数据去重：删除重复的文档或文本片段，以避免模型过度学习重复模式。可以采用模糊去重如SimHash算法，或精确去重通过后缀数组查找字符串之间的精确匹配。

如何控制用户的输入?

## 企业知识库中有哪些内容?

企业知识库通常包含以下内容：

岗位知识：员工执行任务所需的特定知识和技能。
经营管理知识：战略规划、市场分析、项目管理等企业运营相关的知识。
培训资料：新员工培训、技能提升课程、内部分享材料。
客户资料：市场调研、客户案例、销售策略等。
政策制度：企业规章制度、操作手册、法律合规信息。
项目文档：历史项目经验、技术文档、设计规范等。

## 用的是什么框架？对它了解得深入吗？（未做）

langchain-chatchat

ragflow的框架图在这
ragflow/README_zh.md at main · infiniflow/ragflow (github.com)

## 外挂知识库的流程是什么?

数据预处理->分块（这一步骤很关键，有时候也决定了模型的效果）->文本向量化->query向量化->向量检索->重排->query+检索内容输入LLM->输出

query处理；信息检索；选择合适的数据（过滤和重排）；prompt生成；

## 怎么做到在线地流式输入?

## 有一些问题中在向量数据库中找不到怎么办?

## 如何评估rag?

langchain-ai/auto-evaluator (github.com)

## embedding模型是怎么训练的?

## embedding模型的选型? （疑惑）

代码中适配了cohere，jina，bce，bge，openai的接口，为什么接入那么多embedding。
jina-ai/jina: ☁️ Build multimodal AI applications with cloud-native stack (github.com)
cohere.ai (github.com)
BCEmbedding/README_zh.md at master · netease-youdao/BCEmbedding (github.com)

对于embedding和reranker选型的思路
Boosting RAG: Picking the Best Embedding & Reranker models — LlamaIndex, Data Framework for LLM Applications

没什么区别。 英文jina即可。

## 如何进行隐私防控？数据清洗该怎么搞？敏感词怎么去除（重大疑惑）

## 添加知识库通常的作用是什么？

- 克服遗忘问题
- 提升回答的准确性、权威性、时效性
- 解决通用模型针对一些小众领域没有涉猎的问题
- 提高可控性和可解释性，提高模型的可信度和安全性

自己的想法：
数据可以实时更新，检索数据的策略，过滤的策略可方便地更新，非常高效。

## 用了什么reranker？什么原理？为什么用这个reranker？

这篇文章关于中文的写得不错
中文 Emebedding & Reranker 模型选型 | blog (ninehills.tech)
Reranker API (jina.ai)
jina reranker
两个参数：一个是相似度的阈值，过滤的文件。
1. Extracts the content of each document.
2. Computes scores for each document using the reranker.
3. Assigns the computed scores to the metadata of each document.
Sorts the documents based on their scores in descending order.
5. Prints out the top-k documents based on the sample_top value.
Filters out documents based on a sample_threshold if provided.
Prints the number of documents after filtering.

为什么用？
Traditionally, methods like BM25 or tf-idf have been used to rank search results based on keyword matching. Recent methods, such as embedding-based cosine similarity, have been implemented in many vector databases. These methods are straightforward but can sometimes miss the subtleties of language, and most importantly, the interaction between documents and a query's intent.

This is where the "reranker" shines. A reranker is an advanced AI model that takes the initial set of results from a search—often provided by an embeddings/token-based search—and **reevaluates them to ensure they align more closely with the user's intent.** It looks beyond the surface-level matching of terms to consider the deeper interaction between the search query and the content of the documents.
为什么用jina？
jina在英文上的评测非常好。

中文可能得用：bge-reranker-v2-m3

## finetune的劣势?

通用能力会差很多。

## rag的缺点?

非常依赖数据的质量，重新设置prompt比较困难，

关于RAG的一些开源项目：
【OmniParse】全能文件解析和清洗工具为RAG而生 提升回复质量
https://github.com/adithya-s-k/omniparse
【Langchain-Chatchat】基于 Langchain 与 ChatGLM 等语言模型的 RAG 与 Agent 应用
https://github.com/chatchat-space/Langchain-Chatchat
【ZTE】
https://challenge-zte.zhaopin.com/

