深入RAG：构造知识库

前言

这篇技术文章，我第一次用了“深入”这个词。此时的心情，类似于工作了十几年的资深技术人员，在简历里写了“精通某某技术”的感觉。学的越深，知道的越多，越不敢用这些修饰词。之所以，用上“深入”这个词。是觉得，在2025年当下，以及接下来的2~3年内，RAG对业界会有非常深远的影响，而一大堆聪明人不停的对这个技术进行迭代，使之焕发新春。对读者而言，希望务必搞懂RAG，打牢基础。

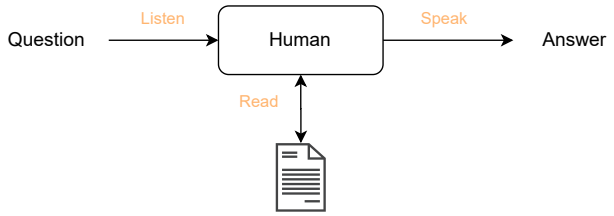
何为RAG

提出问题

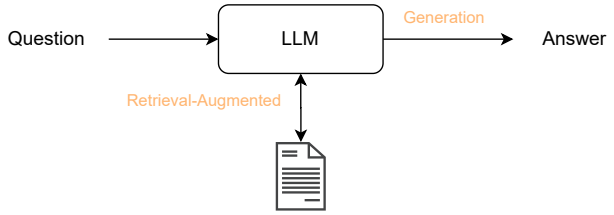
先抛开定义，让我们先看看在运用LLM解决实际问题时，遇到了哪些”痛点“。
以最常见的AI智能客服（AI Chatbot）为例，最初的问题是对人类自然语言的理解不够好，后来随着LLM技术的发展，特别是BERT的大规模应用，已经很好的解决了这个问题（即NLU问题）。接着，新的问题出现了，生成式LLM开始“一本正经的胡说八道”了，即产生了“幻觉”问题。如何解决这个“一本正经的胡说八道”问题呢？RAG技术的诞生，很大程度就是基于此问题背景的。

解决问题

我们回到人类的场景，很多正式场合，比如外交部答记者问现场直播，是不允许发言人说错话的。这个时候，外交部发言人的手头往往都会有一份书面的文档，上面预先准备好大量的问题和答案。发言人在回答记者问题时，需要先看一下手头的文件，然后回答记者的问题。用一个图来表示这个过程，如下图所示：



现在我们把这个过程模拟到LLM技术中，来解决幻觉问题，即用LLM来代替人脑。



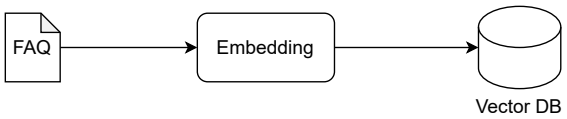
上图这个过程，其实这就是RAG技术。现在给出正式的RAG定义：

> **RAG(Retrieval-Augmented Generation):** 是一种结合信息检索（Retrieval）和生成（Generation）的技术架构，主要用于构建智能问答系统或知识增强型生成任务。它通过检索外部知识库中的相关信息来增强生成模型的能力，从而提高生成结果的准确性和知识覆盖范围。

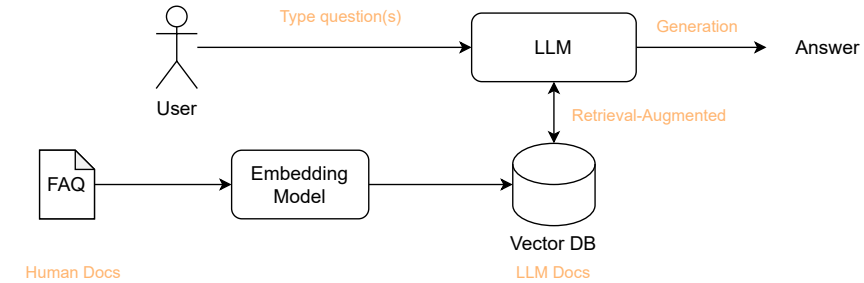
仔细观察上图，我们很容易得知RAG包含两个核心流程，一个是增强检索（Retrieval-Augmented），另一个是生成(Generation)。学习RAG技术，就是要把这两个核心流程搞清楚。工程领域的学习，最好的方式就是边学边做。现在，我们开始动手构造一个简单的RAG系统。

构造简单的RAG系统

在打造RAG系统之前，需要先解决一个问题——如何构造文档?
回到外交部发言人的例子，在正式开发布会前，需要花大量的精力准备记者们的提问，也就是准备常见问题的答案集（FAQ）。同样的，在计算机系统里，我们也需要准备让LLM能读得懂的FAQ文档。也就是帮LLM准备FAQ文档，这一过程，一般是人预先准备的，所以这个文档也是人类可读的。想让计算机理解人类的文档，就是必须要用数学来表示文档。这里，我们采用最常用的向量来表示文档，即将文档向量化（Word Embedding）技术。如果读者想了解其中缘由，建议阅读我之前的文章《词表达发展史》。而业界把处理好的文档储存在向量数据库中，下图描述了这个过程。



这样一个简单的RAG系统就构造好了，如下图所示。

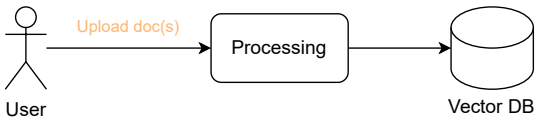


在真实世界中，运行的商业RAG系统要远远比这个复杂。接下来，我们就逐步改造这个简单的RAG系统，达到商业级别的架构。上面这个流程最大的问题就是需要对FAQ文档进行处理，每一次FAQ文档的更改，都需要对系统进行改动，耦合性非常严重。大多数RAG系统，对这一部分就进行了解耦。参照业界RAG系统的普遍做法，我们现在就对这个例子进行升级。

