**那你讲一下rag的话有哪几个步骤？**

rag的步骤主要分为两个部分，先做准备，然后再做运行。

首先是把文本分割成小块，接着用编码模型把这些文本块嵌入成向量，然后给这些向量建立一个索引，最后给大模型写提示，告诉它要根据后续搜索到的上下文来回答用户的问题。

到运行阶段的时候，就和用文本嵌入时相同的编码器，把用户的查询也转化成向量，然后根据我们的索引搜索这个查询向量，找到前k个最相关的结果，然后再从我们的向量数据库里边调出对应的文本块，把这些文本块作为我们的上下文放入大模型的提示里，最后大模型总结就能输出答案了。

**那你讲一下这个rag的标准流程，在实际的项目当中里面你用过哪些优化的一些技巧呢？**

实际项目里主要的优化也是两个方面，首先就是召回环节用了多路召回，像稀疏召回、语义召回、字面召回这些都用过了。不过多路召回会有截断和分数对齐的问题，所以我们在召回之后会加一个重排序阶段，既能减少召回的数，又能提高召回的质量。

二是针对模型做微调，比如我们的embedding模型、再重排序模型还有生成模型，会根据系统问答的这个指标情况，做进一步的针对性微调，让效果能够更好一点。

**你刚才提到这个评价的指标，那你讲一下这个rag一般是怎么样去做这个效果的这个评估的？**

rag的评估主要针对检索和生成的两个环节。检索环节用的指标有MRR﻿（平均倒排率：查询（或推荐请求）的排名倒数），hits rate（﻿命中率：前 k 项中，包含正确信息的项的数目占比），还有我们的这个NDCG的排序指标，这些指标都挺不错的，这些指标都能判断到内容准不准，排序合不合理。

生成环节的评估主要分为三类，一类是量化指标，比如我们的﻿Rouge-L、文本相似度、关键词重合度，这些东西都是有用的，看我们生成内容和正确答案的匹配度。

第二就是多样性，检查我们的模型能不能生成多种合理又相关的答案，避免只会说一种。

第三是人类评估，一般让项目的产品经理、测试人员或者说内测用户来我们一起来做，给模型回答的质量、准确度和连贯性都进行一个打分。

**那你觉得这个rag的话，为什么会出现这种幻觉的问题啊？**

rag的幻觉其实也分为两类吧，它的原因也是不一样的。

第一类首先是生成结果和数据源对不上，可能是训练数据和源数据没有对齐，有偏差，也可能是编码器没有理解好内容，都有可能，或者解码器甚至在生成策略的时候也可能会出错。

第二类是用户问的问题超过大模型的认知范围了，语言模型本身就不懂这个问题，它自然就容易瞎答，所以出现幻觉了。

**那针对这种幻觉的问题啊，你有什么解决的这种思路呢？**

主要还是两个思路吧，第一个就是优化数据源和模型的逻辑，找到更精准的知识库，删掉虚假数据，减少偏差，还可以增加我们的纠偏规则，比如用react的这个思想啊，让大模型输出结果后反思一下，看看是不是跟上下文一致。

第二个就是集成我们的知识图谱，不再只靠向量数据库匹配，召回时既看文档块，也看知识图谱的三元组，用图谱里的结构化数据帮rag系统提升我们的推理能力，从而减少幻觉。

**那在实际的这种项目当中的话，有没有遇到这种各种边界的这种case，一般的话是怎么解决的呢？**

会按case的类型分情况处理吧，我们先把case分类。

第一种就是无效问题，数据库没有相关答案的问题，这个时候得先做做一个这个准入判别，要么用二分类模型，要么用大模型加prompt的方式判断要不要回答，要是无效问题，就让大模型输出预设的兜底话术。

第二种就是处理随时间变化的问题，这种问题答案容易过期，不好保证准确，所以在推理模块加了规则和提示工程，让模型不确定的时候就说根据当前的信息不足以回答这个问题，就让它别乱猜嘛。

第三种就是格式错误，比如模型没按预定的这个呃格式生成答案，导致没法解析，这个时候我们就准备了一个备份的代理大模型，一旦解析失败，就用这个备用的代理大模型直接生成一个简洁准确的总结就可以了。

﻿**RAG 的检索阶段，常见的向量检索模型有哪些？**

ANN算法

乘积向量

暴力搜索

hnswlib

KD树