RAG 技术详解与实战应用

第13讲：RAG+多模态：图片、表格通吃的问答系统

目录

上节回顾

向量数据库的应用

持久化存储：使用ChromaDB和Milvus作为存储后端，实现文档的持久化存储。

- ChromaDB：轻量级，适合调试和小规模应用。
- Milvus：高性能，适合大规模应用，支持分布式部署。

高速向量检索：通过配置store_conf参数，可以选择使用ChromaDB或Milvus作为存储后端。LazyLLM提供了对存储数据的增删改查接口，支持通过文档管理服务进行数据管理。

自定义索引的应用

- **自定义Index：**

定义了一个基于关键词的索引类KeywordIndex，通过提取每段话的关键词进行索引，实现快速检索。

- **注册和使用：**

通过register_index方法将自定义索引注册到LazyLLM框架中，并在RAG系统中使用。

- **使用milvus的向量索引：**

通过smart_embedding_index，修改向量索引为milvus

优化大模型推理

- **更换推理引擎**
- **使用量化的模型**

目录

为什么要引入多模态

在实际应用中，我们常常需要从合同、报告、产品说明书等多种形式的文档中提取有用信息。这些文档不仅包含丰富的文字内容，还可能包括图片、图表、表格等非文本信息。下面是某页PDF原始视图的信息展示。

打印机使用说明与参数信息

产品简介

本设备是一款多功能喷墨打印机，专为家庭办公环境设计。它具备打印、复印、扫描等功能，支持无线连接与移动打印，满足日常文档输出与学习资料打印需求，尤其适合家庭用户、学生和远程办公人群。

推荐使用场景

- 居家办公：支持日常文件、合同等材料的打印与扫描
- 学习资料输出：高质量打印教辅材料、作业、试卷等
- 多设备协同：支持手机、平板、电脑无线打印

产品参数一览表

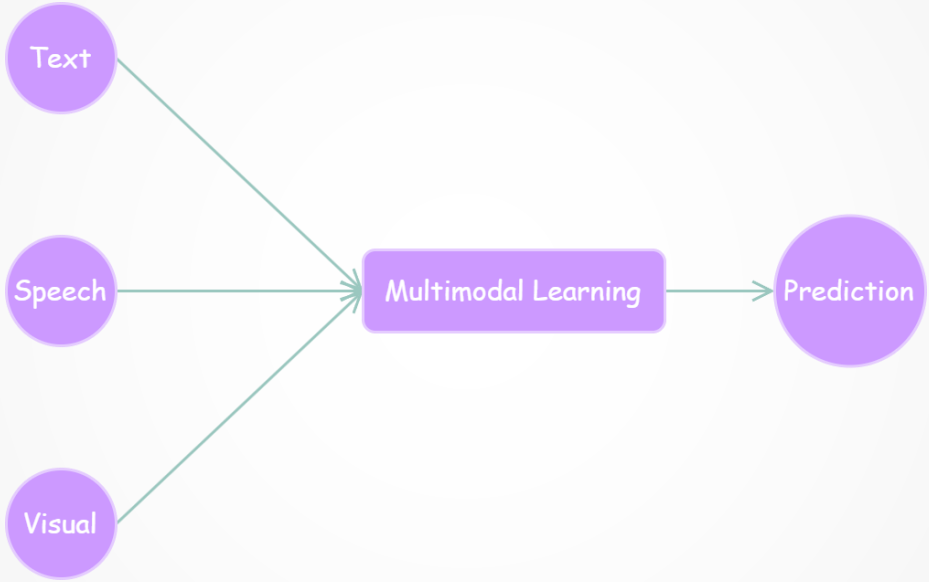
项目	参数说明
打印方式	热喷墨打印
最大分辨率	4800 x 1200 dpi
打印速度（黑/彩）	黑白约 10 页/分钟，彩色约 7 页/分钟
扫描功能	支持 A4 平板式扫描，分辨率 1200 dpi
复印功能	支持等比/缩放复印

类型	示例文本格式（结构化后的内容）	说明
普通 RAG	标题：打印机使用说明与参数信息	正文：本设备是一款多功能喷墨打印机，专为家庭办公环境设计…
	（⚠ 表格和图片信息缺失）	仅能处理纯文本部分，图片与表格内容被忽略或丢失
多模态 RAG	标题：打印机使用说明与参数信息	正文：本设备是一款多功能喷墨打印机，专为家庭办公环境设计…
	图片与表格内容识别	图像信息和表格内容被识别与解析，提供完整的多模态数据结构化输出

对比普通模型与多模态模型在读取图文混合 PDF 时生成的可用文本格式。可以看到，传统 RAG 仅依赖文本处理，难以识别图像和图表中的关键信息，导致内容缺失，尤其在处理复杂文档时问题突出。因此，引入多模态能力势在必行，通过结合 OCR 和图像理解，使模型具备“读图识意”的能力，从而实现更智能的问答与生成。

多模态大模型 – 简介

在现实世界中，信息从来不是单一模态的。人类的日常感知依赖于多种模态的融合输入，包括视觉、听觉、语言、触觉等，这些信息共同构建了我们对世界的理解。



- 当研究任务或数据同时涉及多种模态时，我们将其称为**多模态问题**（Multimodal Problem）。解决这类问题，是推动人工智能系统更贴近人类认知方式、实现 “类人智能” 的关键一步。
- 正因如此，**多模态大型语言模型**（MLLM）应运而生。它融合了语言大模型（LLM）的推理与理解能力，以及视觉大模型（LVM）的感知能力，弥补了单一模态处理的局限。通过联合建模和跨模态对齐等技术，MLLM能同时理解文本与图像，实现更强的多模态问答、检索与生成能力，推动人工智能向更全面的认知迈进。

多模态大模型 – 原理

1. 感知层：模态感知与特征提取

模态感知与特征提取不同模态数据（图像、语音、文本、视频）结构不同，需要各自的编码器提取特征，如图像用CNN或ViT，音频用频谱Transformer，文本用预训练语言模型（如BERT、GPT）。这一步的目标是**将不同模态的数据转化为统一的向量表示**（embedding）。

2. 对齐层：表示对齐与语义映射

表示对齐与语义映射由于各模态的向量来自不同空间，需通过连接器对齐语义表示。常用方法包括：MLP映射（将图像/音频向量转到文本空间）、跨模态注意力（使用Transformer建立模态间交互）、内部融合机制（比如图文混合输入）。这一层的关键是**让模型理解模态之间的语义联系**。

