RAG 技术详解与实战应用

第14讲：实战：构建一个支持复杂学术论文问答的RAG系统

目录

1. 上节回顾
2. 传统RAG的论文系统
3. 朴素多模态RAG的论文系统
3. 论文系统的效果优化
4. 总结拓展

上节回顾

多模态RAG相关技术

- **多模态大模型：**
- 感知层：模态感知与特征提取
- 对齐层：表示对齐与语义映射
- 理解与生成层：统一语义建模与跨模态推理
- **在LazyLLM中使用多模态大模型**
- OnlineChatModule + WebModule 实现一键启动和web服务

多模态 RAG

- **PDF编码：**
- 首先通过布局识别模型对文档段落、标题以及图标等其他元素进行识别和提取，调用视觉模型对图片进行描述，统一模态为文本，进行统一编码。
- **生成：**
- 通过图表链接格式化和针对图片的提示词优化等策略，支持图文并茂的生成

提高多模态编码准确率

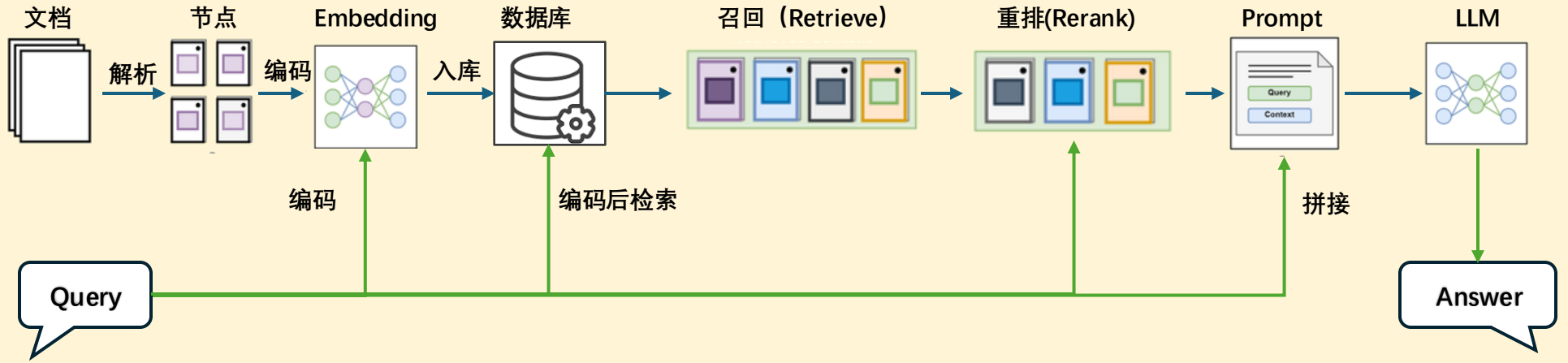
- **文本补全**
- 结合标题与注解等文本信息
- **结构化生成**
- 预先从多模态数据中提取 QA 对
- **上下文增强**
- 将上下文一同编码
- **微调多模态大模型**

目录

1. 上节回顾
2. 传统RAG的论文系统
3. 朴素多模态RAG的论文系统
3. 论文系统的效果优化
4. 总结拓展

传统RAG的论文系统

召回(Retrieve)-> 重排(Rerank) -> 问答(llm)



传统RAG的论文系统 代码展示

```
# 定义documents
documents = lazyllm.Document(dataset_path=tmp_dir.rag_dir, embed=OnlineEmbeddingModule(), manager=False,
            store_conf=milvus_store_conf, doc_fields=doc_fields)
# 注册自定义reader 和 文档切分方式
documents.add_reader("*.pdf", MagicPDFReader)
documents.create_node_group(name="block", transform=lambda s: s.split("\n") if s else '')
# rag pipeline
with lazyllm.pipeline() as ppl:
    ppl.retriever = lazyllm.Retriever(doc=documents, group_name="block", topk=3)
    ppl.reranker = lazyllm.Reranker(name='ModuleReranker',
                                    model="bge-reranker-large",
                                    topk=1,
                                    output_format='content',
                                    join=True) | bind(query=ppl.input)

    ppl.formatter = (
        lambda nodes, query: dict(context_str=nodes, query=query)
    ) | bind(query=ppl.input)

    ppl.llm = OnlineChatModule(stream=False).prompt(
        lazyllm.ChatPrompter(instruction=prompt, extra_keys=['context_str']))
# 启动 web 服务
lazyllm.WebModule(ppl, port=23456, static_paths=get_cache_path()).start().wait()
```

传统RAG的论文系统 效果展示

PaperQA_basic.py — zhaoshe [SSH: test0224]

PaperQA_basic.py

my_rag > chapter14 > PaperQA_basic.py > ...

```python

if __name__ == "__main__":
    prompt = 'You will play the role of an AI Q&A assistant and complete a dialogue task.'\
        ' In this task, you need to provide your answer based on the given context and question.'\
        ' If an image can better convey the information being expressed, please include the image reference'\
        ' in the text in Markdown format. Keep the image path in its original format.'

    documents = lazyllm.Document(dataset_path=tmp_dir.rag_dir,
                                 embed=lazyllm.TrainableModule("bge-large-zh-v1.5"),
                                 manager=False,
                                 store_conf=milvus_store_conf,
                                 doc_fields=doc_fields)

    documents.add_reader("*.pdf", MagicPDFReader)
    documents.create_node_group(name="block", transform=lambda s: s.split("\n") if s else '')

    with lazyllm.pipeline() as ppl:
        ppl.retriever = lazyllm.Retriever(doc=documents, group_name="block", topk=3)
        ppl.reranker = lazyllm.Reranker(name='ModuleReranker',
                                        model="bge-reranker-large",
                                        topk=1,
                                        output_format='content',
                                        join=True) | bind(query=ppl.input)

        ppl.formatter = (
            lambda nodes, query: dict(context_str=nodes, query=query)
        ) | bind(query=ppl.input)
        ppl.func = func
        ppl.llm = lazyllm.TrainableModule('internlm2-chat-7b').prompt(
            lazyllm.ChatPrompter(instruction=prompt, extra_keys=['context_str']))
```

问题 输出 调试控制台 终端 端口 11

zsh - LazyLLM

```
(lazyllm) → LazyLLM git:(addTrainableModule) ✗
```

SSH: test0224 addTrainableModule* 0 0 11 行 458，列 85 空格: 4 UTF-8 LF Python 3.10.9 ('lazyllm': conda)

