北京師範大學
BEIJING NORMAL UNIVERSITY

RAG 智能问答系统

姓名 何嘉凯　　学号 202211079261

姓名 何芯玥　　学号 202211079126

姓名 黄子豪　　学号 202211079185

2025 年 6 月 15 日

目录

1 引言

1.1 研究背景

随着自然语言处理技术的迅猛发展，智能问答系统（Question ing, QA）已经成为人工智能的重要应用方向之一。其核心目标是让计算机能够自动理解用户以自然语言形式提出的问题，并基于相关知识准确作答。

早期问答系统多为任务驱动型闭域系统，如 60 年代的 BASEBALL 系统、LUNAR 系统，用户问题需转化为结构化查询语句（如 SQL），并从后端数据库中匹配结果。这类系统结构清晰但泛化能力差，难以处理开放问题。20 实际 90 年代开始，随着搜索引擎与文本检索技术的发展，问答系统如 IBM Watson 逐渐从结构化知识转向非结构化文本。系统先通过信息检索模块（IR）从大量文档中找出相关段落，再由阅读理解模块（RC）在候选片段中找出答案。2000 年后，TREC QA、SQuAD 等数据集推动了该方向快速发展。进入深度预训练时代，BERT、GPT、T5 等大型语言模型具备了“理解 + 生成”的强大能力，极大简化了问答系统架构。如今的开放域问答不再依赖人工构建知识库，而是通过模型自身或结合检索模块，直接完成问答任务。

尽管大模型拥有强大的生成能力，但面对专业、细粒度、实时性强的任务场景，仍需借助外部知识。其中，检索增强生成方法（RAG）可以保证答案内容的可追溯性，以避免大模型“幻觉”问题；同时，RAG 支持文档的高效语义搜索，易于实现文档问答、法规问答、产品手册问答等实际场景。因此，RAG 成为当前主流、有效的问答系统范式。

本次实验中，我们使用检索增强生成式问答框架，以《汽车介绍手册》PDF 文件为知识源，完成了以大语言模型（如 Qwen-72B）为核心的问答系统，并结合 LangChain 框架搭建完整的 RAG 流程，包括文档加载、切分、向量构建、检索、压缩、生成等模块。

1.2 研究目的

本实验旨在探究如何基于给定领域文档（汽车介绍手册.pdf）和大语言模型构建一个高效、准确、可溯源的问答系统，最终生成答案并进行效果的优化。主要包括以下问题：1）结合文档结构和内容特点，实现面向中文 PDF 手册的问答系统设计；2）通过引入大语言模型（LLM）提升问答的理解能力和表达能力；3）利用多路召回与重排序策略，提升文档检索的相关性与有效性；4）探索不同的优化方法，综合考察生成答案的内容质量与文献引用的准确性。

1.3 研究思路

本次实验参考 CCF 第七届 AIOps 挑战赛 [1] 方案与 Enterprise RAG Challenge 挑战赛方案 https://abdullin.com/ilya/how-to-build-best-rag/，在其方案上针对本

问题做了相应调整。

具体研究思路如下：

1. **基础问答系统设计**
   - **数据准备**：采用两种方案处理《汽车介绍手册》PDF 文件。第一种方案直接使用 PyPDFLoader 提取内容；第二种方案通过 Docling 等工具将 PDF 转换为 Markdown 文档，并进行人工校对，确保文本结构清晰、内容准确。最终选择第一种方案作为主要数据处理方式。
   - **模型完善**：基于 Qwen2.5-72B 模型设计提示词模板，结合语境学习思想，确保生成的答案符合要求。模板中明确规定了无法回答问题的场景，并提供了示例以引导模型行为。
   - **分词处理**：使用对中文语料进行分词，适配 ROUGE 和 BLEU 评测指标的计算需求。
2. **评测方法设计**
   - **ROUGE 与 BLEU 分数**：通过计算生成答案与参考答案的 n-gram 重叠度，评估答案的召回率和相似性。
   - **Reference 准确率校验**：定义命中率等指标，验证生成答案的引用准确性。
3. **优化策略**
   - **多路召回**：结合 BM25 检索与向量相似度检索（如 m3e-small），通过粗排 RRF（倒数排名融合）实现混合检索，提升召回效果。
   - **重排序**：采用 BGE-Reranker 系列模型（如 BGE-Reranker-Base）对召回结果进行精细化排序，确保最相关的文档优先传递给 LLM。
   - **其他优化**：尝试扩展嵌入模型（如 M3E-Base、xiaobu-embedding-v2 ）和 Reranker 模型（如 BGE-Reranker-Large），探索不同组合对系统性能的影响。
4. **实验分析**
   - **粗排结果**：对比不同数据处理方式和分块策略对 ROUGE、BLEU 分数的影响。
   - **多路召回与重排序**：分析优化策略对答案质量的提升效果，讨论其有效性和改进空间。
   - **其他优化效果**：全面评估各项优化策略的贡献与局限性。
5. **结论与展望**
   - 总结系统性能表现及优化成果，提出未来研究方向，如更先进的召回算法、跨领域适应性增强等。

1.4 人员工作安排

- 何嘉凯：负责主体代码编写、Prompt 设计、超参数与模型设置、文档编辑校对、报告撰写

- 何芯玥：负责结果评测、PDF 转 Markdown 方式探索与设计、运行实验、报告撰写、PPT 制作
- 黄子豪：负责结果分析、运行实验、报告撰写

2 研究方法

2.1 基础问答系统设计

2.1.1 数据准备

《汽车介绍手册》是一份双栏的 PDF 手册，包含引导语、目录和各章节。其中，每个章节的组成形式大致如下：

- 一页标题；
- 一页空白页；
- 随后开始正文部分。

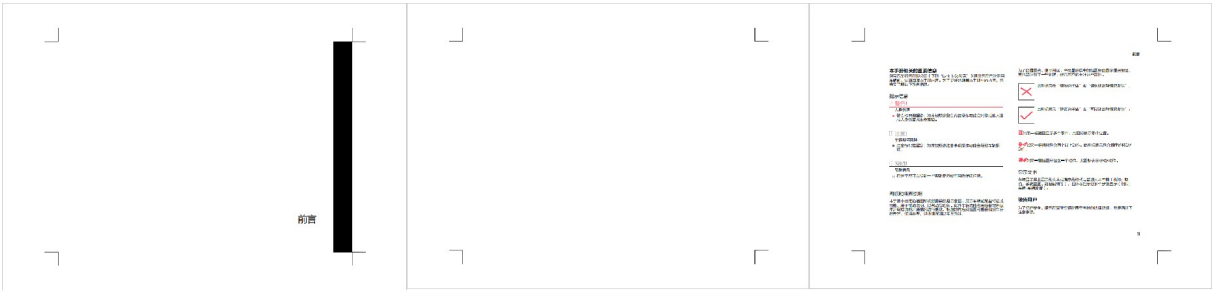

图 1: 《汽车介绍手册》概览

我们小组采用了两种方案对此文档进行处理，第二种方案我们在后续优化中没有采用 (后续会解释原因)，但该方法仍是最广泛有效的优化方式，对实际场景存在一定意义，故我们仍在此进行详细介绍：

1. 使用 LangChain 自带的 PyPDFLoader 模块，通过 PyPDF 库进行 PDF 内容提取。这个方法比较简单实用，而且效果不差；
2. 考虑到手册本身为双栏结构，且图片与文字混杂 (文字中甚至夹杂小图标)，这会使得 PDF 解析存在困难，有可能出现错位、颠倒、文字错误等问题。因此，我们想先对原有的 PDF 进行解析，随后进行内容校对，再将校对完的内容重新做 Embedding。有一个可编辑的文档会是好的选择，我们希望将原本的 PDF 转为 Markdown 文档，方便进行编辑校对。我们使用Docling对原本的 PDF 文档进行解析，得到初步的 Markdown 文档，再通过人工校对，得到结构化的干净的 Markdown 文本。

Docling 是 IBM 公司开发的一个功能强大的文档解析和转换工具，能够将 PDF, DOCX, PPTX, XLSX, Images, HTML, AsciiDoc 和 Markdown 文档转换为 HTML、Markdown 和 JSON 格式。支持 OCR（EasyOCR、Tesseract 等），安装配置简单。

图 2: PDF 转 Markdown 文档

此外，Docling 还具有出色的表格提取能力，通过使用专门用于表格结构识别的 AI 模型，可以完整地将 PDF 的表格转化为 Markdown 格式的表格，以供 LLM 向用户重现相同的表格，并且将其以人类可读的格式呈现。