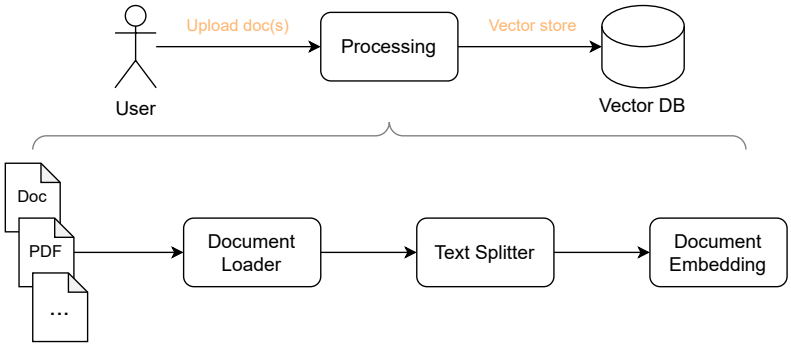
构造知识库系统

离线流程：知识入库

上文提到，在真实的RAG系统中，我们不会“hard code”用户的文档，而是让用户自己上传文档，也就是说，将大规模的知识库（Knowledge Base）存储到向量数据库中，这个过程叫做**知识入库**。下图描述了这一过程。



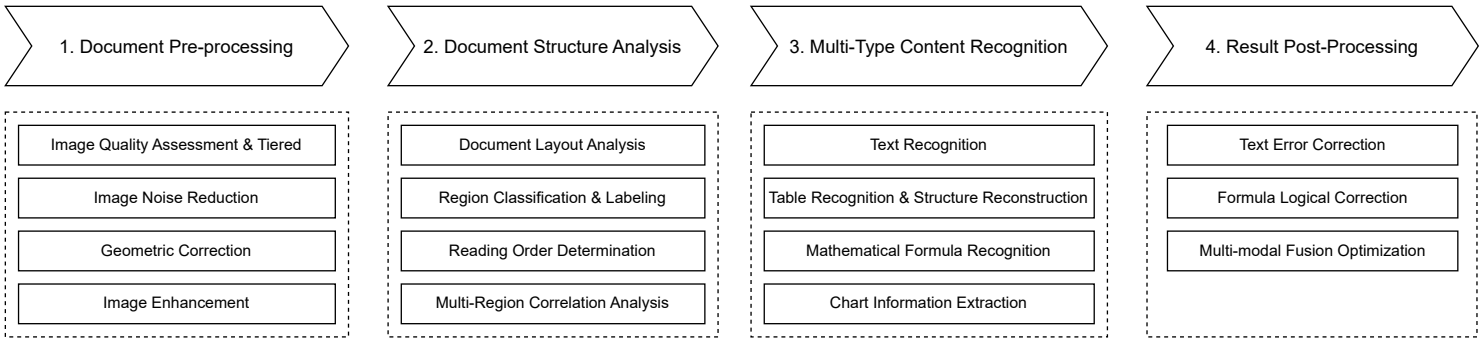
新的问题又来了，每个LLM都有一个最大Token的限制（2025年流行的LLM最大Token数一般是几千到几万。），而一篇小说，至少10万字，远远超过LLM最大Token数的限制。而文档又有各种格式，同时含有目录，表格，图片等，所以，我们需要先进行**Step 1. 文档解析**，再进行**Step 2. 文档切片**，然后将**Step 3. 文档切片向量化**，最后将切片的向量记录插入到向量数据库中。下图简单的阐述了这个过程。



接下来，我们将分别讨论文档解析，文档分片和文档向量化存储这三个过程。

Step 1. 文档解析

文档解析，顾名思义，就是将各种数据源及其格式（如txt，doc，pdf，json等格式）解析成统一格式。最简单的统一格式就是纯文本。根据我的实际工程经验，至少在2025年，绝大多数场景下，这项工作LLM没有技术优势。
因为本文着重介绍AI相关的技术，对传统的文件解析，就一笔带过了，建议有兴趣的读者看一下LangChain Document Loaders文档。
想要介绍AI模型如何在文档解析部分的运用，就要用到一些复杂的场景。这里，我们挑一个“硬骨头”例子来啃。一个含有文字+图片+表格/图表的PDF文档，这类文档常见于财报，商业研究和学术论文等。为了挑战难度，直接采用“扫描版”的PDF文件。
针对这一类文档，哪些单纯依赖规则的，传统的解析库是无法实现这一目标的，目前主持这些功能的多为基于深度学习的开源库等。处理这类文档，要经过下面四个步骤：**预处理，结构解析，多类型内容识别，文档后处理及优化。**下图是一个处理复杂文档的总览图：



接下来，我们对几个步骤做一个简要的介绍。

1. 文档预处理
   预处理的第一步是评估图像质量，根据质量等级采取不同的处理策略：

- 高质量图像（分辨率≥300DPI，清晰无噪声）：直接进入结构解析阶段
- 中等质量图像（200DPI<分辨率<300DPI，轻微噪声或倾斜）：进行基本的噪声消除和几何校正
- 低质量图像（分辨率<200DPI，严重噪声、模糊或倾斜）：进行超分辨率重建、去模糊、复杂噪声消除等处理

2. 文档结构解析
   文档布局分析(Document Layout Analysis) 是理解文档结构的关键步骤，现在流行的布局检测（Layout Detection）都是基于LLM的，比如微软的LayoutLMv3模型。一些流行的OCR集成开源软件，比如MinerU，还对LayoutLMv3做了微调，还用了传统的YOLO模型。将PDF文档各个区域，比如文本，表格，公式，图表，图像及注释等做分类和标记，以便于后续用不同的模型来对这些区域做OCR。这个过程就是**区域分类与标记(Region Classification & Labeling)**。
   确定文档区域的阅读顺序(Reading Order Determination) 是保证输出内容逻辑连贯的重要步骤，就是对离散的文本段落进行排序和拼接。最后还要使用LM从语义上分析各区域内容的上下文关系，确定关联强度，这一过程就是**多区域关联分析(Multi-Region Correlation Analysis)**。
3. 多类型内容识别
   内容识别是OCR系统的核心功能，需要针对不同语言和字体提供高精度识别。主要针对文字，表格和图表的识别。
   文字识别(Text Recognition)，技术很成熟了，就不多费口舌了。中文的话，PaddleOCR就不错。其他语言的话，可以考虑Tesseract。
   表格识别(Table Recognition)，当前技术比较有挑战性，需要识别表格格式和表格中的内容，并且将图像中的表格转化为HTML/Markdown/LaTeX等格式。目前有两种解决思路，一种是直接上一个大模型，一步到位输出转化好的格式。另一种是分解为2个模型，一个负责文字部分，一个负责表格识别和重建表格结构。在2025年，就实际效果来说，第二种思路好一点。但是，我觉得第一种思路是未来的方向。
   公式识别(Fomula Recognition)，技术成熟度介于两者之间，打印体的公式识别率已经很高了，当然手写的公式要差一点。其实公式OCR难点在公式检测和分割，建议训练一个专门YOLO去完成这部分工作。
   图表信息提取(Chart Information Extraction)，这部分技术，除了对标准的柱状图（Bar Chart），折线图（Line Chart）和饼图（Pie Chart）识别，马马虎虎能用。其他图表的识别准确率离产品化，还有点距离。2025年，市面上现有产品中，大多数把这些统计图，直接当作普通图片处理，不做内容识别和重建。
4. 文档后处理及优化
   在最终文档输出前，要做一些优化，比如文本纠错，表格结构优化，公式逻辑修正等。这部分比较杂，原理不难，有兴趣的读者，自己相应找一些材料看一下即可。也有不少软件把这部分修正(Correction)功能分散放在前面的步骤中。