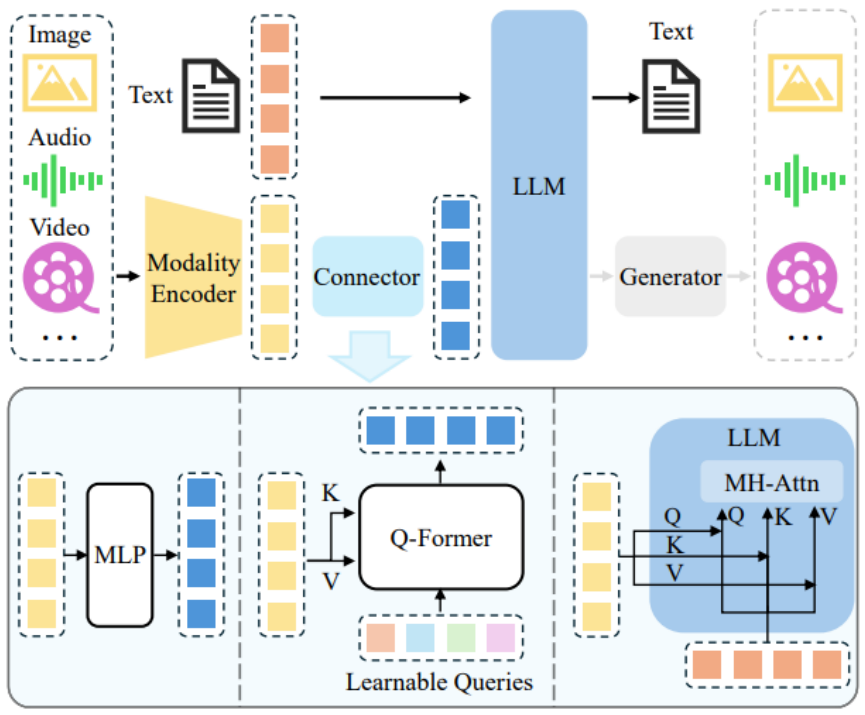
3. 理解与生成层：统一语义建模与跨模态推理

统一语义建模与跨模态推理所有模态的表示被映射到统一空间后，输入到LLM中进行整合、理解和生成。LLM作为推理核心，支持问答、描述、对话等任务，最终实现类人式的多模态理解与生成。

💡 **总结：**多模态大模型并非只是简单地把图像和文本“拼接”在一起，而是通过**编码 → 对齐 → 理解**的流程，实现了模态间的语义融合与统一推理。这种设计使得 MLLM 具备类人认知能力：能看图说话、听音理解、跨模态问答，甚至进行多模态创作。

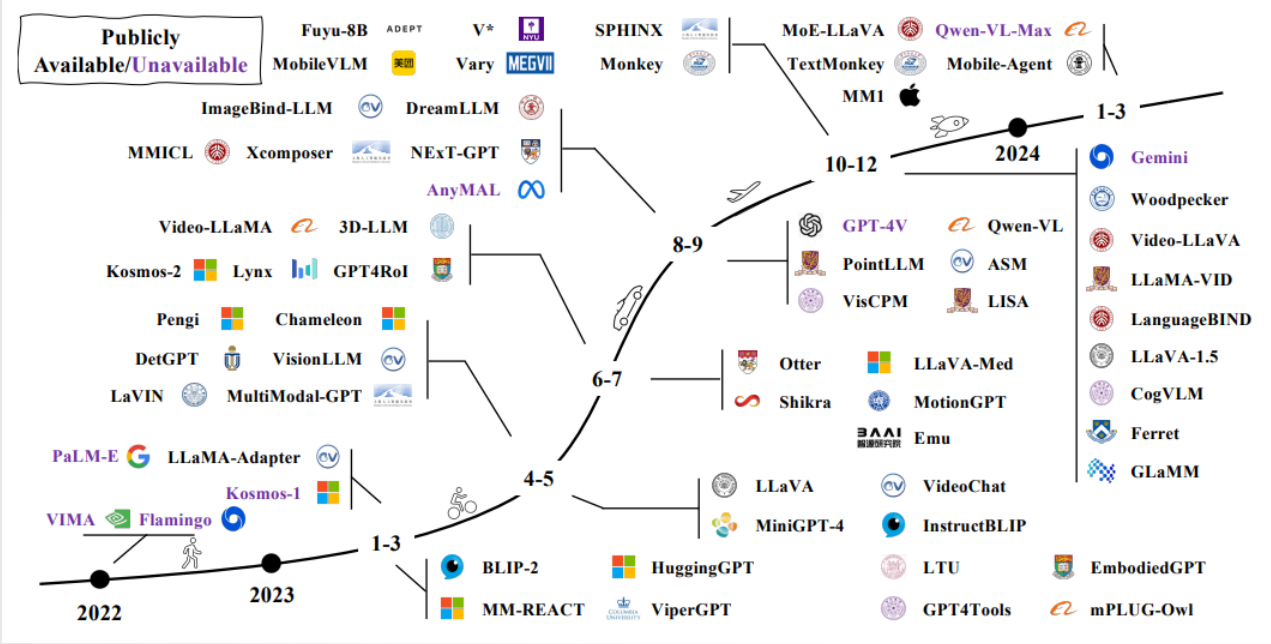
多模态大模型 – 架构

下图为多模态大模型的整体结构，通常包括三大核心模块：模态编码器（Modality Encoder）、连接器（Connector）以及语言模型（LLM）。



- **模态编码器（Modality Encoder）**
  - **作用：**将不同模态的数据（如图像、音频、视频、文本）转化为向量表示（embedding）。
  - **特点：**每种模态使用专门的模型进行编码，如 CNN/ViT 处理图像、语音模型处理音频、语言模型处理文本。
- **连接器（Connector）**
  - **作用：**将非文本模态的向量映射到与文本 embedding 相容或可对齐的空间，解决语义空间不一致的问题。
  - **常见连接方式：**MLP 映射：通过多层感知机进行空间转换；Cross-Attention 映射：使用跨模态注意力机制建立模态间交互；模型内部融合：如将图像 patch 与文本 token 混合输入，实现底层结构级融合。
- **语言模型（LLM）**
  - **作用：**接收统一后的向量表示，执行理解与生成任务；
  - **特点：**作为推理核心，LLM 支持图文对话、问答、描述等任务，是整个多模态系统的关键部分。

多模态大模型 – 主流开源模型



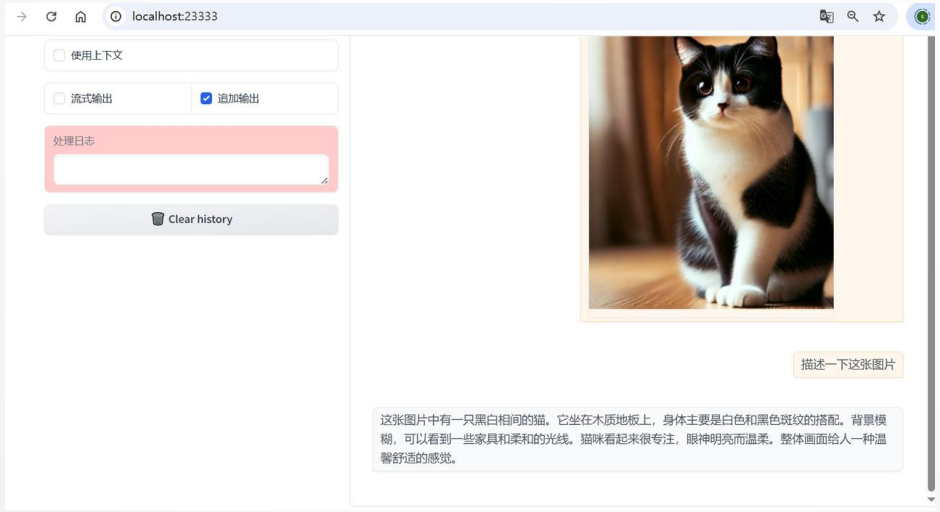
- **判别式范式（如 CLIP）**：判别式范式（如 CLIP）：侧重图文对齐，用于图像分类、检索等；
- **生成式范式（如 OFA、VL-T5、Flamingo）**：侧重跨模态生成，如图像描述、视觉问答。

- MLLM 延续生成式路线，借助强大语言模型和新训练范式，大幅拓展能力边界。
- 主流开源模型多采用 **“图像编码器 + 连接器 + LLM”** 的结构，如用 CLIP-ViT、BLIP 编码图像，映射至 LLaMA、Vicuna 等语言模型。通过多模态指令微调，模型具备看图说话、图文对话、跨模态推理等能力。
- 典型项目有 LLaVA、MiniGPT-4、InstructBLIP、Fuyu，持续推动多模态模型在实际应用中的发展。