图 3: 转化表格

将 PDF 转化为 Markdown 文档还有一个好处，我们可以使用标题将文档分成连贯的片段。在读取 PDF 文档并将其转换为 Markdown 后，我们可以使用 LangChain 的 RecursiveCharacterTextSplitter 根据特定的 Markdown 语法进行分块。

```
text_splitter = RecursiveCharacterTextSplitter.from_language(
    language=Language.MARKDOWN,
    chunk_size=300,
    chunk_overlap=100,
)
```

```
documents = text_splitter.create_documents(texts=[docling_text])
```

最后，我们检查了所有问题，发现所有问题均与图片内容无关，且图像无法给问题回答提供帮助，故我们将所有图像进行了剔除，得到最终的 Markdown 文档。

2.1.2 基于 Qwen2.5-72B 的模型完善

此部分主要基于 Qwen2.5-72B 模型进行 Prompt Engineering 的实践。结合标准答案的设置和语境学习思想，我们设计如下提示词模板：

基于以下已知信息，请专业地回答用户的问题：
已知内容：{context}
问题：{input}

要求：
1. 不要乱回答，如果无法从已知信息中找到答案，请回答“结合给定的资料，无法回答问题。”
2. 诚实地告诉用户。
3. 结合内容回答，不要输出'#' 和'*' 等符号，输出内容简洁。
4. 当回答“什么是 xxx”的问题时，只回答该术语定义即可。特别的，当询问以下特定术语时，请回答“结合给定的资料，无法回答问题。”，术语包括：“用车前准备”、“装载货物”、“上车和下车”、“驾驶前的准备”、“仪表和灯光”、“安全出行”、“启动和驾驶”、“驾驶辅助”、“空调”、“泊车”、“中央显示屏”、“OTA 升级”、“Lynk&Co App”、“高压系统”、“保养和维护”。[a]

example1: 操作多媒体娱乐功能需要注意什么?
1: 操作多媒体娱乐功能时，确保将车辆停驻在安全地点，将挡位切换至驻车挡（P）并使用驻车制动。

example2: Lynk & Co App 要多少钱?
2: 结合给定的资料，无法回答问题。

example3: 如何更换车内的后视镜?
3: 结合给定的资料，无法回答问题。

[a]对于这条 prompt，我们在 3.3 并没有用到，我们后续会解释原因

2.1.3 针对中文语料的分词处理

在评测中，由于需要使用 Rouge 与 BLEU 指标对 RAG 方案性能进行测试，因此需要进行 gram 的定义。在中文语料中，单个 gram 通常设置为词语粒度，如 “我喜欢自然语言处理”，对应的 1-gram 分别为 [’ 我’,’ 喜欢’,’ 自然语言处理’]。我们使用 jieba 进行中文分词，便于后续的 Rouge 分数与 BLEU 分数计算。

2.2 评测方法

2.2.1 ROUGE 分数与 BLEU 分数计算

为了客观评估自动问答系统中生成答案的质量，本实验采用了 ROUGE 分数和 BLEU 分数作为自动评测指标，分别用于衡量生成内容与参考答案之间的重合度与质量接近度。

ROUGE 是一组召回导向的自动评测指标，主要用于自动摘要和问答系统中，用于评估生成文本与参考文本之间的重合程度。我们计算了其中的 ROUGE-N 指标（包括 ROUGE-1 和 ROUGE-2），并以召回率作为最终结果：

$$\text{Rouge} - \text{N}_{\text{Recall}} = \frac{\text{重叠的 n-gram 数量}}{\text{参考文本中的 n-gram 数量}}$$

其中，n-gram 的大小由参数 n 决定。例如，n=1 表示 unigram（单个单词或字符），n=2 表示 bigram（两个连续的单词或字符）。计算中将文本分割成连续的 n 个单词或字符的序列，然后统计每个 n-gram 在文本中出现的次数。由于本实验语料为中文语料，我们将 gram 设置为词语的粒度。

BLEU 通过计算 n-gram 的重叠程度自动评估生成文本与参考文本的相似性。对于每个 n-gram，计算生成答案中 n-gram 和参考答案 n-gram 的精确匹配数，用于计算精确准确率：

$$P_n = \frac{\text{匹配的 n-gram 数量}}{\text{候选译文中的 n-gram 总数量}}$$

为了惩罚候选译文过短的情况，引入简短惩罚因子 BP：

$$BP = \begin{cases} 1 & \text{如果候选译文长度} \geq \text{参考译文长度} \\ \exp\left(1 - \frac{\text{参考译文长度}}{\text{候选译文长度}}\right) & \text{如果候选译文长度} < \text{参考译文长度} \end{cases}$$

BLEU 分数是所有 n-gram 精确率的加权几何平均值，乘以简短惩罚因子 BP。公式如下：

$$\text{BLEU} = BP \times \exp\left(\sum_{n=1}^{N} w_n \log P_n\right)$$

其中，常用 $w_n = \frac{1}{N}$ 作为权重。

由于 Rouge 库是基于英文语料进行设计的，因此需要进行中文适配。我们使用了 Rouge-Chinese 库，这是一个对中文语料专门适配后的测试库，使用方法与 Rouge

库基本一致。得到 LLM 后，我们将 LLM 与标准通过 JIEBA 分词得到 gram，随后使用该进行 ROUGE 分数计算。对于 BLEU 分数，我们使用 `nltk.translate.bleu_score` 进行计算，考虑了 1-gram 到 4-gram 的精确匹配，并使用了平滑方法来避免零概率问题。

2.2.2 Reference 准确率校验

我们通过比较参考答案和预测答案的页码是否匹配判断预测是否正确，计算所有回答的准确率。

2.3 基本符号介绍

为了书写方便，先介绍一些组件编号：

数据 ⓪ 代表使用未处理的 PDF 文件，① 代表使用经过人为调整后的 markdown 文件。

粗排 ⓪ 代表 M3E-Small，① 代表 M3E-Base，② 代表 xiaomi-embedding-v2，③ 代表 Gte_Qwen2-7B-instruct。

重排 ⓪ 代表 BGE-Reranker-Base，① 代表 BGE-Reranker-Large，② 代表 BGE-Reranker-v2-m3。

融合 ⓪ 代表相似度检索与 BM25 检索权重为 [0.5,0.5]，① 代表相似度检索与 BM25 检索权重为 [0.3,0.7]。

2.4 基本检索 (粗排)

粗排并不是我们实验的重点，因此我们仅使用 m3e-small 进行粗排，即仅根据文本嵌入的向量相似度进行检索. 我们对 PDF 和 Markdown 在同样参数下的粗排效果进行简要对比。

m3e 是 **MokaAI** 开发的一个强大的文本嵌入模型，其全称为 **Moka Massive Mixed Embedding**[2]。其“Massive”体现在它在千万级（2200 万 +）中文句对数据集和 1450 万英文三元组数据集上进行训练，使其在处理大规模文本数据方面表现出色。“Mixed”则表示它支持中文和英文双语的同质文本相似度计算和异质文本检索。M3E 的目标是提供一个通用的文本嵌入模型，能够覆盖多种应用场景，例如同质句子相似度判断和异质文本检索。该模型使用 `hfl` 实验室的 `Roberta` 系列模型进行训练，并通过 in-batch 负采样的对比学习方式在句对数据集上进行训练。它同样兼容 `sentence-transformers` 库，易于使用。核心能力包括文本相似度计算、文本检索、文本分类，并具备一定的跨语言检索能力。M3E 提供 `small` 和 `base` 两个版本，它广泛应用于需要处理中文和英文文本的语义匹配、信息检索、推荐系统、智能问答等场景。

向量相似度检索是一种通过将文本（如单词、句子、文档等）转换为向量表示，并利用这些向量之间的相似度来进行检索的方法。在向量空间模型中，文本被表示为高维

向量，文本之间的相似性可以通过计算它们在向量空间中的距离或角度来衡量。

余弦相似度是向量相似度检索中最常用的一种相似度度量方法，特别适用于文本检索。它通过计算两个向量之间的夹角余弦来衡量它们的相似度。余弦相似度的值范围从 -1 到 1，其中 1 表示两个向量完全相似，-1 表示完全相反，0 表示没有相似性。

余弦相似度的公式为：

$$\text{cosine similarity}(A, B) = \frac{A \cdot B}{\|A\|\|B\|}$$

其中：

- $A \cdot B$ 是向量 A 和向量 B 的点积；
- $\|A\|$ 和 $\|B\|$ 分别是向量 A 和向量 B 的欧几里得范数（即向量的模）。

余弦相似度关注的是两个文本的方向（语义相似性），而不是文本的长度。因此，它能够有效地衡量不同长度的文本之间的相似度。其计算简单，特别适用于大规模文档检索。

检索流程　密集检索器对于一个查询 q，具体的文档检索流程如下：

1. 文档编码：将所有文本块输入模型进行编码得到表征，存入Chroma向量数据库
2. 查询编码：利用查询提示模板将 q 转换为 Embedding 模型的查询输入，利用模型进行编码
3. 相似度召回：在检索时，使用余弦相似度进行匹配，召回 Top-k 个文本块。