Step 2. 文档分片

在RAG系统中，**文档分片（Document Chunking）** 是连接 “文档存储” 与 “高效检索” 的关键环节。其核心目标是将冗长、结构复杂的原始文档（如 PDF、长文、报告等）拆解为大小适中、语义完整的文本片段（Chunk），既保证检索时能精准匹配用户query的核心信息，又避免因片段过碎导致语义丢失。大致有5种方式分片策略：

1. 固定长度分片(Fixed-size Chunking)
2. 语义分片(Semantic Chunking)
3. 递归分片(Recursive Chunking)
4. 基于文档结构分片(Document Structure-based Chunking)
5. 基于LLM分片(LLM-based Chunking)

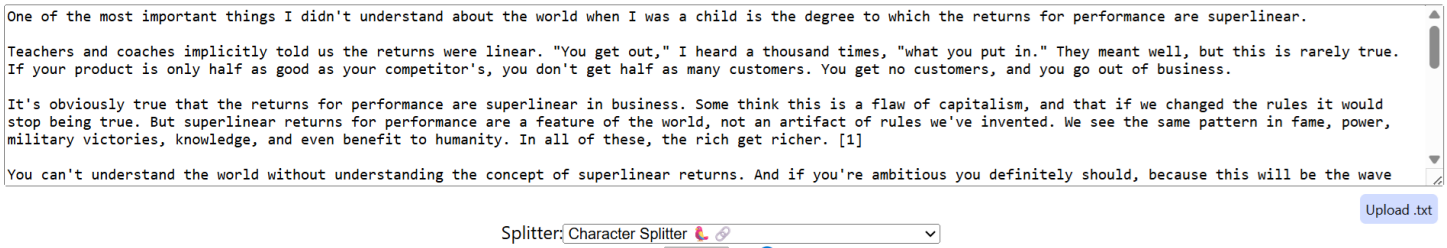
从实用性的角度来说，我介绍其中的两种，即固定长度分片和递归分片技术。

固定长度分片

固定长度分片技术(Fixed-size Chunking) 是最基础、最早期的文档拆分方法。它通过设定固定的长度阈值（如字符数或Token数），将原始文档切割为大小相对统一的片段（Chunk），是理解更复杂分片策略的基础。其中，有两个关键参数需要掌握：

1. **固定长度阈值（Chunk Size）**，核心参数，决定每个片段的最大长度。需匹配检索模型和大模型的输入限制。
2. **重叠长度（Chunk Overlap）**，可选参数，用于缓解 “切割点语义割裂” 问题，确保相邻片段在切割边界处有内容重叠，避免关键信息被 “一分为二”。通常设置为阈值的 10%-20%。

介绍个可视化工具ChunkViz，读者可以试一下这个工具，体会一下固定长度分片技术。



固定长度分片技术优点就是简单，高效。算法时间复杂度为O(0)。在非结构化短文本场景（如聊天记录等）很适用。但是，缺点也非常明显，最突出的就是语义割裂严重。对结构化文档（比如带标题的文档）也不够友好。现在，越来越多的RAG系统采用递归分片技术了，以达到更高的准确率。