在LazyLLM中使用多模态大模型

LazyLLM 已支持多模态大模型接入（如图文问答、图文理解等场景），可通过以下方式快速启动并使用。以智谱的多模态模型 glm-4v-flash 为例，使用 LazyLLM 启动服务：

```
1. import lazyllm
2. chat = lazyllm.OnlineChatModule(source="glm", model="glm-4v-flash")
3. lazyllm.WebModule(chat, port=23333, files_target=chat).start().wait()
```

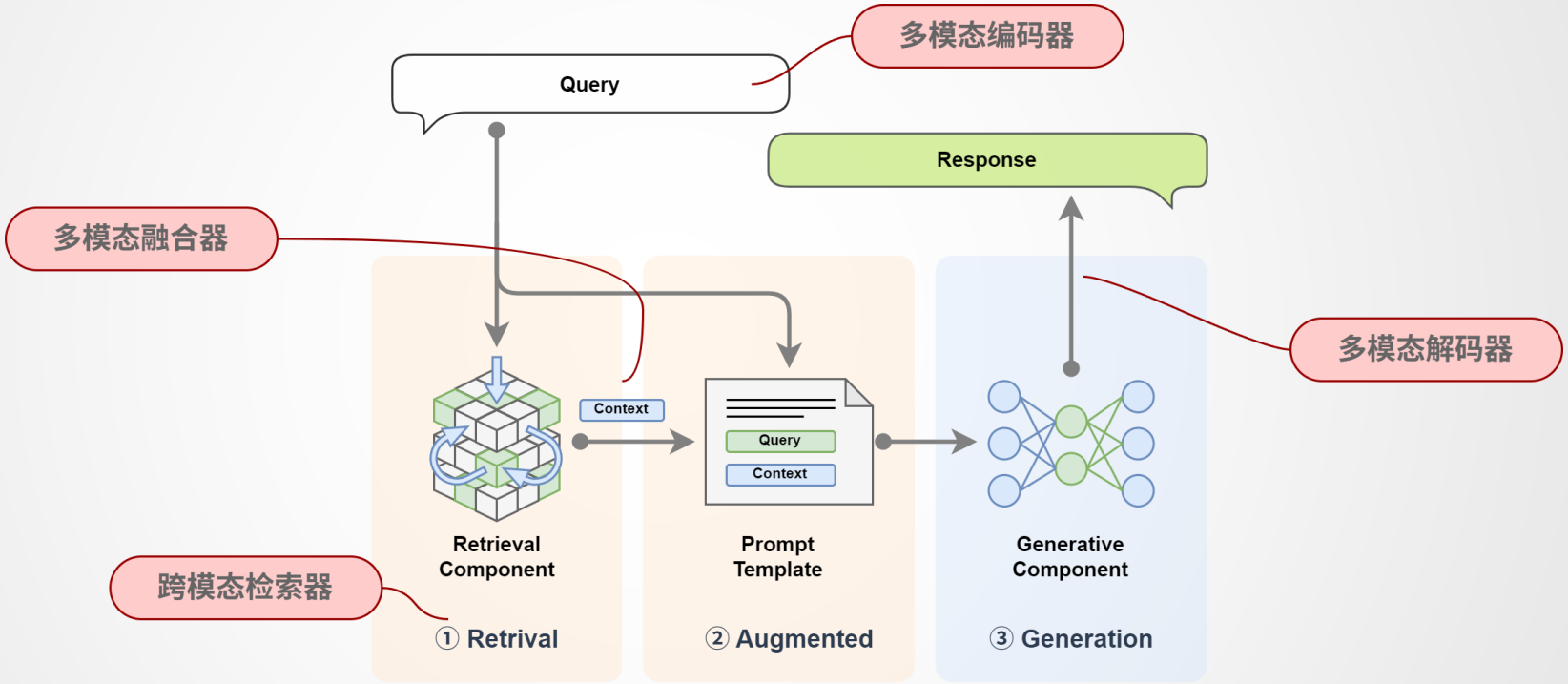


除了 glm-4v-flash，LazyLLM 同样支持多种主流多模态大模型，如Qwen-VL 系列、OpenAI GPT-4V等，但需要用户自行申请和配置API Key 。

目录

多模态RAG架构

在前面的 RAG 架构图中，我们主要针对文本数据进行了检索与生成。然而，要支持多模态输入和输出，需要对原有 RAG 架构进行一定的调整和扩展。

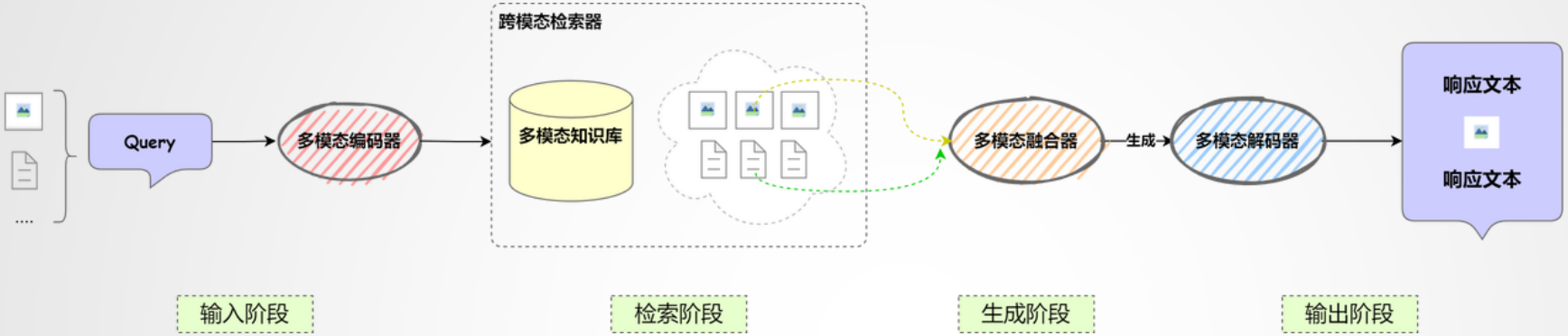


需要修改的模块包括：

- 检索模块：原本仅支持文本检索，现在需要扩展为支持多模态检索，例如图像、音频等信息的索引和匹配。
- 生成模块：原始 RAG 仅针对文本生成，现在需要扩展支持多模态输出，如文本结合图像、音频的生成能力。

多模态RAG核心组件 – 总述

- 为了支持多模态输入和输出，调整和扩展后的多模态RAG 流程图如下。



- 需要新增的模块包括以下四个部分：

模块	功能
多模态编码器	用于对不同模态的数据（文本、图像、音频等）进行编码，以便统一表示并用于检索和生成。
多模态融合器	用于融合不同模态的信息，使其能够协同作用，提高生成内容的准确性和丰富度。
跨模态检索器	支持输入多种数据格式，并能在多模态知识库中找到相关信息。
多模态解码器	负责将生成结果解码为多种形式，如文本、图片、语音等，以适应不同的输出需求。

多模态RAG核心组件 - 编码与融合

- 下面是常用编码模型：

1. **文本编码常用模型：**
   - BAAI/bge系列（如 bge-m3：支持多语言和长文本处理，生成高质量语义嵌入）
   - Cohere Embed（提供高效的语义编码和上下文理解，适用于文本检索与生成任务）
   - GTE-large（专注于高精度语义匹配与检索，适合大规模文本数据处理）
2. **多模态编码常用模型：**
   - 图像编码：NFNet-F6、ViT、CLIP ViT
   - 音频编码：Whisper、CLAP
   - 视频编码：CMVC