目录

1. 上节回顾

2. 传统RAG的论文系统

3. 朴素多模态RAG的论文系统

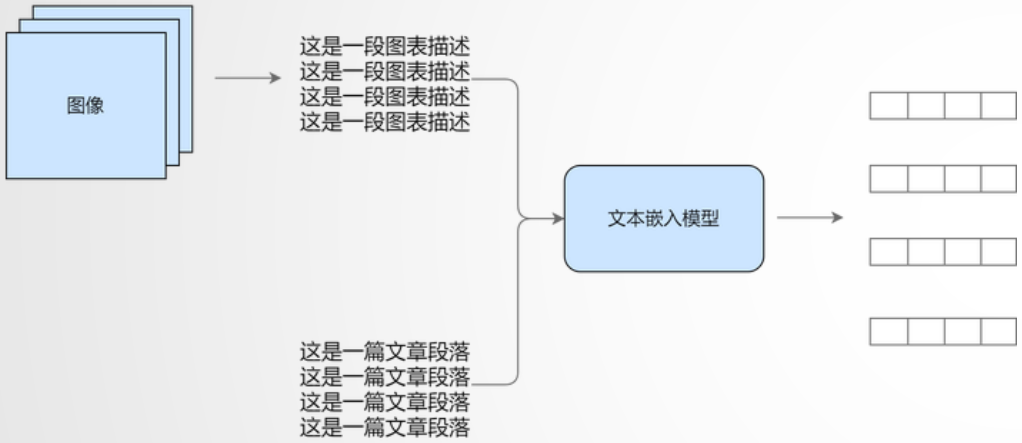
3. 论文系统的效果优化

4. 总结拓展

多模态RAG的论文系统

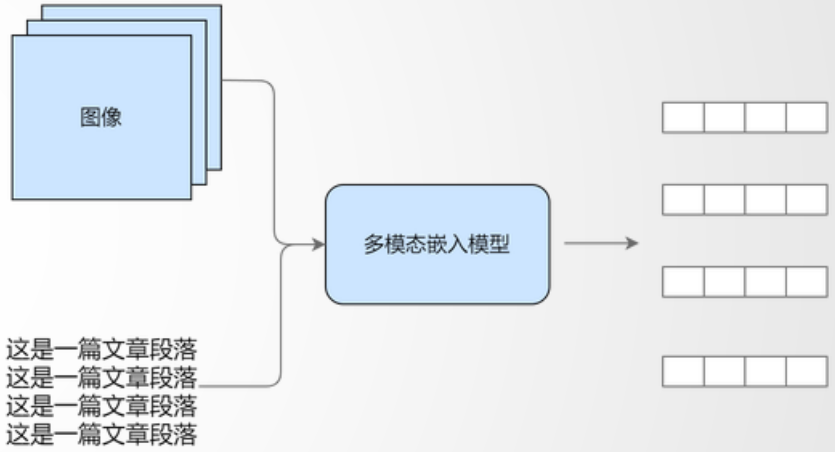
多模态嵌入的回顾

方案1



统一到文本模态，然后进行向量嵌入

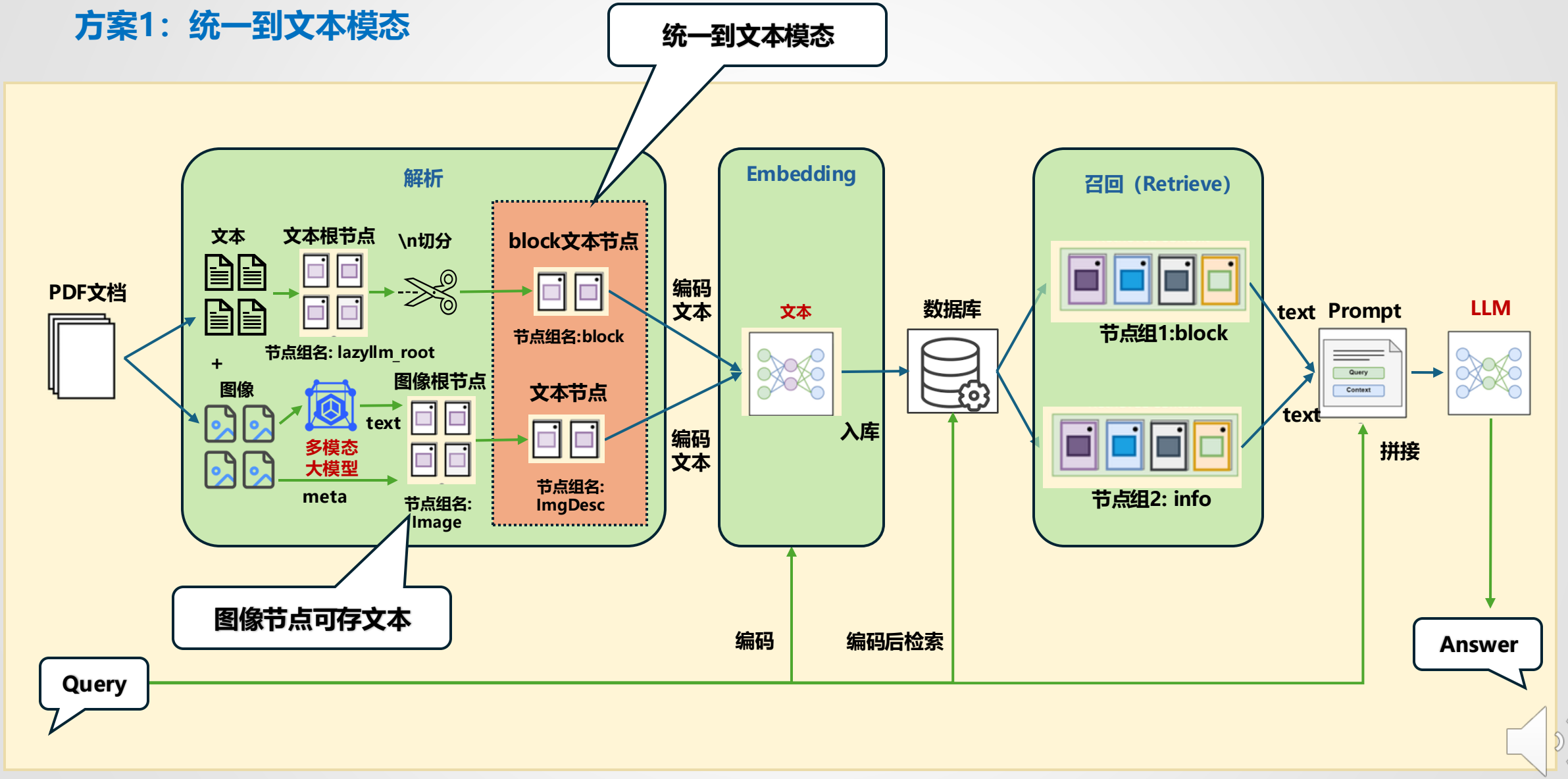
方案2



统一到向量空间：利用多模态模型进行映射

多模态RAG的论文系统

方案1：统一到文本模态



多模态RAG的论文系统

方案1：统一到文本模态 —— 解析的过程构建ImageDocNode节点

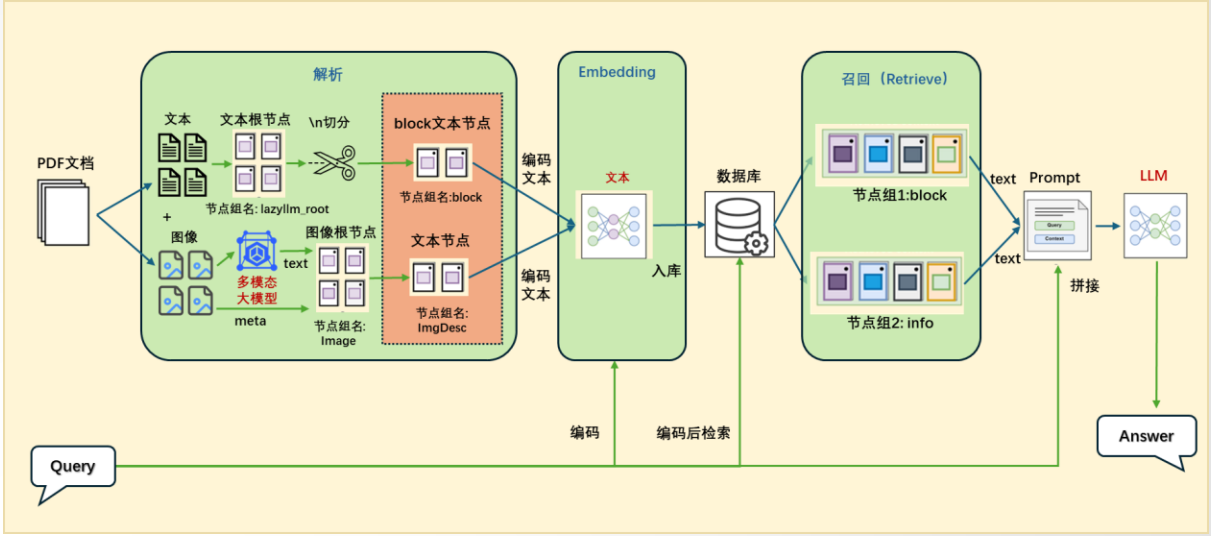
```
1.  class MagicPDFReader(ReaderBase):
2.          def __init__(self):
3.              self.image_save_path = get_image_path()
4.              self.model = None
5.              self.vlm = lazyllm.TrainableModule( 'internvl-chat-v1-5' ).start()    # 初始化一个多模态大模型

6.          def _result_extract(self, content_list):
                ......
7.              if not content["img_path"]:
                    ......
8.                  block['img_desc'] = self.vlm(formatted_query(block["image_path"])) # 利用VLM对图像内容进行解析生成图像的文本描述

9.          def _load_data(self, file: Path, split_documents: Optional[bool] = True, ) -> List[DocNode]:
                ......
10.             for k, v in element.items(): # 对其中的一个block
                    ......
11.                 if "text" in element:
12.                     docs.append(DocNode(text=element["text"], metadata=metadata))
13.                 elif "img_desc" in element:
14.                     image_node = ImageDocNode(text=element["img_desc"],
15.                                               image_path=element[ "image_path" ], global_metadata=metadata)# 构建ImageDocNode节点
16.                     docs.append(image_node)
17.                 else:
18.                     docs.append(DocNode(text="", metadata=metadata))
```

多模态RAG的论文系统

方案1：统一到文本模态 —— 应用编排实现




```
1   documents = lazyllm.Document(
2       dataset_path=tmp_dir.rag_dir,
3       embed=lazyllm.TrainableModule("bge-m3"),
4       manager=False)
5   documents.add_reader("*.pdf", MagicPDFReader)
6   documents.create_node_group(name="block", transform=lambda s: s.split("\n") if s else '')
7
8   with lazyllm.pipeline() as ppl:
9       with lazyllm.parallel().sum as ppl.prl:
10          ppl.prl.retriever1 = lazyllm.Retriever(documents, group_name="block", similarity="cosine", topk=1)
11          ppl.prl.retriever2 = lazyllm.Retriever(documents, lazyllm.Document.ImgDesc, similarity="cosine", topk=1)
12      ppl.prompt = build_vlm_prompt | bind(_0, ppl.input)
13      ppl.vlm = lazyllm.OnlineChatModule()
14
15  lazyllm.WebModule(ppl, port=range(23468, 23470), static_paths=get_image_path()).start().wait()
```

多模态RAG的论文系统

方案1：效果展示

介绍一下deepseek-r1

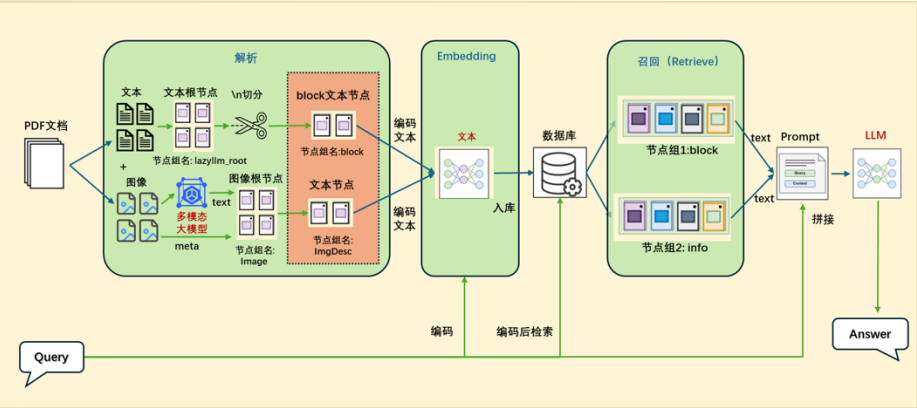
DeepSeek-R1是第一代的推理模型，它被设计用来提升推理能力。这一模型是在尝试解决其前身DeepSeek-R1-Zero所面临的一些挑战，比如可读性差和语言混合问题。DeepSeek-R1-Zero主要是通过大规模的强化学习（RL）训练，而没有采用监督式微调（SFT）作为预备步骤。尽管DeepSeek-R1-Zero展现出了许多强大且有趣的推理行为，但它也遇到了一些问题。

为了解决这些问题并进一步提升推理性能，DeepSeek-R1引入了多阶段训练和冷启动数据作为RL前的处理步骤。这样的改进使得DeepSeek-R1在推理任务上的表现能够与OpenAI的o1-1217模型相媲美。

此外，为了支持研究社区，DeepSeek-R1-Zero、DeepSeek-R1，以及基于Qwen和Llama从DeepSeek-R1中提炼出的六个密集模型（规模分别为1.5B、7B、8B、14B、32B、70B）都被开源。

在性能方面，有一张柱状图直观地展示了DeepSeek-R1与其它模型如OpenAI-01-1217、DeepSeek-R1-32B、OpenAI-01-mini和DeepSeek-V3在多个模型性能指标上的对比。这些模型在不同的测试如AIMER 2024、Codeforces、GPOA Diamond、MATH-500、MMLU和SWE-bench Verified上的表现以不同颜色的条形图表示，反映了它们各自的性能百分比。

整体而言，DeepSeek-R1体现了一个通过多阶段训练和数据处理优化来提升模型推理能力和性能的方法，且通过开源来促进相关领域的研究与发展。



block节点组召回内容：

We introduce our frst-generation reasoning models, DeepSeek-R1-Zero and DeepSeek-R1. DeepSeek-R1-Zero, a model trained via large-scale reinforcement learning (RL) without supervised fne-tuning (SFT) as a preliminary step, demonstrates remarkable reasoning capabilities. Through RL, DeepSeek-R1-Zero naturally emerges with numerous powerful and intriguing reasoning behaviors. However, it encounters challenges such as poor readability, and language mixing. To address these issues and further enhance reasoning performance, we introduce DeepSeek-R1, which incorporates multi-stage training and cold-start data before RL. DeepSeekR1 achieves performance comparable to OpenAI-o1-1217 on reasoning tasks. To support the research community, we open-source DeepSeek-R1-Zero, DeepSeek-R1, and six dense models (1.5B, 7B, 8B, 14B, 32B, 70B) distilled from DeepSeek-R1 based on Qwen and Llama.

info节点组召回内容：

这张图片是一张柱状图，标题为“DeepSeek-R1 OpenAI-01-1217 DeepSeek-R1-32B OpenAI-01-mini DeepSeek-V3”。图表展示了六种不同模型的性能百分比，每种模型都以不同颜色的条形表示。

从左到右，第一列条形代表“AIMER 2024”模型，第二列代表“Codeforces”模型，第三列代表“GPOA Diamond”模型，第四列代表“MATH-500”模型，第五列代表“MMLU”模型，最后一列代表“SWE-bench Verified”模型。

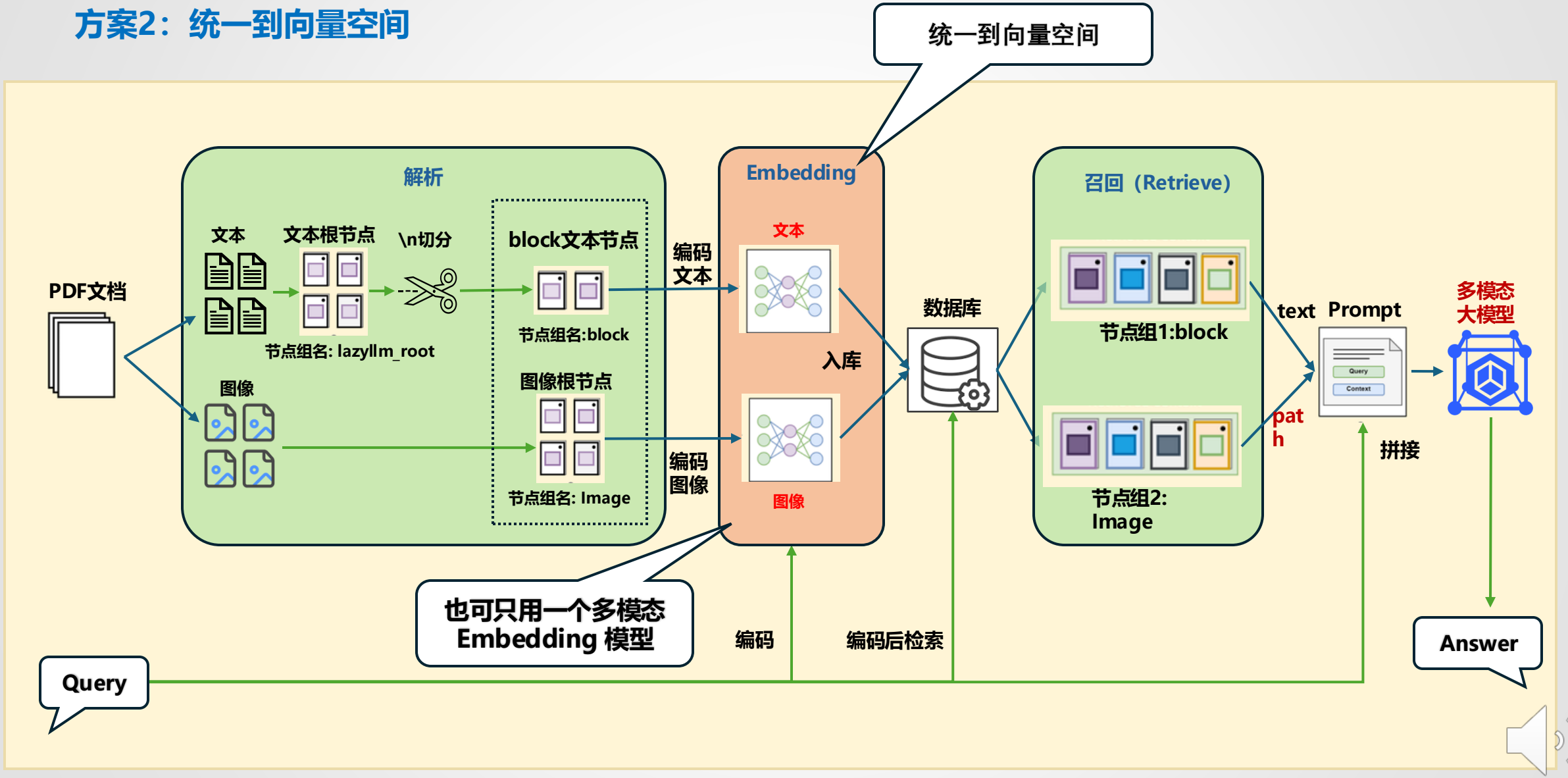
每组条形都有五条，颜色分别为蓝色、灰色、橙色、黄色和紫色，分别对应不同的性能指标。

