# 大模型面试八股知识手册

RAG · Agent · 微调 · 推理部署 · 评估

| 作者 | Modular RAG 项目组 |
| --- | --- |
| 版本 | v1.0 — 2026 年 2 月 |
| 页数 | 30+ 页 |
| 用途 | QA 测试长文档 / Section O |

# 目录

# 第一章 Transformer 与注意力机制

## 1.1 自注意力机制原理

自注意力机制（Self-Attention）是 Transformer 的核心组件。它允许模型在处理每个 Token 时，能够关注输入序列中的所有其他 Token，从而捕获长距离依赖关系。计算过程可以描述为：首先将输入向量通过三个线性变换分别映射为 Query（Q）、Key（K）和 Value（V），然后计算 Q 与 K 的点积并进行缩放和 Softmax 归一化，最后与 V 加权求和得到输出。

## 1.2 多头注意力

多头注意力（Multi-Head Attention）将输入空间划分为 h 个子空间，在每个子空间独立计算注意力，最后拼接所有子空间的输出。这使模型能够同时关注不同位置的不同类型的信息。例如，一个 Head 可能关注语法关系，另一个 Head 关注语义相似性。GPT-4 等现代模型通常使用 32-128 个 Head。

## 1.3 位置编码

由于 Self-Attention 本身不包含位置信息，需要额外添加位置编码（Positional Encoding）。原始 Transformer 使用正弦余弦函数生成固定位置编码；后续研究提出了旋转位置编码（RoPE）和 ALiBi 等方案，在支持更长上下文的同时保持良好的泛化能力。RoPE 通过旋转向量的方式将相对位置信息编码到注意力计算中，被 LLaMA、GPT-NeoX 等模型广泛采用。

## 1.4 KV Cache 与推理优化

在自回归生成过程中，每一步都需要计算当前 Token 对所有之前 Token 的注意力。KV Cache 技术将已计算的 Key 和 Value 向量缓存起来，避免重复计算，将推理的时间复杂度从 $O(n^2)$ 降低到 $O(n)$。但 KV Cache 的显存占用随序列长度线性增长，成为处理长上下文的主要瓶颈之一。常见优化方案包括 GQA（Grouped Query Attention）、MQA（Multi-Query Attention）等。

## 本章小结

以上介绍了 Transformer 与注意力机制的核心概念和关键技术要点。理解这些知识对于构建高质量的 RAG 系统至关重要。在实际面试中，面试官往往会从基础概念出发，逐步深入到实现细节和工程经验。建议读者在阅读本章内容的基础上，结合项目代码进行实践验证，加深对各项技术的理解。同时，注意关注该领域的最新进展，因为大模型技术发展迅速，新的方法和工具不断涌现。

在准备面试时，建议将本章的知识点与实际项目经验相结合。不仅要能够清晰地解释技术原理，还要能够说明在实际项目中如何选型、如何调优、遇到了哪些问题以及如何解决。这种理论与实践结合的回答方式，能够给面试官留下深刻印象。此外，建议准备一些具体的数据和指标来支撑你的技术决策，例如选择某个 Embedding 模型后检索精度提升了多少、引入 Reranker 后 Top-3 命中率从 X 提升到 Y 等。

## 延伸思考

学习技术知识不能仅停留在表面，需要深入思考每项技术背后的设计动机和取舍。例如，为什么 RAG 系统要采用两阶段检索（粗排 + 精排）？这是因为精排模型（如 Cross-Encoder）虽然精度高，但计算成本大，无法对全库进行打分。通过粗排快速筛选候选集，再用精排做精细评估，既保证了精度又控制了延迟。这种'漏斗式'架构思想在搜索引擎、推荐系统等领域都有广泛应用。面试中如果能展示出这种跨领域的技术视野和工程直觉，会非常加分。同时，也要关注各种技术的局限性和适用边界。没有银弹，每种方案都有其最佳应用场景。能够根据具体需求选择合适的技术方案，是高级工程师的核心能力。