- 常见编码方法有三种：
  - **统一模态**：将非文本模态（如图像、音频、表格）转换为描述性文本，再用文本编码器统一编码，得到语义向量。
  - **共享向量空间**：通过多模态编码器（如CLIP、CLAP）直接将多模态数据编码到同一向量空间，无需中间转换，保留原始信息，统一接口，工程友好。
  - **分离检索**：不同模态使用专门模型分别编码，检索阶段融合结果，灵活性强，适配性好，适合复杂场景。

多模态RAG核心组件 – 检索

1. 基础检索

- 处理单一模态的查询，如文本到文本、图像到图像的匹配。如图（左）。

输入	输出
文本查询："一只黑白相间的小猫"	相关文本段落，如"黑白相间的小猫通常是美国短毛猫"
	相关图像，如下： 

输入	输出
文本查询："一只黑白相间的小猫"	相关图像，如一张黑白猫的图片，如下： 
这是什么猫 	根据相关文本段落，这种黑白相间的小猫通常是美国短毛猫

- 检索通常依赖嵌入相似度计算，如余弦相似度：$\text{sim}(q,d) = \frac{\mathbf{q}\cdot\mathbf{d}}{\|\mathbf{q}\|\|\mathbf{d}\|}$ 。 其中q是查询文档的嵌入向量，d是候选文档的嵌入向量。如果计算出的相似度高于某个阈值（例如0.9），则认为该文档相关。

2. 跨模态检索

- 支持跨模态查询，例如文本查询匹配图像，或图像查询匹配文本。如图（右）。

多模态RAG核心组件 – 生成

- 原有 RAG 仅支持文本生成，现在扩展为支持多模态输出，例如生成带有图片的回答或结合语音的解释。

输入	输出
文本查询："黑白相间的猫长什么样？"	- 图文回答：文本 + 一张黑白猫的图片，如下：  这里是一只黑白相间的小猫的图片！它有柔软的毛发，圆圆的眼睛充满好奇，耳朵竖起，显得十分机灵。它安静地坐在木地板上，小爪子整齐地收在身下，尾巴轻轻地盘绕在身侧，背景则是一个温馨的家居环境。希望你喜欢！

- 其中，x是输入的检索内容，yt是当前生成的词，$P(y|x) = \prod_{t=1}^{T} P(y_t|y_{<t}, x)$ 。 y<t是前面已经生成的词。假设模型可能按步骤生成了P("A")=0.8；P("black")=0.9；P("and")=0.85；P("white")=0.95，最终的输出则为"A black and white cat"。

多模态 – 三阶段示例

接下来以「旅行问答」场景为例，介绍多模态RAG三个阶段的结果输出。

1. 编码阶段 - 输入：

- 用户查询（文本+图）："这张照片里的建筑是什么风格？" + [拜占庭风格教堂照片]
- 知识库内容：
  - 文本1：《拜占庭建筑特点》文档（含穹顶、马赛克等关键词）
  - 图片2：哥特式教堂照片
  - 图片3：威尼斯圣马可大教堂照片
  - ...

【query图片】

【图片2】

【图片3】

多模态 – 三阶段示例

1. 编码阶段 - 编码结果示例

```
# 多模态编码 （如CLIP混合编码器实现）
1. query_text_embed = [0.23, -0.57, ..., 0.89]  # query的embedding，均为512维
2. doc1_text_embed = [0.20, -0.52, ..., 0.91]   # 文本1的embedding， 与 query_text_embed 相似

3. query_img_embed = [0.67, 0.12, ..., -0.33]  # query中图片的embedding
4. img2_embed = [0.02, 0.45, ..., 0.11]        # 图片2的embedding， 与 query_img_embed 差异较大
5. img3_embed = [0.63, 0.09, ..., -0.30]       # 图片3的embedding， 与 query_img_embed 相似

# 多模态融合表示（以加权平均为例）
6. query_fused = [0.45, -0.22, ..., 0.28]      # 文本0.4 + 图像0.6权重
```

多模态 – 三阶段示例

2. 检索阶段 - 向量数据库操作：

- 分别计算 query_fused 与所有知识库向量的余弦相似度；
- 返回Top3结果：

排名	内容类型	相似度	片段示意
1	图片3	0.93	圣马可大教堂（拜占庭风格照片）
2	文本1	0.82	"拜占庭建筑以穹顶和金色马赛克..."
3	图片2	0.31	哥特式教堂尖顶照片

- 检索结果的json表示：

```
retrieved_results = [
    {
        "image_path": "圣马可大教堂.jpg",
        "score": 0.93,
        "modality": "multimodal"
    },
    {
        "content": "拜占庭建筑特点包括...",
        "score": 0.82,
        "modality": "text"
    }
]
```

多模态 – 三阶段示例

3. 生成阶段

- 多模态LLM输入：

[USER_QUERY]
文本: "这张照片里的建筑是什么风格？"
图像: <拜占庭教堂照片>

[CONTEXT]
1. [图片] <圣马可大教堂照片>
2. [文本] 拜占庭建筑特点: "圆形穹顶、金色马赛克装饰..."

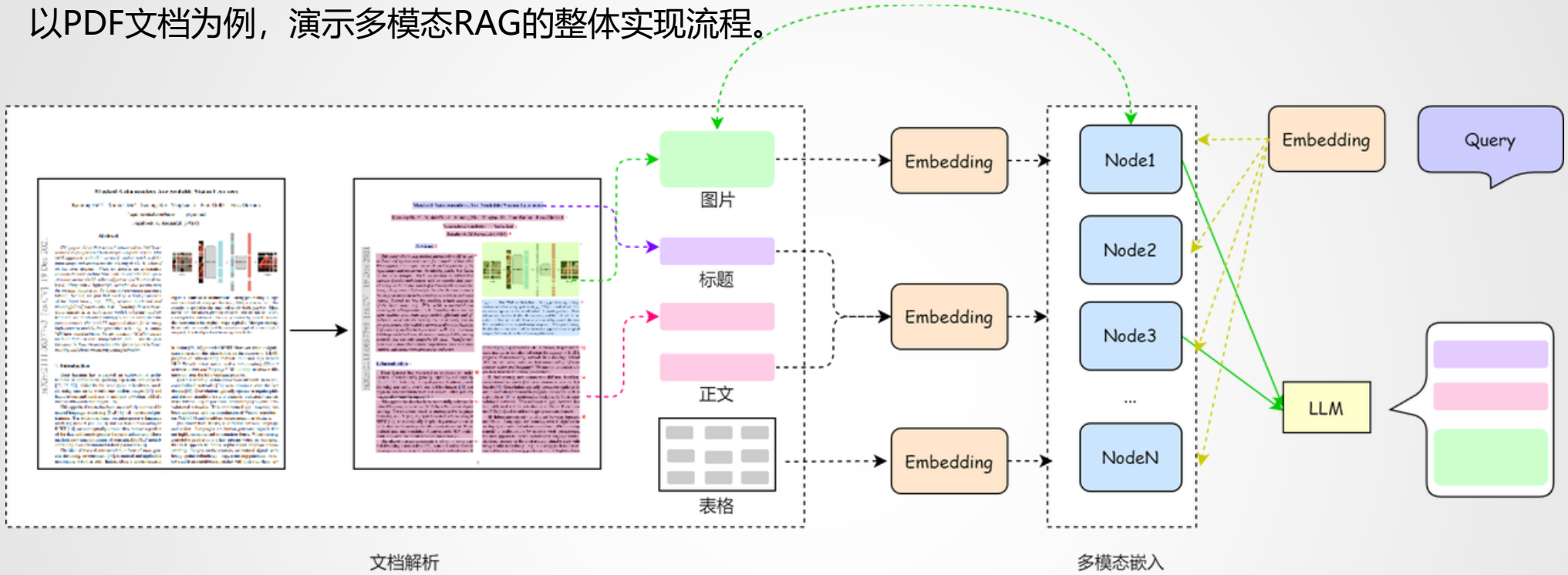
[OUTPUT]

- 生成结果：

> "您照片中的建筑具有典型的拜占庭风格特征：
> 1. 穹顶结构（如圣马可大教堂的中央穹顶）
> 2. 马赛克装饰（可见照片中金色反光部分）
> ...