在图表下方，有五个百分比数值，分别对应每个模型的性能指标。这些数值以从左到右的顺序排列，与条形的颜色相对应。

整个图表以白色为背景，柱状图的颜色与背景形成对比，易于区分。

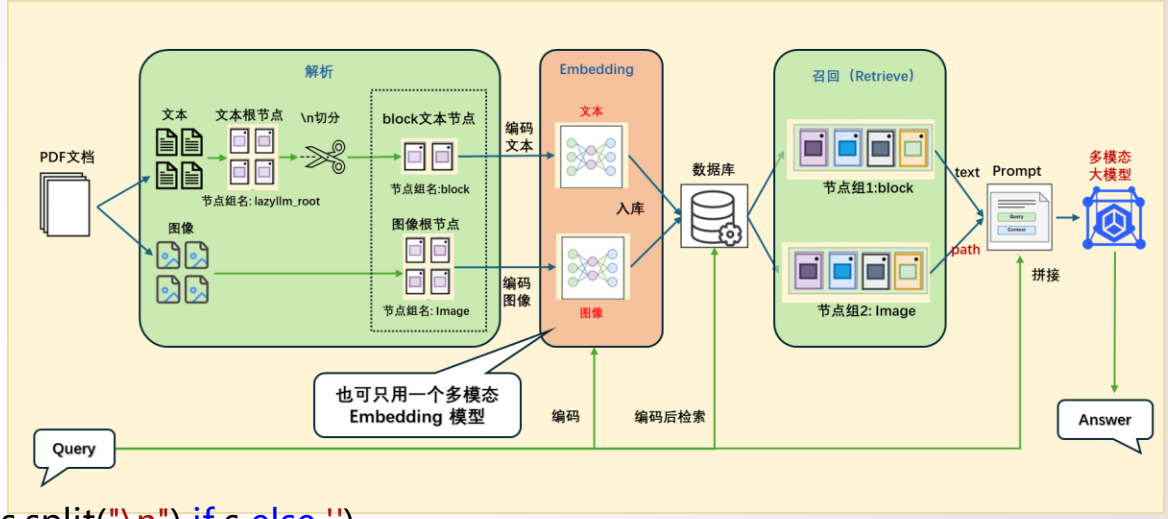
多模态RAG的论文系统

方案2：统一到向量空间



多模态RAG的论文系统

方案2：统一到向量空间 ——应用编排实现




```
1  embed_multimodal = lazyllm.TrainableModule("colqwen2-v0.1")
2  embed_text = lazyllm.OnlineEmbeddingModule(
3      source='qwen', embed_model_name='text-embedding-v1')
4  embeds = {'vec1': embed_text, 'vec2': embed_multimodal}
5  documents = lazyllm.Document(
6      dataset_path=tmp_dir.rag_dir, embed=embeds, manager=False)
7  documents.add_reader("*.pdf", MagicPDFReader)
8  documents.create_node_group(name="block", transform=lambda s: s.split("\n") if s else '')
9
10 with lazyllm.pipeline() as ppl:
11     with lazyllm.parallel().sum as ppl.prl:
12         ppl.prl.retriever1 = lazyllm.Retriever(documents, group_name="block", embed_keys=['vec1'], similarity="cosine", topk=1)
13         ppl.prl.retriever2 = lazyllm.Retriever(documents, group_name="Image", embed_keys=['vec2'], similarity="maxsim", topk=2)
14
15     ppl.prompt = build_vlm_prompt | bind(_0, ppl.input)
16     ppl.vlm = lazyllm.OnlineChatModule(source="sensenova", model="SenseNova-V6-Turbo").prompt(lazyllm.ChatPrompter(gen_prompt))
17
18 lazyllm.WebModule(ppl, port=range(23468, 23470), static_paths=get_image_path()).start().wait()
```

多模态RAG的论文系统

方案2：效果展示

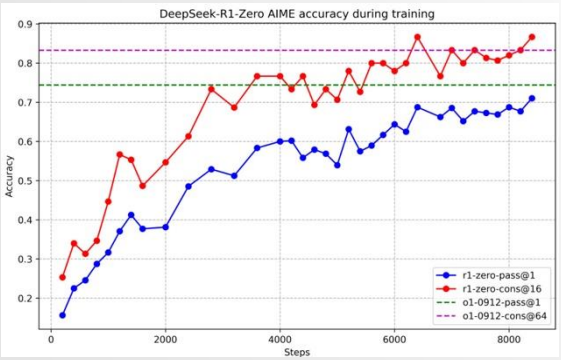
block节点组召回内容：

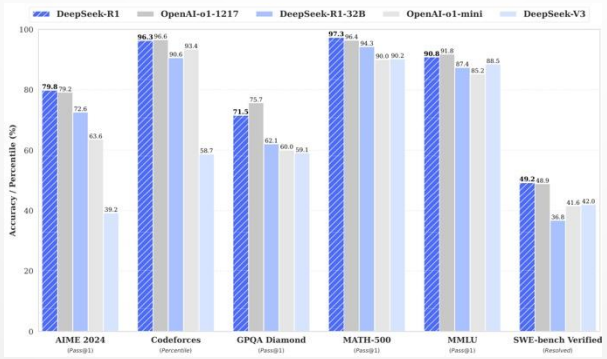
For education-oriented knowledge benchmarks such as MMLU, MMLU-Pro, and GPQA Diamond, DeepSeek-R1 demonstrates superior performance compared to DeepSeek-V3. This improvement is primarily attributed to enhanced accuracy in STEM-related questions, where significant gains are achieved through large-scale reinforcement learning. Additionally, DeepSeek-R1 excels on FRAMES, a long-context-dependent QA task, showcasing its strong document analysis capabilities. This highlights the potential of reasoning models in AI-driven search and data analysis tasks. On the factual benchmark SimpleQA, DeepSeek-R1 outperforms DeepSeek-V3, demonstrating its capability in handling fact-based queries. A similar trend is observed where OpenAI-o1 surpasses GPT-4o on this benchmark. However, DeepSeek-R1 performs worse than DeepSeek-V3 on the Chinese SimpleQA benchmark, primarily due to its tendency to refuse answering certain queries after safety RL. Without safety RL, DeepSeek-R1 could achieve an accuracy of over 70%.

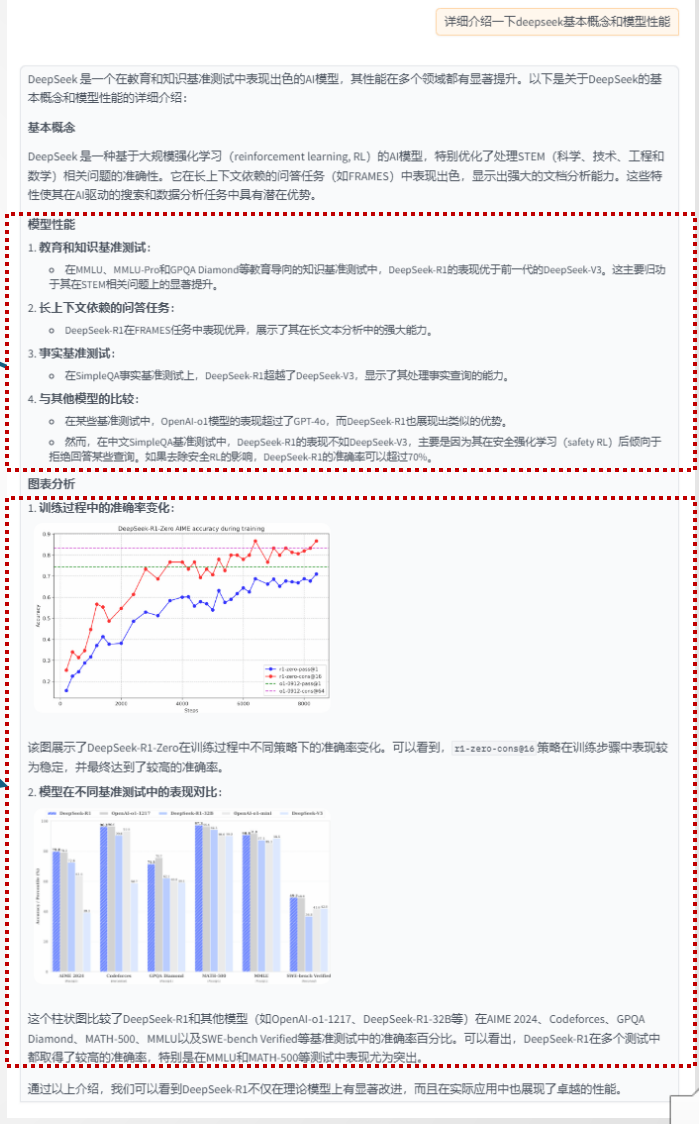
Image 图像节点组召回内容：

/home/mnt/sunxiaoye/.lazyllm/rag_for_qa/images/c357bd57757e4c544fdda3ad32066e64c5d01bb8540066f1a722c8872d664183.jpg

/home/mnt/sunxiaoye/.lazyllm/rag_for_qa/images/2c6271b8cecc68d5b3c22e552f407a5e97d34030f91e15452c595bc8a76e291c.jpg