# 第二章 预训练与微调技术

## 2.1 预训练范式

大语言模型的预训练通常采用自回归语言建模目标（Causal Language Modeling），即预测下一个 Token。训练数据来自互联网大规模文本语料（如 Common Crawl、Wikipedia、GitHub 代码等），Token 数量通常在万亿级别。预训练阶段需要数千张 GPU 运行数周甚至数月。

## 2.2 SFT 监督微调

预训练完成后，模型具备了基础的语言理解和生成能力，但无法很好地遵循人类指令。监督微调（Supervised Fine-Tuning, SFT）使用高质量的指令-回答对数据集对模型进行二次训练，使其学会按照指令格式回答问题。SFT 数据质量远比数量重要。

## 2.3 RLHF 与 DPO

人类反馈强化学习（RLHF）通过训练奖励模型（Reward Model）来评估模型输出的质量，然后使用 PPO 算法优化模型策略使其获得更高的奖励。DPO（Direct Preference Optimization）则省去了奖励模型的训练，直接从人类偏好数据中学习策略，简化了训练流程。

## 2.4 LoRA 与 QLoRA

全参数微调需要巨大的计算资源，LoRA（Low-Rank Adaptation）通过在原始权重矩阵旁添加低秩分解矩阵（A·B，rank << d），仅训练这些新增参数，大幅降低显存和计算需求。QLoRA 进一步将模型权重量化为 4-bit，配合 LoRA 在消费级 GPU 上也能微调大模型。

## 本章小结

以上介绍了 预训练与微调技术的核心概念和关键技术要点。理解这些知识对于构建高质量的 RAG 系统至关重要。在实际面试中，面试官往往会从基础概念出发，逐步深入到实现细节和工程经验。建议读者在阅读本章内容的基础上，结合项目代码进行实践验证，加深对各项技术的理解。同时，注意关注该领域的最新进展，因为大模型技术发展迅速，新的方法和工具不断涌现。

在准备面试时，建议将本章的知识点与实际项目经验相结合。不仅要能够清晰地解释技术原理，还要能够说明在实际项目中如何选型、如何调优、遇到了哪些问题以及如何解决。这种理论与实践结合的回答方式，能够给面试官留下深刻印象。此外，建议准备一些具体的数据和指标来支撑你的技术决策，例如选择某个 Embedding 模型后检索精度提升了多少、引入 Reranker 后 Top-3 命中率从 X 提升到 Y 等。

## 延伸思考

学习技术知识不能仅停留在表面，需要深入思考每项技术背后的设计动机和取舍。例如，为什么 RAG 系统要采用两阶段检索（粗排 + 精排）？这是因为精排模型（如 Cross-Encoder）虽然精度高，但计算成本大，无法对全库进行打分。通过粗排快速筛选候选集，再用精排做精细评估，既保证了精度又控制了延迟。这种'漏斗式'架构思想在搜索引擎、推荐系统等领域都有广泛应用。面试中如果能展示出这种跨领域的技术视野和工程直觉，会非常加分。同时，也要关注各种技术的局限性和适用边界。没有银弹，每种方案都有其最佳应用场景。能够根据具体需求选择合适的技术方案，是高级工程师的核心能力。

# 第三章 Prompt Engineering

## 3.1 Prompt 设计原则

有效的 Prompt 应当清晰、具体，包含足够的上下文信息。常见的设计原则包括：提供明确的任务说明（你是一个 XX 专家）、给出输出格式要求（请用 JSON 格式输出）、提供示例（Few-shot）、以及步骤化引导（先分析…再总结…最后给出建议）。

## 3.2 思维链 (Chain-of-Thought)

思维链（CoT）提示通过要求模型「逐步思考」来提升推理质量。研究表明，在 Prompt 中加入「Let's think step by step」等引导语，可以显著提升数学推理、逻辑推理等任务的准确率。变种包括 Self-Consistency（多次采样取多数投票）和 Tree-of-Thought（树状分支探索）。