基于 PDF 文档的多模态 RAG

- 以PDF文档为例，演示多模态RAG的整体实现流程。



为了对PDF文档进行完整的嵌入，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。上图展示了基于OCR文档解析的图文RAG系统流程，接下来我们将对各个步骤进行简要介绍，包括文档解析、多模态嵌入以及查询与生成三个关键步骤。

文档解析 – PDF 格式概述

PDF（Portable Document Format，便携式文档格式）是一种独立于应用程序、硬件和操作系统的文件格式，主要用于共享和查看文档。

- PDF 文档的主要特点包括：
  - 具有固定的布局和格式，在不同设备和操作系统上保持一致。
  - 可包含文本、图像、链接、按钮、表单字段、音频、视频及业务逻辑等内容。
  - 适用于文档存储、交换和打印，广泛应用于办公和学术领域。
- 根据来源不同，PDF 可分为以下两类：

类型	特点	解析方法
机器生成的 PDF	- 由专业软件生成（如 Adobe Acrobat、Word、Typora） - 包含可选中、可搜索、可编辑的内容 - 适合结构化处理	- 可直接使用解析工具如 `pdfminer`、`pdfplumber` - 支持提取文本、图像、表格等信息
扫描生成的 PDF	- 由扫描仪或拍摄图片生成 - 内容为位图，无法直接编辑或搜索 - 需借助 OCR 技术提取文字	- 使用 OCR 工具如 `Tesseract OCR`、`LayoutLM` - 可选云端 OCR 服务，适用于复杂版式、多语言场景

通过选择合适的解析方法，可以高效地从 PDF 文档中提取有价值的信息，为文档理解与处理提供支持。

文档解析 – PDF 解析与结构化处理

PDF 解析旨在从不可编辑文档中提取文本并保留结构，如标题、段落、表格等。因排版复杂，直接提取易丢失结构信息，影响语义理解和推理。因此，它通常被视为广义 OCR，包含文本识别（OCR）与布局分析两步：

- **OCR 提取文本：**OCR（光学字符识别）技术用于将纸质文档或扫描图片中的文字转换为可编辑、可搜索的电子文本。例如，PaddleOCR 和 EasyOCR 等开源工具能够高效识别 PDF 中不可编辑的文本，并将其存储为纯文本（TXT）格式。
- **布局分析：**布局分析的目标是恢复 PDF 的原始文本组织结构，识别各文本块之间的关系，并按照文档原有格式组织文本。例如，LayoutLM 模型能够联合建模文档中的文本、布局和图像信息，准确识别文本在页面中的位置及其排版结构。此外，对于包含表格、公式等复杂内容的 PDF，可结合专门的表格分析模型进行解析，以保留文档的完整信息。

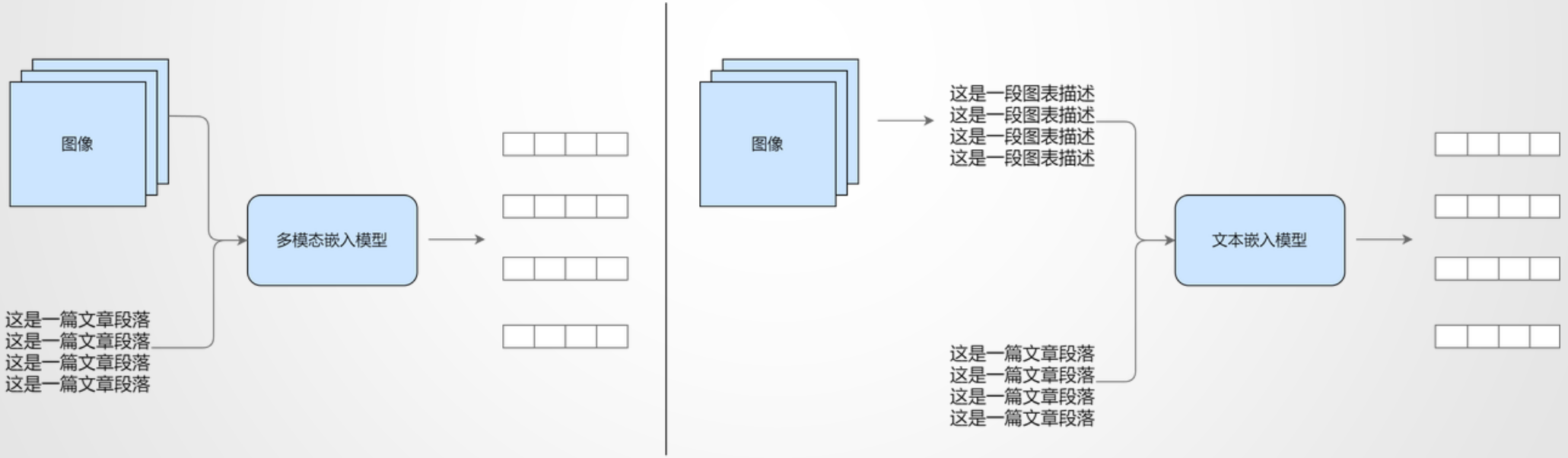
💡 MinerU：一个强大的 PDF 解析工具

它封装了文本提取、布局恢复、表格和公式分析等功能，并提供了便捷的 Python 接口。通过使用 MinerU，可以显著简化 PDF 解析流程，提高多模态信息处理的效率和准确性，为后续的文档理解、检索和生成任务提供更丰富的支持。

多模态嵌入

在完成文档解析后，最基础的方式是将提取出的文本和图像分别嵌入，但这会导致模态间（如文本与图像）缺乏语义关联，需分别检索并重排，或者需要训练一个跨模态融合模块，复杂且效率低。为提升系统性能，可以：

1. 使用多模态嵌入模型直接映射到同一空间
2. 离线阶段将多模态信息映射先转换成同一模态，然后映射到同一向量空间，使在线检索更高效，避免多模态独立处理，提升响应速度和任务表现。



左侧为利用多模态模型进行映射方法，右侧为统一数据模态后进行向量嵌入方法

- 上图展示了两种方法：直接统一嵌入或先模态转换再嵌入。

多模态嵌入

（1）多模态直接嵌入统一空间

- 可使用 CLIP、VisualBERT 等模型将文本与图像映射到同一向量空间，便于跨模态的语义对齐与统一检索，无需分开处理各模态，提升效率与准确性。

（2）模态转换后再嵌入

- 先将图像、表格等转为文本（如图像生成描述、表格转结构化文本），再用文本模型（如 BERT、T5）进行嵌入。此法可借助成熟文本模型处理多模态信息，但可能丢失原模态的细节和语义。

