检索增强生成（RAG）体系：信息注入、向量管线与微调融合

2025 年 10 月 25 日

1　检索增强原理与信息注入机制

1.1　核心理念

检索增强生成（Retrieval-Augmented Generation, RAG）通过外部知识库弥补大模型的知识盲点和时效性问题。其基本流程如图?? 所示：用户查询首先编码为向量，经过向量搜索检索相关文档，再将检索到的证据与原始查询拼接或重写，最后输入到 LLM 生成带来源的回答。

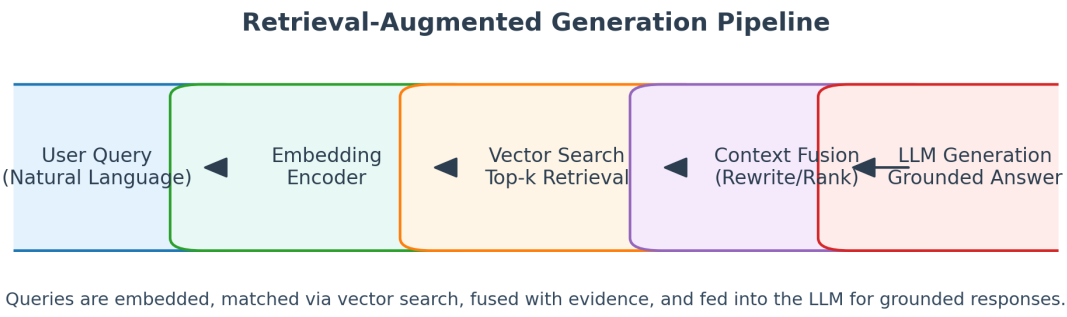

图 1: RAG 推理流水线：查询嵌入、向量检索、上下文融合、LLM 带引用回答。

1.2　检索与生成的闭环

- **双向依赖：**检索结果的质量直接决定生成上限，而生成阶段的指令（如引用要求、回答结构）反过来影响对证据的需求。
- **信息注入策略：**常见方法包括直接拼接（concatenation）、模板化摘要（structured prompt）、与查询融合（query rewriting）以及面向段落的排序权重（re-ranking）。

- **反馈机制**：通过生成结果对检索阶段提供反馈（如未找到答案时触发“虚无检索”或改写查询），形成自适应回路。

1.3 典型架构拓扑

类型	描述	应用场景
经典 RAG	查询 → 检索 → 拼接 → 生成	FAQ、知识库问答、客服机器人
双塔 + 重排	首先使用轻量向量检索，然后使用交叉编码器重排	法律、医疗等需要高精度匹配的场景
自适应检索	LLM 根据需要决定检索次数和查询改写	工具增强型 Agent、实时信息查询
文档先摘要	对大文档预生成摘要或结构化节点，查询时检索摘要再回溯全文	长文档阅读、报告生成

2 文档分块与向量嵌入（Embeddings）

2.1 分块策略

在向量化之前需要将文档切分为语义单元，图?? 展示了从原始文档到向量库的步骤。

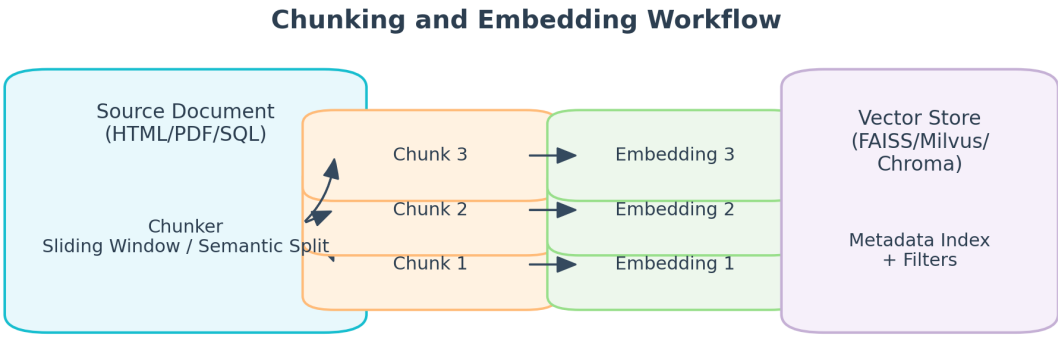

图 2: 分块与嵌入工作流：原始文档经滑动窗口/语义分段生成 Chunk，再编码为向量并存入向量库。

常用分块策略：

- **固定窗口 + 重叠**：适合结构化程度较低的文本，一般设置 200–512 token 的窗口，重叠 10–20% 防止断句。
- **语义分段**：基于句子、段落或语义相似度分段，采用 TextTiling、语义聚类、标题检测等算法。
- **结构化切分**：对表格、代码、知识图谱等，按字段或节点组织，保留元信息。

2.2 嵌入模型选择

- **通用模型**：如 OpenAI `text-embedding-3-large`、bge-large、m3e 等，在多语言场景表现稳定。
- **领域模型**：针对法律、医疗、金融的专用模型，通常由开源模型 LoRA 微调（如 BAAI/bge-law）。
- **稀疏表示**：结合 BM25、ColBERT 等稀疏/混合检索方式，补充关键词匹配能力。

