

RAG问答机器人项目该如何做？

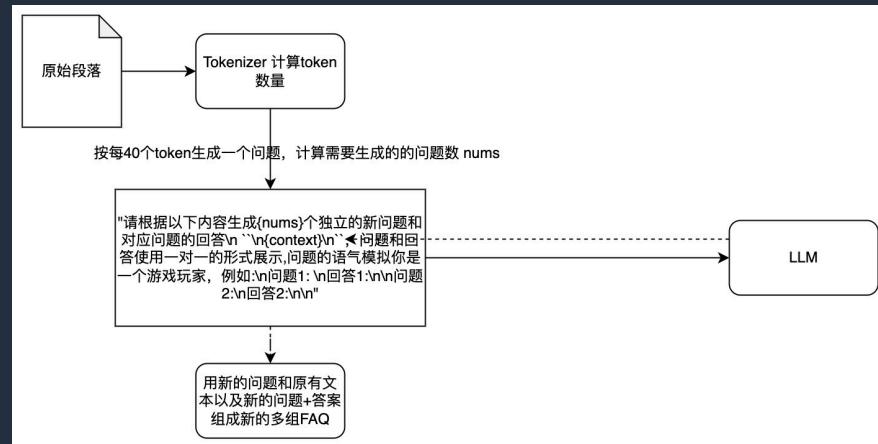
Agenda

- 技术层面经验总结
 - 前置重点 - 知识质量不好咋办？格式复杂还有表格咋办？量大注入慢咋办？
 - 核心工作 - 知识召回效果差咋办？LLM幻觉咋办？怎么判断超出知识库范围？
 - 上线必备 - 如何判断LLM能支持多少QPS？中国区没G5咋办？
 - 经验洞察 - 大模型选哪个？大模型微调有用吗？Streaming response咋支持？Badcase 怎么排查？Bedrock怎么集成？

前置工作 – 知识质量不好咋办?

- ✓ 利用商业LLM做知识增强(原始FAQ/文档都适用), 建议结合人审

- 解决手段:



- 参考效果:

问: 什么是 Amazon EMR?
 答: Amazon EMR 是行业领先的云大数据平台, 适用于使用多种开源框架进行数据处理、交互分析和机器学习, 例如 Apache Spark、Apache Hive、Presto。借助 EMR, 您可以用不到传统本地解决方案一半的成本运行 PB 级分析, 并且其速度比标准 Apache Spark 快 1.7 倍以上。

- 实践检验

某ENT客户的相关项目投标中作为一个独立价值展示, 助力项目投标成功。缓解了客户对于知识质量方面的担忧

- 核心目的: 知识点更加具体。生成多角度提问, 利于召回。

- 参考实现: [Enhance_FAQ.py](#) [Enhance_Doc.py](#)

问: Amazon EMR适用于哪些开源框架进行数据处理、交互分析和机器学习? ,
 答: Amazon EMR适用于使用多种开源框架进行数据处理、交互分析和机器学习, 例如 Apache Spark、Apache Hive、Presto。

问: 通过使用Amazon EMR, 可以以什么样的成本运行PB级分析? ,
 答: 通过使用Amazon EMR, 可以用不到传统本地解决方案一半的成本运行PB级分析。

问: Amazon EMR相比标准Apache Spark有什么优势? ,
 答: Amazon EMR相比标准Apache Spark的速度更快, 快速度比标准Apache Spark快1.7倍以上。

前置工作 – 格式复杂还有表格咋办？

✓ 结合开源代码+Amazon AIML SAAS服务

□ 实践检验：

解决海外一个金融客户POC中，针对基金相关PDF的表格信息提问的问题

□ 解决方案

- 利用Langchain开源代码实现PDF转HTML，可保留字号和像素位置信息，按字号进行合并，提高分段质量
- 利用Textract提取表格，实现Textract输出到Json信息的转换
- 利用前两步结果进行位置映射，克服Textract不支持中文的问题
- 利用NSP模型优化分段效果(研究中)

□ 使用说明：

- [PDF SPLITER README](#)
- [Workshop \(第三步实验\)](#)

□ 参考效果：

The screenshot shows a factsheet for the Allspring Common Stock Fund. It includes sections such as 'Competitive advantages' (with bullet points about PMV and Opportunistic core approach), 'Sector allocation (%)' (with a table comparing the fund's allocation across various sectors like Industrials, Technology, and Healthcare against the Russell 2500® Index²), and 'Annual Returns' (with a table of historical returns from 3 Month to 10Y). The page also features the Allspring logo and some legal disclaimers at the bottom.

```

1 "content":"FUND STRATEGY • Public equity markets are often driven by emotion, requiring successful investors to have conviction in individual securities and diversification across sectors. • Our team's conviction comes from an in-depth private market valuation (PMV), the price an acquirer would pay to purchase the entire company) process of analyzing the business model, competitive positioning, key trends, management, and other proprietary metrics. • We believe that the PMV of a company is much more stable than its associated public market stock price.",  
  "font_size":9,  
  "doc_title":"Common Stock Fund "  
,  
@object{...},  
@object{...},  
@  
2 "content":"Competitive advantages • Private market valuation (PMV) approach: By constantly measuring a company's "private market value," the team is better able to assess a company's worth and act decisively when "market emotion" drives the price of a solid business down to discount levels. Additionally, the PMV investment process helps to discern differences between mispriced stocks and those with cheap valuations, improving the team's likelihood to generate alpha. • Opportunistic core approach: The PMV investment approach is designed to be growth- and value-neutral, with the flexibility to opportunistically invest in the best ideas at either end of the growth and value spectrum.",  
  "font_size":12,  
  "doc_title":"Common Stock Fund "  
,  
@object{...},  
@  
3 "content":  
  "table": "Sector allocation (%)",  
  "footer": "Sector allocation is subject to change and may have changed since the date specified. Percent total may not add to 100% due to rounding.",  
  "data":  
    @  
      "row_key": "Industrials ",  
      "Fund": "24 ",  
      "Russell 2500® Index2 10 ":"Russell 2500® Index2 10 "  
    ),  
    @  
      "row_key": "Information technology ",  
      "Fund": "18 ",  
      "Russell 2500® Index2 10 ":"14 "  
    ),  
    @  
      "row_key": "Consumer discretionary ",  
      "Fund": "13 ",  
      "Russell 2500® Index2 10 ":"11 "  
    ),  
    @  
      "row_key": "Health care ",  
      "Fund": "13 ",  
      "Russell 2500® Index2 10 ":"13 "  
    ),  
    @  
      "row_key": "Financials ",  
      "Fund": "11 ",  
      "Russell 2500® Index2 10 ":"16 "  
    ),  
    @  
      "row_key": "Real estate ",  
      "Fund": "9 ",  
      "Russell 2500® Index2 10 ":"8 "  
    ),  
    @  
      "row_key": "Materials ",  
      "Fund": "8 ",  
      "Russell 2500® Index2 10 ":"6 "
}

```

© 2021-2023 JsonTool.cn, All rights reserved.

前置工作 – 文档量大注入慢?