递归分片

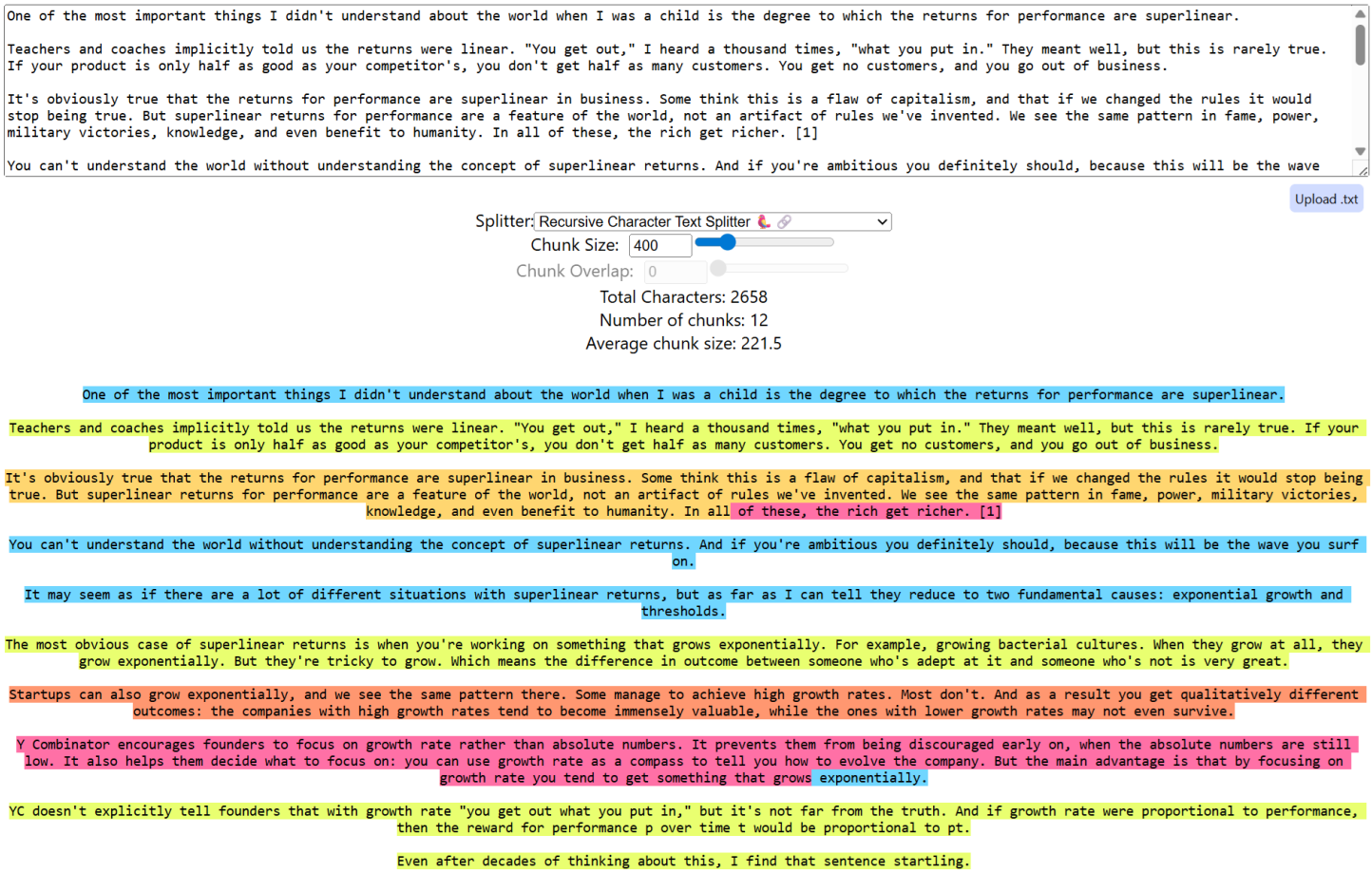
递归分片是一种**自上而下（Top-down）的分层拆分方法**：以完整文档为初始节点，按照预设的 “语义分隔符优先级” 逐层切割文本，若切割后的片段仍超过设定的最大长度阈值，则对该片段重复执行切割逻辑，直至所有最终片段均满足长度要求且语义相对完整。
简单来说，它模拟了人类阅读长文档的逻辑 —— 先看章节（大语义块），再看段落（中语义块），最后看句子（小语义块），不会粗暴地从某个句子中间截断。
其中，有几个关键参数需要掌握：

1. 最大片段长度阈值（Max Chunk Size）
2. 最小片段长度阈值（Min Chunk Size）
3. 重叠长度（Chunk Overlap）
4. 分隔符优先级列表（Separator List）

前三个参数和固定长度分片没区别，重点放在理解Separator List上。这个参数的核心作用是决定片段的 “语义边界贴合度”。具体的说，根据自然语言的语义层级，预先定义一组 “切割锚点”（分隔符），并按 “语义颗粒度从大到小” 排序。不同类型的文档可定制不同的分隔符列表，常见的通用优先级，从高到低，如下（以英文为例，中文可类比）：

- 文档级分隔符：分页符（`\f`）、章节标题分隔（如 `##`、`###`）
- 段落级分隔符：空行（`\n\n`）、段落标记
- 句子级分隔符：句点（`.`）、感叹号（`!`）、问号（`?`）
- 短语级分隔符：分号（`;`）、逗号（`,`）
- 字符级分隔符：空格（最后兜底，避免无法切割）

同样的，我们也在ChunkViz试一下递归切片技术。读者可以将下面这张图的结果和固定长度切片的结果对照一下，体会一下不同。



针对传统分片策略（如固定字符数分片、固定段落分片）存在明显局限，递归分片则针对性解决了这些问题，总结如下：

评估维度	传统固定长度分片	递归分片技术
语义完整性	易从句子、短语中间截断，导致片段语义破碎	优先按自然语义边界切割，最大程度保留语义
适配性	对不同结构（如论文、小说、报告）适配性差	通过动态分隔符适配多种文档结构
检索精准度	可能将相关信息拆入不同片段，导致漏检	相关信息集中在同一语义块，检索召回更精准
长文档处理能力	长文档中深层语义易被割裂	分层拆分，兼顾 “宏观结构” 与 “微观细节”

尽管递归分片是当前最优策略之一，但仍存在局限性，依赖分隔符质量，若文档格式不规范（如无空行、标题混乱），分隔符可能失效，导致切割效果下降。仅依赖 “表面分隔符”，无法识别 “隐性语义关联”（如跨段落的同一主题）。
综上所述，文档递归分片技术通过 “分层切割 + 语义优先” 的逻辑，完美平衡了RAG系统中 “片段长度合规性” 与 “语义完整性” 的核心矛盾，是处理长文档、复杂结构文档的 “标配技术”。其本质是将 “机械切割” 升级为 “智能适配”，通过定制分隔符与调优参数，可最大限度提升后续检索环节的召回精度，最终支撑大模型生成更准确、更具逻辑性的回答。在实际应用中，需结合具体文档场景（如学术、法律、企业报告）定制策略，并搭配结构解析、语义嵌入等技术进一步优化效果。