2.5 优化策略

2.5.1 多路召回

我们将 **BM25 检索**与**向量相似度检索**结合，实现基于**粗排 RRF 融合**的混合检索，作为我们的多路召回方式。此外，我们设置了三组不同的权重进行对照试验，探究不同权重对结果可能带来的影响。

BM25（Best Matching 25）是一种基于概率论的经典信息检索模型，广泛应用于搜索引擎和文本检索系统中。BM25 属于一种基于词频（TF）和文档频率（DF）的加权模型，其核心思想是通过计算查询词与文档之间的匹配程度，来评估文档的相关性。

BM25 的计算公式如下：

$$\text{score}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{\text{TF}(q_i, D) \cdot (k_1 + 1)}{\text{TF}(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

其中：

- q_i 表示查询词 q 中的第 i 个词；
- $\text{TF}(q_i, D)$ 是查询词 q_i 在文档 D 中的词频；
- k_1 和 b 是模型的调节参数，通常设置为 $k_1 = 1.5$ 和 $b = 0.75$；
- $|D|$ 是文档 D 的长度（即文档中的词数）；

- avgdl 是文档集合中的平均文档长度;
- $\mathrm{IDF}(q_i)$ 是逆文档频率（IDF），用于衡量词 q_i 的重要性，计算公式为:

$$\mathrm{IDF}(q_i) = \log\left(\frac{N - \mathrm{DF}(q_i) + 0.5}{\mathrm{DF}(q_i) + 0.5} + 1.0\right)$$

其中 $\mathrm{DF}(q_i)$ 是包含查询词 q_i 的文档数量，N 是总文档数量。

BM25 模型的优点是简单高效，且能够处理长文档和短文档之间的差异。通过参数 k_1 和 b 的调节，BM25 能够平衡词频和文档长度的影响。具体实现中，我们使用 LangChain 自带的 BM25Retriver 进行 BM25 检索。

粗排 RRF(倒数排名融合):

RRF 融合的基本思想是：对于每个文档，计算它在每个检索器中的排名位置的倒数，然后将倒数值加权求和。最终，根据求和后的得分对文档进行排序。

假设我们有 n 个检索器，每个检索器返回了一个文档排名列表。在每个排名列表中，文档的排名位置为 r_i，其中 r_i 表示该文档在第 i 个检索器中的排名（排名从 1 开始）。RRF 的计算公式如下:

$$S_{final}(d) = \sum_{i=1}^{n} \frac{1}{r_i(d) + k}$$

其中:

- $S_{final}(d)$ 是文档 d 在最终排序中的得分。
- $r_i(d)$ 是文档 d 在第 i 个检索器中的排名。排名越小，文档越相关。
- k 是一个常数，用于平滑处理排名较大的情况，通常 $k = 60$。
- n 是参与融合的检索器数量。

其优点包括:

- 简单有效：RRF 方法相对简单，不需要对文档内容进行深度分析，仅基于检索结果的排名信息进行融合。
- 增强多路检索性能：RRF 能够有效结合多个检索器的优势，弥补单一检索器的缺点，从而提高检索结果的相关性和准确性。
- 无偏性：RRF 采用排名的倒数作为加权依据，避免了过分依赖某一检索器的评分标准，增强了结果的公平性。

我们使用 LangChain 提供的 EmsembleRetriver 进行混合检索。

2.5.2 重排序 (重排)

重排序是信息检索系统的**第二阶段**，旨在对初步检索（召回）到的文档进行**精细化排序**，以确保最相关的文档排在结果列表的最前面。

使用重排序的原因:

- **弥补召回器不足:** 召回器（如向量搜索）为了效率，通常只进行粗略匹配，难以捕捉深层语义交互。

- **提升精度：**确保传递给用户或下游任务（如 RAG 中的 LLM）的信息是最高质量的，从而提高准确性，减少“幻觉”。

重排序的基本原理：

- **交叉编码器架构：**交叉编码器将查询和召回文档拼接成一个单一的输入序列（例如：”[CLS] 查询文本 [SEP] 文档文本 [SEP]”），然后将其送入一个 Transformer 编码器（如 BERT、RoBERTa 等）。编码器在处理这个拼接序列时，能够直接观察到查询和文档中所有词元（token）之间的交互关系。它不是单独理解查询和文档，而是同时考虑两者来判断它们之间的关联程度。经过编码后，模型通常会输出一个单一的相似性分数（例如，通过 [CLS] 令牌的输出向量，再经过一个 Sigmoid 层），直接表示查询和文档的相关性。这个分数考虑了查询和文档的整体语义和上下文，因此比简单的向量距离更准确。
- **对比双编码器：**双编码器独立地将查询和文档编码成向量，然后计算向量之间的相似度。这种方法速度快，因为文档向量可以预先计算并存储，但可能无法捕捉细致的交互，通常用于召回；交叉编码器联合编码，虽然计算成本高（需要对每个查询-文档对进行一次完整的模型推理），但其深度交互能力带来了更高的精度。

重排序有如下作用：

- **显著提升相关性：**将最相关的文档推到结果前列。
- **优化上下文质量：**为 RAG 系统中的 LLM 提供高质量、无噪声的上下文，提升生成回复的准确性和可信度。
- **提高用户体验：**帮助用户更快找到所需信息。

我们采用 BGE-Reranker 系列模型作为 Reranker[3]。BGE-Reranker 是由北京智源人工智能研究院 (BAAI) 开发的一系列高效、高性能的文本重排序 (Reranking) 模型。它们采用交叉编码器 (Cross-encoder) 架构，旨在提升信息检索、问答系统等场景中的文档相关性排序准确率。这些模型在 RAG (Retrieval Augmented Generation) 系统中尤为关键，用于在检索阶段后对召回的文档进行精细化排序，从而为大型语言模型 (LLM) 提供更优质的上下文。我们采用如下模型: BGE-Reranker-Base 进行测试。

`BGE-Reranker-Base` 是 BGE 重排序模型系列的基础版本，参数量为 278M，它在保持高效推理速度的同时，提供了良好的重排序精度，非常适合资源受限或需要快速部署的通用信息检索和问答系统应用。

2.5.3 其他优化策略

说明除了多路召回和重排序之外的其他优化尝试，例如数据增强、模型微调、更复杂的提示工程等。

1. 增强粗排准确性

我们扩展了嵌入模型，分别引入如下三种嵌入模型进行计算：

- `M3E-Base`: 此前已有介绍，不再赘述;

- xiaobu-embedding-v2 是一个高级的中文文本嵌入模型,它是对 piccolo-embedding 架构的改进版本，在 xiaobu-embedding-v1 的基础上积累了更多数据。该模型采用了一种统一的 Circle Loss 方法来处理多种中文多任务嵌入基准（CMTEB）任务，这有助于更好地利用原始数据集中的多个正例，并降低管理多个损失函数权重的复杂性。它专注于中文语言处理的优化，在中文文本的各种任务上表现出色，并且基于 sentence-transformers 框架实现，方便用户集成和使用。其核心能力包括在分类任务上表现出色、有效的文本相似度测量、鲁棒的聚类能力以及高质量的文本检索。它适用于中文语义搜索、文本聚类、分类和问答系统等需要理解中文文本语义的任务。
- Gte_Qwen2-7B-instruct 是 **阿里巴巴-NLP** 推出的 gte (General Text Embedding) 模型家族成员，在文本嵌入任务中表现卓越。该模型以 Qwen2-7B 作为基础，继承了 Qwen2 系列在自然语言处理方面的强大能力。它在 MTEB (Massive Text Embedding Benchmark) 和 C-MTEB (Chinese MTEB) 评估中，在英语和中文任务上均表现出色，且支持法语、波兰语等多种语言。模型引入了双向注意力机制，这有助于更好地理解输入文本的上下文信息，并专门对查询侧进行了指令微调以提高处理效率。它支持高达 32,000 个 tokens 的最大输入长度，能够处理较长的文档和文本。通过在涵盖不同领域和场景的庞大多语言文本语料库上进行训练，确保了模型在多种语言和广泛的下游任务中的适用性。其核心能力包括卓越的多语言文本嵌入生成、长上下文理解、高效的语义搜索和检索、跨语言文本相似度分析以及文档分类和聚类。它广泛应用于需要处理长文本、多语言文本，并进行语义搜索、文本分类、信息检索、推荐系统、问答系统等任务的场景。