## 3.3 RAG 场景的 Prompt

在 RAG 系统中，Prompt 需要将检索到的上下文信息与用户查询结合。典型模板为：

'你是一个知识助手。根据以下参考资料回答用户的问题。如果资料中没有相关信息，请如实告知无法回答。

参考资料：{context}

用户问题：{query}'

需要注意上下文的排列顺序、长度控制和格式标注，以提升 LLM 的答案质量。

## 3.4 Prompt 注入与安全

Prompt 注入是指恶意用户通过精心构造的输入来绕过系统 Prompt 的约束，使 LLM 执行非预期的行为。防御措施包括：输入过滤与 sanitization、系统 Prompt 与用户输入的明确分隔、输出格式约束检验、以及使用 Constitutional AI 等方法训练模型的拒绝能力。

## 本章小结

以上介绍了 Prompt Engineering的核心概念和关键技术要点。理解这些知识对于构建高质量的 RAG 系统至关重要。在实际面试中，面试官往往会从基础概念出发，逐步深入到实现细节和工程经验。建议读者在阅读本章内容的基础上，结合项目代码进行实践验证，加深对各项技术的理解。同时，注意关注该领域的最新进展，因为大模型技术发展迅速，新的方法和工具不断涌现。

在准备面试时，建议将本章的知识点与实际项目经验相结合。不仅要能够清晰地解释技术原理，还要能够说明在实际项目中如何选型、如何调优、遇到了哪些问题以及如何解决。这种理论与实践结合的回答方式，能够给面试官留下深刻印象。此外，建议准备一些具体的数据和指标来支撑你的技

术决策，例如选择某个 Embedding 模型后检索精度提升了多少、引入 Reranker 后 Top-3 命中率从 X 提升到 Y 等。

## 延伸思考

学习技术知识不能仅停留在表面，需要深入思考每项技术背后的设计动机和取舍。例如，为什么 RAG 系统要采用两阶段检索（粗排 + 精排）？这是因为精排模型（如 Cross-Encoder）虽然精度高，但计算成本大，无法对全库进行打分。通过粗排快速筛选候选集，再用精排做精细评估，既保证了精度又控制了延迟。这种'漏斗式'架构思想在搜索引擎、推荐系统等领域都有广泛应用。面试中如果能展示出这种跨领域的技术视野和工程直觉，会非常加分。同时，也要关注各种技术的局限性和适用边界。没有银弹，每种方案都有其最佳应用场景。能够根据具体需求选择合适的技术方案，是高级工程师的核心能力。

# 第四章 检索增强生成（RAG）

## 4.1 RAG 概述

RAG 通过在 LLM 生成前引入信息检索阶段，将相关外部知识注入到生成上下文中，从而减少幻觉、提供实时知识更新、并使回答可溯源。RAG 已成为企业级 AI 应用的标准架构模式，广泛应用于知识库问答、文档分析、客服对话等场景。

## 4.2 RAG 链路分解

完整的 RAG 链路包含以下核心阶段：

1. 数据摄取（Ingestion）：解析文档 → 分块 → 增强 → 向量化 → 存储

2. 查询处理（Query Processing）：查询改写 → 关键词提取

3. 检索（Retrieval）：稠密检索 + 稀疏检索 → 融合

4. 重排序（Reranking）：Cross-Encoder 或 LLM 精排

5. 生成（Generation）：Prompt 构建 → LLM 生成 → 引用标注

## 4.3 RAG 常见问题

常见的 RAG 失败模式包括：检索不到相关文档（召回率低）、检索到的文档不精确（精确率低）、LLM 未能正确利用上下文（忠实度低）、以及分块不合理导致关键信息被截断。针对这些问题，可以从分块策略、检索方法、重排序和 Prompt 设计等多个维度进行优化。