2.3 嵌入与元数据

每个 chunk 向量通常关联以下元数据：

- 文档 ID、段落编号、标题、创建时间、权重、标签；
- 结构化字段（如产品类别、地理位置）用于过滤；
- 原始文本以及压缩摘要，用于模型阅读或展示。

合理设计元数据可以在检索阶段进行布尔过滤、时间筛选、权重排序，提高结果相关性。

2.4 批量编码示例

Listing 1: 使用 SentenceTransformers 批量编码文档 chunk

```python
from sentence_transformers import SentenceTransformer
from datasets import load_dataset

model = SentenceTransformer("BAAI/bge-large-zh-v1.5")
dataset = load_dataset("json", data_files="chunks.jsonl")["train"]

def encode_batch(batch):
    embeddings = model.encode(batch["text"], batch_size=64,
        show_progress_bar=False, normalize_embeddings=True)
    batch["embedding"] = [emb.tolist() for emb in embeddings]
```

```
    return batch

encoded = dataset.map(encode_batch, batched=True, batch_size=512)
encoded.to_parquet("chunks_with_embeddings.parquet")
```

3 向量数据库（FAISS, Milvus, Chroma）

3.1 核心功能对比

引擎	特点	适用场景
FAISS	Facebook AI 研发，支持 IVF、HNSW、PQ 等索引；内存型高性能	单机/内存充足，需自定义部署与调优的工程团队
Milvus	云原生架构，支持分布式存储、向量 + 标量过滤、CDC；提供 Milvus Lite	企业级多副本、高可用、与对象存储集成
Chroma	轻量级、嵌入式，支持 SQL API、持久化到 SQLite/Postgres	快速原型、桌面应用、少量数据场景

3.2 索引策略

- **暴力检索（Flat）：**精确但耗时，适合小规模或精确召回阶段（如重排前筛选）。
- **近似最近邻（ANN）：** IVF、HNSW、ScaNN 等方法，通过分桶或图结构实现 $O(\log n)$ 检索。
- **量化压缩：**产品量化（PQ）、OPQ、LSQ 等降低内存占用，适合海量向量。

在工程实践中常采用“双阶段”策略：先用 ANN 快速召回，再使用精确距离或交叉编码器重排。

3.3 Milvus 查询示例

Listing 2: 在 Milvus 中执行向量检索

```
from pymilvus import connections, Collection, utility
import numpy as np
```

```
3
4  connections.connect("default", uri="http://localhost:19530")
5
6  if not utility.has_collection("rag_chunks"):
7      raise RuntimeError("Collection rag_chunks does not exist")
8
9  collection = Collection("rag_chunks")
10 collection.load()
11
12 query_vector = np.load("query_embedding.npy")
13 search_params = {"metric_type": "IP", "params": {"nprobe": 32}}
14
15 results = collection.search(
16     data=[query_vector.tolist()],
17     anns_field="embedding",
18     param=search_params,
19     limit=5,
20     output_fields=["doc_id", "text", "score", "tags"]
21 )
22
23 for hit in results[0]:
24     print(hit.id, hit.distance, hit.entity.get("doc_id"), hit.entity.
         get("tags"))
```

4 RAG 与 Fine-tuning 的结合方式

4.1 互补关系

- **RAG 解决知识更新**：通过检索引入最新数据，降低微调频率与推理成本。
- **微调提升生成风格**：对 RAG 输出的格式、语气、推理深度进行定制化；
- **联合优化**：通过微调模型使其更善于阅读检索上下文、引用来源并区分多段证据。

4.2 常见组合方案

- **Instruction Tuning + RAG**：先对模型进行指令微调，使其遵守回答模板，再用 RAG 注入实时知识。
- **Retriever Fine-tuning**：使用对比学习（如 Contriever、ColBERT）对嵌入模型微调，提高召回质量。

- **Generator Fine-tuning：**采用 RAG 样本（问题、证据、答案）对 LLM 做监督微调或 DPO，提升引用准确性。
- **End-to-End RAG Fine-tuning：**在检索和生成之间引入可微分模块（如 RETRO、Atlas）实现联合训练。

4.3 训练数据构建

- **构造三元组：** (q, d^+, d^-) 对，用于训练嵌入模型；d^- 可来自随机抽样或难负样本挖掘。
- **证据标注：**对生成答案标记引用的文档和段落，为监督微调提供对齐信号。
- **自监督数据：**通过让模型对检索内容回答问题，再将模型自评得分用于强化学习或过滤低质量样本。

4.4 组合评估策略

指标类别	指标	关注点	工具
检索指标	Recall@k、MRR、nDCG	覆盖率与排序质量	BEIR、LlamaIndex Eval
生成指标	BLEU、ROUGE、Factuality	表达质量与真实性	GPT-judge、FactScore
引用指标	Precision@k（引用）、覆盖度	引用准确性、幻觉率	自定义正则表达式、Heuristic
端到端	用户反馈、工单转化率	业务 KPI	A/B Testing、在线实验

实践建议

- 设计结构化 Prompt，将检索到的证据标号呈现，指导模型引用具体段落。
- 定期刷新向量库与嵌入模型，针对新增数据执行增量索引与热度调节。
- 使用离线与在线评测联动，确保检索与生成的优化同步推进。
- 为 RAG + 微调流程配置监控，跟踪检索命中率、回答准确率与安全误报。

参考文献

- Lewis et al. “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.” NeurIPS, 2020.
- Izacard et al. “Few-shot Learning with RETRO.” arXiv, 2022.
- Humeau et al. “Poly-encoders: Architectures and Pre-training Strategies for Fast and Accurate Multi-sentence Scoring.” ICLR, 2020.
- Xu et al. “BGE: BE Better in Embedding.” arXiv, 2023.
- Chen et al. “Atlas: Few-shot Learning with Retrieval Augmented Language Models.” ICML, 2022.