在我们实验期间，阿里的通义团队推出了基于 Qwen3 的 Embedding 模型 Qwen3-Embedding[4]，该模型专门设计用于文本嵌入和排序任务。基于 Qwen3 系列的密集基础模型，它提供了各种大小（0.6B、4B 和 8B）的全面文本嵌入和重新排序模型。该系列继承了其基础模型卓越的多语言能力、长文本理解和推理技能。Qwen3 Embedding 系列在多个文本嵌入和排序任务中代表了重大进步，包括文本检索、代码检索、文本分类、文本聚类和双语文本挖掘。8B 模型在 MTEB 上取得了 SOTA。但迫于实验时间有限，无法完整地对这些嵌入模型进行测试，我们仅对某些配置情况进行了初步探索。

2. 基于 BERT-Reranker 的探索

此外，我们扩展了 Reranker 模型库，尝试比对三种 reranker 模型的效果：

- BGE-Reranker-Base，此前已提到，此处不过多赘述；
- BGE-Reranker-Large 是 BGE 重排序模型系列中的大型版本，它拥有更多的参数 (560M)，旨在通过更强大的语义理解能力和交互建模，在各种复杂查询和文档对上实现更高的重排序准确率，但代价是更高的计算资源消耗和推理延迟；
- BGE-Reranker-v2-m3[5] 代表了 BGE 重排序模型系列的第二代改进版本，参数量达到 568M，其中”m3” 指其在多任务、多语言和/或多模态数据上的训练优化，使其在处理多样化的语言和领域数据时展现出更强的泛化能力和鲁棒的重排序性能。

同样的，通义团队也推出了 `Qwen3-Reranking` 模型，但由于该模型上线时间较晚，我们未对此模型进行测试。

3 实验分析

3.1 最终结果汇总

最终，基础问答系统效果（answer1.json）、多路召回与重排序优化效果（answer2.json）和其它优化效果（answer3.json）如下表所示：

表 1: 最终结果汇总

文件名称	Rouge-1	Rouge-2	BLEU	命中率
answer1.json	60.36	48.02	41.34	48.15
answer2.json	84.46	72.95	56.21	77.78
answer3.json	86.30	75.65	54.44	76.54

3.2 密集检索粗排结果展示

3.2.1 测评分数

在粗排实验中，主要对比了使用未经处理的 PDF 文件（数据 0）和经过人工调整的 Markdown 文件（数据 1）在相同参数下的性能。实验结果如表 1 所示，分块策略均采用 chunk_size=300, chunk_overlap=100，粗排模型为 m3e-small，返回 Top-1 结果，且未进行重排。

表 2: 粗排实验结果，分块列中两个数字分别代表 chunk_size 和 chunk_overlap，序号 3 实验采用 markdwon 符号帮助分割；粗排旁边的 1 表示返回向量相似度检索的 Top-1

序号	数据	分块	粗排	重排	Rouge-1	Rouge-2	BLEU	命中率
0	⓪	300,100	⓪,1	无	60.36	48.02	41.34	48.15
1	①	300,100	⓪,1	无	66.57	52.95	38.77	-
2	①	300,100*	⓪,1	无	65.65	52.42	38.31	-

从表 1 可以看出，使用 Markdown 文档 (1-2) 作为知识源的系统在 ROUGE-1 和 ROUGE-2 分数上均优于使用原始 PDF 文件 (0) 的系统。具体而言，(1) 的 ROUGE-1 分数达到 66.57，ROUGE-2 分数达到 52.95，而 (0) 分别为 60.36 和 48.02 。这表明经过人工校对和结构化处理的 Markdown 文档，能够提供更高质量的检索内容，从而使得模

型生成的答案与参考答案在 1-gram 和 2-gram 层面有更高的重合度。另外，我们对比了传统使用 chunk_size 和 chunk_overlap 进行语块切分与结合 Markdown 标记符辅助切分的结果，发现二者无显著性差异，一方面可能是因为检索模型具有鲁棒性，对于切分差异并不敏感；另一方面，可能是因为 Markdown 标记符的语义粒度与 chunk_size 重合度高，本身的切分已经能够在一定程度上捕获到这些语义块。

然而，在 BLEU 分数方面，(0) (41.34) 略高于 (1) 的结果 (38.77)。BLEU 分数侧重于精确率，对生成答案的流畅度和语法结构有更高要求。这可能是因为虽然 Markdown 文档提供了更准确的内容片段，但原始 PDF 文档的文本可能在某些情况下更接近参考答案的表达方式，或者在分词和 N-gram 匹配上存在一些偶然的优势。

值得注意的是，由于 Markdown 文档没有页码，因此无法衡量其 Reference 准确率。而使用原始 PDF 文档的系统，其命中率为 48.15%。我们将 (0) 的结果作为 1.json 提交。

3.2.2 问题与挑战

经小组讨论，我们认为当前的基线结果有以下不足：

- **单一粗排的局限性**：仅使用 m3e-small 进行向量相似度检索（粗排），虽然能捕捉语义相似性，但可能无法全面覆盖所有相关信息，尤其是在面对多样化查询或知识源内容复杂时，召回效果存在上限。
- **缺乏重排序机制**：在粗排阶段没有引入重排序，导致召回的文档质量未经过进一步筛选和优化，直接传递给 LLM，可能会引入不那么相关的上下文，影响最终答案的生成质量和准确性。

3.3 多路召回与重排序优化效果

3.3.1 测评分数

表 2 展示了结合多路召回（BM25 检索与向量相似度检索融合）和重排序策略后的实验结果。粗排阶段 BM25 检索器和向量相似度检索器均返回 Top-5 结果。

表 3: 多路召回与重排序实验结果，粗排旁边的 5,5 表示 BM25 检索器和向量相似度检索器均返回 Top-5 结果

序号	数据	分块	粗排	重排	融合	Rouge-1	Rouge-2	BLEU	命中率
0	⓪	300,100	⓪,5,5	无	⓪	86.97	75.18	54.99	61.73
1	⓪	300,100	⓪,5,5	⓪	⓪	84.46	72.95	56.21	77.78
2	①	300,100	⓪,5,5	⓪	⓪	79.84	66.95	50.63	-

3.3.2 策略分析与讨论

对比表 1 和表 2 的数据，可以明显看出多路召回与重排序策略对系统性能的显著提升：

- **ROUGE 分数显著提升**：在⓪（原始 PDF）上引入多路召回和重排序后 (1)，ROUGE-1 从 60.36 大幅提升至 84.46，ROUGE-2 从 48.02 提升至 72.95(对比基线模型)。这表明多路召回（结合 BM25 和向量相似度检索）能够更全面地召回相关文档，而重排序（使用 BGE-Reranker-Base）则能对召回结果进行精细化排序，确保最相关的文档优先传递给大模型，从而大幅提升了生成答案的召回率和与参考答案的重合度
- **BLEU 分数也有提升**：对比基线，(1) 的 BLEU 分数从 41.34 提升至 56.21 ，这说明重排序后的文档不仅内容相关性更高，也使得模型生成的答案在流畅度和语法结构上更接近标准答案。
- **Reference 准确率大幅提高**：最显著的提升体现在 Reference 准确率上，从基础粗排的 48.15% 提升到 61.73%(结合 BM25 混合检索)，再通过结合重排序模型提升至 77.78% 。这直接证明了多路召回和重排序在提供高质量、可追溯的上下文方面的有效性，大大降低了模型产生“幻觉”的风险。
- **奇怪的现象——Markdown 文档表现的局限性**：尽管多路召回和重排序对 PDF 数据⓪带来了显著提升，但对于 Markdown 文档①而言，其提升效果不如预期。一方面，这可能受到大模型输出的随机性影响，另一方面，这可能也与 Markdown 文档的结构化处理方式有关，在粗排阶段已能较好地捕捉信息，而多路召回和重排序的边际效应递减。

由于 Markdown 文档的效果不及预期，且无法提供命中率指标，故在后续实验中，我们均以 PDF 文档进行实验。对于实验二，我们将 (1) 的结果作为 2.json 提交。

3.4 其他优化策略效果

3.4.1 实验配置

为了深入探究不同超参数对 RAG 系统性能的影响，我们设计并进行了一系列详尽的实验。实验主要围绕以下超参数组合展开：

- **Embedding Models**：`xiaobu-embedding-v2`, `m3e-base`, `gte_Qwen2-7B-instruct`
- **Reranker Models**: `bge-reranker-base`, `bge-reranker-large`, `bge-reranker-v2-m3`
- **Chunking Strategy**: [300, 100](chunk_size=300,chunk_overlap=100，进行 288 次实验), [500,150](进行 72 次实验)
- **Ensemble Weights**: [0.5, 0.5](同等权重)，[0.7, 0.3](BM25 权重更高)
- **BM25 召回数**：3，5
- **向量检索召回数量**：3，5