## 4.4 面试高频问题

- Q: RAG 和微调的区别？各自适用场景？

- Q: 如何评估 RAG 系统的质量？有哪些评估指标？

- Q: RAG 系统中如何处理多跳推理问题？

- Q: 如何解决 RAG 中的幻觉问题？

- Q: 混合检索相比纯向量检索有什么优势？

## 本章小结

以上介绍了 检索增强生成（RAG）的核心概念和关键技术要点。理解这些知识对于构建高质量的 RAG 系统至关重要。在实际面试中，面试官往往会从基础概念出发，逐步深入到实现细节和工程经验。建议读者在阅读本章内容的基础上，结合项目代码进行实践验证，加深对各项技术的理解。同时，注意关注该领域的最新进展，因为大模型技术发展迅速，新的方法和工具不断涌现。

在准备面试时，建议将本章的知识点与实际项目经验相结合。不仅要能够清晰地解释技术原理，还要能够说明在实际项目中如何选型、如何调优、遇到了哪些问题以及如何解决。这种理论与实践结合的回答方式，能够给面试官留下深刻印象。此外，建议准备一些具体的数据和指标来支撑你的技术决策，例如选择某个 Embedding 模型后检索精度提升了多少、引入 Reranker 后 Top-3 命中率从 X 提升到 Y 等。

## 延伸思考

学习技术知识不能仅停留在表面，需要深入思考每项技术背后的设计动机和取舍。例如，为什么 RAG 系统要采用两阶段检索（粗排 + 精排）？这是因为精排模型（如 Cross-Encoder）虽然精度高，但计算成本大，无法对全库进行打分。通过粗排快速筛选候选集，再用精排做精细评估，既保证了精度又控制了延迟。这种'漏斗式'架构思想在搜索引擎、推荐系统等领域都有广泛应用。面试中如果能展示出这种跨领域的技术视野和工程直觉，会非常加分。同时，也要关注各种技术的局限性和适用边界。没有银弹，每种方案都有其最佳应用场景。能够根据具体需求选择合适的技术方案，是高级工程师的核心能力。

# 第五章 向量数据库与 Embedding

## 5.1 Embedding 模型选型

选择 Embedding 模型时需要综合考虑以下因素：向量维度（影响存储和计算成本）、推理速度（影响摄取和查询延迟）、语义质量（尤其是中文和多语言能力）、以及部署方式（云端 API vs 本地推理）。对于中文场景，BGE-large-zh 和 GTE-large-zh 在 MTEB 中文榜单上表现最佳。

## 5.2 向量数据库对比

主流向量数据库包括 ChromaDB（轻量级、适合快速原型）、FAISS（Facebook 开源、高性能 ANN 搜索）、Milvus（分布式、适合大规模生产环境）、Pinecone（全托管云服务）、和 Weaviate（支持混合搜索和 GraphQL）。本项目选择 ChromaDB，因其简单易用、支持持久化、且与 Python 生态集成良好。

## 5.3 ANN 索引算法

近似最近邻（ANN）算法是向量检索的核心。常见方案包括：

- HNSW（Hierarchical Navigable Small World）：多层图结构，查询快但构建慢

- IVF（Inverted File Index）：倒排索引 + 聚类，适合大规模数据

- PQ（Product Quantization）：向量压缩，大幅降低存储

ChromaDB 默认使用 HNSW 算法，对于本项目的数据规模是最优选择。

## 5.4 Embedding 优化技巧

提升 Embedding 质量的技巧：

1. 在 Chunk 前添加 Instruction Prefix（如「为检索优化：」）

2. 使用 Matryoshka Embedding 按需截断维度

3. 对查询做 HyDE（Hypothetical Document Embedding）扩展

4. 使用领域微调的 Embedding 模型提升专业领域效果

## 本章小结

以上介绍了向量数据库与Embedding的核心概念和关键技术要点。理解这些知识对于构建高质量的 RAG 系统至关重要。在实际面试中，面试官往往会从基础概念出发，逐步深入到实现细节和工程经验。建议读者在阅读本章内容的基础上，结合项目代码进行实践验证，加深对各项技术的理解。同时，注意关注该领域的最新进展，因为大模型技术发展迅速，新的方法和工具不断涌现。