目录

1. 上节回顾

2. 传统RAG的论文系统

3. 朴素多模态RAG的论文系统

3. 论文系统的效果优化

4. 总结拓展

改进方案

针对上述论文问答助手，我们还可以从以下几个角度进行优化：

优化0. 多路召回

- **问题**：

由于不同类型的文档结构复杂、信息密度不同，单一的召回策略往往难以适应所有场景。

- **解决方案**：

并行多路召回

优化1. 文本QA对总结提取

- **问题**：

召回的文档往往包含大量冗余信息，直接传递给 LLM 可能会影响生成质量，导致回答不够精准或聚焦。

- **解决方案**：

预先使用 LLM 对解析后的文段进行处理，让 LLM 自动生成摘要标题和相关问题，并构造高质量的问答对（QA Pairs）和文档总结（Summary）。

优化2. 图像QA对提取

- **问题**：

纯文本的回答有时不能很好的解决用户的问题，与文本相比，很多图表更加清晰。

解决方案：

1. 利用多模态大模型对图像内容进行解析，生成图像的文字描述。

2.将图像文字描述、路径等信息存储在ImageDocNode中。

3.利用Lazyllm的LLM_Parse根据图像的文字描述生成QA对。

优化3. PDF转图化繁为简

- **问题**：

PDF解析过于复杂，代码功能设计繁琐。

解决方案：

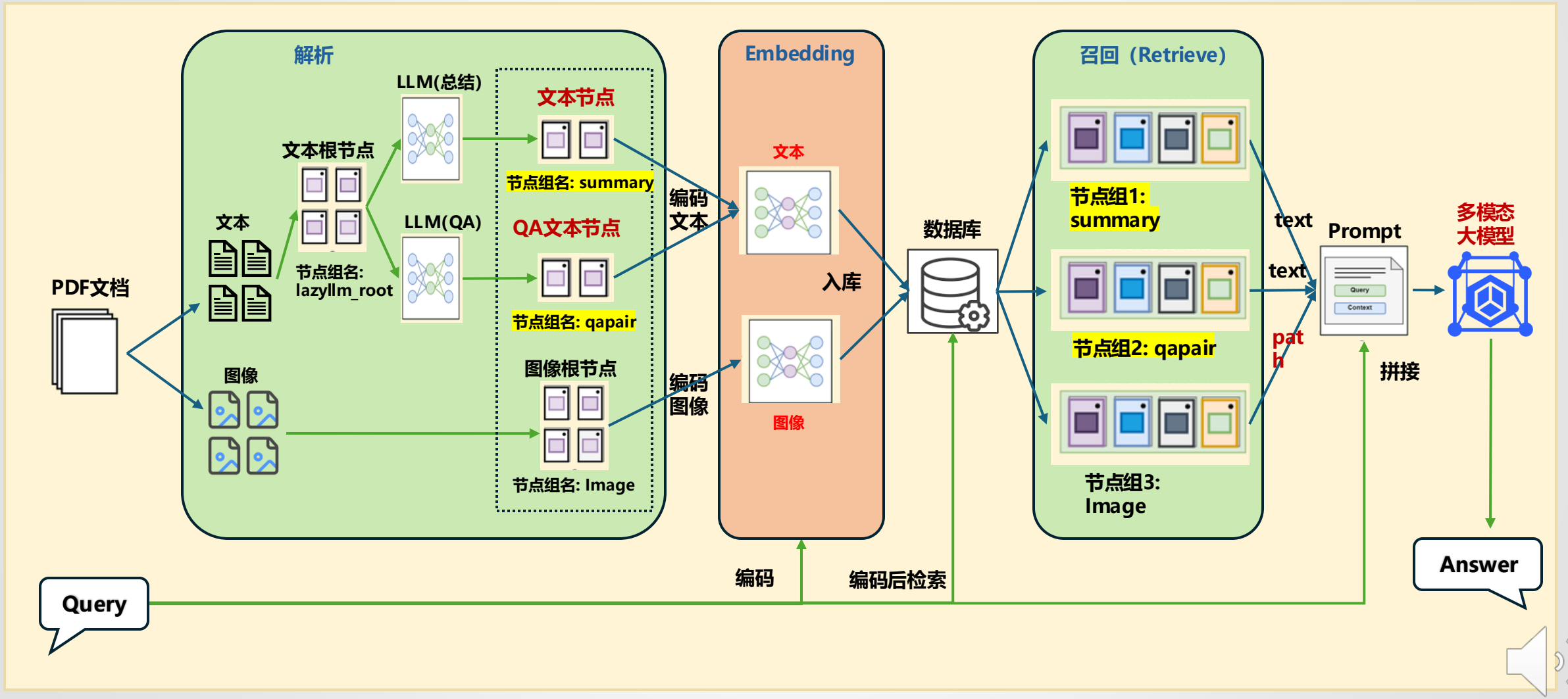
1. 将PDF直接换行为图片；

2. 使用专门针对图文混合版式的多模态嵌入模型对其进行向量化；

3. 将和query匹配到的图像与query本身送给多模态大模型来做回答。

论文系统的效果优化1

优化1： + 文本QA对 & Summary提取 方案



论文系统的效果优化1

代码实现 —— 应用编排

```
embed_mltimodal = lazyllm.TrainableModule("colqwen2-v0.1")
embed_text = lazyllm.TrainableModule("bge-m3")
embeds = {'vec1': embed_text, 'vec2': embed_mltimodal}

qapair_llm = lazyllm.LLMParser(lazyllm.OnlineChatModule(stream=False), language="zh", task_type="qa")
summary_llm = lazyllm.LLMParser(lazyllm.OnlineChatModule(stream=False), language="zh", task_type="summary")

documents = lazyllm.Document(dataset_path=tmp_dir.rag_dir, embed=embeds, manager=False)
documents.add_reader("*.pdf", MagicPDFReader)
documents.create_node_group(name="summary", transform=lambda d: summary_llm(d), trans_node=True)
documents.create_node_group(name='qapair', transform=lambda d: qapair_llm(d), trans_node=True)

with lazyllm.pipeline() as ppl:
    with lazyllm.parallel().sum as ppl.prl:
        ppl.prl.retriever1 = lazyllm.Retriever(documents, group_name="qapair", embed_keys=['vec1'], similarity="cosine", topk=1)
        ppl.prl.retriever2 = lazyllm.Retriever(documents, group_name="summary", embed_keys=['vec1'], similarity="cosine", topk=1)
        ppl.prl.retriever3 = lazyllm.Retriever(documents, group_name="Image", embed_keys=['vec2'], similarity="maxsim", topk=2)
    ppl.prompt = build_vlm_prompt | bind(_0, ppl.input)
    ppl.vlm = lazyllm.OnlineChatModule(source="sensenova", model="SenseNova-V6-Turbo").prompt(lazyllm.ChatPrompter(gen_prompt))

lazyllm.WebModule(ppl, port=range(23468, 23470), static_paths=get_image_path()).start().wait()
```

论文系统的效果优化1

效果展示

qapair节点组召回内容：
query:
What capabilities does DeepSeek-R1-Zero demonstrate?
answer
DeepSeek-R1-Zero demonstrates remarkable reasoning capabilities and naturally emerges with numerous powerful and intriguing reasoning behaviors through RL.

summary节点组召回内容：
性能对比表格摘要:
- 不同模型在多个基准测试中的表现，包括MMLU、MMLU-Redux、MMLU-Pro、DROP、IF-Eval、GPQA Diamond、SimpleQA、FRAMES、AlpacaEval2.0、ArenaHard等英语理解测试，以及LiveCodeBench、Codeforces、SWE Verified、Aider-Polyglot等代码能力测试和AIME、MATH-500、CNMO等数学能力测试。
- Claude-3.5-Sonnet-1022 0513、GPT-4o DeepSeek V3、OpenAI OpenAI 01-mini o1-1217、DeepSeek R1等模型在不同测试中各有优劣，例如DeepSeek R1在多数测试中表现优异，尤其在Codeforces和Aider-Polyglot中表现突出。
- 各模型在架构、激活参数、总参数和MMLU (Pass@1)等指标上的差异。
- 中文评估中，各模型在C-Eval和C-SimpleQA上的表现。

Image 图像节点组召回内容：
/path/to/images/2c6271b8cecc68d5b3c22e552f407a5e97d34030f91e15452c595bc8a76e291c.jpg
/path/to/images/b671779ae926ef62c9a0136380a1116f31136c3fd1ed3fedc0e3e05b90925c20.jpg

介绍DeepSeek-R1模型在哪些方面表现出色?

DeepSeek-R1模型在多个领域表现出色，具体如下：

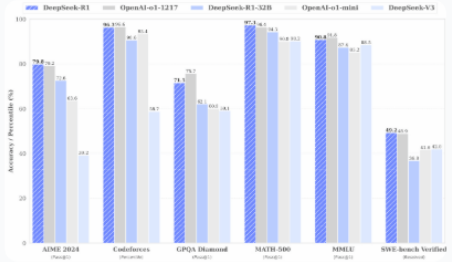
1. **推理能力**：根据参考资料中的描述，DeepSeek-R1-Zero展现了卓越的推理能力，并通过强化学习（RL）自然地发展出许多强大且有趣的推理行为。

2. **代码能力**：在性能对比中，DeepSeek-R1在Codeforces和SWE-bench Verified测试中表现突出，显示出其在解决复杂编程问题方面的能力。

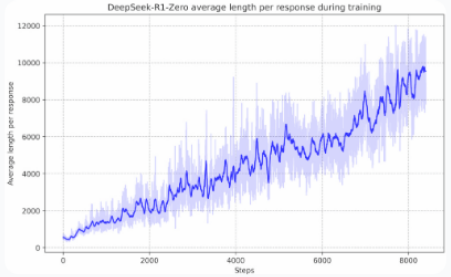
3. **数学能力**：在AIME 2024和MATH-500等数学测试中，DeepSeek-R1也取得了优异的成绩，尤其是在高难度的数学问题上。

4. **多领域理解**：在MMLU、GPQA Diamond、MATH-500等多个基准测试中，DeepSeek-R1都显示出了较高的准确率，表明其在广泛的知识领域内具有良好的理解和应答能力。

这些结果如图所示：



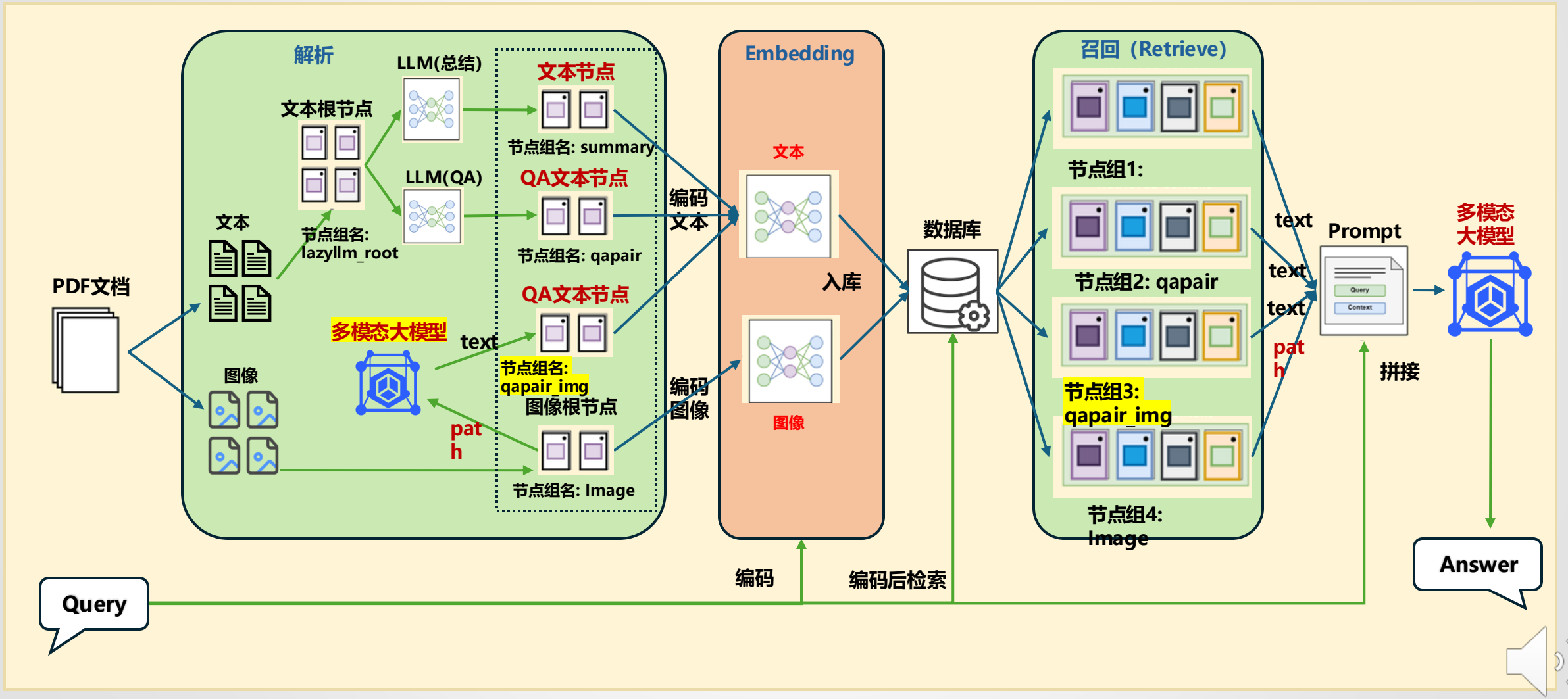
此外，图表还展示了DeepSeek-R1在训练过程中每次响应的平均长度变化，这反映了模型在处理任务时生成输出的能力和效率。



综上所述，DeepSeek-R1在推理、数学、编码以及多领域知识理解方面均表现出色。

论文系统的效果优化2

优化2：+ 图片QA对 方案



论文系统的效果优化2

代码实现 ——应用编排

```
embed_mltimodal = lazyllm.TrainableModule("colqwen2-v0.1")
embed_text = lazyllm.TrainableModule("bge-m3")
embeds = {'vec1': embed_text, 'vec2': embed_mltimodal}

qapair_llm = lazyllm.LLMParser(lazyllm.OnlineChatModule(stream=False), language="zh", task_type="qa")
qapair_img_llm = lazyllm.LLMParser(
    lazyllm.OnlineChatModule(source="sensenova", model="SenseNova-V6-Turbo"), language="zh", task_type="qa_img")
summary_llm = lazyllm.LLMParser(lazyllm.OnlineChatModule(stream=False), language="zh", task_type="summary")

documents = lazyllm.Document(dataset_path=tmp_dir.rag_dir, embed=embeds, manager=False)
documents.add_reader("*.pdf", MagicPDFReader)
documents.create_node_group(name="summary", transform=lambda d: summary_llm(d), trans_node=True)
documents.create_node_group(name='qapair', transform=lambda d: qapair_llm(d), trans_node=True)
documents.create_node_group(name='qapair_img', transform=lambda d: qapair_img_llm(d), trans_node=True, parent='Image')

with lazyllm.pipeline() as ppl:
    with lazyllm.parallel().sum as ppl.prl:
        ppl.prl.retriever1 = lazyllm.Retriever(documents, group_name="summary", embed_keys=['vec1'], similarity="cosine", topk=1)
        ppl.prl.retriever2 = lazyllm.Retriever(documents, group_name="Image", embed_keys=['vec2'], similarity="maxsim", topk=2)
        ppl.prl.retriever3 = lazyllm.Retriever(documents, group_name="qapair", embed_keys=['vec1'], similarity="cosine", topk=1)
        ppl.prl.retriever4 = lazyllm.Retriever(documents, group_name="qapair_img", embed_keys=['vec1'], similarity="cosine", topk=1)

    ppl.prompt = build_vlm_prompt | bind(_0, ppl.input)
    ppl.vlm = lazyllm.OnlineChatModule(source="sensenova", model="SenseNova-V6-Turbo").prompt(lazyllm.ChatPrompter(gen_prompt))

lazyllm.WebModule(ppl, port=range(23468, 23470), static_paths=get_image_path()).start().wait()
```