✓ 结合OpenSearch调优最佳实践 + Glue + 向量模型推理Batch化

□ 面临问题:

- 摄入速度慢, 对面行业文档, 注入几十个小时注入不进去
- 摄入不完整, 反复摄入问题, 知识重复或者不完整

□ 实践检验:

医疗行业的一个客户的POC中, 目标把5w篇文档导入AOS, 最初耗时几十个小时, 仅有13w条向量导入进去了, 合每篇文档仅2.6条记录。折腾了几周也没有解决摄入速度和完整性。

替换方案后, 摄入速度大大提高, 几个小时完成摄入, 完整性也没有问题。

□ 技术总结

1. [基于大语言模型知识问答应用落地实践 – 知识库构建 \(上\)](#)
2. [基于大语言模型知识问答应用落地实践 – 知识库构建 \(下\)](#)

□ 参考实现: [batch_upload_docs.py](#)

Indices (5)									
Index	Health	Managed by policy	Status	Total size	Size of primary...	Total document...	Deleted document...	Primaries	Replicas
oa-5w-0622	Green	No	Open	7.1gb	3.5gb	130311	58539	5	1
oa-5000-0616	Green	No	Open	3.1gb	1.5gb	85624	0	5	1
.opendistro_security	Green	No	Open	149.1kb	74.5kb	10	3	1	1
.kibana_92668751_admin_1	Green	No	Open	39.5kb	18.1kb	3	0	1	1
.kibana_1	Green	No	Open	10.3kb	5.1kb	1	0	1	1

□ 方案Benchmark

1万篇文档的数据规模, 摄入时间1h内, 完整性100%

□ 方案思路

- 大规模文档注入速度方面
 - AOS 多客户端注入 + 集群/索引 参数调优
 - 向量模型多节点部署 + Client Side Batching
- 文档摄入完整性方面
 - 利用Glue 克服大文件超时 (Lambda 15m 限制)
 - 利用Glue Retry机制, 克服AOS负载的波动

核心工作 – 知识召回效果差咋办?

(信息量大)

□ 有效手段

- ✓ BM25打分调优
- ✓ 更优的向量模型选型
- ✓ 多路召回： 向量召回 + 倒排召回
- ✓ 多种召回范式： 对称召回(Query-Question) + 非对称召回(Query-Document)
- ✓ 微调向量模型
- ✓ 引入Rerank模型

□ 面临问题：

- 正确的知识没召回，导致回答没有引用到知识
- 召回了不正确的知识导致LLM产生了误解

□ 技术总结

1. [基于大语言模型知识问答应用落地实践 - 知识召回调优 \(上\)](#)
2. [基于大语言模型知识问答应用落地实践 - 知识召回调优 \(下\)](#)

核心工作 – 知识召回效果差咋办？

✓ BM25打分调优

□ 面临问题：

- 用户的垂直数据中可能某些专词与停用词词频差不多导致IDF失真，引起得分计算有误
- 有些特定‘黑话’，数据不足时语义向量也无法解决

□ 有效手段：

- 构建停用词表，使得停用词均不参与BM25得分
- 构建同义词表，定向解决‘黑话’问题

□ 技术输出

1. [blog 基于大语言模型知识问答应用落地实践 – 知识召回调优（上）](#)

□ 视频Demo

[如何调整BM25倒排优化知识召回效果](#)

核心工作 – 知识召回效果差咋办？

✓ 更好的向量模型选型

□ 面临问题：

- 公开数据集上的表现，在垂直领域没有特别大的参考性
- 在自己数据场景中，通过少量case手工测试无法获取全面信息，难以客观比较

□ 价值提供：

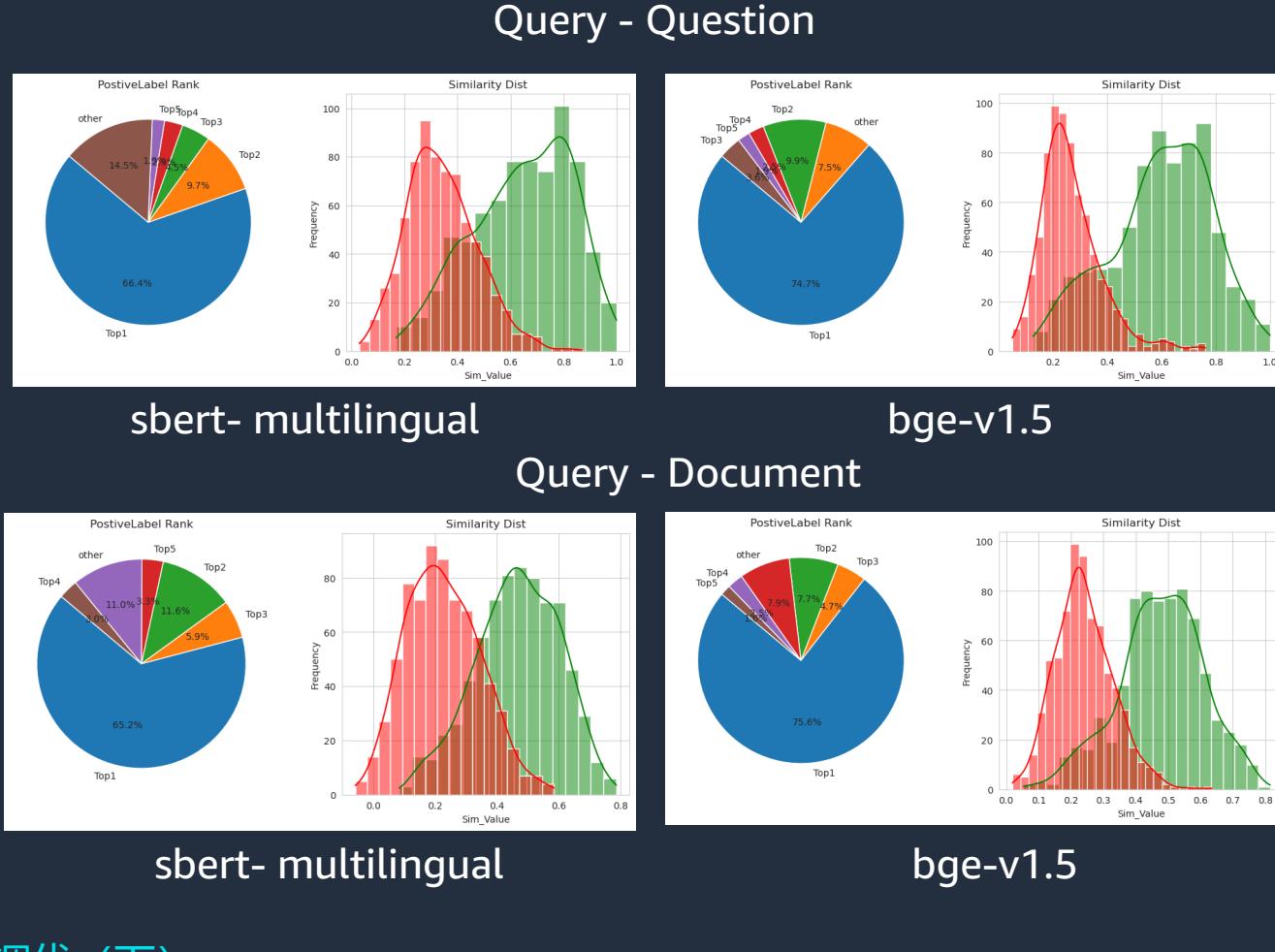
- 给出标准化的评估方式和可视化方法

□ 基本结论：

- 优选bge-large-zh-v1.5
- 优选bge-large-en-v1.5

□ 技术输出

1. 代码实现 [bge_zh_research.ipynb](#)
2. blog [基于大语言模型知识问答应用落地实践 – 知识召回调优（下）](#)



核心工作 – 知识召回效果差咋办?