在准备面试时，建议将本章的知识点与实际项目经验相结合。不仅要能够清晰地解释技术原理，还要能够说明在实际项目中如何选型、如何调优、遇到了哪些问题以及如何解决。这种理论与实践结合的回答方式，能够给面试官留下深刻印象。此外，建议准备一些具体的数据和指标来支撑你的技术决策，例如选择某个 Embedding 模型后检索精度提升了多少、引入 Reranker 后 Top-3 命中率从 X 提升到 Y 等。

## 延伸思考

学习技术知识不能仅停留在表面，需要深入思考每项技术背后的设计动机和取舍。例如，为什么 RAG 系统要采用两阶段检索（粗排 + 精排）？这是因为精排模型（如 Cross-Encoder）虽然精度高，但计算成本大，无法对全库进行打分。通过粗排快速筛选候选集，再用精排做精细评估，既保证了精度又控制了延迟。这种'漏斗式'架构思想在搜索引擎、推荐系统等领域都有广泛应用。面试中如果能展示出这种跨领域的技术视野和工程直觉，会非常加分。同时，也要关注各种技术的局限性和适用边界。没有银弹，每种方案都有其最佳应用场景。能够根据具体需求选择合适的技术方案，是高级工程师的核心能力。

# 第六章 混合检索与重排序

## 6.1 稠密检索 vs 稀疏检索

稠密检索基于语义向量相似度，擅长处理同义词、近义词和语义理解；稀疏检索基于关键词匹配（如 BM25），擅长处理专有名词、精确数字和低频词汇。两者具有天然的互补性：稠密检索解决「语义漂移」问题，稀疏检索解决「精确匹配」问题。

## 6.2 RRF 融合策略

RRF（Reciprocal Rank Fusion）是最简单有效的排名融合方法。公式：score(d) = Σ 1/(k + rank_i(d))，其中 k 通常取 60。RRF 的优势是与分数量纲无关，可以直接融合不同系统的排名结果。相比于 CombSUM、CombMNZ 等权重方案，RRF 不需要调参，鲁棒性更好。

## 6.3 Cross-Encoder Reranker

Cross-Encoder 将查询和文档拼接为一个序列输入 BERT 类模型，通过全交互注意力计算精细的相关性分数。常用模型包括 ms-marco-MiniLM-L-12-v2 和 BGE-reranker。Cross-Encoder 精度高但速度慢（O(n) 次前向传播），通常用于对粗排 Top-50 结果进行精排，最终取 Top-3~5 作为最终结果。

## 6.4 LLM Reranker

使用 LLM（如 GPT-4o）作为 Reranker，通过 Prompt 让模型对每个 Chunk 与 Query 的相关性打分（1-10 分），然后按分数重排。优势是精度高、可解释性强；缺点是延迟高、成本大。适合对准确率要求极高、对延迟不敏感的场景。

## 本章小结

以上介绍了 混合检索与重排序的核心概念和关键技术要点。理解这些知识对于构建高质量的 RAG 系统至关重要。在实际面试中，面试官往往会从基础概念出发，逐步深入到实现细节和工程经验。建议读者在阅读本章内容的基础上，结合项目代码进行实践验证，加深对各项技术的理解。同时，注意关注该领域的最新进展，因为大模型技术发展迅速，新的方法和工具不断涌现。

在准备面试时，建议将本章的知识点与实际项目经验相结合。不仅要能够清晰地解释技术原理，还要能够说明在实际项目中如何选型、如何调优、遇到了哪些问题以及如何解决。这种理论与实践结合的回答方式，能够给面试官留下深刻印象。此外，建议准备一些具体的数据和指标来支撑你的技术决策，例如选择某个 Embedding 模型后检索精度提升了多少、引入 Reranker 后 Top-3 命中率从 X 提升到 Y 等。

## 延伸思考

学习技术知识不能仅停留在表面，需要深入思考每项技术背后的设计动机和取舍。例如，为什么 RAG 系统要采用两阶段检索（粗排 + 精排）？这是因为精排模型（如 Cross-Encoder）虽然精度高，但计算成本大，无法对全库进行打分。通过粗排快速筛选候选集，再用精排做精细评估，既保