## 文本分割器用什么？其他的分割器了解哪些？原理是什么？

RecursiveCharacterTextSplitter

定个chunk size，chunk overlap就可以了；也有tokenSplitter，tokenSplitter是根据token的数量大小来对文本进行分割；

它的原理是什么?
先根据\n分割，然后merge到不超过chunk size，若\n分割不足够，然后再根据空格分割，然后merge到不超过chunk size。

文档拆分非常重要，因为它需要确保语义相关的内容在同一块中组合起来。
1. CharacterTextSplitter
一个基本的拆分器，它基于单个字符分隔符（如空格或换行符）拆分文本。在处理结构不清晰的文本或想要在特定点拆分文本时，此拆分器非常有用。

2.RecursiveCharacterTextSplitter
用于通用文本拆分，它根据分隔符的层次结构拆分文本，从双换行符开始，然后是单换行符 、空格，最后是单个字符。这种方法旨在通过优先考虑段落和句子等自然边界的拆分来保持文本的结构和连贯性。RecursiveCharacterTextSplitternnn

3.TokenTextSplitter
根据标记计数而不是字符计数拆分文本，因为许多语言模型都具有由标记计数而不是字符计数指定的上下文窗口。标记的长度通常约为四个字符，因此基于标记计数进行拆分可以更好地表示语言模型将如何处理文本。TokenTextSplitter

4.MarkdownHeaderTextSplitter
旨在根据标题结构拆分 Markdown 文档。它将标头元数据保留在生成的块中，从而允许上下文感知拆分和使用文档结构的潜在下游任务。MarkdownHeaderTextSplitter

langchain还有一个context-aware拆分。

也有通过LLM来辅助进行文本分割的
LLM based context splitter for large documents | by Ayham Boucher | Medium
就是在RecursiveCharacterTextSplitter分割之后的文本，再对各个文本进行相似度计算，若相似度差不多，那么就合并为一个文本块。

多进程处理各个文档

```
with Pool(processes=os.cpu_count()) as pool:
```

## 怎么做切分的？chunk size多少？

这里集中了许多相关的hacker news中的文章。
How to choose your chunk size for summarizing large documents using LLM | Hacker News (ycombinator.com)
这篇文章是为了解决最后一个chunk过短的解决方案。
VectifyAI/LargeDocumentSummarization: Optimal Chunk Size for Large Document Summarization (github.com)
通常是128,256,512，这个没什么太多的文章，但很自然可以想到比较零散的信息，要用小的chunk size，比较“长篇大论”的文章，就用大的chunk size。

## 你知道其他知识库的框架有哪些吗？框架上有什么对比？

Langchain-Chatchat， Qanything，AnythingLLM

## 一秒能承受多少并发？

需要多少块显卡？
qwen14B 2张。

模型vllm加速，1.5s就可以返回。
qwen14b；4*3090；2张卡；
分词分块的长度，交集的长度？

遇到过什么问题？幻觉问题？

为什么用qwen14？
为什么用vllm加速方法？

意图识别得不准怎么办？
几百条就够了，可以灵活加。
通常的数据量是多少呢？
几万条。
lora微调多少会有效果？
几千条，几万条。
llama模型知道吗？优化器？
adamw。

## 大语言模型通用问题

### 大模型的幻觉问题、复读机问题有没有什么解决办法（疑惑：应该还有更好的办法把）

针对幻觉问题：引入外挂知识库，加入一些纠偏规则，限制输出长度等

针对复读机问题：

丰富数据集的多样性，预处理时尽量过滤重复无意义的文本
同义词替换等做数据增强
温度参数调整
后处理与过滤

**怎么进行后处理呢**

**模型量化工具?**

模型量化是使用低精度数据类型（例如 8 位整数 (int8)）而不是传统的 32 位浮点 (float32) 表示来表示模型中的权重、偏差和激活的过程。通过这样做，它可以明显减少推理过程中的内存占用和计算需求，从而能够在资源受限的设备上进行部署。模型量化在计算效率和模型精度之间取得微妙的平衡。目前主要使用的 LLM 开源量化工具主要有：bnb，GPTQ，AWQ