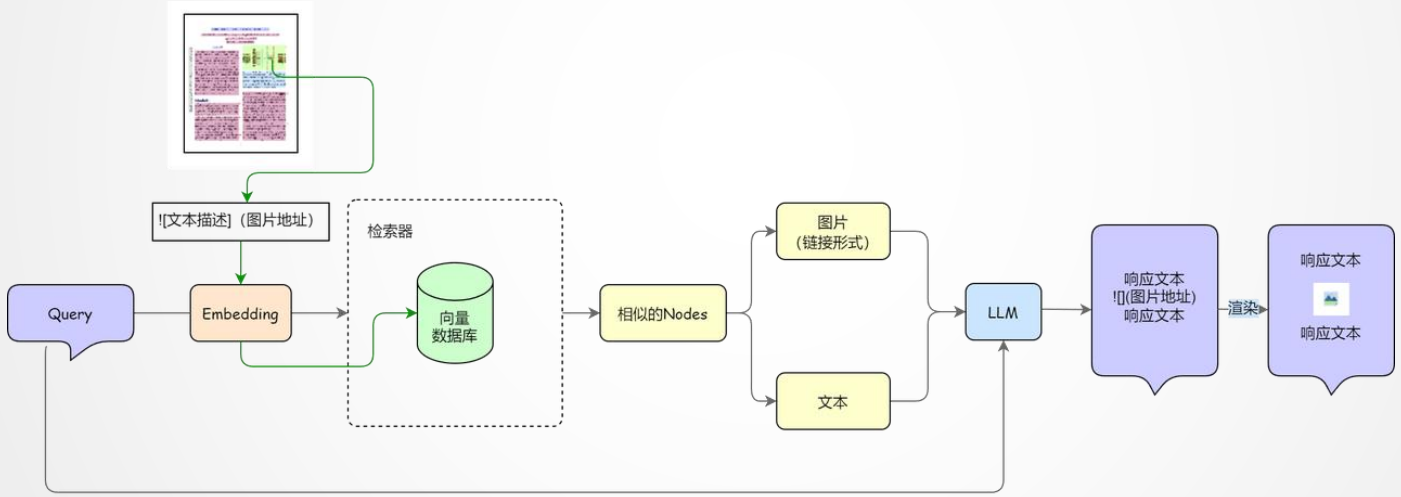
利用多模态模型解析图片：自动提取信息并生成QA对

除了上文中提到的两种图片解析方法，在解析图片时，我们还可以使用多模态模型如： InternVL-Chat-V1-5 来自动提取图片中的关键信息，并生成相应的问答（QA）对。这种方法的核心思路是：首先，多模态模型结合视觉与文本理解，分析图片内容，包括物体、文字、场景、结构等；然后，基于提取的信息，生成相关问题及答案，帮助用户快速理解图片的核心内容。例如，在解析一张论文中的示意图或实验结果图时，模型可以识别出图片的标题、数据趋势、关键结论等信息，并自动生成「该实验的主要结论是什么？」等问题及对应答案，从而提升论文内容的理解和信息获取的效率。

生成图文并茂的响应

实现图文并茂输出，可在 RAG 系统中加入以下步骤：

- 图表链接格式化：文档解析时提取图像及其描述，保存为 Markdown 格式，如 ![描述](图像链接)；
- 节点分类：检索阶段识别图像节点，为生成阶段提供图像提示；
- 提示词优化：为大模型提供指引，如“如有图像，请展示其链接并附解释”。



通过上述方法，RAG 系统可在生成答案时引入图像或图表，增强表达和可理解性。这是将原图纳入答案的一种简单方式，若需实现图表生成等更复杂功能，可关注后续 Agent 教学内容❗

目录

提高准确率

多模态内容向量化效果优化技巧

图像、表格等非文本模态的信息在编码过程中常会出现“**语义缺失**”或“**语义漂移**”等问题，导致向量表达与原始内容理解不一致，影响最终检索效果。

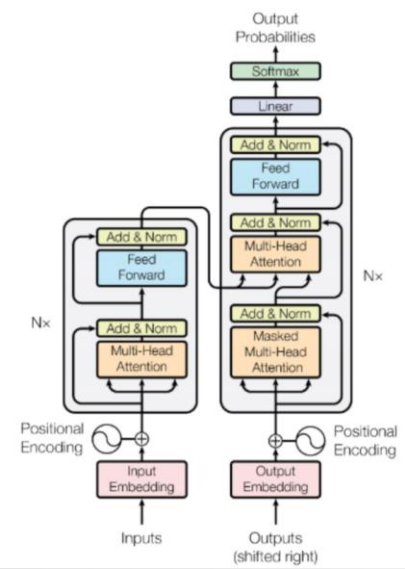
- 为提高多模态向量化的准确性与可用性，可以从以下几个方面入手优化：
  - **文本补全**：结合图片的标题与图注等文本信息
  - **结构化生成**：预先从多模态数据中提取 QA 对
  - **上下文增强**：将上下文一同编码
  - **微调多模态模型**：提升模型在特定多模态领域内的适应能力

提高准确率

1. 文本补全：结合图像标题与注解等文本信息

优化策略示例：

- 将生成的图片描述与图片注解等拼接后再做统一编码；
- 将图像本身与其标题、图注等文本拼接后作为联合输入，通过联合编码模型（如 CLIP）进行向量生成；



标题：　*“Figure 1: The Transformer - model architecture.”*

图注：　*“...illustrating the self-attention and feed-forward layers in both encoder and decoder blocks.”*

拼接后语义增强点：明确了**模型类型**（Transformer）、**关键算法**（self-attention / feed-forward）、**结构组件**（encoder / decoder blocks）等内容。

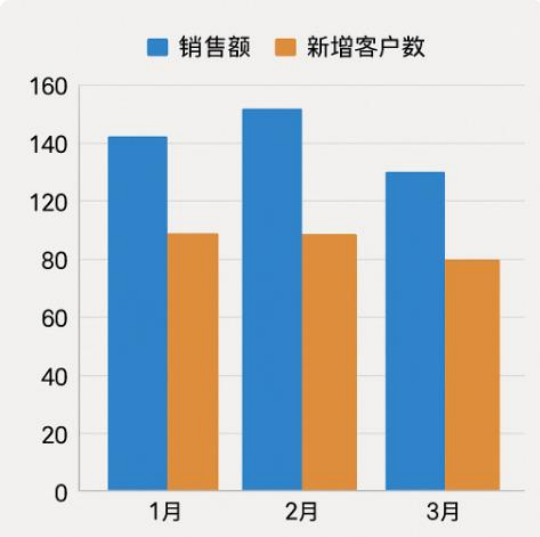
- 直接编码仅能捕捉 “框架图” 或 “神经网络类图” 的模糊语义

提高准确率

2. 结构化生成：预先从多模态数据中提取 QA 对

优化策略示例：

- 使用 OCR、图像理解模型或表格抽取模型对图片/表格进行解析；
- 结合 LLM 从解析结果中自动生成“**问题-答案对（QA对）**”或“摘要性文本”；
- 将这些结构化 QA 对作为知识增强材料与原始图像联合编码，或将这些 QA 对加入向量数据库。



❓: 2024年哪个月的销售额最高?

✅ : 2月，销售额为147万元。

❓: 哪个月新增客户最多?

✅ : 2月，新增客户数为76人。

❓: 3月相比2月的销售额下降了多少?