Step 3. 文档向量化存储

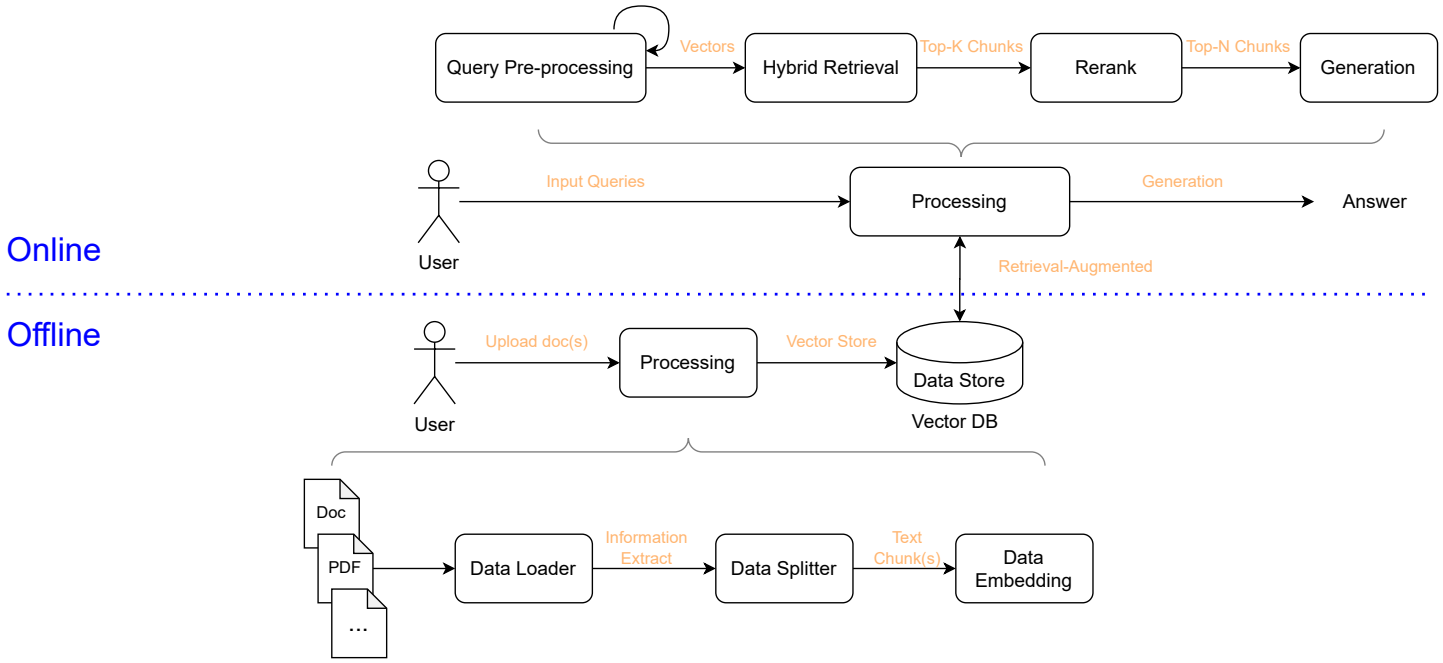
文档完成分片后，就需要转化成计算机能理解的数学语言。这里就要用到文本嵌入模型（Text Embedding Models）了，其核心思想是将文本（如单词、句子、段落或文档）转换为低维稠密向量（嵌入向量）的模型。这些向量能够捕获文本的语义信息，使得语义相似的文本在向量空间中具有较近的距离。业界有很多Text Embedding Model，就实际效果来说，在2025年，我推荐BGE（中文）和BCEmbedding（中英文）。在RAG系统里，一般会有把这些向量存在专门的**向量数据库(Vector DB)** 来管理。向量数据库的功能就是存储分片的向量表示，支持高效的语义检索，一般采用快速近邻搜索（ANN）算法，篇幅原因就不展开了。
Tips：如果读者想要了解Word Embedding背后的原理，建议读一下我的《词表达发展史》和《详谈Skip-gram》这两篇文章。

在线流程：检索增强问答

讲完前面的预备知识，现在我们回归到RAG系统的两个核心流程中来。
我在开篇文章《从生物大脑到数字大脑》中，提到阿甘的一句话——现代AI算法都是在模拟人脑，发现模拟的越来越像，效果越好。这个观点在一次一次的算法演进中被证明。在RAG系统中，也不例外。

先举个身边的例子，研发团队打算招聘一个开发工程师。在企业里的标准流程：
Step 1： 写一个招聘职位的要求（Job Description），给到HR
Step 2： HR将这个JD发到各个渠道，比如招聘网站，猎头等，来收集合适的候选人简历（Resume）
Step 3： HR团队**粗筛**这些简历，并把筛选出来的候选人简历给到研发招聘负责人
Step 4： 研发招聘负责人就**精准筛选**这些候选人，并根据简历安排合适候选人进行下一步的面试流程。

RAG系统的，检索增强问答过程和上述例子中的Step 3和Step 4非常像。和人类处理实际问题一样，一个优秀的RAG系统，也需要**两阶段检索**这个流程，而RAG系统的核心就在于检索层。其功能就是根据用户查询，从向量库中召回最相关的分片，核心是“精准+高效”。
在展开讨论这个两阶段检索前，先大致先复习一下整体流程，如下图所示。其中offline部分就是，我们在前面提到的入库流程；而online部分，就是RAG系统工作时的主要流程。



如图所示，一个商用级别的较完善的RAG系统要经过四个步骤，即**用户问题的预处理、混合检索、重排、答案生成**这四个阶段。

Query理解和重写

在**用户问题的预处理**阶段，主要的目的就是更好的理解用户的意图，并把用户的问题改写成LLM更加容易理解的Prompt。通常有**意图识别**、**问题拆解**、**复杂问题判定**、**多轮改写**等核心环节，具体含义补充如下：

- **意图识别**: 是判断用户查询的核心目的（如“信息检索”、“事实确认”、“逻辑推理”），为后续检索策略提供方向；做的好的RAG系统，会在这一步加入CoT，对用户模糊问题进行进一步的澄清(Clarify)，已达到最大限度的识别用户的实际意图。
- **问题拆解**: 针对长句或多意图查询，将其拆分为多个子问题（如“介绍BGE和BCEmbedding的区别”拆分为 “BGE 的特点”、“BCEmbedding 的特点”、“二者差异对比”），提升检索精准度；
- **复杂问题判定**: 则通过规则或模型识别查询是否涉及多步骤推理、跨文档关联等复杂场景，进而启用更深度的检索逻辑（如多轮检索、重排序增强）。
- **多轮改写**: 指结合对话历史，对用户当前查询进行多次优化（如补充上下文、修正表述），使其更适配检索需求；

混合检索

上文提到的两阶段检索，就是对应的**混合检索**和**重排**。
在实际项目中，一般先采用VectorDB的自带的语义检索，即通过向量相似度召回Top-K chunks。在准确度要求高的RAG系统里，还要加入关键字检索（如ElasticSearch的BM25算法），解决 “语义漂移” 问题。这种结合BM25关键字检索和基于向量相似度检索的方式，称之为**混合检索**。而这个阶段的输出，就是Top-K的文档切片（chunks）。

Tips:

1. Top-K和Top-P的区别，请参考我的《提示词工程》一文。

2. 早期RAG系统时只做基础的语义检索，基于我的工程实践经验，是推荐加入BM25算法的，工程代价很低，越来越多VectorDB都自带的。效果很不错，最终能提高1~2%的准确率。