## 了解vllm的原理吗

vLLM：给大模型提提速，支持高并发吞吐量提高24倍，同时推理速度最少提高 8 倍_vllm 并发-CSDN博客

KV cache会导致不连续的内存占用，造成大量内存浪费。采用了pageAttention算法。

## langchain相关

【浪潮面试】自我介绍、项目、分工、做的什么、编程语言情况、了解什么算法、意向城市等，差不多就这些

## 2024.7.10

用中兴比赛练手。

组委会的消息：这道题的意图是考察同学们如何在较小参数量的基础上实现优质的AI应用，因此限制了大模型的参数量。提升AI应用能力的方式有很多，可以采用RAG、fine-tuning、few-shot等技术。

另外的一个考点是语料的清洗和转换，提供的数据集是原始的PDF，也是考察大家的AI工程应用能力，优质的语料服务是提升AI应用能力的重要因素～ 而其中有两个难点：一是试题可能出自非同一语种的语料，二是试题选自的原始语料体量较大。所以大家也要考虑较好的检索算法～另外算法性能虽然不在AB榜得分体现，但在专家评审中也是一个关键考察点哦

xinference还是能把qwen7b和qwen14b的模型启动起来的，不知道为什么不能启动glm模型，很奇怪。

可以启动glm4，比如使用命令`xinference launch --model-engine vllm --model-name glm4-chat --size-in-billions 9 --model-format pytorch --quantization none`下载之后，在UI界面启动

# 2024.6.30之前

内容：知识库、大模型使用、模型加速、模型训练、实际项目流程梳理、实际业务需求介绍和解决方案说明、分布式训练。
/home/cider/proj/llm_proj/Langchain-Chatchat/server/chat/knowledge_base_chat.py

写成校企合作项目

llava微调，k vqa

医学知识库问答

查询向量数据库，rerank精排，prompt

用到了多个向量数据库，表a还是用表b，意图识别。
企业做数字人，医院做数字人客服，；路线，医院内部的规范等，三类就有三个知识库。三种知识库。
问题做bert分类。

敏感词识别->意图识别->不同的类别对应不同的prompt，数据库 ->（rerank）->LLM输出

意图识别也可以用小模型bert也可以用LLM。

pdf通过ocr的工具转成文本，

如何做数据：
知识库里会有pdf，通过ocr工具转成word，输给kimi得到了一问一答的形式，存入知识库。然后再本地部署的时候，可以用一个小模型，这样可以做到成本可控。

这种知识库能够保证效果，减少幻觉，不会出错，比如在意识形态方面。

通过知识库上传文件，

原本是知识库上传文件，文件切块，
也可以是

文件是一块块的，既有问题又有答案，如何生成embedding，选问题，选答案，还是选问题和答案。问题和答案分别进行生成emedding，既会在q里查询，又会在文件里查询。

http是接口，输入是问题和答案，接受之后就通过embedding模型，生成embedding，这是一个很实用的功能，就是能及时添加没有的问题。

mysql要看，大概要会。向量数据库，缓存数据库和关系型数据库。

企业的数据是什么呢？ 博物馆的数据是什么呢?

这些数据产生的模型真的对他们的工作有帮助吗？能提升生产力吗?

做数据的时候，问题和答案就会存储起来。

多一个接口，直接在这个函数里的入参加就可以了。
knowledge_base_chat

举例说明：

哪个用户问的哪个问题?

## 0626 文档阅读记录

langchain agent

pseudo code of the agent.
```
next_action = agent.get_action(...)
while next_action != AgentFinish:
    observation = run(next_action)
    next_action = agent.get_action(..., next_action, observation)
return next_action
```
it's trying to solve these problems.