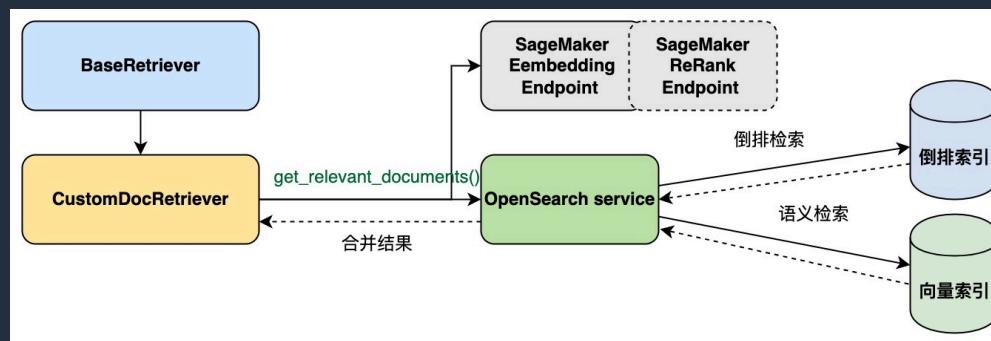
✓ 多路召回 : 向量召回 + 倒排召回

□ 面临问题:

- 向量模型不理解垂直领域专词
- 向量模型有时出现语义相似但主题不相似情况
- 可解释性弱, 不易通过补丁解决bad case
- 倒排召回, 缺乏语义信息, 仅靠关键词匹配

□ 工程实现:

- 重载Langchain Retriaver接口, 融合多路召回结果



□ 优势样例:

用户的原始问题中“**城市外观粉色**”的信息，在向量召回的结果中并没有（前面2条 $score < 1$ 的结果），向量调优难以解决该 Case，但是在第3条倒排召回 ($score > 1$) 的知识中，含有相关信息。

我看有的城市外观粉色的！我的为什么没有呢

您好，关于您的城市外观没有粉色这个问题，可能是在创建城市时选择了其他的城市外观。默认城市外观是根据指挥官性别有两种颜色，男性是蓝色，女性是粉色。但是您可以选择其他的城市外观来改变城市的外观和属性。如果您需要了解如何更改城市外观，我可以为您提供帮助。[['baichuan-stream']]

#Reference

```

Doc[1]:{"ai-content": "topwarfaq230908.faq"}-[""]
{"doc": "Question: 城市外观的拥有属性\nAnswer: 城市外观的拥有属性，是指解锁该城市外观后，其城市外观的拥有属性就会生效，并且可以和其他城市外观的拥有属性叠加生效。", "score": 0.7095267}
Doc[2]:{"ai-content": "cleaned_topwar_enrich_faq_0911.faq"}-[""]
{"doc": "Question: 我的城市外观有哪些使用属性？\nAnswer: 您的城市外观可以拥有许多不同的使用属性，例如美观度、抗风化、防护性等等。这些属性会在城市的建设、升级和维护中发挥重要作用。", "score": 0.7164252}
Doc[3]: {"ai-content": "topwarfaq230908.faq"}-[""]
{"doc": "Question: 默认城市外观\nAnswer: 默认城市外观是根据指挥官性别有两种颜色，男性是蓝色，女性是粉色。\\n默认城市外观没有属性", "score": 20.781532}
  
```

核心工作 – 知识召回效果差咋办?

✓ 多种召回范式： 对称召回 (Query-Question) + 非对称召回(Query-Document)

□ 面临问题：

○ 两种形式各有弊端

- 对于垂直领域做QD召回需要向量模型具备很强的理解能力，需要用这个领域数据的训练过
- 用户query中的一些信息只出现在知识的Document/Answer中，通过Query-Question匹配难度大

□ 例子：

用户的发问角度，或者query-Question的语义相似性不一定高

□ 视频Demo

[知识问答中的对称召回+非对称召回策略](#)

```
1 Question: AWS Clean Rooms的数据源必须在AWS上么?  
2 Answer: 对，目前必须在AWS上，而且必须是同一个region。  
3 ======  
4  
5 user : Clean Rooms的数据源可以不在同一个region么?
```

核心工作 – 知识召回效果差咋办？

✓ 向量模型微调

□ 面临问题：

- 场景数据过于垂直，通用的模型表现不佳

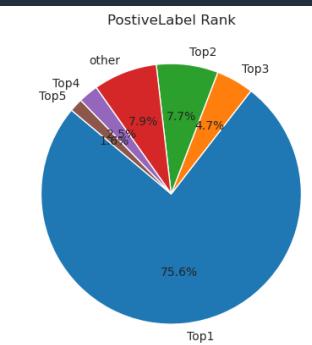
□ 价值解读：

- 在训练集上效果非常好，意味着后续可以通过持续收集用户反馈，并纳入到训练集以更新模型，使得这个效果不断扩大覆盖范围。
- 测试集上效果没有下降，反而有小幅提升，意味着训练没有破坏模型原有语义能力，对于未被训练集覆盖到的场景，模型仍能以优于原模型的性能进行服务

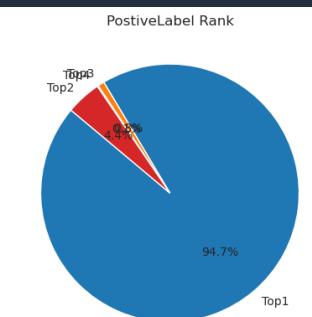
□ 技术输出

1. 代码实现 [bge_zh_research.ipynb](#)，包含训练数据构造，训练部署
2. [blog 基于大语言模型知识问答应用落地实践 – 知识召回调优（下）](#)

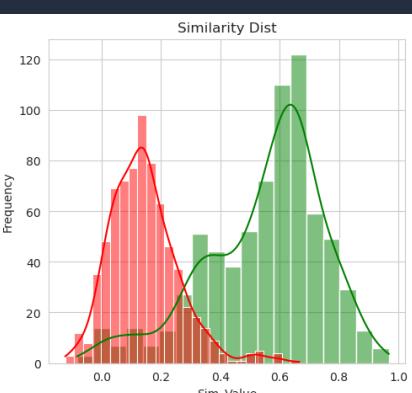
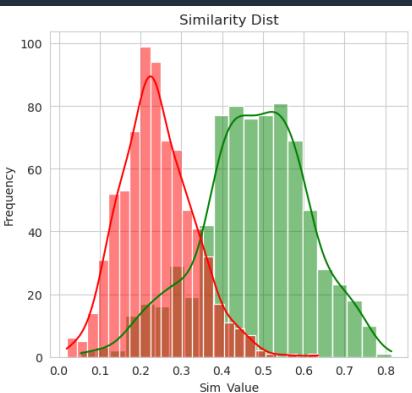
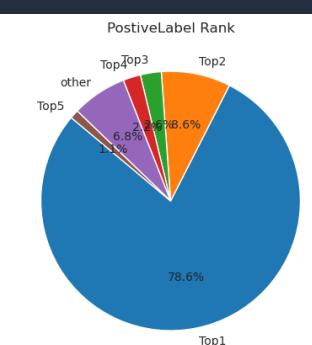
原始