论文系统的效果优化2

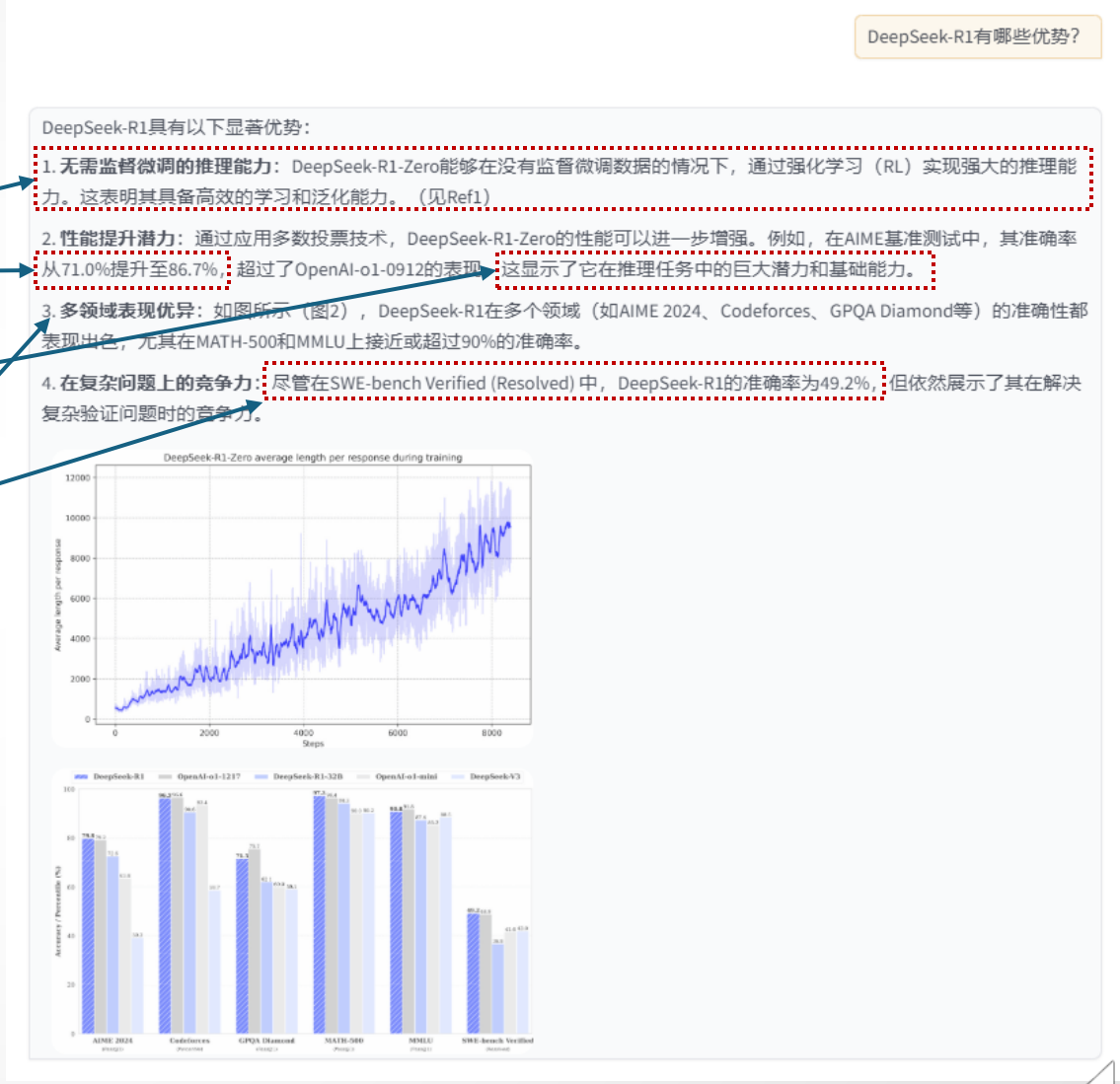
效果展示

summary 节点组召回内容：
DeepSeek-R1-Zero无需监督微调即可实现强大推理能力，通过仅使用强化学习展现出高效学习和泛化能力。应用多数投票可进一步提升其性能，如在AIME基准测试中，其性能从71.0%提升至86.7%，超越OpenAI-o1-0912。这表明DeepSeek-R1-Zero具备强大的基础能力，有潜力在推理任务中实现更多进展。

qapair 节点组召回内容：
query:
DeepSeek-R1-Zero的性能表现说明了什么？
answer
DeepSeek-R1-Zero的性能表现突显了其强大的基础能力和在推理任务中进一步提升的潜力。

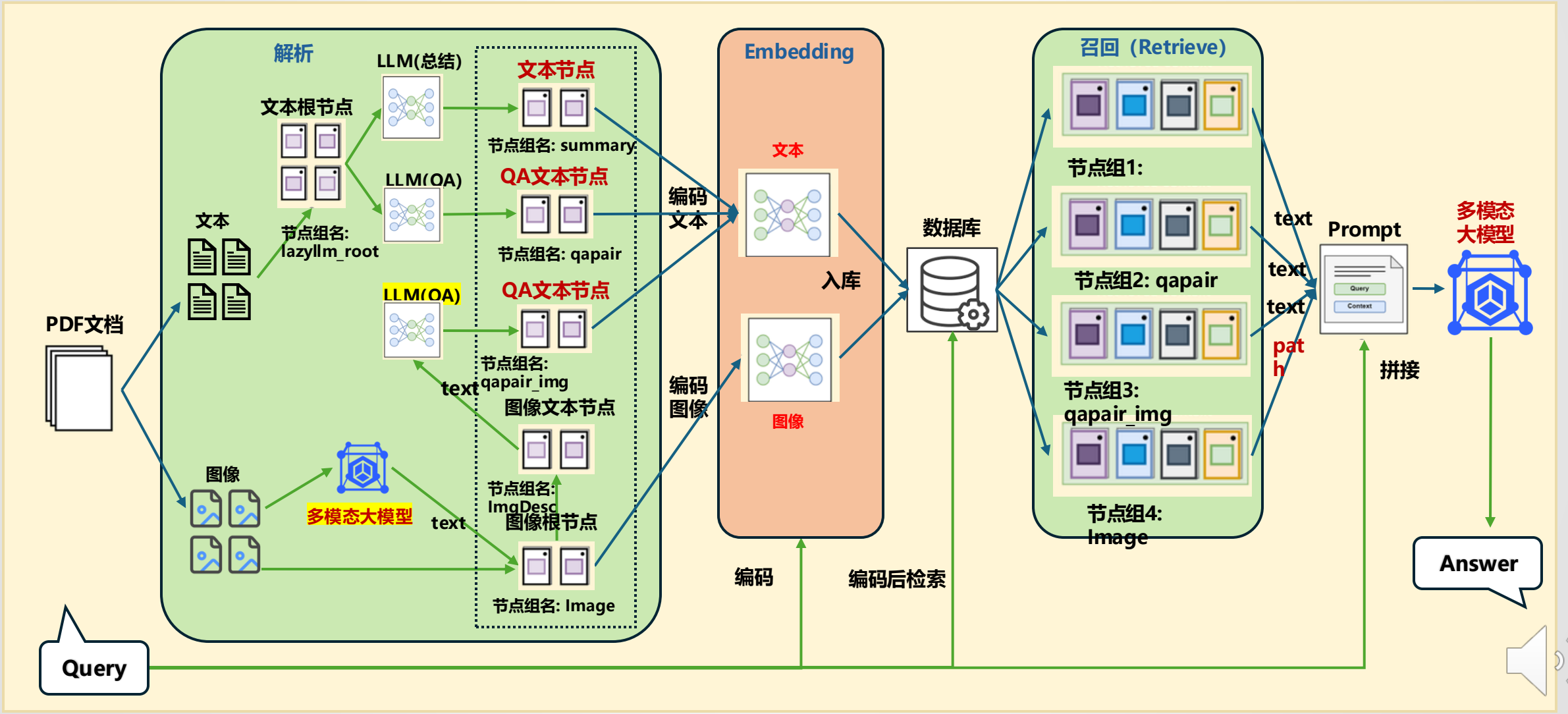
qapair_img 节点组召回内容：
query:
在SWE-bench Verified (Resolved) 中，DeepSeek-R1的准确率是多少？
answer
49.2%

Image 节点组召回内容：
/path/to/images/b671779ae926ef62c9a0136380a1116f31136c3fd1ed3fedc0e3e05b90925c20.jpg
(2)
/path/to/images/2c6271b8cecc68d5b3c22e552f407a5e97d34030f91e15452c595bc8a76e291c.jpg



论文系统的效果优化2

优化2： + 图片QA对(变种) 方案



论文系统的效果优化2

代码实现 ——应用编排（变种）

```
1   embed_mltimodal = lazyllm.TrainableModule("colqwen2-v0.1")
2   embed_text = lazyllm.TrainableModule("bge-m3")
3   embeds = {'vec1': embed_text, 'vec2': embed_mltimodal}
4
5   qapair_llm = lazyllm.LLMParser(lazyllm.OnlineChatModule(stream=False), language="zh", task_type="qa")
6   summary_llm = lazyllm.LLMParser(lazyllm.OnlineChatModule(stream=False), language="zh", task_type="summary")
7
8   documents = lazyllm.Document(dataset_path=tmp_dir.rag_dir, embed=embeds, manager=False)
9   documents.add_reader("*.pdf", MagicPDFReader)
10  documents.create_node_group(name="summary", transform=lambda d: summary_llm(d), trans_node=True)
11  documents.create_node_group(name='qapair', transform=lambda d: qapair_llm(d), trans_node=True)
12  documents.create_node_group(name='qapair_img', transform=lambda d: qapair_llm(d), trans_node=True, parent='ImgDesc')
13
14  with lazyllm.pipeline() as ppl:
15      with lazyllm.parallel().sum as ppl.prl:
16          ppl.prl.retriever1 = lazyllm.Retriever(documents, group_name="summary", embed_keys=['vec1'], similarity="cosine", topk=1)
17          ppl.prl.retriever2 = lazyllm.Retriever(documents, group_name="Image", embed_keys=['vec2'], similarity="maxsim", topk=2)
18          ppl.prl.retriever3 = lazyllm.Retriever(documents, group_name="qapair", embed_keys=['vec1'], similarity="cosine", topk=1)
19          ppl.prl.retriever4 = lazyllm.Retriever(documents, group_name="qapair_img", embed_keys=['vec1'], similarity="cosine", topk=1)
20
21      ppl.prompt = build_vlm_prompt | bind(_0, ppl.input)
22      ppl.vlm = lazyllm.OnlineChatModule(source="sensenova", model="SenseNova-V6-Turbo").prompt(lazyllm.ChatPrompter(gen_prompt))
23
24  lazyllm.WebModule(ppl, port=range(23468, 23470), static_paths=get_image_path()).start().wait()
```