1. Handling cases where the agent selects a non-existent tool
2. Handling cases where the tool errors

3. Handling cases where the agent produces output that cannot be parsed into a tool invocation
4. Logging and observability at all levels (agent decisions, tool calls) to stdout and/or to LangSmith.

tools have two components

1. input schema
2. the function to call

two keys to consider when using tools:

1. the right access to the function
2. the right description of using the tool

extract structure output from unstructured data

Extracting structured output | 🦜🔗 LangChain

Handle Long Text | 🦜🔗 LangChain

handle long text

three methods:

- LLM with longer context window
- chunk, extract information from each part, and merge
- rag

trade offs

- Chunking content means that the LLM can fail to extract information if the information is spread across multiple chunks.
- Large chunk overlap may cause the same information to be extracted twice, so be prepared to de-duplicate!
- LLMs can make up data. If looking for a single fact across a large text and using a brute force approach, you may end up getting more made up data.

pydantic is just a data validation tool

general guidelines of extracting the structure output

Set the model temperature to 0.

Improve the prompt. The prompt should be precise and to the point.

Document the schema: Make sure the schema is documented to provide more information to the LLM.
Provide reference examples! Diverse examples can help, including examples where nothing should be extracted.
If you have a lot of examples, use a retriever to retrieve the most relevant examples.
Benchmark with the best available LLM/Chat Model (e.g., gpt-4, claude-3, etc) -- check with the model provider which one is the latest and greatest!
If the schema is very large, try breaking it into multiple smaller schemas, run separate extractions and merge the results.
Make sure that the schema allows the model to REJECT extracting information. If it doesn't, the model will be forced to make up information!
Add verification/correction steps (ask an LLM to correct or verify the results of the extraction).

the former is to benchmark the language model, the latter create the langchain data.
langchain-ai/langchain-benchmarks: 🦜💯 Flex those feathers! (github.com)
Get started with LangSmith | 🦜🛠️ LangSmith (langchain.com)

Tool call
Tool/function calling | 🦜🔗 LangChain

other local knowledgebase solutions
GanymedeNil/document.ai: 基于向量数据库与GPT3.5的通用本地知识库方案(A universal local knowledge base solution based on vector database and GPT3.5) (github.com)

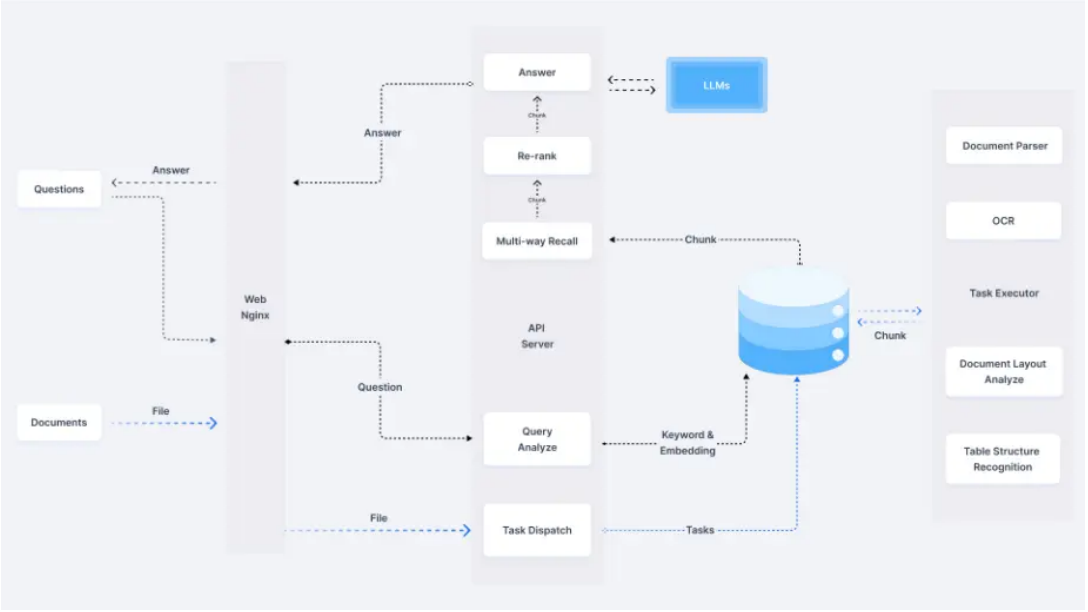
webws/embedding-knowledge-base: a local knowledge base based on chatgpt Embedding and qdrant, supporting data import and Q&A (github.com)

infiniflow/ragflow: RAGFlow is an open-source RAG (Retrieval-Augmented Generation) engine based on deep document understanding. (github.com)