✅ : 下降了22万元。

❓: 一季度总销售额是多少?

✅ : 共404万元。

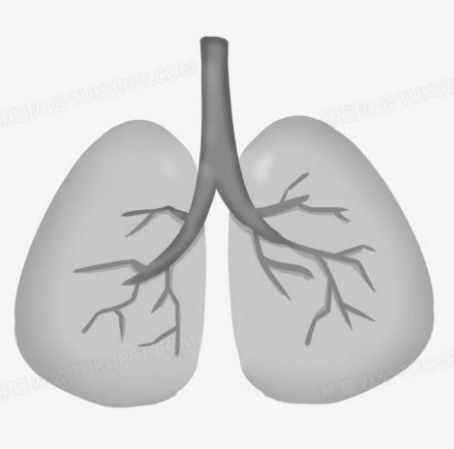
- 直接编码仅会得到“图表”、“数值”等粗糙标签。

提高准确率

3. 上下文增强：将上下文一同编码

优化策略示例：

- 在输入图像编码模型前，将其相关上下文文本一并拼接或融合输入，构建“图+文上下文窗口”；
- 使用多模态混合编码模型或统一模态后对其进行联合编码；



图文问答：肺片是否异常？是否有肺部疾病的迹象？

❌ 图文分开：

模型可感知肺部阴影，但缺乏上下文信息，无法确定其含义或严重程度。

✅ 图文联合输入后：

模型结合阴影特征与文本中的症状描述和诊断提示，能够更准确判断该影像可能为肺炎的初步表现，给出具备医学推理能力的回答。

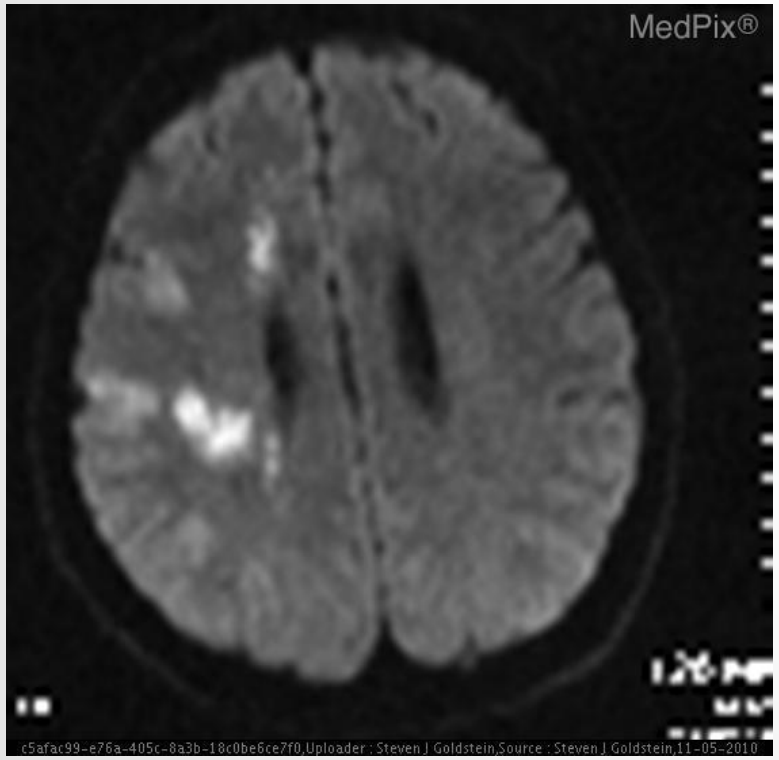
上下文信息：“患者男性，年龄42岁，抱怨轻微咳嗽和胸部不适。胸部X光检查显示右肺区域有轻微阴影，考虑可能是感染或肺炎的初步迹象。

提高准确率

4. 微调多模态模型：提升模型在特定多模态领域内的适应能力

4.1 示例：

- 医学图像领域，基于多模态模型实现简单的诊断。甚至就不需要额外上下文信息。



图文问答：“大脑的区域是否梗塞？”。

❌ 普通通用模型：

无法做出诊断。

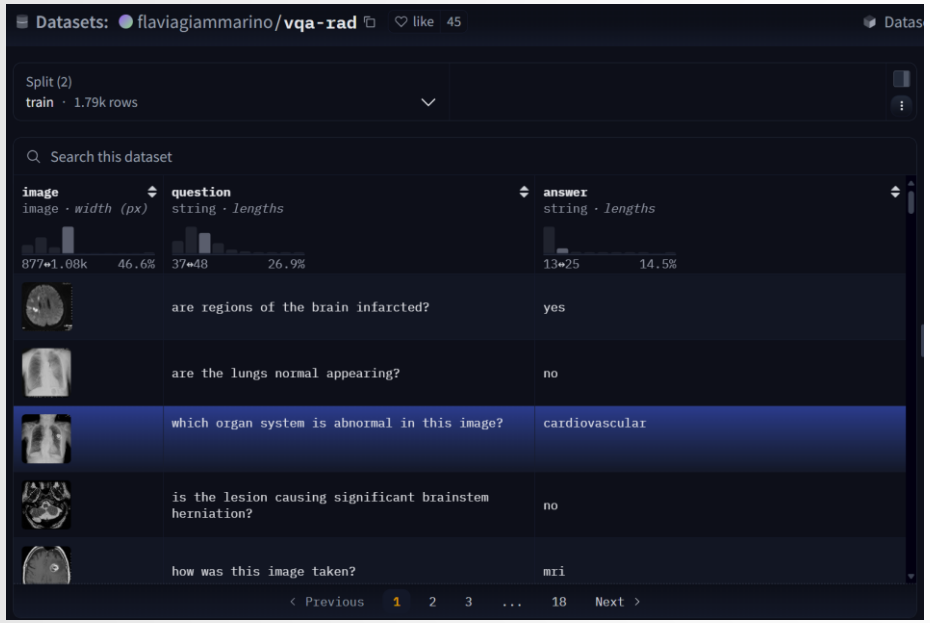
✅ 微调后的模型：

是的。**大脑的区域存在梗塞。**

提高准确率

4. 微调多模态模型：提升模型在特定多模态领域内的适应能力

4.2 数据简介：



VQA-RAD 是一个关于放射影像的问题-答案对数据集。

数据集用途

1. 训练和测试医学影像VQA（视觉问答）系统
2. 支持**开放式问题**（如“病灶位置？”）和**二元问题**（如“是否存在肿瘤？”）

数据来源

1. 基于**MedPix**（开放医学影像数据库）
2. 由临床医生**手动标注**，确保专业性

核心优势

1. 首个专注放射影像的VQA数据集
2. 结构清晰，覆盖临床常见问题类型

	训练集	测试集
问题	1,793	451
图像	313	203

提高准确率

4. 微调多模态模型：提升模型在特定多模态领域内的适应能力

4.3 数据处理：

（1）数据获取：

```
1. from datasets import load_dataset
2. dataset = load_dataset("flaviagiammarino/vqa-rad")
```

（2）处理前：

```
{
    "image":
<PIL.JpegImagePlugin.JpegImageFile image
mode=RGB size=566x555>,
    "question": 'are regions of the brain
infarcted?',
    "answer": 'yes'
}
```

（3）处理后（OpenAI格式）：

```
[
  {
    "messages": [
      {
        "content": "<image>are regions of the brain infarcted?",
        "role": "user"
      },
      {
        "content": "yes",
        "role": "assistant"
      }
    ],
    "images": [
      path/to/train_image_0.jpg"
    ]
  },
...
```