证了精度又控制了延迟。这种'漏斗式'架构思想在搜索引擎、推荐系统等领域都有广泛应用。面试中如果能展示出这种跨领域的技术视野和工程直觉，会非常加分。同时，也要关注各种技术的局限性和适用边界。没有银弹，每种方案都有其最佳应用场景。能够根据具体需求选择合适的技术方案，是高级工程师的核心能力。

# 第七章 文档处理与分块策略

## 7.1 文档解析

文档解析是 RAG 数据摄取的第一步。PDF 解析面临的挑战包括：表格识别、多栏排版、扫描件 OCR、图片提取、以及页眉页脚过滤。本项目使用 MarkItDown 库将 PDF 转换为 Markdown 格式，保留标题层级和格式信息，为后续分块提供结构化线索。

## 7.2 分块策略选择

不同类型的文档适合不同的分块策略：结构化技术文档适合按标题层级切分；长篇叙述性文本适合语义分块（基于 Embedding 相似度检测主题转换点）；代码文档适合按函数/类为单位切分。本项目默认使用递归字符切分（chunk_size=1000, overlap=200），并可通过配置切换。

## 7.3 元数据增强

为 Chunk 附加丰富的元数据可以显著提升检索效果。本项目通过 LLM 为每个 Chunk 生成：- title：概括 Chunk 主题的短标题

- summary：50-100 字的内容摘要

- tags：3-5 个关键词标签

这些元数据可用于检索时的辅助过滤、在搜索结果中提供快速预览。

## 7.4 图片处理流程

对于包含图片的 PDF，系统会：

1. 在解析阶段提取嵌入图片，以 SHA256 哈希命名存储

2. 调用 Vision LLM（如 GPT-4o）生成中文图片描述

3. 将图片描述注入对应 Chunk 的 metadata

4. 在查询时可通过文字描述匹配到相关图片

5. MCP 工具返回结果时包含 Base64 编码的图片预览

## 本章小结

以上介绍了 文档处理与分块策略的核心概念和关键技术要点。理解这些知识对于构建高质量的 RAG 系统至关重要。在实际面试中，面试官往往会从基础概念出发，逐步深入到实现细节和工程经验。建议读者在阅读本章内容的基础上，结合项目代码进行实践验证，加深对各项技术的理解。同时，注意关注该领域的最新进展，因为大模型技术发展迅速，新的方法和工具不断涌现。

在准备面试时，建议将本章的知识点与实际项目经验相结合。不仅要能够清晰地解释技术原理，还要能够说明在实际项目中如何选型、如何调优、遇到了哪些问题以及如何解决。这种理论与实践结合的回答方式，能够给面试官留下深刻印象。此外，建议准备一些具体的数据和指标来支撑你的技术决策，例如选择某个 Embedding 模型后检索精度提升了多少、引入 Reranker 后 Top-3 命中率从 X 提升到 Y 等。

## 延伸思考

学习技术知识不能仅停留在表面，需要深入思考每项技术背后的设计动机和取舍。例如，为什么 RAG 系统要采用两阶段检索（粗排 + 精排）？这是因为精排模型（如 Cross-Encoder）虽然精度高，但计算成本大，无法对全库进行打分。通过粗排快速筛选候选集，再用精排做精细评估，既保证了精度又控制了延迟。这种'漏斗式'架构思想在搜索引擎、推荐系统等领域都有广泛应用。面试中如果能展示出这种跨领域的技术视野和工程直觉，会非常加分。同时，也要关注各种技术的局限性和适用边界。没有银弹，每种方案都有其最佳应用场景。能够根据具体需求选择合适的技术方案，是高级工程师的核心能力。

# 第八章 Agent 与工具调用

## 8.1 Agent 基础概念

Agent 是能够自主感知环境、规划任务、执行动作并根据反馈调整策略的 AI 系统。与简单的 Prompt → Response 模式不同，Agent 引入了工具调用、记忆管理和反思机制，使 LLM 能够处理需要多步推理和外部交互的复杂任务。