微调后训练集



微调后测试集



核心工作 – 知识召回效果差咋办？

✓ 引入Rerank模型

□ 面临问题：

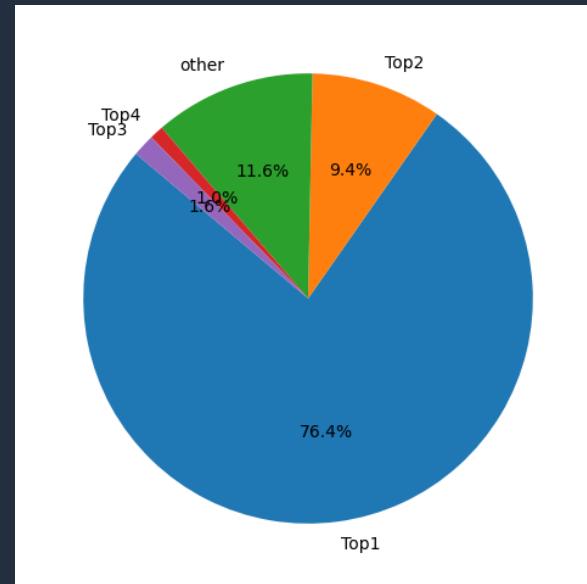
- 向量召回与倒排召回的评分体系不一致，只能随便各取 TopK，缺乏依据

□ 价值解读：

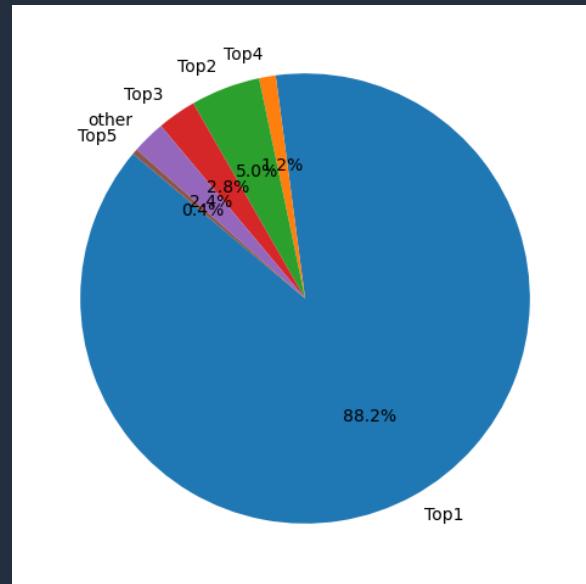
- 解决两路合并各取多少的问题。
- 进一步提高知识召回质量，作为一个独立插拔模块解决向量模型微调解决不了的问题

□ 技术输出

1. 代码实现 [bge_zh_research.ipynb](#)，包含难样本挖掘，训练部署等
2. blog [基于大语言模型知识问答应用落地实践 – 知识召回优 \(下\)](#)



无Rerank的正例排名



有Rerank的正例排名

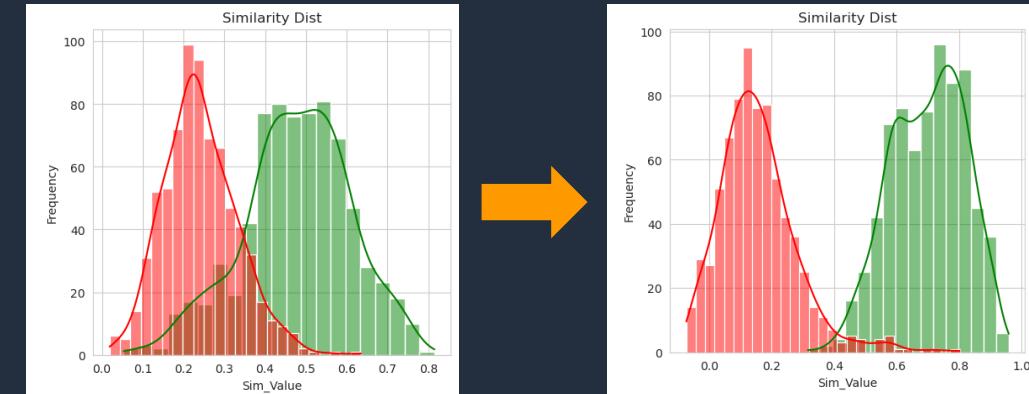
核心工作 – LLM幻觉咋办？怎么判断超出知识库范围？

✓ 根据多级召回阈值采取灵活降级策略并结合意图识别

□ 面临问题：

- 由于LLM幻觉编造一些内容可能误导用户，在某些情景下造成的负面影响非常大（跟钱，账单相关的）
- 通过Prompt提示LLM，要它依据知识不要胡说的方法是不靠谱
- 用户可能还会问一些不在服务范围的话题，滥用服务引起GPU浪费。

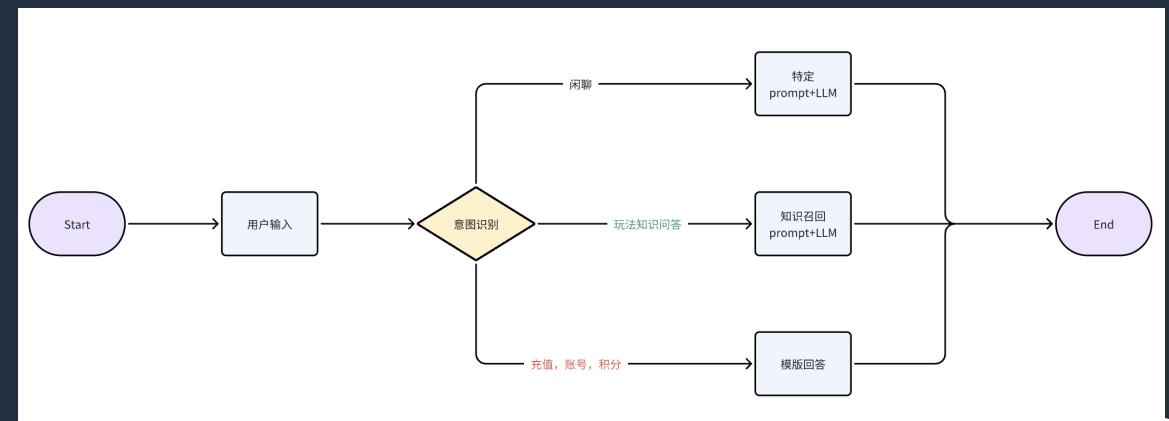
拉开值域分布



□ 有效手段：

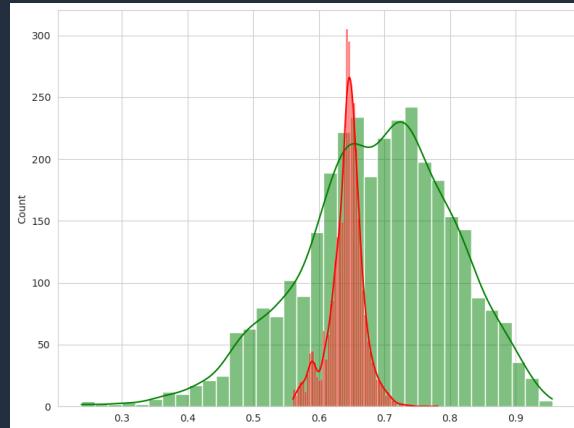
- 通过收集反馈数据，微调向量模型叠加微调Rerank模型，拉开相关召回和非相关召回的得分的值域分布。得分分为多级，最置信的走LLM，次置信的提示LLM如果不相关进行拒答，不置信的仅返回召回TopK或直接拒答。
- 意图识别进行场景分流，敏感场景避免LLM介入直接走预制答案。

意图识别避免敏感场景幻觉的示意图



核心工作 – 如何找出那个阈值?