所有实验均基于原始 PDF 数据⓪进行，并采用之前验证过的多路召回与重排序架构。通过穷举不同的超参数组合，并进行多次重复实验以确保结果的可靠性，我们旨在识别出对 RAG 系统性能影响最显著的超参数，并找出最优的配置方案。另外，为了适配更通用的场景，同时避免 prompt 指令中可能的泄露测试集问题，我们将原本 prompt 的第四条要求删去，得到未污染的结果。

3.4.2 结果分析

通过对大量实验结果（详见附件A）的分析和可视化，我们得到了以下发现：

1. Embedding Model 对性能的影响：

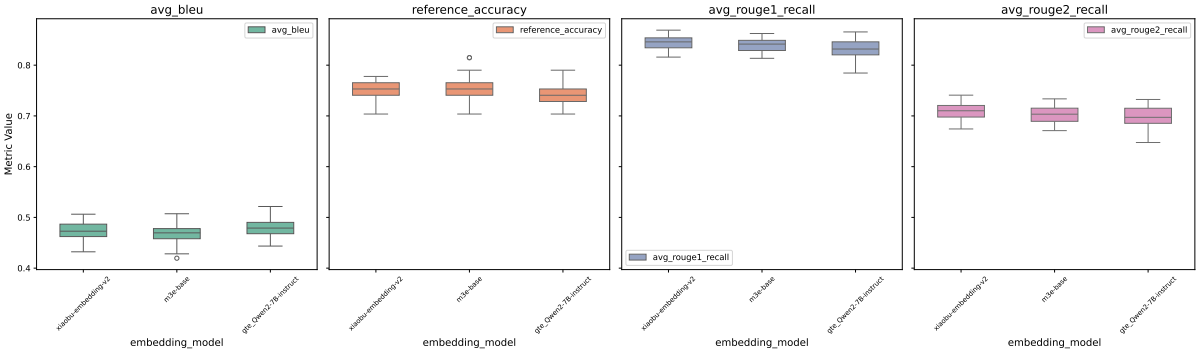

图 4: Embedding Model 影响

- 从图中可以看出，xiaobu-embedding-v2 模型在 Rouge-1 和 Rouge-2 分数上普遍表现最优，其次是 m3e-base，而 gte_Qwen2-7B-instruct 表现相对较弱。这表明 xiaobu-embedding-v2 在中文语义理解和向量表示方面具有更强的能力，能更有效地捕捉文本间的语义相似性，从而为后续的检索和生成提供高质量的上下文。
- 尽管 gte_Qwen2-7B-instruct 也是一个强大的大模型，但在作为 Embedding 模型时，其检索效果似乎不如专门设计的 Embedding 模型。这可能与 Embedding 模型的训练目标和结构有关，更侧重于生成区分度高、语义信息丰富的向量。
- 在 BLEU 分数和 Reference 命中率方面，xiaobu-embedding-v2 同样保持领先。特别是命中率的优势，进一步印证了其在召回相关文档方面的卓越性能。高命中率意味着系统能够更频繁地找到包含正确答案的原文片段，从而显著降低了“幻觉”的风险。

2. Reranker Model 对性能的影响

- 在对比中，我们发现不同模型效果差别并不明显，`bge-reranker-large` 和 `bge-reranker-v2-m3` 的总体效果略好于 `bge-reranker-base`。

3. Chunking Strategy

实验结果显示，(300, 100) 和 (500, 150) 这两种分块策略在 Rouge、BLEU 和命中率等指标上并没有表现出显著的差异。这一现象可能归因于：

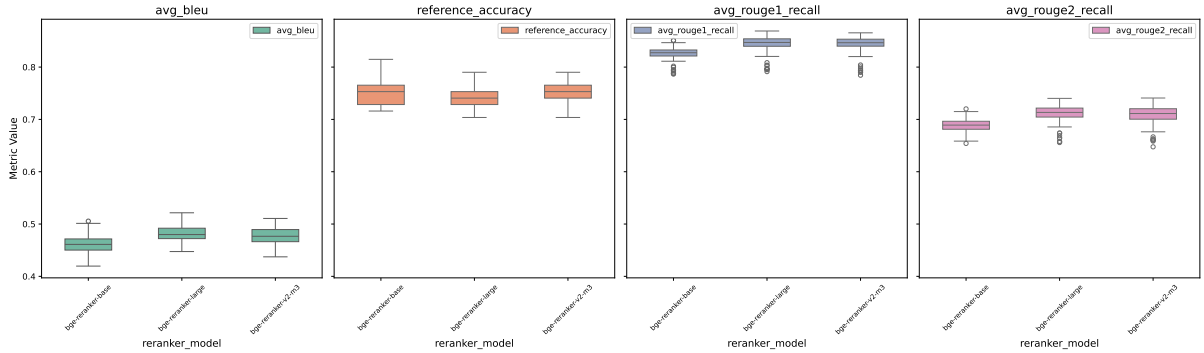

图 5: Reranker Model 影响

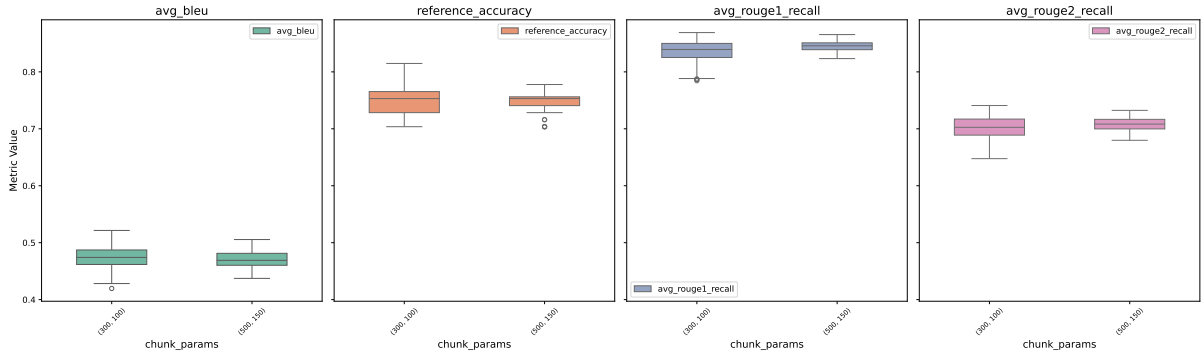

图 6: Chunk Strategy 影响

- 内容粒度与模型鲁棒性：对于本文档的类型，两种大小的语块可能都能够较好地捕获到完整的语义单元和关键信息。同时，Embedding 模型和 Reranker 模型的鲁棒性也可能在一定程度上弥补了分块策略上的细微差异。
- chunk_overlap 的作用：适当的 chunk_overlap 确保了语块间的上下文连续性，即使语块边界划分不完美，重叠部分也能提供足够的相邻信息，减少了信息丢失的风险。
- 数据特性：如果原始文档的平均段落长度或信息密度与这两种 chunk_size 范围大致匹配，那么改变 chunk_size 可能不会带来颠覆性的影响。例如，如果大多数回答所需的知识点都集中在 300-500 字内，那么两种切分方式都能有效地召回相关信息。

4. Ensemble Weights, BM25 k, Vector k 对性能的影响：

- 两种融合权重 (0.5, 0.5 和 0.7, 0.3) 在各项指标上表现出相似的性能，没有显著的优劣之分。这说明在本实验场景下，BM25 和向量检索的相对重要性差异不大，或者 RRF（倒数排名融合）机制本身对权重的敏感度不高，能够在不同权重下保持较好的融合效果。
- BM25-k 和 Vector-k 在 3 和 5 之间变化时，对最终结果的影响也相对较小。这表明在我们的多路召回设置中，召回 Top-3 或 Top-5 的文档已经能够覆盖大部分相关信息，进一步增加召回数量（例如到 Top-10）可能只会引入更多噪音而无法带