论文系统的效果优化2

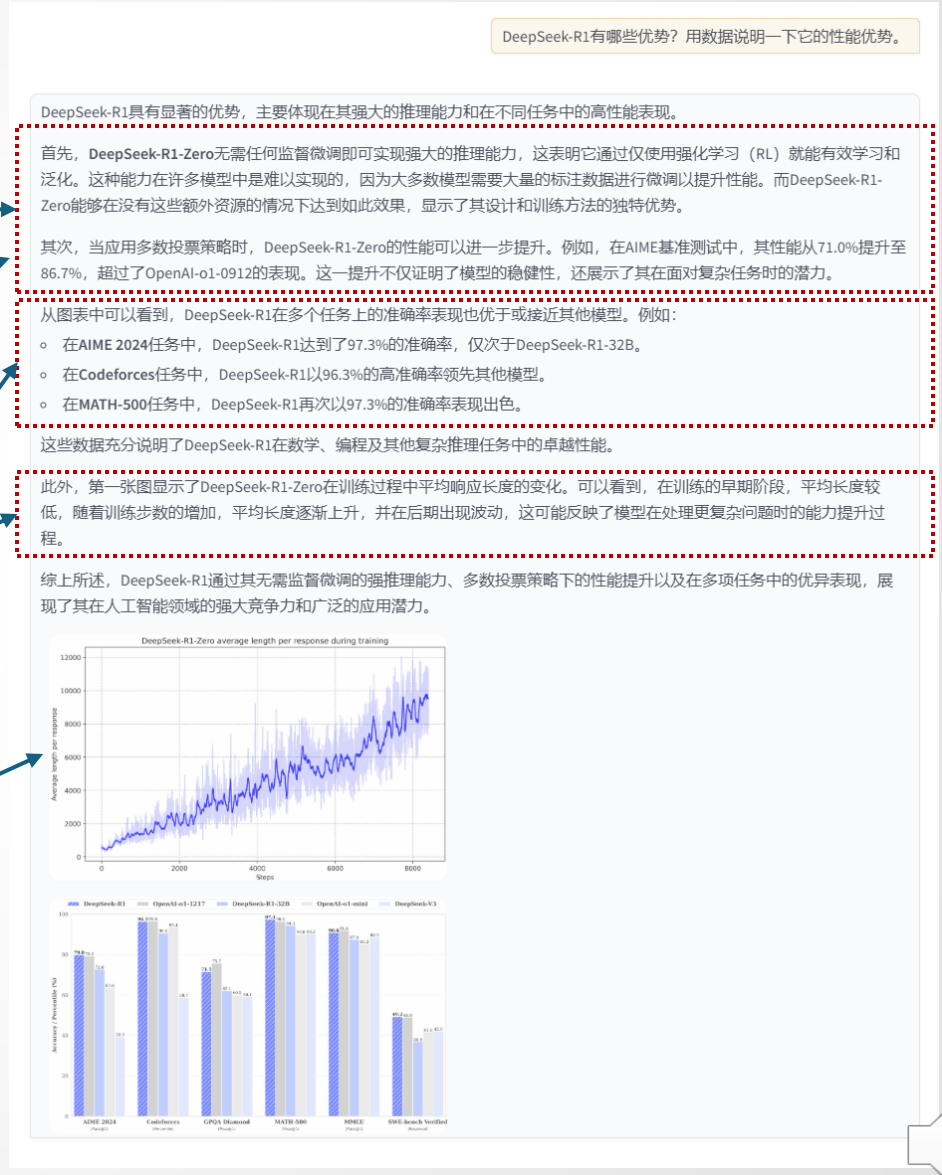
效果展示1

summary 节点组召回内容：
DeepSeek-R1-Zero无需监督微调即可实现强推理能力，通过仅使用强化学习显示其有效学习和泛化能力。采用多数投票可进一步提升其性能，如在AIME基准测试中，性能从71.0%提升至86.7%，超过OpenAI-o1-0912。其有竞争力表现展示了强大的基础能力和在推理任务中进一步发展的潜力。

qapair 节点组召回内容：
query:
DeepSeek-R1-Zero的性能表现说明了什么?
answer
DeepSeek-R1-Zero的性能表现突显了其强大的基础能力和在推理任务中进一步发展的潜力。

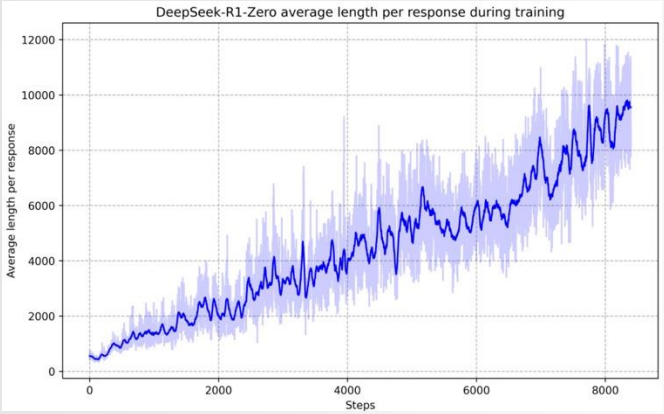
qapair_img 节点组召回内容：
query:
图表显示了DeepSeek-R1-Zero在训练过程中的平均长度如何变化?
answer
在训练的早期阶段，平均长度相对较低，随着步数的增加，平均长度逐渐上升，并在后期阶段出现波动。

Image 节点组召回内容：
/path/to/images/b671779ae926ef62c9a0136380a1116f31136c3fd1ed3fedc0e3e05b90925c20.jpg
/path/to/images/2c6271b8cecc68d5b3c22e552f407a5e97d34030f91e15452c595bc8a76e291c.jpg

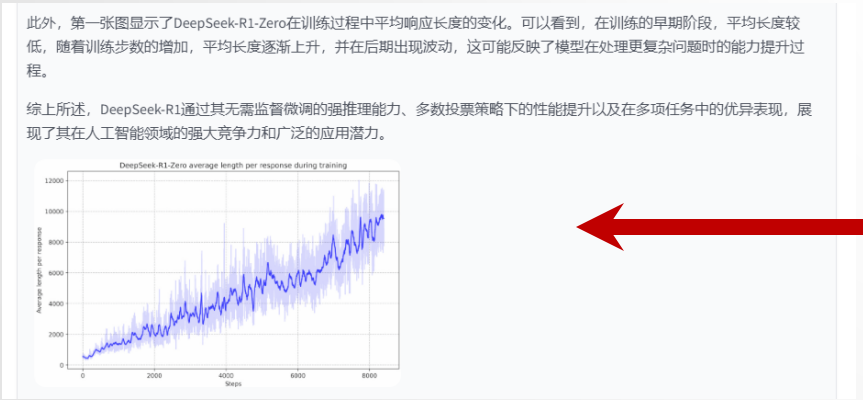


论文系统的效果优化2

效果展示2——图像QA细节展示



✓ 从PDF中解析出的原图像



✓ 最终问答效果

✓ 多模态大模型生成描述

这张图片展示了一个图表，该图表显示了DeepSeek-R1-Zero在训练过程中平均长度随步数增加而变化的趋势。图表中的蓝色线表示平均长度，而浅蓝色的阴影区域可能表示训练过程中的不确定性或误差范围。
在图表中，x轴代表步数，从0到大约8000，而y轴表示平均长度，从0到大约12000。图表显示，在训练的早期阶段，平均长度相对较低，随着步数的增加，平均长度逐渐上升，并在后期阶段出现波动。图表中的线条和阴影区域显示了训练过程中的变化，可能反映了DeepSeek-R1-Zero在处理数据或任务时所遇到的复杂性和不确定性。
请注意，由于图表中没有明确标注x轴和y轴的具体单位，上述描述是基于图表的视觉呈现，并未提供具体的数值信息。

✓ 针对生成的描述利用LLM生成QA对

Q: 这张图片展示了一个图表，它显示了什么内容？
A: 这张图片展示了一个图表，该图表显示了DeepSeek-R1-Zero在训练过程中平均长度随步数增加而变化的趋势。

Q: 图表中的蓝色线表示什么？
A: 图表中的蓝色线表示平均长度。

Q: 图表中的浅蓝色阴影区域可能表示什么？
A: 浅蓝色的阴影区域可能表示训练过程中的不确定性或误差范围。

命中的QA对

Q: 图表的x轴和y轴分别代表什么？
A: x轴代表步数，从0到大约8000，而y轴表示平均长度，从0到大约12000。

Q: 图表显示了DeepSeek-R1-Zero在训练过程中的平均长度如何变化？
A: 在训练的早期阶段，平均长度相对较低，随着步数的增加，平均长度逐渐上升，并在后期阶段出现波动。

Q: 图表中的线条和阴影区域可能反映了什么？
A: 线条和阴影区域可能反映了DeepSeek-R1-Zero在处理数据或任务时所遇到的复杂性和不确定性。

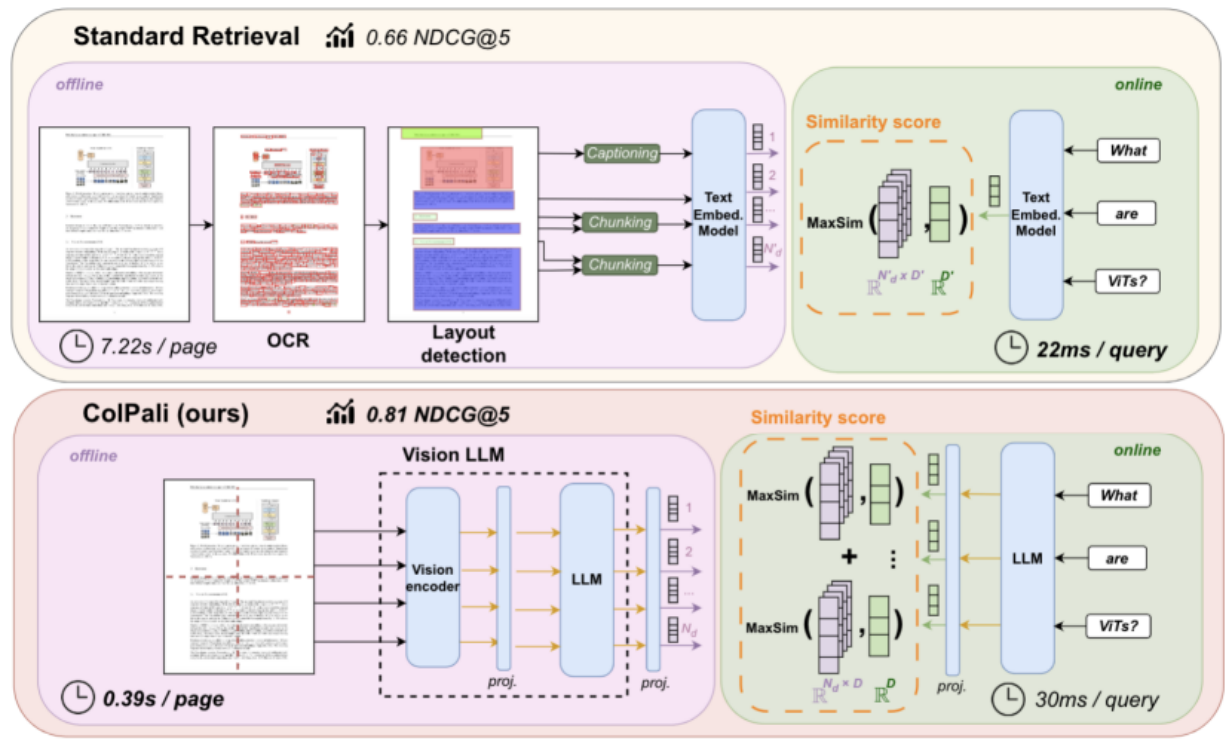
Q: 描述中是否提供了具体的数值信息？
A: 没有，描述中没有提供具体的数值信息，仅基于图表的视觉呈现。

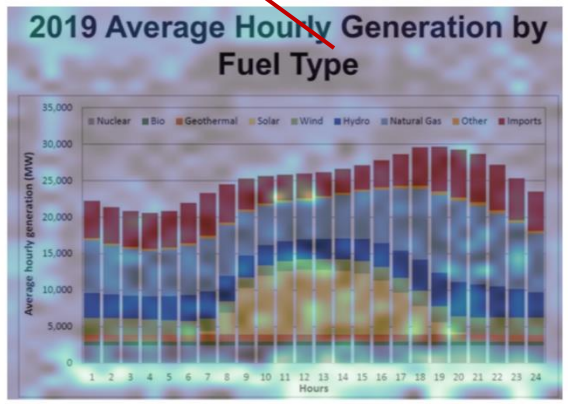
论文系统的效果优化3

优化3：PDF转图化繁为简

不同模型切分文档图像的块数不一样：
ColQwen2-VL模型将图像最大切分为： 768份
ColPali模型将图像最大切分为： 1024份

- 传统方法：将文档结构化解析 → 图像 & 文本分别处理
- 新方向：**将整页文档视为图像**，使用多模态嵌入模型直接编码





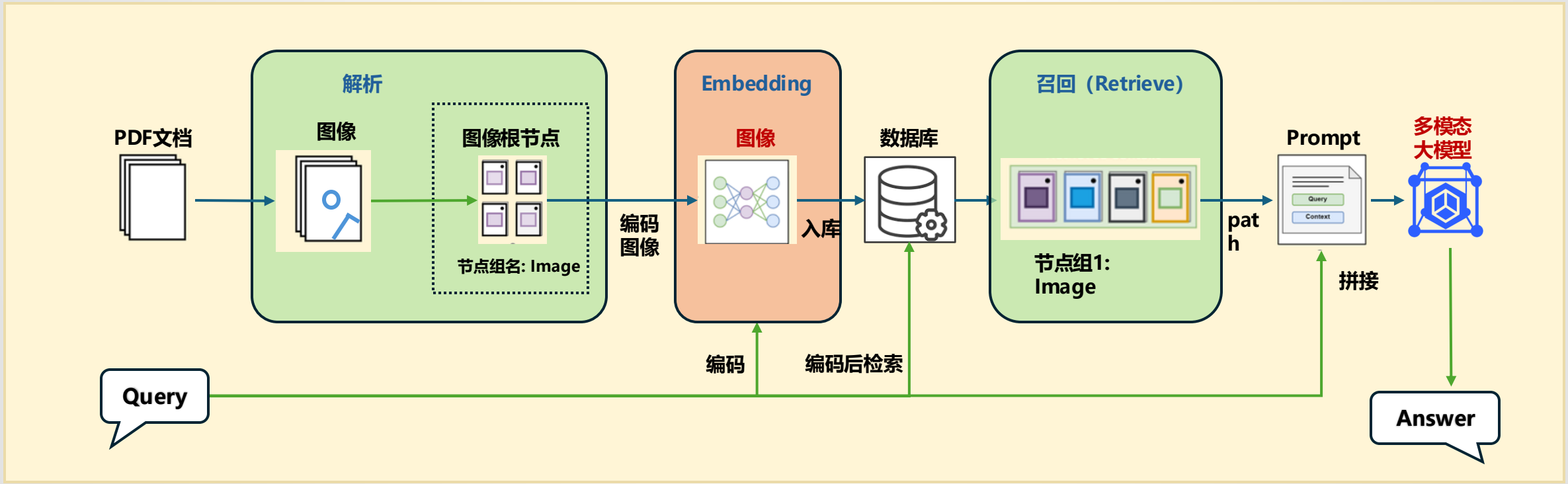
Query: "Which hour of the day had the highest overall eletricity generation in 2019?"