LLM+本地知识库？简单又没那么简单 - 掘金 (juejin.cn)

基于开源大模型快速构建本地知识库应用 - danghl - twt企业IT交流平台 (talkwithtrend.com)

One rag project. it aims to talk deep with the document.
RAGFlow




## 0625阅读文档记录

```
Starting from version 0.3.0, Langchain-Chatchat no longer directly loads models based on the local model path entered by users. The involved model types include LLM, Embedding, Reranker, and the multi-modal models to be supported in the future. Instead, it supports integration with mainstream model inference frameworks such as Xinference, Ollama, LocalAI, FastChat and One API.
Starting from version 0.3.0, Langchain-Chatchat no longer modifies the configuration through local files but uses command-line methods and will add configuration item modification pages in future versions.
```

5个向量数据库比较Chroma, Pinecone, Milvus, FAISS, Annoy
Primary Language and API Support:
Chroma: Python and JavaScript support.
Weaviate: Various modules for integration with platforms like OpenAI, Cohere, and HuggingFace.
Faiss: C++ with full Python/NumPy integration and some GPU execution.
Qdrant: OpenAPI v3 specs and clients for multiple languages.
Pinecone: Integration with LangChain, primarily aimed at data engineers and data scientists.
Unique Features:
Chroma: Easy to build LLM apps by making knowledge, facts, and skills pluggable for LLMs.
Pinecone: Real-time data ingestion and low-latency search.
Weaviate: Speed and flexibility with support for large-scale production.
Faiss: Algorithms capable of handling vector sets that exceed RAM capacity, with GPU execution support.
Qdrant: Advanced filtering and support for diverse data types, built-in Rust for efficiency.

Images:

The 5 Best Vector Databases | A List With Examples | DataCamp

## 0624实验和思考

本地跑pdf阅读的可能性太小了，速度太慢了，无论是上传文档的速度还是建立embedding的速度都达不到好用的地步，这么小的模型也垃圾。
复旦的机器xinference能用，但是26b的模型又起不来，cuda out of memory，它似乎不用多卡来跑？？奇怪；而且langchainchatchat也不能起来，各种报错，不过我用的还是0.3的版本，0.2还没尝试。
cider上的xinference也没有问题，但是chatchat建立20几个pdf文件的embedding时，又会出现线程超量的情况，程序直接卡死。这机器绝对有问题。
用xinference启动模型是真的方便。但是应该只适用于测试模型吧，只适合对模型进行知识库这种外部操作比较方便吧。微调模型基本也不可能。

二开，要多传一个字段进来该怎么办?

本来知识库需要手动上传文件，如何让知识库自动地流式接受在线的数据?

## 环境搭建

vllm-project/vllm: A high-throughput and memory-efficient inference and serving engine for LLMs (github.com)
hiyouga/LLaMA-Factory: Unify Efficient Fine-Tuning of 100+ LLMs (github.com)
chatchat-space/Langchain-Chatchat（原Langchain-ChatGLM）基于 Langchain 与 ChatGLM 等语言模型的本地知识库问答 | Langchain-Chatchat (formerly langchain-ChatGLM), local knowledge based LLM (like ChatGLM) QA app with langchain (github.com)

langchain-chatchat，llama-factory，vllm
本来是打算安装langchainchatchat0.2.0版本的，但是由于报错的问题，我就选择放弃。
最近出了最新版langchain-chatchat0.3.0，然后我的部署就是基于langchain0.3.0。

按照readme照着做即可

vllm
```

```

为了能在cider机器上跑，所以还是用了xinference
```
pip install ”xinference[all]”
pip install tiktoken  sentence-transformers
```

xinference[all] 在cider上有问题，估计是g++ 有问题，暂时不搞，先不用vllm 的backend。

langchain-chatchat。有些包还是得后来装。
```
pip install langchain-chatchat -U
pip install streamlit==1.36.0
pip install rank_bm25 faiss-gpu
```

但是在复旦的机器上需要安装faiss-gpu。
得用conda install faiss-gpu才能成功安装。
后来复旦的机器上还是失败了，是因为以下chatchat 报错

httpcore.RemoteProtocolError: peer closed connection without sending complete message body (incomplete chunked read)