- 实现步骤
 - 收集所有的用户正负反馈，找出应该拒答的query集合
 - 计算这些query和知识库内所有知识的相似分，统计其分布，比如可参考下图设定多级阈值(红色为拒答query的相似分分布)



- 可见区分度不足，如果上图是Rerank分数则微调Rerank模型，如果是向量模型分则微调向量模型

□ 技术输出

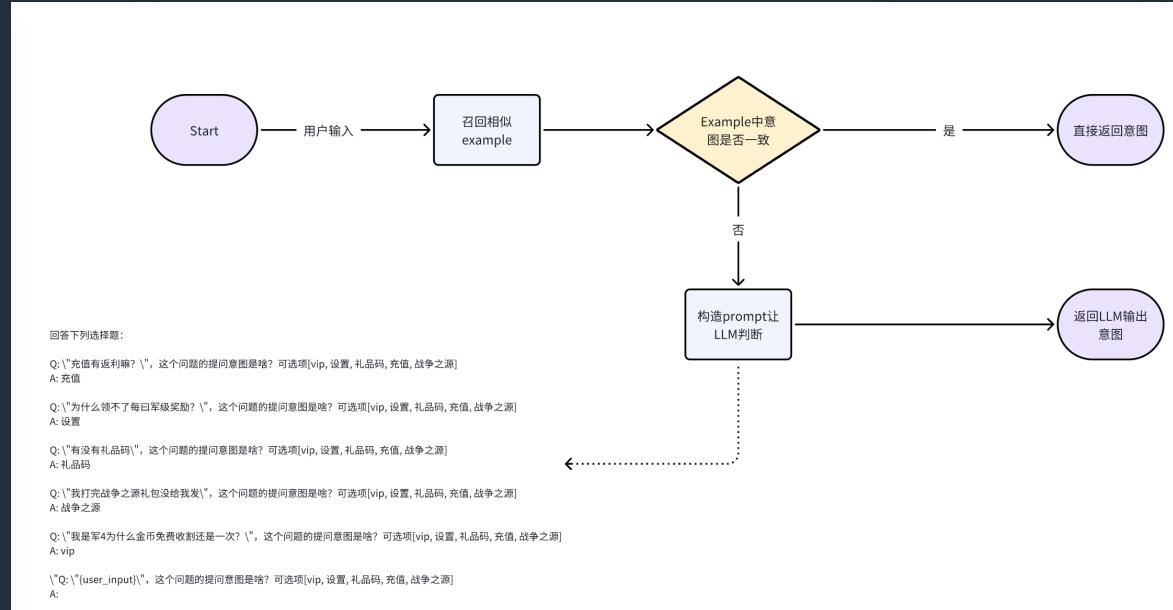
1. 参考代码 [HowToFindThreshold.ipynb](#)
2. Blog [基于大语言模型知识问答应用落地实践 – 知识召回调优 \(下\)](#)

□ 视频Demo

[知识问答中拒答策略及实现方式](#)

核心工作 – 意图识别模块

□ 实现方式



□ 优势特点

- 简单易用，不太需要太多算法能力
- 随着积累example越多越准，越不需要LLM参与，性能越好
- 生活场景的用户输入比较易用，非生活场景可能需要向量微调

□ 实践检验

采用bge_large_en向量模型 + Bedrock Claude大模型结合这个技术方案。

在南区一个IOT客户的意图识别场景的POC中，把之前客户基于OpenAI的准确率从80%+提升到90%+，获得客户认可，客户正在申请Claude权限

□ 技术输出

1. 使用说明 [README.md](#)
2. 参考代码 [intention.py](#)

□ 视频Demo

[知识问答中的意图识别方案](#)

上线必备 – 如何判断LLM能支持多少QPS?

✓ 数量级可以参考下表，具体值需要根据模型+用户prompt实际评估

易懂测试结论

- Metric 参见测试数据中的 Avg RPS(每秒响应)
- 性价比排名：
 - G5 > g4dn > p3
 - 单卡 > 多卡
- SeverSide Batch/Rolling Batch 能大幅提高吞吐量，对延迟影响小
- 决定延迟的主要是Completion token数，而prompt token数影响较小

部分测试数据

测试配置&环境			GPU机型 * 台数			
Prompt_length	max_new_tokens	Bytes	Avg RPS	P90/ms	Avg RPS	P90/ms
12	512	517	2.39	6800	1.59	11000
219	512	17	4.96	1200	4.77	1500
588	512	179	3.45	3600	2.63	5800
12	32	167	3.92	2700	3.16	4300
219	32	17	4.96	1200	4.75	1600
588	32	166	3.63	3300	2.81	5200

更多测试数据

- [若干汇总Quip\(@shishuai\)](#)
- [通义千问Quip\(@seanguo\)](#)

测试方法

Locus示例 <https://github.com/arunprsh/SageMaker-Load-Testing>

上线必备 – 大模型怎么选？中国区没G5怎么办？

✓ 选择量化模型，且优先选择GPTQ量化版本，推理速度更快，性能差别不大。

□ 面临问题：

部署百亿级模型需要G5多卡，甚至是P4系列机型，目前中国区还没有G5，单卡机型部署新出的模型都比较困难，客户使用成本高，选择范围有限。

□ 解决方案：

推荐官方GPTQ量化 或者 AWS CSDC Auto-GPTQ量化的模型，选择量化模型，且优先选择GPTQ量化版本，能获得更好的性能，和更轻微的效果损失。

推荐模型：CSDC Baichuan2 int4 GPTQ量化版，CSDC后续会量化 internLM-20B模型。

- <https://huggingface.co/csdc-atl/Baichuan2-13B-Chat-GPTQ-Int4>
- <https://huggingface.co/csdc-atl/Baichuan2-7B-Chat-GPTQ-Int4>

CSDC Baichuan 2测试结果：量化后GPU内存降低2/3, 推理速度比官方bitandbtyes int4量化版速度快113% (18.5 ->39.2 tokens/s) , 比官方BF16 快24%，性能损失差别不大。

模型版本	参数大小	说明	数据集评测结果				推理速度(A100-40G)
			agieval	ceval	cmmlu	size	
Baichuan2-13B-Chat	13B	官方原版	40.25	56.33	58.44	27.79g	31.55 tokens/s
Baichuan2-13B-Chat-4bits	13B	官方bitandbtyes量化版	39.01	56.63	57.81	9.08g	18.45 tokens/s
GPTQ-4bit-128q	13B	CSDC GPTQ量化版	38.78	56.42	57.78	9.14g	28.74(hf) \ 39.24(autogptq) tokens/s

□ 实践经验：

某客户在中国区做POC时部署Qwen 7B模型，由于g4dn.2x GPU内存较小，只能选择g4dn.12x，造成测试阶段，费用较高，性能过剩。

我们给客户更新GPTQ int4量化版本的部署之后，实测GPU内存使用率降低48%左右，token生成速度提升75% (40 -> 70 tokens/s)

。另外，根据其官方的基准测试测试结果和实际感受，4bit量化版本模型跟16BF模型比起来，效果下降幅度轻微，因此我们推荐客户使用GPTQ量化版本，提升性价比。

经验洞察 – OpenSearch查询时性能有问题怎么办?