对于用户查询中的每一项目，ColPali 会识别出最相关的文档图像区域（被高亮显示的区域），并计算查询与页面之间的匹配分数。

这里显示了“hour”和图像中的“Hours”及其时间高度相关。

论文系统的效果优化3

优化3：PDF转图化繁为简 LazyLLM方案



论文系统的效果优化3

代码实现——PDF转图片

✓ 之前的方案中，文本解析阶段我们采用的是先解析文本、在构建节点的方式，流程相对复杂。这里直接将PDF解析为图像。供给后续多模态大模型使用。

```
# PDF转图像阅读器实现类
class Pdf2ImageReader(ReaderBase):
    def __init__(self, image_save_path="pdf_image_path"):
        super().__init__(); self.image_save_path = image_save_path
    if not os.path.exists(self.image_save_path): os.makedirs(self.image_save_path)

# PDF文件加载和转换核心方法
    def _load_data(self, file: Path, extra_info=None) -> List[ImageDocNode]:
        if not isinstance(file, Path): file = Path(file)
        docs = fitz.open(file); file_path = []
        for page_num in range(docs.page_count):
            metadata = extra_info or {}; metadata["file_name"] = file.name; metadata["file_split"] = page_num
            page = docs.load_page(page_num); pix = page.get_pixmap(dpi=300)
            img = Image.frombytes("RGB", [pix.width, pix.height], pix.samples)
            save_path = f"{self.image_save_path}/{file.name[:-4]}_{page_num}.jpg"
            img.save(save_path); file_path.append(ImageDocNode(image_path=save_path, global_metadata=metadata))
    return file_path
```

通过上述代码，就可以通过 ImageDocNode.get_content()获取图像 PIL 对象，也可以通过 ImageDocNode.image_path 获取对应的存储路径，方便后续进行进一步的操作。

论文系统的效果优化3

优化相似度计算方式——MaxSim相似度计算法

基本概念

MaxSim相似度计算法是基于Token级最大匹配的延迟交互算法

核心原理

- **延迟交互机制**：
  不直接计算全局相似度，而是逐Token交互后聚合
- **最大匹配策略**：
  每个查询Token寻找文档中的最大响应值

计算公式

$$S(q,d) = \sum_{i=1}^{|E_q|} \left(\max_{j=1}^{|E_d|} E_q^{(i)} \cdot E_d^{(j)T} \right)$$

符号说明：

- E_q：查询向量矩阵（每行为一个Token的嵌入向量）
- E_d：文档向量矩阵（每行为Token/图像块的嵌入向量）
- .：向量点积运算

计算步骤分解

1.**Token级交互**：

1. 计算查询每个Token与文档所有Token的点积
2. 形成交互矩阵：$Mij = Eq(i) \cdot Ed(j)T$

2.**最大值提取**：

1. 对每行取最大值：max$jMij$
2. 反映该查询Token在文档中的最佳匹配强度

3. **全局聚合**：

1. 对所有查询Token的最大值求和
2. 最终得分反映整体语义匹配强度

论文系统的效果优化3

优化相似度计算方式——MaxSim相似度计算过程

查询（Query）：2个Token，向量分别为 $Eq = \begin{bmatrix} 0.5 & -0.2 \\ 0.3 & 0.8 \end{bmatrix}$ **文档（Doc）**：3个Token，向量分别为 $Ed = \begin{bmatrix} 0.1 & 0.4 \\ -0.3 & 0.6 \\ 0.7 & -0.5 \end{bmatrix}$

步骤1：通过点积运算（对应元素相乘后求和）：

- $M11 = 0.5 \times 0.1 + (-0.2) \times 0.4 = 0.05 - 0.08 = -0.03$
- $M12 = 0.5 \times (-0.3) + (-0.2) \times 0.6 = -0.15 - 0.12 = -0.27$
- $M13 = 0.5 \times 0.7 + (-0.2) \times (-0.5) = 0.35 + 0.10 = 0.45$
- $M21 = 0.3 \times 0.1 + 0.8 \times 0.4 = 0.03 + 0.32 = 0.35$
- $M22 = 0.3 \times (-0.3) + 0.8 \times 0.6 = -0.09 + 0.48 = 0.39$
- $M23 = 0.3 \times 0.7 + 0.8 \times (-0.5) = 0.21 - 0.40 = -0.19$

$M = \begin{bmatrix} -0.03 & -0.27 & 0.45 \\ 0.35 & 0.39 & -0.19 \end{bmatrix}$

步骤2：取每行最大值：

第一行最大值：$\max(-0.03, -0.27, 0.45) = 0.45$

第二行最大值：$\max(0.35, 0.39, -0.19) = 0.39$

步骤3：全局求和

最终相似度：$S(q,d) = 0.45 + 0.39 = 0.84$

论文系统的效果优化3

代码实现——MaxSim相似度计算法

```
# 注册自定义相似度计算方法(支持批量处理)
@lazyllm.tools.rag.register_similarity(mode='embedding', batch=True)
def maxsim(query, nodes, **kwargs):
    batch_size = 128
    scores_list = []
    query = [torch.Tensor(query) for i in range(len(nodes))]
    nodes_embed = [torch.Tensor(node) for node in nodes]
    for i in range(0, len(query), batch_size):
        scores_batch = []
        query_batch = torch.nn.utils.rnn.pad_sequence(query[i:i + batch_size], batch_first=True, padding_value=0)
        for j in range(0, len(nodes_embed), batch_size):
            nodes_batch = torch.nn.utils.rnn.pad_sequence(
                nodes_embed[j:j + batch_size],
                batch_first=True,
                padding_value=0
            )
            scores_batch.append(torch.einsum("bnd,csd->bcns", query_batch, nodes_batch).max(dim=3)[0].sum(dim=2))
        scores_batch = torch.cat(scores_batch, dim=1).cpu()
        scores_list.append(scores_batch)
    scores = scores_list[0][0].tolist()
    return scores
```

论文系统的效果优化3

代码实现——应用编排

```
# 定义将图像路径转换为markdown格式的函数
def format_markdown_image(text):
    json_part = text[text.index("{"):]; data = json.loads(json_part)
    image_paths = data.get("files", []); return f'\n\n![]({image_paths[0]})'

# 初始化文档处理模块并添加PDF阅读器
    image_file_path = "/content/images"
    documents = lazyllm.Document(dataset_path="/content/doc",
                                 embed=lazyllm.TrainableModule("colqwen2-v0.1"))
    documents.add_reader("*.pdf", Pdf2ImageReader(image_file_path))

# 构建图像检索和处理流水线
    with pipeline() as ppl:
        ppl.retriever = Retriever(doc=documents, group_name="Image", similarity="maxsim", topk=1)
        ppl.formatter1 = lambda nodes : [node.image_path for node in nodes]
        ppl.formatter2 = encode_query_with_filepaths | bind(ppl.input, _0)
        with parallel().sum as ppl.prl:
            ppl.prl.vlm = lazyllm.OnlineChatModule(source="sensenova", model="SenseChat-Vision")
            ppl.prl.post_action = format_markdown_image
# 启动Web服务
lazyllm.WebModule(ppl, static_paths=image_file_path).start().wait()
```

论文系统的效果优化3

效果展示

Chatbot

为我介绍下DeepSeekR1这篇论文

DeepSeek-R1 是由 DeepSeek-AI 开发的一组新型推理模型，旨在通过强化学习（RL）激励大语言模型（LLMs）的推理能力。这篇论文介绍了两个主要模型：DeepSeek-R1-Zero 和 DeepSeek-R1。

主要内容概述：

1. **模型介绍：**
   - **DeepSeek-R1-Zero**：该模型通过大规模强化学习训练，而不是依赖监督微调（SFT）。它展示了出色的推理能力，并自然涌现出许多强大且有趣的推理行为。
   - **DeepSeek-R1**：为了解决 DeepSeek-R1-Zero 遇到的问题（如可读性差、语言混杂等），DeepSeek-R1 引入了多阶段训练和在进行RL之前的冷启动数据。
2. **性能表现：**
   - DeepSeek-R1 在推理任务上的表现与 OpenAI 的 o1-1217 模型相当。
   - 论文中的图表展示了 DeepSeek-R1 在多个基准测试中的表现，包括 AIME 2024、Codeforces、GPQA Diamond、MATH-500、MMLU 和 SWE-bench Verified。这些测试结果表明 DeepSeek-R1 具有很强的推理和问题解决能力。
3. **开源贡献：**
   - 为了支持研究社区，DeepSeek-AI 开源了 DeepSeek-R1-Zero、DeepSeek-R1 以及六个从 DeepSeek-R1 基于 Qwen 和 Llama 模型蒸馏出的密集模型（1.5B, 7B, 8B, 14B, 32B, 70B）。

图表分析：

图1 展示了 DeepSeek-R1 在多个基准测试中的性能。以下是一些关键观察点：

- 在 AIME 2024 (Pass@1) 中，DeepSeek-R1 和 OpenAI-o1-1217 表现接近，均超过 79%。
- 在 Codeforces (Percentile) 中，DeepSeek-R1 超过了 96%，显著高于其他模型。
- GPQA Diamond (Pass@1) 中，DeepSeek-R1-32B 表现优于 OpenAI-o1-mini。

Chatbot

为了支持研究社区，DeepSeek-AI 开源了 DeepSeek-R1-Zero、DeepSeek-R1 以及六个从 DeepSeek-R1 基于 Qwen 和 Llama 模型蒸馏出的密集模型（1.5B, 7B, 8B, 14B, 32B, 70B）。

图表分析：

图1 展示了 DeepSeek-R1 在多个基准测试中的性能。以下是一些关键观察点：

- 在 AIME 2024 (Pass@1) 中，DeepSeek-R1 和 OpenAI-o1-1217 表现接近，均超过 79%。
- 在 Codeforces (Percentile) 中，DeepSeek-R1 超过了 96%，显著高于其他模型。
- GPQA Diamond (Pass@1) 中，DeepSeek-R1-32B 表现优于 OpenAI-o1-mini。
- MATH-500 (Pass@1) 中，DeepSeek-R1 和 OpenAI-o1-1217 的表现相近，均在 96% 左右。
- MMLU (Pass@1) 中，DeepSeek-R1 略低于 OpenAI-o1-1217，但仍然表现出色。
- SWE-bench Verified (Resolved) 中，DeepSeek-R1 的表现略逊于其他模型，但依然具有竞争力。