混合检索，如果对应上面招聘的例子，就是Step 3：HR粗筛/初筛简历阶段。结合现实中的例子，HR给到业务部门的候选人简历，往往只是做了候选人简历（Resume）和招聘职位描述（Job Description）的大致匹配，无法做到非常精准的匹配。业务部门还需要对这些候选人做进一步的筛选和面试，来对候选人做修正和排序，确保把合适的（基于技能/薪水等）候选人排在最前列。
同样的，在RAG系统中，初轮检索（如向量检索的 “近似最近邻搜索”、BM25的“关键词匹配”）虽能快速从海量分片中召回一批相关候选，但存在天然局限性：

- **向量检索**：依赖单向量（如段落均值向量）表征语义，难以捕捉长文本的细节逻辑或多义词的语境差异，可能将 “形似神不似” 的片段召回。
- **BM25等关键词检索**：仅基于词频、逆文档频率等统计特征，无法理解 “语义关联”（如 “人工智能”与“AI” 的等价性），易遗漏无关键词但语义相关的片段。
- **效率优先的妥协**：初轮检索为了速度，通常采用 “近似算法”（如向量检索的HNSW索引），牺牲了部分匹配精度。
  那么，在RAG系统中，有没有类似于“面试”这类流程，对初轮召回的Top-K chunks进行深度语义分析，修正排序偏差，确保最相关的片段排在最前列？答案是，有的，就是重排(Rerank)。

重排

Rerank的本质，用一个专门的打分模型，是对“查询（Query）-文档分片（Chunk)”对进行精细化语义相关性打分。结合上面招聘的例子，就是业务部门仔细的把HR推荐过来的候选人，面试完毕后，给每个候选人打分，分数低于阈值的，不符合职位要求。符合的若干候选人，给出综合打分和排名。HR团队会根据排名从高到低，进行录取。

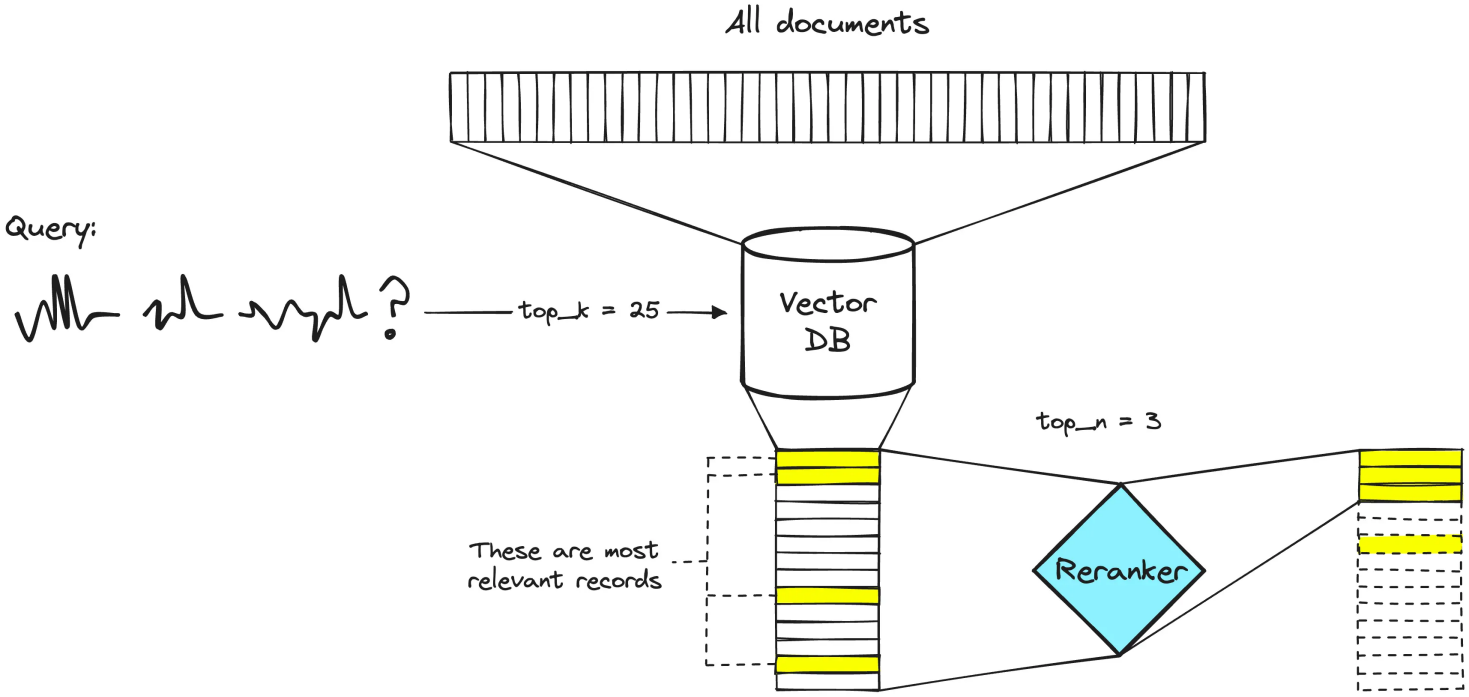
与初轮检索的**单向量独立编码**不同，Rerank模型采用**交叉编码（Cross-encoding）** 机制，其核心特点是 “将查询与片段拼接为一个整体输入，让模型在编码过程中实现二者的语义交互”，具体步骤：

- **输入拼接**：将用户查询（Query）与候选片段（Chunk）按固定格式拼接，例如：`"Query: {用户问题} Context: {候选片段内容}"`。
- **整体编码**：用预训练语言模型（如BERT）对拼接后的文本进行整体编码，捕捉查询与片段之间的细粒度关联（如“查询中的实体是否在片段中被详细解释”、“片段的逻辑是否能回答查询”）。
- **相关性打分**：取模型[CLS]位置的向量（或对所有Token向量做池化），通过一个全连接层输出标量分数，该分数直接表征 “查询与片段的语义相关程度”。

举个例子：
Query: “BGE 模型的核心优势是什么”
Chunk 1: “BGE 是 BAAI 开发的嵌入模型，支持多语言且语义匹配精度高”
Chunk 2: “Sentence-BERT 是基于 BERT 的句子嵌入模型”。