✓ 参考以下经验总结进行参数调整

□ 面临问题：

1. 莫名其妙的查询延迟达到10s。
2. 不清楚各种参数在各种场景下该如何调整？

□ 一手经验：

- 在bulk和update的过程中不进行查询，而是结束后的一段时间后进行查询，则首次查询会有超时。需要定期预热，定期发查询，回避首次延迟问题。
- 总数据量稳定且不大（实际案例包括1400w过滤出5千-几万）的情况下，所需SLA 500ms 以下（常规100-200ms左右），Exact KNN就可以了，相对来说资源使用也更少些。
- 如果是Pre-Filter + Exact KNN：建议关闭ANN功能（"knn":false），查询会稳定
- 如果启动ANN（"knn":true），可以调整参数“knn.algo_param.index_thread_qty”为更高值，查询更稳定
- 在HNSW算法中，调整参数m相较于调整参数ef_construction 以及 ef_search 获得的收益更高。M=64，ef_construction从512增加到2048，延迟从524ms增加到836ms，召回从0.989到0.9923。ef_construction=512，m从64增加到100，延迟从524ms到457ms，召回从0.989。

□ 实践检验：

1. 客户A最开始走的ANN, 偶发资源不够超时，后面参照我们的建议调整为Pre-filter+KNN后，查询更稳定了
2. 客户B之前的实践中一直调整ef_construction，我们建议他调整参数m后，召回和latency双双提升
3. 客户C之前偶发秒级延迟，经过我们分析发现集群sizing不合理，调整后情况明显改善

经验洞察 – 该不该微调，如何微调？

- ✓ 目前RAG场景下，LLM主要是依赖Prompt中提供的知识片段做回答，用非常少量的客户数据基于Lora方法的微调意义不大，但是对于指令的理解或者回答风格有一定作用

以下是CSDC对baichuan做context instruct微调的prompt模板，

以下context内的文本内容为背景知识：

```
<context>  
{context}  
</context>  
请根据背景知识, 回答这个问题: {question}
```

Stuff方式生成答案

这是原始问题: {question}
已有的回答: {existing_answer}

现在context内的还有一些文本内容，（如果有需要）你可以根据它们完善现有的回答。

```
<context>  
{context}  
</context>  
请根据新的文段，进一步完善你的回答。
```

Refine方式生成答案

- 模型地址：<https://huggingface.co/csdc-atl/buffer-instruct-baichuan-001>

经验洞察 – BadCase 怎么排查?

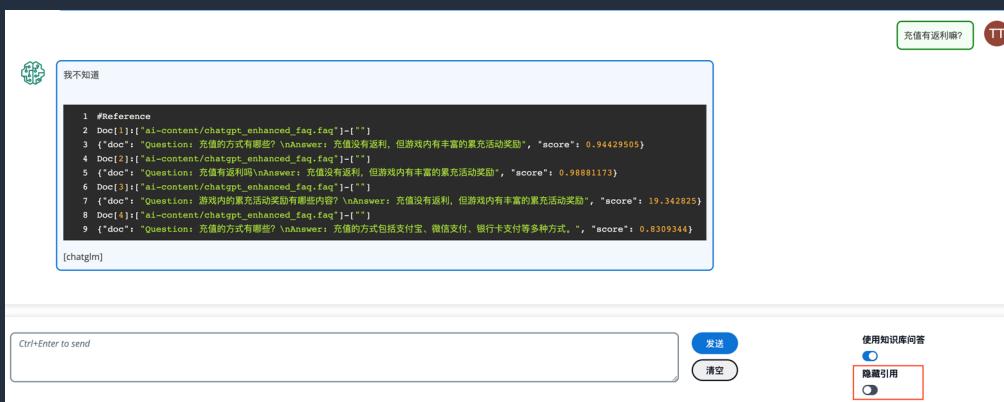
✓ 全流程日志输出并自动回流 + AOS做日志检索 + 前端信息显示

□ 面临问题:

- RAG项目不是一蹴而就的，需要长期的优化调优
- 链路中多阶段的中间结果需要独立调优

□ 前端信息显示:

- 主要用于快速了解召回质量，提升排查速度



LLM 版本

用户的query

The screenshot shows the Amazon OpenSearch Dashboard Discover tab. It displays a table of log entries with columns: Time, detect_query_type, knowledges, LLM_input, query, and opensearch_knn_doc.

识别的意图: Points to the 'detect_query_type' column, which shows values like 'QueryType_Conversation'.

构造的Prompt: Points to the 'LLM_input' column, which contains JSON objects representing the constructed prompts for different LLM versions.

各路召回的知识&打分: Points to the 'opensearch_knn_doc' column, which shows the retrieved documents and their scores.

Annotations on the right side of the table highlight specific parts of the log entries:

- "**LLM版本**" points to the LLM input section of the first log entry.
- "**用户的query**" points to the user query section of the first log entry.

The logs also contain detailed descriptions of the LLM configuration and resource usage for each query.

□ 技术材料

- [Workshop第5节](#)

经验洞察 - Serverless架构中如何实现Stream输出?

✓ 使用WebSocket API Gateway

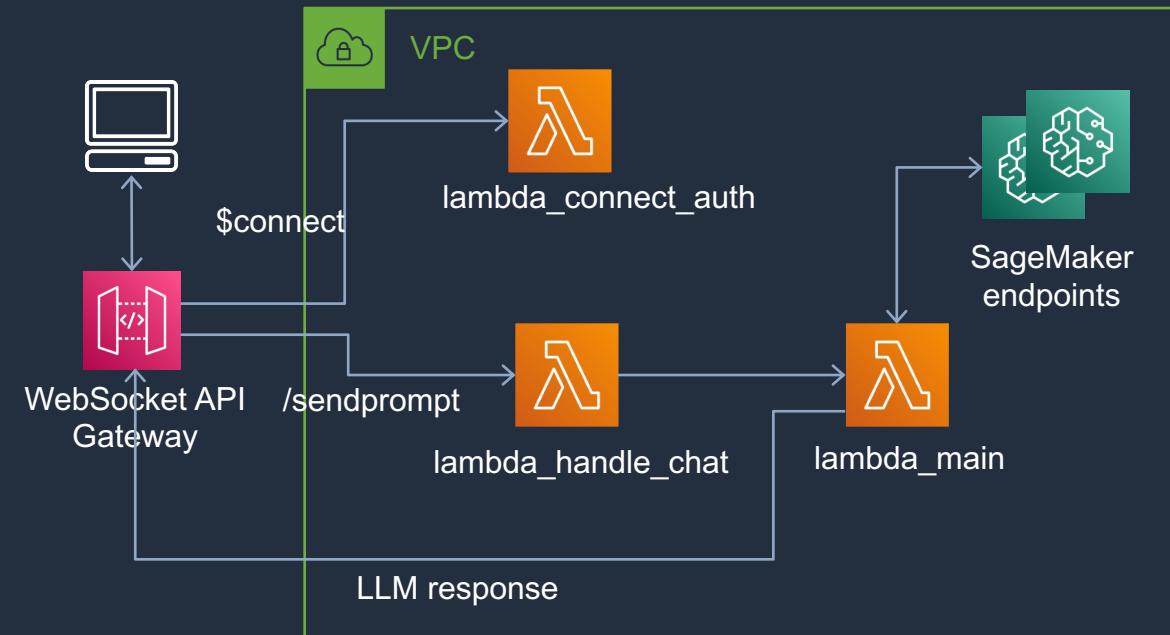
□ 面临问题:

1. Serverless 架构中一般会用到Rest API gateway, 但是 Rest API gateway 目前不支持SSE(Server Side Events), 无法实现流式响应。
2. Rest API gateway的最大响应超时为30s, 如果LLM生成的内容较长, 可能会超出这个时限。

□ 解决方法:

- 在WebSocket API Gateway实现两个Route, \$connect 和 /sendprompt, 分别和两个lambda函数集成。前者用于建立链接, 并获取connection ID, 后者用于把前端的输入和 connection ID传到后端主lambda函数, 主lambda函数中把LLM的token输出直接通过WebSocket API Gateway接口回传给前端。

➤ 架构示意图 (核心部分)



经验洞察- WebSocket和SageMaker Stream如何跟Langchain无缝集成?