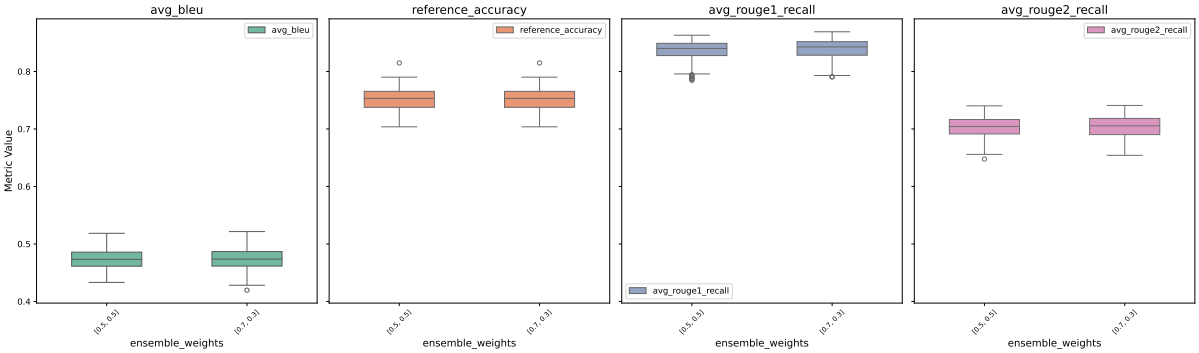

图 7: Ensemble Weight 影响

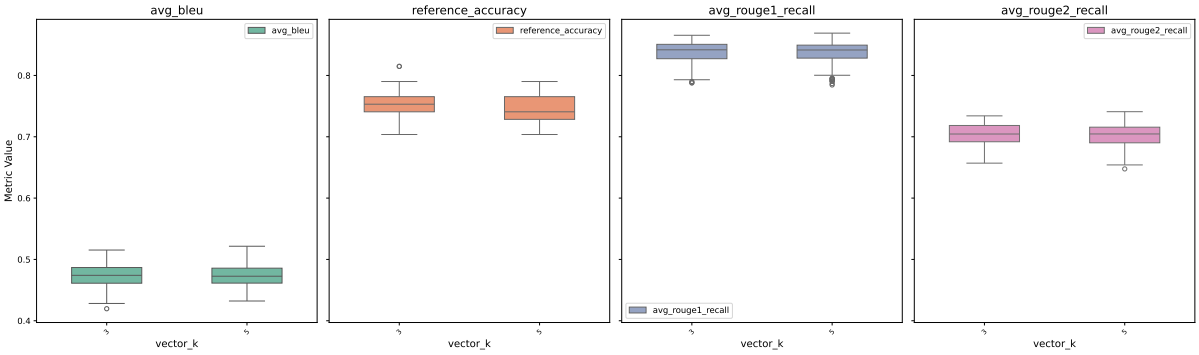

图 8: Vector-k 影响

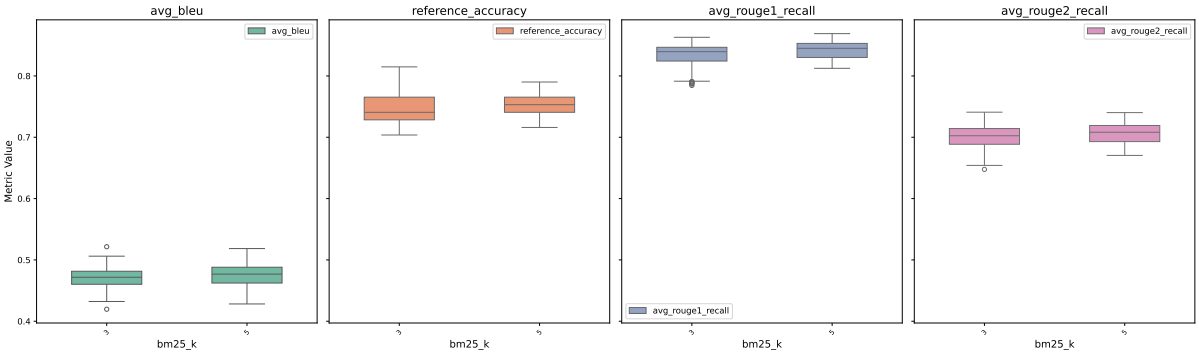

图 9: BM25-k 影响

来显著的性能提升。这与 Reranker top_n 设置为 3 的策略是相符的，即我们最终只希望将最精炼和相关的 3 个文档传递给大模型。

3.4.3 优化策略结果评估

在实验中，有时会出现命中率高但 Rouge 分数低的结果或 Rouge 分数高但命中率低的结果。为了综合考虑各项指标，我们通过对各指标求和排序找到上述实验中每组超参数对应的最好结果：

$$\text{sum} = \text{Rouge-1} + \text{Rouge-2} + \text{BLEU} + \text{Hit\_Rate} \tag{1}$$

表 4: 优化策略结果

序号	数据	分块	粗排	重排	融合	Rouge-1	Rouge-2	BLEU	命中率
0	⓪	300,100	②,3,3	②	⓪	86.30	75.65	54.44	76.54
1	⓪	300,100	①,3,3	⓪	⓪	82.16	68.85	46.12	81.48

其中序号 1 为命中率最高的结果。综合考虑，最终我们将序号 0 作为最终的 answer3.json。

3.4.4 Qwen3 结果

我们还使用 Qwen3-Embedding-0.6B 和 Qwen3-Embedding-4B 进行了相关实验，控制其他超参数与最优的策略一致。得到结果如下：

表 5: Qwen3-Embedding 结果

序号	数据	分块	粗排	重排	融合	Rouge-1	Rouge-2	BLEU	命中率
2	⓪	300,100	0.6B,3,3	②	⓪	83.26	71.83	53.90	77.78
3	⓪	300,100	4B,3,3	②	⓪	83.63	73.04	55.80	76.54

虽然效果还可以，但是整体效果并没有如我们预期的变得更好，我们猜测可能是由于给定的语料和需要回答的问题过少，所以效果不明显。

4 结论与展望

4.1 研究结论

本研究聚焦于基于检索增强生成（RAG）方法，结合大语言模型（如 Qwen-72B）和结构化文档的问答系统的优化策略与实现。通过一系列实验，本文探索了如何利用《汽车介绍手册》这一特定领域的 PDF 文档，搭建高效、准确、可溯源的问答系统。以下是主要研究结论：

1. 多路召回与重排序的优化效果显著：本实验展示了多路召回策略与重排序模型显著提高了问答系统的性能。通过结合 BM25 检索和向量相似度检索，使用倒数排名融合（RRF）方法，成功提升了 ROUGE 分数与 BLEU 分数，尤其是在召回率和答案质量方面。具体而言，采用多路召回和重排序后，ROUGE-1 和 ROUGE-2 的得分从基线的 60.36 和 48.02 提高到 86.97 和 75.18，且 BLEU 分数也有显著提升。
2. 文档结构的影响与优化：对比不同数据处理方式（PDF 与 Markdown），我们发现，经过人工校对和结构化处理的 Markdown 文档，能够提供更高质量的检索内容，尤其在 ROUGE 指标上表现较好。然而，Markdown 文档对于生成答案的语法和流畅度影响较小，表明在某些情况下，原始 PDF 文档的非结构化信息可能更贴合参考答案的表达方式。
3. 参考准确性和命中率的提升：实验结果表明，结合多路召回与重排序策略，Reference 准确率大幅提高，特别是在使用 PDF 文档时，命中率从原始数据的 48.15% 提升至最高 77.78%。这表明，精细的文档重排序策略能够有效地减少无关信息的干扰，确保生成的答案更加准确且可追溯。
4. 优化策略对系统性能的提升：除了多路召回与重排序策略外，扩展嵌入模型（如 xiaobu-embedding-v2 和 gte_Qwen2-7B-instruct）对系统的文本理解和检索性能有重要影响。尤其是 xiaobu-embedding-v2 模型，在 ROUGE 和 BLEU 分数上表现优异，有效提升了语义理解和检索精度。

总体而言，本研究表明，通过合理的优化策略和模型选择，可以显著提升基于 RAG 框架的问答系统的性能。通过细化每个环节的策略，能够有效提高召回的质量与生成答案的准确性，进一步推动智能问答系统在实际场景中的应用。

4.2 未来工作展望

基于当前 RAG 智能问答系统的研究与实践，我们识别出以下几个方向作为未来工作的重点，以期进一步提升系统的性能、鲁棒性和用户体验：

1. **探索更先进的知识图谱构建构建检索：**考虑将非结构化文本转化为结构化知识（如知识图谱），通过实体关系和事件抽取，能够提供更精准、更具推理能力的检索上下文，从而提升复杂问答和多跳问答的能力。在实验中，我们曾经考虑过使用

GraphRAG 进行检索，但由于评测指标强调与原始内容的匹配程度，且 QApairs 里面的问题大多不涉及到实体属性，我们最终放弃了此想法，但我们认为这个方法有一定的现实应用空间。