## 8.2 工具调用机制

工具调用（Function Calling / Tool Use）是 Agent 的核心能力。现代 LLM 支持在对话中生成结构化的工具调用请求（JSON 格式），由运行时执行对应功能后将结果返回给模型。常见工具类型包括：搜索引擎、数据库查询、计算器、代码执行器、文件操作等。

## 8.3 MCP 协议详解

Model Context Protocol (MCP) 标准化了 LLM 与工具之间的通信协议。核心接口包括：

- tools/list：返回可用工具列表及其参数 Schema

- tools/call：执行指定工具并返回结果

- resources/list：列出可访问的资源（如文件、数据库）

本项目的 MCP Server 通过 Stdio 传输层与 VS Code Copilot 集成。

## 8.4 Agent 设计模式

常见的 Agent 设计模式包括：

- ReAct：Reasoning + Acting 交替进行

- Plan-and-Execute：先制定完整计划再逐步执行

- Reflection：执行后自我反思并修正

- Multi-Agent：多个 Agent 协作完成任务，各自负责不同职能

## 本章小结

以上介绍了 Agent 与工具调用的核心概念和关键技术要点。理解这些知识对于构建高质量的 RAG 系统至关重要。在实际面试中，面试官往往会从基础概念出发，逐步深入到实现细节和工程经验。建议读者在阅读本章内容的基础上，结合项目代码进行实践验证，加深对各项技术的理解。同时，注意关注该领域的最新进展，因为大模型技术发展迅速，新的方法和工具不断涌现。

在准备面试时，建议将本章的知识点与实际项目经验相结合。不仅要能够清晰地解释技术原理，还要能够说明在实际项目中如何选型、如何调优、遇到了哪些问题以及如何解决。这种理论与实践结

合的回答方式，能够给面试官留下深刻印象。此外，建议准备一些具体的数据和指标来支撑你的技术决策，例如选择某个 Embedding 模型后检索精度提升了多少、引入 Reranker 后 Top-3 命中率从 X 提升到 Y 等。

## 延伸思考

学习技术知识不能仅停留在表面，需要深入思考每项技术背后的设计动机和取舍。例如，为什么 RAG 系统要采用两阶段检索（粗排 + 精排）？这是因为精排模型（如 Cross-Encoder）虽然精度高，但计算成本大，无法对全库进行打分。通过粗排快速筛选候选集，再用精排做精细评估，既保证了精度又控制了延迟。这种'漏斗式'架构思想在搜索引擎、推荐系统等领域都有广泛应用。面试中如果能展示出这种跨领域的技术视野和工程直觉，会非常加分。同时，也要关注各种技术的局限性和适用边界。没有银弹，每种方案都有其最佳应用场景。能够根据具体需求选择合适的技术方案，是高级工程师的核心能力。

# 第九章 模型评估与基准测试

## 9.1 RAG 评估框架

RAG 系统的评估需要同时考虑检索质量和生成质量两个维度。检索维度关注的是能否找到正确的文档（Hit Rate、MRR、NDCG）；生成维度关注的是 LLM 输出的质量（Faithfulness、Answer Relevancy、Context Precision）。

## 9.2 Ragas 评估工具

Ragas 是一个专门用于 RAG 系统评估的开源框架。它提供了一系列基于 LLM-as-Judge 的评估指标：Faithfulness 衡量回答是否忠实于检索上下文，Answer Relevancy 衡量回答与问题的相关度，Context Precision 衡量检索上下文中相关信息的占比。Ragas 通过调用 LLM 对回答进行自动评分，无需人工标注。

## 9.3 Golden Test Set

Golden Test Set 是一组预定义的查询-期望答案对，用于批量评估系统的端到端质量。构建 Golden Test Set 时应覆盖不同类型的查询：事实性问题、推理性问题、多跳问题、否定查询等。本项目在 tests/fixtures/golden_test_set.json 中维护了测试集。