总体而言，DeepSeek-R1 通过创新的强化学习方法显著提升了语言模型的推理能力，并在多个基准测试中展现了卓越的表现。开源这些模型也为进一步的研究提供了宝贵的资源。

deepseek

DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning

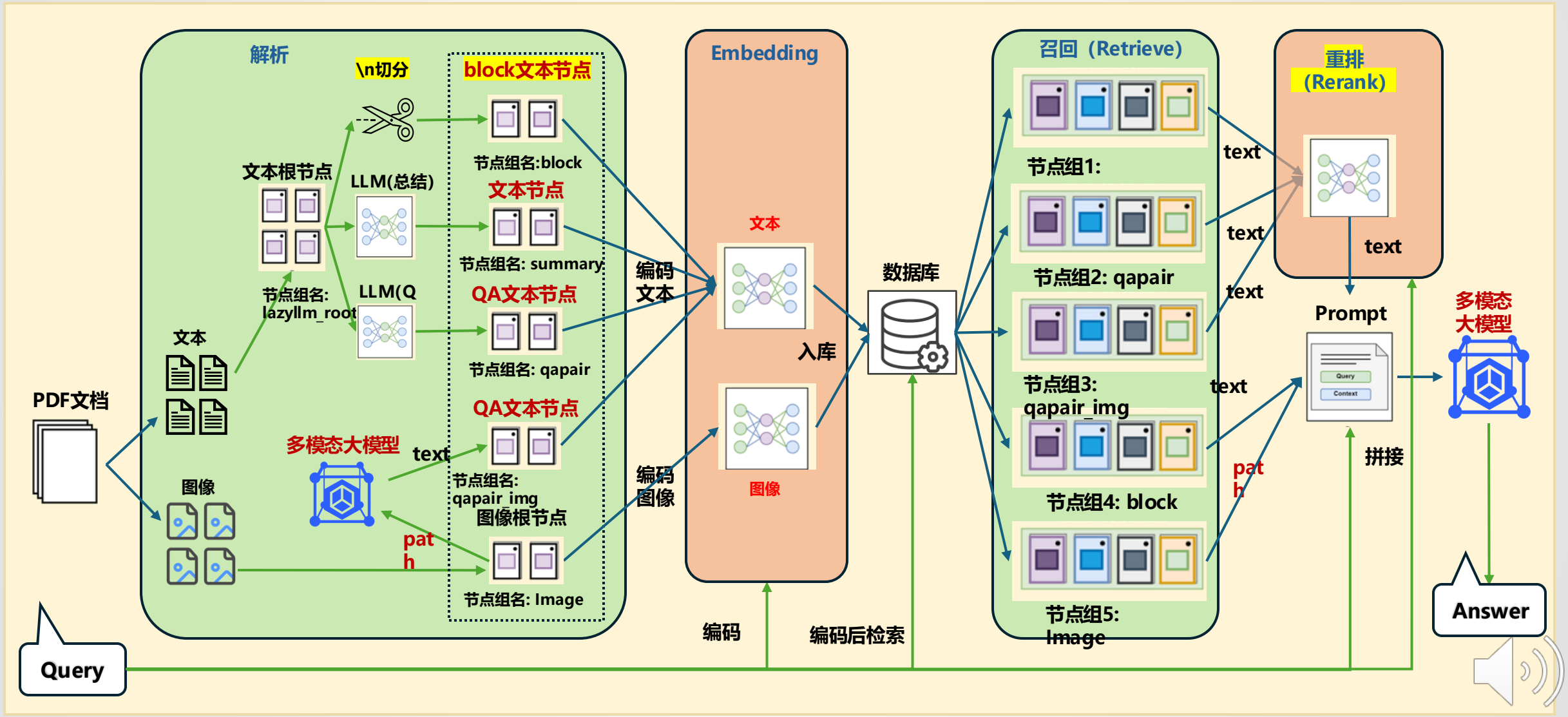
DeepSeek-AI

Abstract

We introduce our first-generation reasoning models, DeepSeek-R1-Zero and DeepSeek-R1. DeepSeek-R1-Zero, a model trained via large-scale reinforcement learning (RL) without supervised fine-tuning (SFT) as a preliminary step, demonstrates remarkable reasoning capabilities. Through RL, DeepSeek-R1-Zero naturally emerges with numerous powerful and intriguing reasoning behaviors. However, it encounters challenges such as poor readability, and language mixing. To address these issues and further enhance reasoning performance, we introduce DeepSeek-R1, which incorporates multi-stage training and cold-start data before RL. DeepSeek-R1 achieves performance comparable to OpenAI-o1-1217 on reasoning tasks. To support the research community, we open-source DeepSeek-R1-Zero, DeepSeek-R1, and six dense models (1.5B, 7B, 8B, 14B, 32B, 70B) distilled from DeepSeek-R1 based on Qwen and Llama.

论文系统综合方案

加上Rerank、按段落切分等基础策略



论文系统综合方案

代码实现 ——应用编排

```
embed_mltimodal = lazyllm.TrainableModule("colqwen2-v0.1")
embed_text = lazyllm.TrainableModule("bge-m3")
embeds = {'vec1': embed_text, 'vec2': embed_mltimodal}

qapair_llm = lazyllm.LLMParser(lazyllm.OnlineChatModule(stream=False), language="zh", task_type="qa")
qapair_img_llm = lazyllm.LLMParser(
    lazyllm.OnlineChatModule(source="sensenova", model="SenseNova-V6-Turbo"), language="zh", task_type="qa_img")
summary_llm = lazyllm.LLMParser(lazyllm.OnlineChatModule(stream=False), language="zh", task_type="summary")

documents = lazyllm.Document(dataset_path=tmp_dir.rag_dir, embed=embeds, manager=False)
documents.add_reader("*.pdf", MagicPDFReader)
documents.create_node_group(name="block", transform=lambda s: s.split("\n") if s else '')
documents.create_node_group(name="summary", transform=lambda d: summary_llm(d), trans_node=True)
documents.create_node_group(name='qapair', transform=lambda d: qapair_llm(d), trans_node=True)
documents.create_node_group(name='qapair_img', transform=lambda d: qapair_img_llm(d), trans_node=True, parent='Image')

with lazyllm.pipeline() as ppl:
    with lazyllm.parallel().sum as ppl.mix:
        with lazyllm.pipeline() as ppl.mix.rank:
            with lazyllm.parallel().sum as ppl.mix.rank.short:
                ppl.mix.rank.short.retriever1 = lazyllm.Retriever(documents, group_name="summary", embed_keys=['vec1'], similarity="cosine", topk=4)
                ppl.mix.rank.short.retriever2 = lazyllm.Retriever(documents, group_name="qapair", embed_keys=['vec1'], similarity="cosine", topk=4)
                ppl.mix.rank.short.retriever3 = lazyllm.Retriever(documents, group_name="qapair_img", embed_keys=['vec1'], similarity="cosine", topk=4)
            ppl.mix.rank.reranker = lazyllm.Reranker("ModuleReranker", model="bge-reranker-large", topk=3) | bind(query=ppl.mix.rank.input)
        ppl.mix.retriever4 = lazyllm.Retriever(documents, group_name="block", embed_keys=['vec1'], similarity="cosine", topk=2)
        ppl.mix.retriever5 = lazyllm.Retriever(documents, group_name="Image", embed_keys=['vec2'], similarity="maxsim", topk=2)

    ppl.prompt = build_vlm_prompt | bind(_0, ppl.input)
    ppl.vlm = lazyllm.OnlineChatModule(source="sensenova", model="SenseNova-V6-Turbo").prompt(lazyllm.ChatPrompter(gen_prompt))

lazyllm.WebModule(ppl, port=range(23468, 23470), static_paths=get_image_path()).start().wait()
```

论文系统的效果优化2

效果展示1

召回内容：

内容1：DeepSeek-R1 outperforms DeepSeek-V3 on education benchmarks and long-context QA, excelling in STEM and factual queries via reinforcement learning. However, it underperforms on Chinese SimpleQA due to safety RL restrictions.

内容2：DeepSeek-R1在IF-Eval、AlpacaEval2.0和ArenaHard上表现出色，得益于SFT和RL训练中包含的指令跟随数据。其优于DeepSeek-V3，展示了大型RL的泛化优势，提升推理能力和跨领域表现。生成的摘要长度平均为689个token（ArenaHard）和2218个字符（AlpacaEval2.0），表明其简洁性。

内容3：各模型在多种基准测试中的性能对比，包括MMLU、Codeforces、AIME等，显示不同模型在英语、代码和数学等领域的差异化表现。

内容4： We introduce our frst-generation reasoning models, DeepSeek-R1-Zero and DeepSeek-R1. DeepSeek-R1-Zero, a model trained via large-scale reinforcement learning (RL) without supervised fne-tuning (SFT) as a preliminary step, demonstrates remarkable reasoning capabilities. Through RL, DeepSeek-R1-Zero naturally emerges with numerous powerful and intriguing reasoning behaviors. However, it encounters challenges such as poor readability, and language mixing. To address these issues and further enhance reasoning performance, we introduce DeepSeek-R1, which incorporates multi-stage training and cold-start data before RL. DeepSeekR1 achieves performance comparable to OpenAI-o1-1217 on reasoning tasks. To support the research community, we open-source DeepSeek-R1-Zero, DeepSeek-R1, and six dense models (1.5B, 7B, 8B, 14B, 32B, 70B) distilled from DeepSeek-R1 based on Qwen and Llama.

内容5：| | Benchmark (Metric) | Claude-3.5- Sonnet-1022 0513 | GPT-4o DeepSeek V3 | | OpenAI OpenAI 01-mini o1-1217 | DeepSeek R1 |

Image **节点组召回内容：**

/path/to/images/2c6271b8cecc68d5b3c22e552f407a5e97d34030f91e15452c595bc8a76e291c.jpg
/path/to/images/b671779ae926ef62c9a0136380a1116f31136c3fd1ed3fedc0e3e05b90925c20.jpg

目录

1. 上节回顾

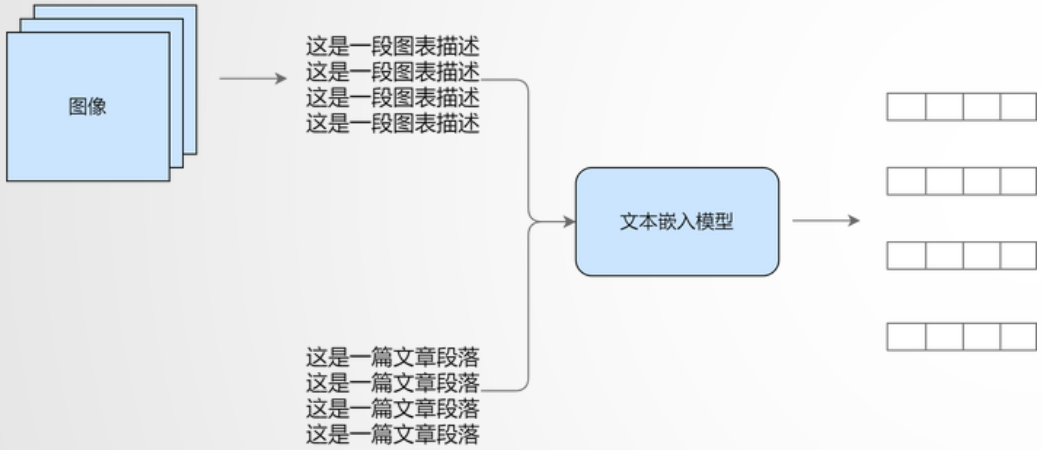
2. 传统RAG的论文系统

3. 朴素多模态RAG的论文系统

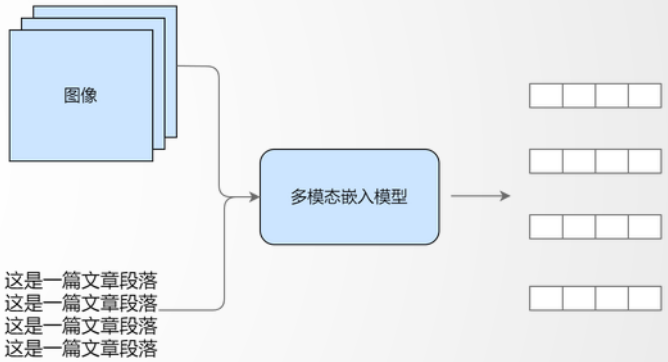
3. 论文系统的效果优化

4. 总结拓展

关键技术点



统一到文本模态，然后进行向量嵌入



统一到向量空间：利用多模态模型进行映射

优化策略

- **多路召回**：通过并行多路召回策略，系统能够结合多种检索方法，提高召回的全面性和鲁棒性。
- **图像QA对提取**：通过多模态大模型提取图像描述并生成高质量的问答对（QA Pairs），系统能够在召回阶段将匹配范围从文本扩展到图像，提升生成结果的丰富度和准确性。
- **QA文本对提取**：通过LLM生成高质量的问答对（QA Pairs），系统能够在召回阶段更精准地匹配用户查询，提升生成结果的相关性和准确性。
- **PDF转图化繁为简**：使用专门针对图文混合版式的多模态嵌入模型对转换为图的PDF文档进行向量化，省去了对PDF文档进行复杂解析和处理的逻辑，大大简化了开发工作。

Q&A

1.嵌入编码时，上传图片非RGB图像时，是否有多通道处理图像方式

感谢聆听

Thanks for Listening