2. **优化多路召回与融合策略：**虽然当前的多路召回策略已经取得了显著效果，但仍有优化空间。未来可以尝试引入更多元化的召回通道，例如基于关键词扩展、领域知识库、或用户历史行为的召回。在融合策略上，除了 RRF，可以探索更复杂的学习排序（Learning-to-Rank）模型，利用机器学习方法自动学习不同召回源的最佳融合权重，甚至根据查询类型和文档特性进行动态调整，以实现更优的召回融合效果。
3. **引入 LLM-Reranker 作为重排序模型：**当前使用的 BGE 系列 Reranker 模型虽然效果优秀，但其本质是交叉编码器，主要依赖于文本匹配。未来一个重要的方向是探索采用大型语言模型（LLM）作为重排序模型（LLM-Reranker）。LLM 具备强大的语言理解、推理和上下文建模能力，能够更深入地理解查询和文档之间的语义关联、逻辑关系，甚至进行一定程度的常识推理。通过将召回的候选文档输入给 LLM 进行打分排序，可以显著提升重排序的准确性和对复杂查询的理解能力，从而筛选出最符合用户意图和问题上下文的文档片段。这将有助于进一步减少无关信息的干扰，为最终的答案生成提供最高质量的上下文。本实验原本考虑使用 `bge-reranker-v2-minicpm-layerwise`(一个基于 MiniCPM-2B-dpo-bf16 在混合的多个多语言排序数据集上训练的 LLM Reranker，此模型在中英文上具有先进的排序效果) 作为 LLM Reranker，但由于 LangChain 尚未兼容而作罢。
4. **构建全面的评测体系与用户反馈机制：**当前的 ROUGE、BLEU 和命中率等指标虽然重要，但无法完全衡量 RAG 系统的用户体验和在实际应用中的效果。未来应建立更全面的评测体系，例如引入答案的连贯性、完整性、简洁性、安全性等定性指标，并结合人工评估。同时，建立有效的用户反馈机制，收集用户对答案质量、相关性、满意度等方面的反馈，形成数据闭环，指导模型的持续优化和迭代。
5. **强化答案生成与幻觉抑制：**虽然 RAG 系统能有效降低幻觉，但仍无法完全避免。未来需要重点研究如何进一步提升 LLM 基于检索到的上下文生成答案的准确性和事实性。包括但不限于引入信任度或证据链机制、多文档信息整合与冲突解决与可控文本生成机制。
6. **探索更多的融合方法：**在 [1] 中还提到了采用粗排简单合并、重排融合中的生成前融合、生成后选答案更长融合、生成后答案拼接融合等不同的融合策略，由于实验时间有限，我们没有对这些不同策略的效果进行测试。

参考文献

[1] Zhangchi Feng, Kuang Dongdong, Wang Zhongyuan, Nie Zhijie, Zheng Yaowei, and Richong Zhang. Easyrag: Efficient retrieval-augmented generation framework for au-

tomated network operations. *arXiv preprint arXiv:2410.10315*, 2024.

[2] Wang Yuxin, Sun Qingxuan, and He Sicheng. M3e: Moka massive mixed embedding model, 2023.

[3] Chaofan Li, Zheng Liu, Shitao Xiao, and Yingxia Shao. Making large language models a better foundation for dense retrieval, 2023.

[4] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

[5] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, 2024.

A 不同参数结果

bm25_k	vec_k	重排	分块	粗排	融合	BLEU	命中率	Rouge-1	Rouge-2
3	3	⓪	(500, 150)	③	①	0.5055	0.7284	0.8463	0.7201
5	5	②	(500, 150)	③	①	0.5025	0.7654	0.8607	0.723
3	3	②	(500, 150)	③	⓪	0.5024	0.7654	0.834	0.7046
5	5	①	(300, 100)	③	⓪	0.501875	0.73455	0.844075	0.710875
3	5	⓪	(500, 150)	③	⓪	0.5013	0.7284	0.8506	0.715
5	3	①	(300, 100)	③	⓪	0.5008	0.749975	0.841925	0.709775
5	5	①	(300, 100)	②	⓪	0.4981	0.749975	0.852325	0.72115
5	3	②	(300, 100)	③	⓪	0.495425	0.768525	0.845875	0.712275
5	3	②	(300, 100)	③	①	0.4948	0.768525	0.844	0.710975
3	3	①	(500, 150)	③	①	0.4937	0.7284	0.8566	0.7315
5	3	①	(300, 100)	③	①	0.49345	0.749975	0.840825	0.7087
5	5	①	(300, 100)	③	①	0.49295	0.73455	0.8399	0.704275
5	5	①	(500, 150)	①	①	0.4923	0.7407	0.8575	0.7154
5	3	②	(500, 150)	③	⓪	0.4916	0.7654	0.8538	0.7253
5	5	②	(300, 100)	②	①	0.489975	0.7407	0.85355	0.7201
3	5	②	(300, 100)	②	①	0.4899	0.737625	0.85155	0.72125
3	5	⓪	(500, 150)	③	①	0.4899	0.7284	0.8408	0.7029
3	3	①	(500, 150)	③	⓪	0.4895	0.7284	0.8524	0.725
5	3	②	(300, 100)	②	⓪	0.489375	0.7407	0.852525	0.7179
3	3	①	(500, 150)	①	⓪	0.4893	0.7037	0.8383	0.713
3	5	①	(500, 150)	③	①	0.4885	0.716	0.8451	0.7087
5	5	①	(300, 100)	②	①	0.48835	0.749975	0.860625	0.72605
3	5	①	(300, 100)	②	⓪	0.4881	0.743825	0.850925	0.71905
5	3	①	(300, 100)	①	①	0.4862	0.759225	0.8474	0.71565
5	3	①	(300, 100)	①	⓪	0.486175	0.759225	0.84665	0.713575
3	5	①	(500, 150)	③	⓪	0.4859	0.716	0.8437	0.7099
3	5	①	(300, 100)	②	①	0.4853	0.743825	0.85835	0.7237
5	5	②	(300, 100)	②	⓪	0.48505	0.7407	0.849975	0.716875

bm25_k	vec_k	重排	分块	粗排	融合	BLEU	命中率	Rouge-1	Rouge-2
5	5	②	(300, 100)	③	⓪	0.4849	0.7716	0.8398	0.7013
5	5	②	(300, 100)	③	①	0.48445	0.7716	0.84175	0.70715
5	3	①	(300, 100)	②	①	0.48415	0.749975	0.85135	0.7172
3	3	①	(300, 100)	①	①	0.484025	0.7685	0.845	0.709075
3	3	①	(300, 100)	②	①	0.4839	0.7469	0.84645	0.71395
5	5	②	(300, 100)	①	①	0.4839	0.7469	0.8526	0.71845
5	3	①	(500, 150)	②	①	0.4832	0.7407	0.8574	0.7225
3	5	②	(300, 100)	②	⓪	0.483175	0.737625	0.852975	0.722175
5	3	①	(500, 150)	③	①	0.4828	0.7531	0.8568	0.7204
5	3	⓪	(500, 150)	③	⓪	0.4824	0.7407	0.8462	0.7042
3	5	②	(500, 150)	③	①	0.4819	0.7531	0.8459	0.7117
5	5	①	(500, 150)	③	①	0.4817	0.7531	0.8587	0.7282
5	3	①	(300, 100)	②	⓪	0.481525	0.749975	0.8501	0.712825
3	3	①	(300, 100)	②	⓪	0.48145	0.7469	0.84975	0.7161
3	5	①	(300, 100)	③	①	0.481325	0.71295	0.820725	0.6864
5	3	⓪	(500, 150)	③	①	0.4813	0.7407	0.8449	0.7034
5	3	①	(500, 150)	③	⓪	0.4813	0.7531	0.8541	0.7165
5	5	①	(500, 150)	②	⓪	0.4803	0.7407	0.8509	0.7191
5	3	⓪	(300, 100)	③	⓪	0.4791	0.75615	0.830875	0.69825
5	3	①	(500, 150)	①	①	0.4789	0.7407	0.8522	0.723
3	3	①	(300, 100)	①	⓪	0.478575	0.7685	0.847925	0.717525
3	3	②	(300, 100)	②	⓪	0.478375	0.740725	0.8532	0.71855
3	5	①	(500, 150)	②	①	0.4783	0.7407	0.8428	0.7118
5	3	②	(300, 100)	②	①	0.478125	0.7407	0.84925	0.713175
5	3	②	(300, 100)	①	①	0.478125	0.75305	0.851675	0.71545
5	3	②	(300, 100)	①	⓪	0.477975	0.75305	0.849275	0.711125
5	5	⓪	(500, 150)	③	①	0.4778	0.7407	0.8466	0.7112
3	5	①	(300, 100)	③	⓪	0.477525	0.71295	0.817	0.683925
3	5	⓪	(500, 150)	①	⓪	0.4773	0.7531	0.8403	0.7023
3	3	⓪	(500, 150)	②	①	0.4773	0.7654	0.8467	0.7072
3	5	①	(500, 150)	①	⓪	0.4772	0.7037	0.8458	0.7101
3	5	②	(300, 100)	③	⓪	0.4771	0.75	0.80985	0.67695
3	3	⓪	(500, 150)	③	⓪	0.477	0.7284	0.8329	0.6912
3	3	①	(300, 100)	③	⓪	0.47695	0.734575	0.818975	0.689875
5	5	②	(500, 150)	③	⓪	0.4769	0.7654	0.8614	0.7193
3	3	②	(300, 100)	②	①	0.476875	0.740725	0.8476	0.71545
5	5	②	(300, 100)	①	⓪	0.476825	0.7469	0.8471	0.71245
3	3	②	(500, 150)	②	⓪	0.4765	0.7778	0.8458	0.7044
3	3	②	(300, 100)	①	①	0.4757	0.749975	0.847175	0.71505
3	3	②	(300, 100)	③	①	0.47515	0.743825	0.81605	0.6864
3	5	①	(300, 100)	①	①	0.475025	0.75	0.84785	0.707675
3	5	②	(500, 150)	①	⓪	0.474	0.7407	0.8297	0.6908
5	5	①	(500, 150)	③	⓪	0.4738	0.7531	0.8609	0.7324
3	3	⓪	(300, 100)	③	①	0.472675	0.73765	0.8105	0.681125
5	5	⓪	(300, 100)	③	①	0.472625	0.743825	0.826775	0.69
5	5	①	(300, 100)	①	①	0.472625	0.75615	0.847775	0.71475
3	3	②	(500, 150)	③	①	0.4726	0.7654	0.8469	0.7185
3	5	①	(300, 100)	①	⓪	0.4725	0.75	0.84055	0.7
3	5	②	(300, 100)	①	①	0.47235	0.7438	0.847225	0.710525
5	3	⓪	(500, 150)	①	①	0.4716	0.7531	0.8513	0.7126
5	3	⓪	(500, 150)	②	⓪	0.4715	0.7531	0.8372	0.6989
5	3	⓪	(300, 100)	③	①	0.470975	0.75615	0.82655	0.6907
3	3	①	(300, 100)	③	①	0.4707	0.734575	0.81365	0.6826
5	3	①	(500, 150)	②	⓪	0.4706	0.7407	0.8476	0.7192
3	3	②	(300, 100)	③	⓪	0.47055	0.743825	0.8124	0.684675