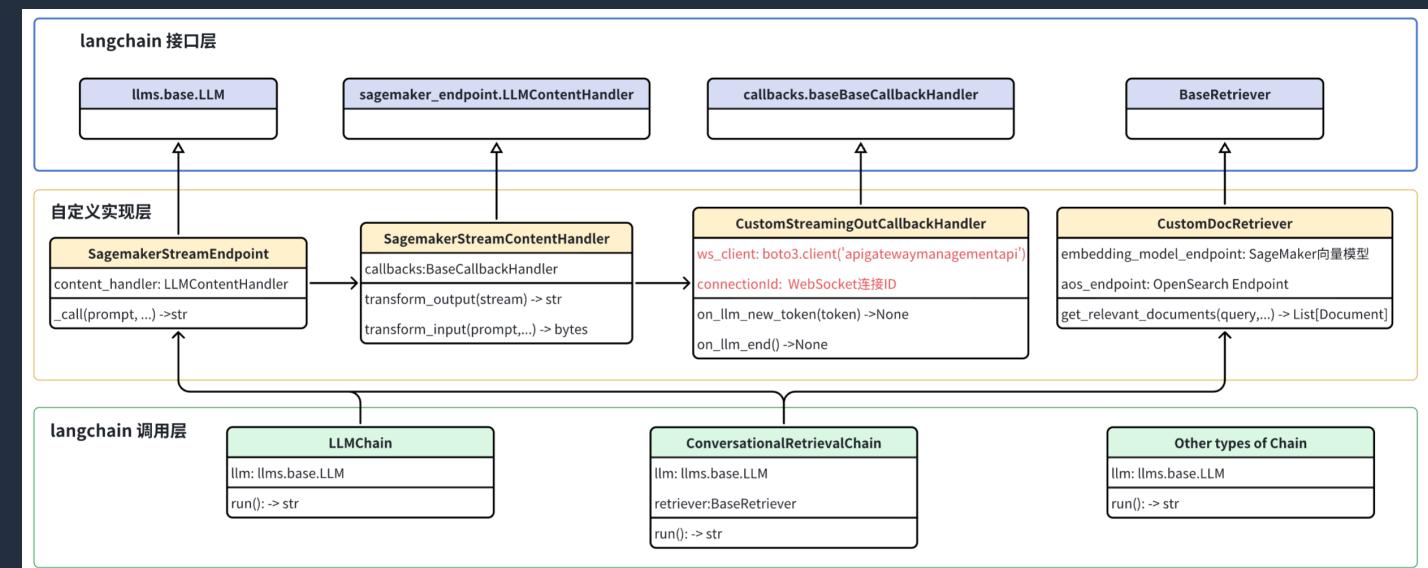
- ✓ 用Langchain接口自定义实现业务逻辑，支持Sagemaker endpoint的流式响应和WebSocket API Gateway调用，从而实现了与Langchain的无缝集成。

□ 面临问题：

为了简化LLM应用开发和多种LLM模型的集成流程，我们采用了LangChain作为基本的开发框架。Langchain框架提供了对Sagemaker Endpoint 非stream响应的标准封装，但是暂不支持Sagemaker Endpoint stream Response 和 WebSocket 输出。

□ 解决方法：

Langchain将大模型的调用接口统一封装在LLM接口层中，因此只要继承这些接口，重新实现SageMaker endpoint的stream输出处理和WebSocket API Gateway调用，就可以和Langchain实现无缝集成



□ 参考实现：代码

经验洞察 - 如何将Bedrock等商用大模型及SageMaker 部署本地模型跟Langchain无缝集成?

- ✓ 由于我们已经用Langchain接口重新实现方法, 只需要根据用户选择的模型切换Langchain调用的LLM对象即可

□ 参考实现

IF Bedrock (Claude-v1):

```
logger.info("llm_model_name : {}".format(llm_model_name))
llm = None
stream_callback = CustomStreamingOutCallbackHandler(wsclient,msgid, session_id,llm_model_name)
if llm_model_name == 'claude':
    ACCESS_KEY, SECRET_KEY=get_bedrock_aksk()

boto3_bedrock = boto3.client(
    service_name="bedrock",
    region_name="us-east-1",
    endpoint_url="https://bedrock.us-east-1.amazonaws.com",
    aws_access_key_id=ACCESS_KEY,
    aws_secret_access_key=SECRET_KEY
)

parameters = {
    "max_tokens_to_sample": max_tokens,
    "stop_sequences":STOP,
    "temperature":temperature,
    "top_p":0.95
}

llm = Bedrock(model_id="anthropic.claude-v1", client=boto3_bedrock, model_kwargs=parameters)
```

ELSE SageMaker:

```
elif llm_model_name.endswith('stream'):
    use_stream = True
    parameters = {
        "max_length": max_tokens,
        "temperature": temperature,
        "top_p":0.95
    }
    llmcontent_handler = SagemakerStreamContentHandler(
        callbacks=stream_callback
    )

    model_kwargs={'parameters':parameters,'history':[],'image':imgurl,'stream':use_stream}
    logging.info(f"model_kwargs:{model_kwargs}")
    llm = SagemakerStreamEndpoint(endpoint_name=llm_model_endpoint,
        region_name=region,
        model_kwargs=model_kwargs,
        content_handler=llmcontent_handler,
        endpoint_kwarg={'CustomAttributes':'accept_eula=true'} ##for llama2
    )
```

ELSE ChatGPT:

```
elif llm_model_name.startswith('gpt-3.5-turbo'):
    use_stream = True
    global openai_api_key
    llm=ChatOpenAI(model = llm_model_name,
                    openai_api_key = openai_api_key,
                    streaming = True,
                    callbacks=[stream_callback],
                    temperature = temperature)
```

经验洞察 - 如何实现知识问答结果的图文并茂?