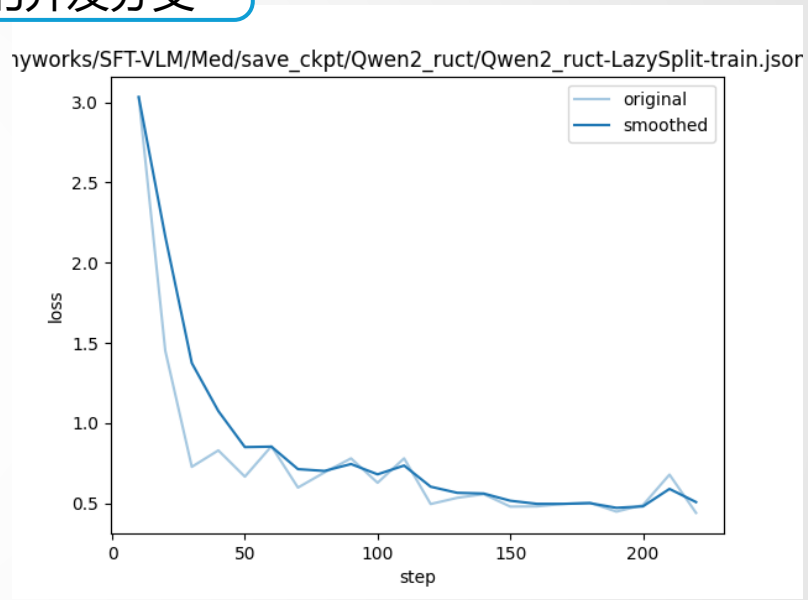
提高准确率

4. 微调多模态模型：提升模型在特定多模态领域内的适应能力

4.4 微调模型：

⚠ 需要将环境中的transformers和llama-factory升级到最新的开发分支

```
1.  import lazyllm
2.  model_path = 'path/to/Qwen2.5-VL-3B-Instruct'
3.  data_path = 'path/to/vqa_rad_processed/train.json'
4.  m = lazyllm.TrainableModule(model_path)\
5.      .mode('finetune')\
6.      .trainset(data_path)\
7.      .finetune_method(
8.          (lazyllm.finetune.llamafactory,{
9.              'learning_rate': 1e-4,
10.             'cutoff_len': 5120,
11.             'max_samples': 20000,
12.             'val_size': 0.01,
13.             'num_train_epochs': 2.0,
14.             'per_device_train_batch_size': 16,
15.         }))\
16. m.update()
```



提高准确率

4. 微调多模态模型：提升模型在特定多模态领域内的适应能力

4.5 微调评测：

Qwen2.5-VL-3B-Instruct	微调前	微调后
精确匹配率	0.0%	55.43%
语义相似度	31.85%	80.64%

共 451 题


```
    "query": "is the liver visible in the image?",
    "true": "no",
    "infer": "yes, the liver is visible in the image. it appears as a large, dark gray structure located in the upper left quadrant of the abdomen.",
    "exact_score": 0,
    "cosine_score": 0.3227266048281184
}
```

```
{
    "query": "is the liver visible in the image?",
    "true": "no",
    "infer": "no",
    "exact_score": 1,
    "cosine_score": 1.0
}
```

扩展学习：ColPali 中的多模态 RAG

不区分模态，直接整体处理是否可行?

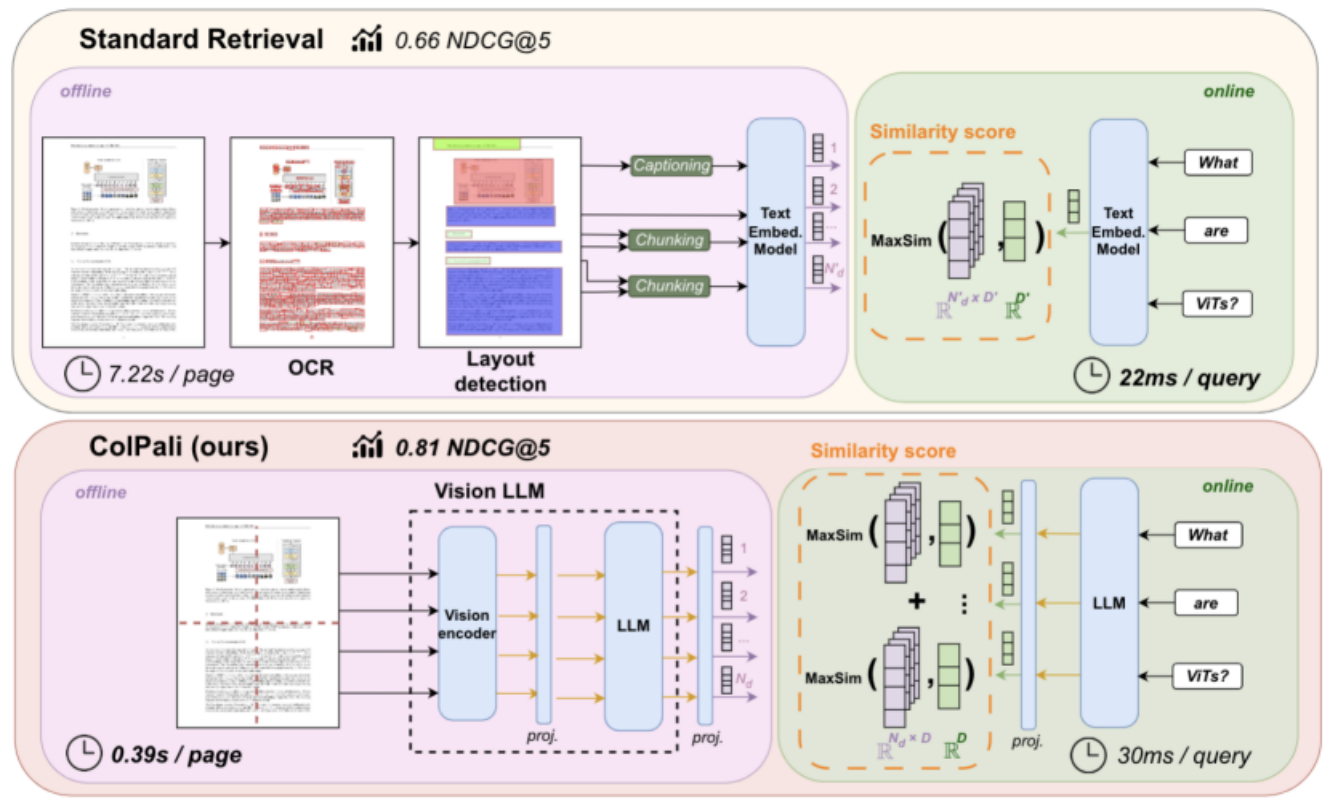
- 传统方法：将文档结构化解析 → 图像 & 文本分别处理
- 新方向：**将整页文档视为图像**，使用多模态模型直接编码

方法代表：ColPali

- **ColPali（Contextualized Late Interaction over PaliGemma）**使用**PaliGemma**（一个强大的多模态预训练模型）作为多模态编码器，对文档进行嵌入。
- 将某一页文档看成n个图像块，每个块都对应一个向量。
- 在计算相似度时利用嵌入列表，采用 "MaxSim "操作来计算两个文本之间的相似度。

$$S_{q,d} = \sum_{i \in |E_q|} \max_{j \in |E_d|} E_{q_i} \cdot E_{d_j}^T$$

扩展学习：ColPali 中的多模态 RAG



感谢聆听

Thanks for Listening