两个模型下载
qwen1.5-14b-chat  bge-large-zh-v1.5

pip install modelscope

```
  from modelscope import snapshot_download
  model_dir = snapshot_download('iic/Whisper-large-v3')
```

## 如何配置

xinference+chatchat
主要参考内容

Langchain-Chatchat/docs/install/README_xinference.md at master · chatchat-space/Langchain-Chatchat (github.com)

这里是0.2 的参数配置文档
参数配置 · chatchat-space/Langchain-Chatchat Wiki (github.com)
首先是xinference，需要启动webui，启动chat模型和embedding模型。

```
 xinference-local --host 0.0.0.0 --port 7000
```

如果是自己的模型，就填一下本地xinfernce路径，注册一下，如果不是，直接界面选择模型启动即可。

xinfernce启动完成之后，就是langchainchatchat的事了。

首先配置文件全部都在
`/home/szh/miniconda3/envs/lancc03/lib/python3.11/site-packages/chatchat/configs/model_providers.yaml`这里修改。
如下所示。但奇怪的是这里的url的端口号怎么改都没用，估计是在其他地方改的，但我无论如何都找不到。

```
xinference:
  model_credential:
    - model: 'qwen1.5-chat'
      model_type: 'llm'
      model_credentials:
        server_url: 'http://127.0.0.1:9997/'
        model_uid: 'qwen1.5-chat'
    - model: 'my-bge'
      model_type: 'text-embedding'
      model_credentials:
        server_url: 'http://127.0.0.1:9997/'
        model_uid: 'my-bge'
  provider_credential:
    api_key: 'EMPTY'
```

除了以上配置，需要设置一下默认的模型位置。
`chatchat-config model --default_llm_model qwen1.5-chat --default_embedding_model bge-large-en-v1.5`

## 环境报错集锦

pydantic.v1.error_wrappers.ValidationError: 1 validation error for LocalAIEmbeddings
AttributeError: 'NoneType' object has no attribute 'embed_documents'

现在找不到模型-　原来是9997端口没变- 原来是自己的config文件一点儿都没起作用

现在检查cider的机器- 得到结论：site-packages里面的config文件改了是有作用的。

居然还是配置问题。

结论：仔细检查哪里命名错了。。。

module 'streamlit' has no attribute 'experimental_dialog'
pip install streamlit==1.36.0

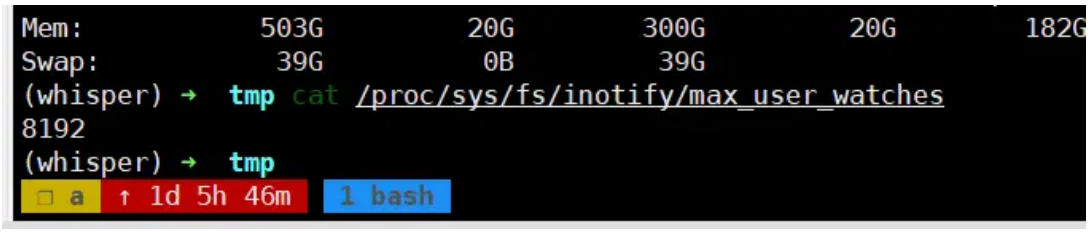
ModuleNotFoundError: No module named 'rank_bm25'
pip install rank_bm25