Rerank会通过整体分析，给Chunk 1打高分（强相关）；给Chunk 2打低分（弱相关）。

下面这张图很好的概括了RAG系统的核心流程（初筛→精排→输出）。对两阶段检索的过程，即初轮召回Top-K + 精排Top-N ” Chunks，也做了很好的描述。



Rerank通过交叉编码的深度语义交互，解决了初轮检索 “粗糙匹配” 的核心痛点，是提升RAG系统检索精度的关键一环。以牺牲少量计算效率为代价，换来Top-K结果相关性的显著提升，直接决定了后续大模型生成回答的准确性和可靠性。在我经历的实际的RAG项目中，Rerank并非 “可选模块”，而是商用RAG系统的标配。当然，在硬件资源受限的情况下，比如端侧LLM，Rerank可以舍弃。不过，我的建议是，哪怕在端侧LLM的场景，依旧推荐采用。

生成回答

在RAG系统中，**生成层（Generation Layer）** 是最终面向用户输出结果的核心模块。它接收检索层返回的相关文档片段（Context，通常为Top-N个chunks，N=3~10）和用户的原始查询（Query），通过大语言模型（LLM）生成符合语境、逻辑连贯且有据可依的回答。生成层的性能直接决定了用户对系统的体验，其核心目标是 “基于检索到的事实性信息，生成准确、易懂、有针对性的回答”。
生成层在RAG系统中扮演“信息整合者”和“自然语言生成器”的双重角色，其工作流程可分为5个关键步骤，形成“接收输入→上下文处理→提示词构建→LLM生成回答→输出后处理与返回”的完整链路。
Step 1. 接收输入
最核心输入是用户查询（Query）、检索层返回的相关文档片段（Context，通常为Top-N个分片，N=3~10）。一个完善的RAG系统的，还会加入很多辅助的输入，比如对话历史（多轮场景）、文档元数据（如来源、时间）、系统配置（如回答风格、长度限制）。
Step 2. 上下文预处理
对检索到的文档片段（Context）进行优化，确保其适合作为LLM的输入。比如，长度截断，去重合并和结构化整理。这里稍微解释一下长度截断和结构化整理。
长度截断的原因是LLM有最大输入限制（max token length），一般直接调整N的值，也有人用LLM先对超长文本做一次总结。
结构化整理，即为每个片段添加唯一标识（如 `[文档1，片段3]`），便于后续来源标注。一般常见于在线RAG，并和搜索功能结合在一起。
Step 3. 提示词构建
将查询、预处理后的上下文及规则约束整合为 LLM 可理解的提示词（Prompt），典型模板结构如下：

```
System：请基于以下提供的上下文回答用户问题，严格遵循以下规则：
1. 仅使用上下文信息，不编造内容；
2. 若上下文信息不足，明确告知无法回答；
3. 回答中需用[来源标识]标注信息来源；
4. 语言简洁明了，分点说明复杂内容。

Context：
[文档1，片段1]：...（具体内容）...
[文档2，片段2]：...（具体内容）...

Conversation History：
User：...（上一轮问题）...
Assistant：...（上一轮回答）...

Answer：{User Query} 回答：
```

Tips: 提示词工程能大幅度提高RAG系统的准确率，防止幻觉(Anti-Hallucination)。有兴趣的读者，可以参考我的另一篇文章《提示词工程》。

Step 4. LLM生成回答
将构建好的提示词输入大语言模型，模型基于以下逻辑生成回答：

理解用户查询的核心意图（如 “事实查询” “步骤指导” “对比分析”）。从上下文中提取与意图相关的关键信息（如数据、观点、逻辑关系）。按提示词中的规则（如格式、来源标注）组织语言，生成初步回答。

Step 5. 输出后处理与返回
对LLM生成的原始回答进行优化，提升用户体验：

- **格式美化**：将纯文本转化为结构化形式（如列表、标题、表格），重点内容加粗或高亮。
- **来源校验**：确保所有事实性陈述均有上下文支持，移除无依据的内容。
- **冗余去除**：删减重复表述或与查询无关的信息，精简回答长度。
- **多模态适配**：若系统支持多模态输出（如图片、图表），将文本中提及的数据转化为可视化内容。

生成层是RAG系统的“最后一公里”，其核心价值在于将检索到的碎片化信息转化为用户可直接理解的高质量回答。它的性能不仅依赖于大语言模型的能力，还取决于提示词工程、上下文管理和防幻觉策略的优化。
在实际落地中，生成层需根据业务场景（如客服问答、学术研究、企业知识库）调整生成策略。例如，客服场景需口语化且简洁，学术场景需严谨且带详细来源。通过持续优化模型选型、提示词设计和后处理逻辑，生成层可显著提升RAG系统的用户体验和可信度。

行文至此，RAG系统的整体介绍已经完毕。本可以潇洒的写一句，Enjoy it~~就可以结束本章内容。但总觉得缺了点什么。决定再加一节，算给本文加点料吧。

给RAG系统加点料

这部分，就写知识图谱吧。这是我们实际工作中，在把RAG准确度提到很高的数字（比如95%以上）时，最想知道的答案——要不要上知识图谱？

> 直接给出我的结论，在2025年这个时间点，在新上的RAG系统里，坚决不上知识图谱。
> Tips：在RAG系统上，要新建一个知识图谱，没有性价比可言。除非，你们的知识图谱是现成的，不用把这部分成本算到RAG系统里，或者你的RAG系统预算多的用不完。

既然提到知识图谱，先做个科普吧。

三元组知识图谱

人类的知识，需要表示成一种数学方式才能让计算机处理。在《词表达发展史》一文中，比较详细的阐述了，如何用向量来表示人类的知识。而用向量表示知识，即Word Embedding技术，是当前NLP问题的基础。我们之前聊到的NLP技术，无一例外都是基于向量的。而知识图谱是不一样的，这种技术是用图(Graph)，来表示知识的。
限于篇幅，只能讲最实用的三元组知识图谱。三元组是知识图谱最基础、最核心的表示单元，通过“实体-关系-实体”或“实体-属性-属性值”的形式，将现实世界的知识结构化存储。

三元组的本质是用三个元素描述一个具体的知识事实，其标准格式为**主谓宾(Subject, Predicate, Object)，简称SPO结构**。三个元素的角色分工明确：