- ✓ 将知识文档中的超链接和图片，转成markdown格式，通过型在输出的时候可以输出markdown格式的内容，通过前端渲染实现图文并茂

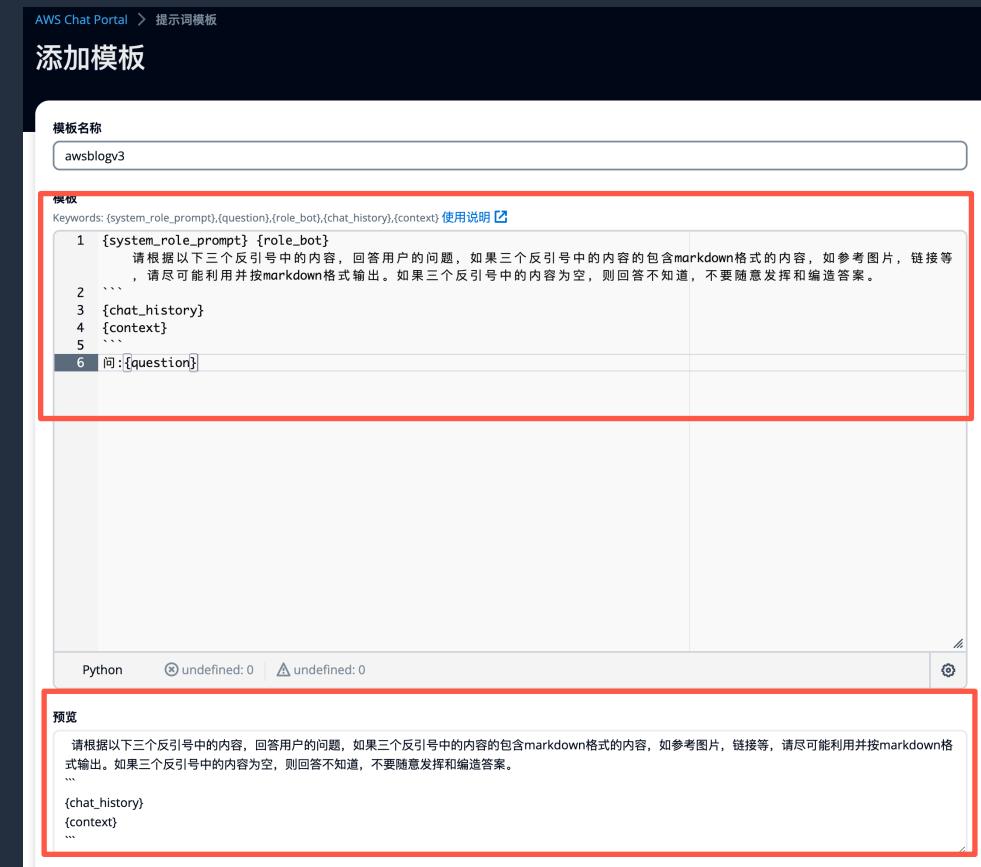
□ 面临问题：

LLM的输出为语言文本，不会凭空生图片，链接等信息，而很多知识内容是包含有图片和超链接的（网站，wiki等），我们需要知识库也能输出相关的图片和链接。

□ 解决方法：

我们将知识文档中的超链接和图片，转成markdown格式，设置提示词模板，让LLM识别并输出原有markdown格式的内容，通过前端渲染实现图文并茂

□ 参考实现：代码



经验洞察 - 知识问答结果的图文并茂-效果展示

- 口 语料来源 AWS 官方博客: [使用 Amazon SageMaker Hugging Face估计器和模型并行库微调 GPT-J](#)

AWS 博客主题 版本 ·

SageMaker 模型并行库带 SageMaker Python SDK。您需要安装 SageMaker Python SDK 才能使用该库，该工具包已经安装在 SageMaker notebook 内核上。要使 PyTorch 训练脚本利用 SMP 库的功能，需要进行以下更改：

- 首先使用 `smp.init()` 调用导入和初始化 SMP 库。
- 初始化后，使用 `smp.DistributedModel` 封装器封装模型，并使用返回的 `DistributedModel` 对象代替用户模型。
- 对于优化器状态，在模型并行库中使用 `smp.DistributedOptimizer` 封装器，使 SMP 能够保存和加载优化器状态。前向和后向传播逻辑可以抽象为一个单独的函数，并向该函数添加 `smp.step` 装饰器。从本质上讲，前向传播和后向传播需要在函数内部运行，上面放置 `smp.step` 装饰器。这样，`smp` 就能在启动训练作业时，将输入函数的数据拆分为固定数量的微批次。
- 接下来，我们可以使用 `torch.cuda.set_device` 然后调用 `to()` API，将输入张量移动到当前进程使用的 GPU。
- 最后，对于后向传播，我们替换了 `torch.Tensor.backward` 和 `torch.autograd.backward`。

请参阅以下代码：

```

@smp.step
def train_step(model, data, target):
    output = model(data)
    loss = F.nll_loss(output, target, reduction="mean")
    model.backward(Loss)

    return output, loss

with smp.tensor_parallelism():
    model = AutoModelForCausalLM.from_config(model_config)

model = smp.DistributedModel(model)
optimizer = smp.DistributedOptimizer(optimizer)

```

AWS 博客主题 版本 ·

张量并行技术
SageMaker 使用张量并行技术对模型进行分层，以便同时在多个设备上运行各个层或模块，从而提高训练速度和性能。下面是一个简单的例子，说明如何使用 SageMaker 的张量并行技术将模型分为两个部分（一部分由两个 GPU 分布），并分布在两个 GPU 之间。本例中的数据并行度为八，因为模型并行配置将每层的张量拆分为八份。

该功能的好处是，您可以选择将张量并行应用于哪些层或子集。要了解更多关于 PyTorch 的张量并行和其他节省内存的功能，以及如何设置管道和张量并行性的组合，请参阅 PyTorch 的 SageMaker 模型并行库的“[扩展功能](#)”。

知识片段

知识片段

机器人回答

机器人回答

Sagemaker如何使用张量并行技术对模型分层

SageMaker 使用张量并行技术对模型进行分层的步骤如下：

- 首先，您需要在 SageMaker 中使用 SMP 库来定义和训练模型。您可以使用 `smp.step` 装饰器定义训练步骤，如下所示：

```

@smp.step
def train_step(model, data, target):
    output = model(data)
    loss = F.nll_loss(output, target, reduction="mean")
    model.backward(Loss)

    return output, loss

```

- 接下来，您需要在使用 SMP 库的上下文中创建模型。您可以使用 `with smp.tensor_parallelism()` 语句来指定使用张量并行性，如下所示：

```

with smp.tensor_parallelism():
    model = AutoModelForCausalLM.from_config(model_config)

```

- 然后，您可以将模型包装在 `smp.DistributedModel` 中，以便在多个 GPU 上进行分布式训练，如下所示：

```

model = smp.DistributedModel(model)

```

- 最后，您还可以使用 `smp.DistributedOptimizer` 来包装优化器，以实现分布式优化，如下所示：

```

optimizer = smp.DistributedOptimizer(optimizer)

```

通过这些步骤，您可以使用 SageMaker 的张量并行技术对模型进行分层，以实现在多个 GPU 上的并行计算和训练。有关更多详细信息，请参阅 [SageMaker 模型并行库的扩展功能](#)。

Sagemaker张量并行技术是怎么对模型分层的

SageMaker 的张量并行技术可以将模型分为多个层或 nn.Modules，并将它们分布在不同的设备上以实现并行计算。通过使用张量并行性，可以在多个设备上同时运行各个层或模块，从而提高训练速度和性能。

以下是一个简单的示例，说明如何使用 SageMaker 的张量并行技术将模型分为四个层，并实现双向张量并行性 ("tensor_parallel_degree": 2)。每个模型副本的层被分成两半，并分布在两个 GPU 之间。此外，该示例中还使用了数据并行性 ("pipeline_parallel_degree": 1 和 "ddp": True)，使得数据在多个 GPU 上进行并行处理。

通过使用 SageMaker 的张量并行技术，您可以选择将张量并行应用于模型的哪些层或子集。要了解更多关于 PyTorch 的张量并行和其他节省内存的功能，以及如何设置管道和张量并行性的组合，请参阅 PyTorch 的 SageMaker 模型并行库的“[扩展功能](#)”。

其他信息

Workshop: <https://catalog.us-east-1.prod.workshops.aws/workshops/158a2497-7cbe-4ba4-8bee-2307cb01c08a>。中英文两个版本，中文是AWS的FAQ做知识库，英文是金融相关的文档做知识库。前者比较成熟，几百人次做过。后者包含了更多的东西，包括PDF文档拆分(表格信息抽取)，CommonCrawl 的信息抽取，LLM的Finetune等方面。

Github Repo: <https://github.com/aws-samples/private-llm-qa-bot>

Blogs:

[<基于大语言模型知识问答应用落地实践 – 知识库构建（上）>](#)

[<基于大语言模型知识问答应用落地实践 – 知识库构建（下）>](#)

[<基于大语言模型知识问答应用落地实践 – 知识召回调优（上）>](#)

[<基于大语言模型知识问答应用落地实践 – 知识召回调优（下）>](#)

视频: [技术细节讲解](#) 成本评估: [AIChat cost evaluation: Chatbot Solution Cost](#)

Thank you!