bm25_k	vec_k	重排	分块	粗排	融合	BLEU	命中率	Rouge-1	Rouge-2
3	5	①	(500, 150)	①	①	0.4693	0.7037	0.8516	0.7111
3	5	②	(300, 100)	①	⓪	0.4691	0.7438	0.84325	0.7046
5	5	①	(300, 100)	①	⓪	0.469025	0.75615	0.84625	0.714175
3	5	②	(300, 100)	③	①	0.468925	0.75	0.814225	0.6835
3	5	①	(500, 150)	②	⓪	0.4687	0.7407	0.8449	0.707
3	3	②	(500, 150)	②	①	0.4687	0.7778	0.8484	0.7073
5	3	②	(500, 150)	②	①	0.4684	0.7654	0.8534	0.7174
5	3	②	(500, 150)	③	①	0.4683	0.7654	0.8655	0.7238
3	3	⓪	(300, 100)	③	⓪	0.4681	0.73765	0.80705	0.67645
3	5	②	(500, 150)	③	⓪	0.468	0.7531	0.8491	0.7162
3	3	①	(500, 150)	②	①	0.4679	0.7407	0.8552	0.7227
5	5	⓪	(500, 150)	①	①	0.4671	0.7531	0.837	0.6945
5	3	①	(500, 150)	①	⓪	0.4667	0.7407	0.8494	0.7115
3	3	②	(300, 100)	①	⓪	0.46625	0.749975	0.848025	0.717075
5	5	⓪	(500, 150)	②	⓪	0.4653	0.7531	0.8455	0.6978
5	5	⓪	(500, 150)	①	⓪	0.4652	0.7531	0.8392	0.6947
3	3	⓪	(500, 150)	①	①	0.465	0.7654	0.8307	0.6876
3	3	⓪	(300, 100)	②	⓪	0.4646	0.740725	0.826175	0.6896
3	3	①	(500, 150)	②	⓪	0.4642	0.7407	0.8479	0.7087
3	5	②	(500, 150)	②	①	0.4639	0.7654	0.8465	0.7079
3	5	⓪	(300, 100)	②	⓪	0.463625	0.753075	0.83005	0.695425
5	5	⓪	(300, 100)	③	⓪	0.462875	0.743825	0.823875	0.686125
5	5	①	(500, 150)	②	①	0.4624	0.7407	0.8452	0.7061
5	5	①	(500, 150)	①	⓪	0.4614	0.7407	0.8471	0.7135
3	3	①	(500, 150)	①	①	0.461	0.7037	0.8467	0.7048
5	3	②	(500, 150)	①	⓪	0.4609	0.7531	0.842	0.699
5	3	⓪	(300, 100)	②	①	0.4607	0.756175	0.82505	0.688925
5	3	②	(500, 150)	②	⓪	0.4604	0.7654	0.8488	0.717
5	5	⓪	(300, 100)	①	①	0.460175	0.756175	0.823375	0.68405
3	3	⓪	(500, 150)	①	⓪	0.46	0.7654	0.8233	0.6815
3	5	⓪	(300, 100)	③	①	0.457925	0.7253	0.806475	0.6724
5	5	②	(500, 150)	①	⓪	0.4576	0.7531	0.8368	0.7065
3	3	⓪	(500, 150)	②	⓪	0.4573	0.7654	0.8425	0.7043
5	5	⓪	(300, 100)	①	⓪	0.4568	0.756175	0.820825	0.678225
5	5	⓪	(300, 100)	②	①	0.456525	0.76235	0.825175	0.689325
5	3	⓪	(500, 150)	①	⓪	0.4563	0.7531	0.8371	0.7003
3	5	⓪	(300, 100)	③	⓪	0.456275	0.7253	0.805425	0.673375
3	3	②	(500, 150)	①	⓪	0.4559	0.7407	0.8367	0.696
5	3	②	(500, 150)	①	①	0.4557	0.7531	0.8375	0.698
3	5	⓪	(300, 100)	②	①	0.4552	0.753075	0.8308	0.69495
3	3	⓪	(300, 100)	①	⓪	0.454975	0.7716	0.8232	0.689425
3	5	⓪	(500, 150)	②	①	0.4543	0.7531	0.8296	0.6798
5	5	⓪	(300, 100)	②	⓪	0.453725	0.76235	0.8266	0.69275
3	5	⓪	(300, 100)	①	①	0.4536	0.746925	0.825775	0.68585
3	3	⓪	(300, 100)	②	①	0.452825	0.740725	0.831025	0.6976
5	3	⓪	(300, 100)	①	⓪	0.45215	0.765425	0.8239	0.683325
3	5	②	(500, 150)	①	①	0.4511	0.7407	0.8369	0.6941
3	5	⓪	(500, 150)	②	⓪	0.451	0.7531	0.8414	0.693
3	5	⓪	(500, 150)	①	①	0.4508	0.7531	0.8283	0.6887
3	5	⓪	(300, 100)	①	⓪	0.4507	0.746925	0.82425	0.6871
5	3	⓪	(500, 150)	②	①	0.4504	0.7531	0.8337	0.6822
5	5	⓪	(500, 150)	③	⓪	0.4504	0.7407	0.8327	0.6914
5	5	②	(500, 150)	①	①	0.4489	0.7531	0.852	0.7145
5	3	⓪	(300, 100)	①	①	0.4477	0.762325	0.826675	0.684725
3	3	②	(500, 150)	①	①	0.4469	0.7407	0.8397	0.7005

bm25_k	vec_k	重排	分块	粗排	融合	BLEU	命中率	Rouge-1	Rouge-2
5	3	⓪	(300, 100)	②	⓪	0.4466	0.756175	0.822125	0.67875
3	5	②	(500, 150)	②	⓪	0.4452	0.7654	0.8392	0.7045
5	5	⓪	(500, 150)	②	①	0.4445	0.7531	0.8281	0.6799
3	3	⓪	(300, 100)	①	①	0.442975	0.7747	0.821275	0.6809
5	5	②	(500, 150)	②	①	0.4419	0.7654	0.8449	0.7101
5	5	②	(500, 150)	②	⓪	0.4373	0.7654	0.8442	0.7001