元素	英文	核心作用	示例
主语（主体）	Subject	知识的描述对象，通常是实体	北京、苹果公司、《三体》
谓词（关系 / 属性）	Predicate	连接主语和宾语，描述两者的关联	是... 的首都、成立于、作者是
宾语（客体）	Object	主语的关联对象，可为实体或属性值	中国、1976 年、刘慈欣

根据宾语类型的不同，三元组主要分为两类：

1. **关系型三元组**：宾语是实体，用于描述两个实体间的关联。
   示例：(北京，是... 的首都，中国)、(《三体》, 作者是，刘慈欣)
2. **属性型三元组**：宾语是属性值，用于描述实体的具体特征。
   示例：(北京，面积，16410.54 平方公里)、(苹果公司，成立时间，1976年4月1日)

三元组之所以成为知识图谱的基础，核心在于其解决了 “知识结构化” 的关键问题，具体价值体现在三点：

1. **简化知识表示**：将复杂的自然语言知识（如 “北京是中国的首都，面积约 1.6 万平方公里”）拆解为标准化的 SPO 结构，机器可直接识别和处理。
2. **支持知识关联**：多个三元组可通过共享实体形成 “知识网络”。例如 (北京，是... 的首都，中国) 和 (中国，位于，亚洲) 共享 “中国”，可串联出 “北京→中国→亚洲” 的关联链。
3. **便于知识推理**：基于三元组的关联关系，可自动推导新知识。例如已知 (A, 父亲是，B) 和 (B, 父亲是，C)，可推理出 (A, 祖父是，C)。

三元组知识图谱的构建

三元组并非直接生成，需从原始数据中提取并验证，典型流程分为三步：

1. **数据采集**：获取非结构化（如新闻、文档）、半结构化（如表格、网页）或结构化（如数据库）数据。
2. **信息抽取**：从数据中提取SPO三元素，核心涉及三个子任务：
   - 实体抽取：识别文本中的主语、宾语（如从 “马云创立阿里巴巴” 中抽取出 “马云” “阿里巴巴”）。
   - 关系抽取：识别实体间的谓词（如从上述句子中抽取出 “创立”）。
   - 属性抽取：识别实体的属性及对应值（如从 “阿里巴巴成立于 1999 年” 中抽取出 “成立时间” 和 “1999 年”）。
3. **知识融合与验证**：消除冗余和冲突（如 “北京” 和 “北京市” 视为同一实体），通过规则或算法验证三元组的准确性（如排除 “北京是美国的首都” 这类错误事实）。

知识图谱与LLM的融合

知识图谱技术在LLM之前就存在很多年了。而当前，知识图谱与大语言模型的融合正成为当前最重要的发展趋势之一。这种融合主要体现在两个方向：

- 利用大语言模型的自然语言理解能力来增强知识图谱的构建和推理能力；
- 利用知识图谱的结构化知识来增强大语言模型的推理能力和可解释性。

我们来讨论一下第一个方向。

- 在知识图谱构建方面，大语言模型可以用于：实体识别和关系抽取，利用预训练语言模型的语义理解能力来提高信息抽取的准确率；知识补全，通过语言模型的推理能力来预测缺失的知识；本体学习，自动从文本中学习领域本体和概念层次关系。
- 在推理增强方面，知识图谱可以为大语言模型提供：结构化的背景知识，帮助模型理解复杂的概念和关系；推理框架，提供可解释的推理路径和逻辑结构；事实核查，验证模型生成内容的准确性，减少幻觉问题。

GraphRAG（Graph-based Retrieval-Augmented Generation）是这种融合的典型代表。它结合了知识图谱的结构化知识和检索增强生成技术，通过图结构来组织和检索相关知识，然后使用大语言模型进行内容生成。从论文数据来看，GraphRAG在多个任务上都取得了显著的性能提升，在众多RAG技术中，可以达到最高的准确率（SOTA）。

知识图谱的缺点

从实际工作经验来说，用LLM来执行实体识别和关系抽取，需要很大一笔开销和以天为单位时间代价。从技术的角度来讲，就是构建图的过程代价有点大。这也是制约知识图谱技术被普遍采用的最根本原因。
展望未来，随着人工智能技术的不断进步和应用需求的持续增长，三元组知识图谱，由衷希望其被更加轻量级的新技术所替代，为实现真正的智能系统提供关键支撑。

参考文献

[1] olmOCR: Unlocking Trillions of Tokens in PDFs with Vision Language Models
pdf/2502.18443
[2] Survey on Question Answering over Visually Rich Documents: Methods, Challenges, and Trends
[3] KITAB-Bench: A Comprehensive Multi-Domain Benchmark for Arabic OCR and Document Understanding
[4] 基于生成式人工智能的图档博资源的集中开发与应用创新
[5] DocParseNet: Advanced Semantic Segmentation and OCR Embeddings for Efficient Scanned Document Annotation
[6] Neural Natural Language Processing for Long Texts: A Survey on Classification and Summarization
[7] Optimizing Nepali PDF Extraction: A Comparative Study of Parser and OCR Technologies
[8] ICDAR 2023 Competition on Robust Layout Segmentation in Corporate Documents
[9] 人工智能技术在企业档案管理中的应用场景、模式创新与未来展望 Application Scenarios,Model Innovation,and Future Prospects of Artificial Intelligence Technology in Enterprise Archive Management
[10] PdfTable: A Unified Toolkit for Deep Learning-Based Table Extraction
[11] ChatGPT赋能档案知识服务:逻辑理路及应用场景 The Empowerment of ChatGPT to Archival Knowledge Services:Logical Reasoning and Application Scenari
[12] OCR Hinders RAG: Evaluating the Cascading Impact of OCR on Retrieval-Augmented Generation
[13] OCR行业2025年趋势-腾讯云开发者社区-腾讯云
[14] OCR 技术的未来:从文字识别到智能文档理解_语义_信息_推理
[15] Rerankers and Two-Stage Retrieval

[16] 知识图谱的表示与推理对自然语言处理中因果性语义逻辑的影响与启示研究
[17] 企业级LLM实战：为什么知识图谱与大语言模型需要融合
[18] 知识图谱与大模型结合实践指南
[19] # GraphRAG对自然语言处理中深层语义分析的革命性影响与未来启示
[20] LangChain Document Loaders
[21] ChunkViz