cuda 安装11.8版本 ，但是我暂时还是没装，想着出问题再说，我目前用的是过去的cuda12，vllm默认用的是cuda12

vllm做模型加速，以openai的格式启动成api接口供使用，vllm版本和启动脚本也有关系，你用的qwen14　1.5版本也稍有区别

vllm启动日志，里面有一个nccl的报错，似乎是可能会影响gpu的加速。

python_python -m vllm.entrypoints.ope_20240619_160257.log




## 其他

```
python -m vllm.entrypoints.openai.api_server --host 0.0.0.0 --port 6008 --model /home/cider/.cache/modelscope/hub/qwen/Qwen1___5-14B-Chat --tokenizer /home/cider/.cache/modelscope/hub/qwen/Qwen1___5-14B-Chat --trust-remote-code --dtype float16 --seed 1234 --worker-use-ray --tensor-parallel-size 2 --served-model-name Qwen-14B-Chat --swap-space 8 --max-model-len 3840
```

```
    },

    "chatglm2-6b": {
        "local_model_path": "THUDM/chatglm2-6b",
        "api_base_url": "http://localhost:8888/v1",  # URL需要与运行fastchat服务端的server_config.FSCHAT_OPENAI_API一致
        "api_key": "EMPTY"
    },

    "chatglm2-6b-32k": {
        "local_model_path": "THUDM/chatglm2-6b-32k",  # "THUDM/chatglm2-6b-32k",
        "api_base_url": "http://localhost:8888/v1",  # "URL需要与运行fastchat服务端的server_config.FSCHAT_OPENAI_API一致
        "api_key": "EMPTY"
    },

    "Qwen-7B-Chat": {
        "local_model_path": "/opt/llm/models/Qwen-7B-Chat",
        "api_base_url": "http://localhost:8888/v1",  # "URL需要与运行fastchat服务端的server_config.FSCHAT_OPENAI_API一致
        "api_key": "EMPTY"
    },
    "Qwen-14B-Chat": {
        "local_model_path": "/opt/llm/Qwen-14B-Chat",
        "api_base_url": "http://10.0.2.200:8888/v1",  # "URL需要与运行fastchat服务端的server_config.FSCHAT_OPENAI_API一致
        "api_key": "EMPTY"
    },
    "Qwen1.5-72B-Chat-GPTQ-Int4": {
        "api_base_url": "http://10.0.2.200:8888/v1",  # "URL需要与运行fastchat服务端的server_config.FSCHAT_OPENAI_API一致
        "api_key": "EMPTY"
    },

    # 调用chatgpt时如果报出： urllib3.exceptions.MaxRetryError: HTTPSConnectionPool(host='api.openai.com', port=443):
    #  Max retries exceeded with url: /v1/chat/completions
    # 则需要将urllib3版本修改为1.25.11
    # 如果依然报urllib3.exceptions.MaxRetryError: HTTPSConnectionPool，则将https改为http
```

```
# API 是否开启跨域，默认为False，如果需要开启，请设置为True
# is open cross domain
OPEN_CROSS_DOMAIN = True

# 各服务器默认绑定host
DEFAULT_BIND_HOST = "0.0.0.0"

# webui.py server
WEBUI_SERVER = {
    "host": DEFAULT_BIND_HOST,
    "port": 8504,
}

# api.py server
API_SERVER = {
    "host": DEFAULT_BIND_HOST,
    "port": 7864,
}

# fastchat openai_api server
FSCHAT_OPENAI_API = {
    "host": DEFAULT_BIND_HOST,
    "port": 8888,  # model_config.llm_model_dict中模型配置的api_base_url需要与这里一致。
}

# fastchat model_worker server
# 这些模型必须是在model_config.llm_model_dict中正确配置的。
# 在启动startup.py时，可用通过`--model-worker --model-name xxxx`指定模型，不指定则为LLM_MODEL
FSCHAT_MODEL_WORKERS = {
    # 所有模型共用的默认配置，可在模型专项配置或llm_model_dict中进行覆盖。
    "default": {
        "host": DEFAULT_BIND_HOST,
```

我发给你的话，这两个截图的话，就是说它是在那llcc的那个配置文件里面，model.py，serve.py因为我们VM对它进行加速了之后的话，它其实启动的模型是那种right for的API接口嘛，就和open AI的那种调用形式就一样了，因为考虑到兼容性的话。其实一般像这样的话会比较好，因为这样的话是那种做成相当于插件式的那种接入方式嘛，然后另外一个配置文件的话，就是说相当于是起那个服务就调动调用那个。嗯，就是那个model model的话是以API的形式接入那个VRM的接口嘛。

model 是以api形式介入vlm的接口。

serve.py 的配置使用接入的model的接口。

本地部署的模型可以直接平替在线平台。

目前的情况是我用vllm启了qwen14的模型，我之前用过xinference，是不是其实用两个都一样， 他们是不是功能类似，只是vllm还有模型加速的功能。

在langchainchatchat中，我修改了两个文件的配置，model