## 9.4 A/B 测试方法

对 RAG 系统的改动（如切换 Embedding 模型、调整分块参数）需要通过 A/B 测试验证效果。方法是在相同的 Golden Test Set 上分别运行改动前后的系统，对比各项指标。Evaluation Panel 页面的 History 功能支持保存和对比历史评估记录。

## 本章小结

以上介绍了 模型评估与基准测试的核心概念和关键技术要点。理解这些知识对于构建高质量的 RAG 系统至关重要。在实际面试中，面试官往往会从基础概念出发，逐步深入到实现细节和工程经验。建议读者在阅读本章内容的基础上，结合项目代码进行实践验证，加深对各项技术的理解。同时，注意关注该领域的最新进展，因为大模型技术发展迅速，新的方法和工具不断涌现。

在准备面试时，建议将本章的知识点与实际项目经验相结合。不仅要能够清晰地解释技术原理，还要能够说明在实际项目中如何选型、如何调优、遇到了哪些问题以及如何解决。这种理论与实践结合的回答方式，能够给面试官留下深刻印象。此外，建议准备一些具体的数据和指标来支撑你的技术决策，例如选择某个 Embedding 模型后检索精度提升了多少、引入 Reranker 后 Top-3 命中率从 X 提升到 Y 等。

## 延伸思考

学习技术知识不能仅停留在表面，需要深入思考每项技术背后的设计动机和取舍。例如，为什么 RAG 系统要采用两阶段检索（粗排 + 精排）？这是因为精排模型（如 Cross-Encoder）虽然精度

高，但计算成本大，无法对全库进行打分。通过粗排快速筛选候选集，再用精排做精细评估，既保证了精度又控制了延迟。这种'漏斗式'架构思想在搜索引擎、推荐系统等领域都有广泛应用。面试中如果能展示出这种跨领域的技术视野和工程直觉，会非常加分。同时，也要关注各种技术的局限性和适用边界。没有银弹，每种方案都有其最佳应用场景。能够根据具体需求选择合适的技术方案，是高级工程师的核心能力。

# 第十章 推理优化与部署

## 10.1 模型量化

模型量化将浮点权重转换为低精度表示（如 INT8、INT4、FP16），大幅降低模型的存储空间和推理计算量。常用方案包括：GPTQ（Post-Training Quantization）、AWQ（Activation-aware Weight Quantization）和 bitsandbytes（NF4 量化）。4-bit 量化通常可将模型大小减少 75%，性能损失在 1-3% 以内。

## 10.2 推理框架

主流的 LLM 推理框架包括：

- vLLM：支持 PagedAttention，内存效率高

- TGI（Text Generation Inference）：HuggingFace 出品

- Ollama：本地部署一键运行

- LM Studio：图形化本地部署工具

- TensorRT-LLM：NVIDIA 官方优化框架

选择框架时需综合考虑吞吐量需求、硬件条件和易用性。

## 10.3 Serving 策略

生产环境中的 LLM Serving 需要考虑：并发请求管理（请求队列 + 批处理）、流式输出（Server-Sent Events / WebSocket）、负载均衡（多实例 + Gateway）、以及故障恢复（健康检查 + 自动重启）。对于 RAG 应用，还需要考虑 Embedding 服务和向量数据库的独立扩展。

## 10.4 成本优化

使用云端 API 的成本优化策略：

1. 语义缓存减少重复调用

2. 合理设置 max_tokens 避免浪费

3. 使用较小模型做初筛，大模型做精排

4. 对非关键功能使用更便宜的模型（如 Chunk Refiner 用较小模型）

5. 批量 Embedding 减少 API 调用次数

## 本章小结

以上介绍了 推理优化与部署的核心概念和关键技术要点。理解这些知识对于构建高质量的 RAG 系统至关重要。在实际面试中，面试官往往会从基础概念出发，逐步深入到实现细节和工程经验。建议读者在阅读本章内容的基础上，结合项目代码进行实践验证，加深对各项技术的理解。同时，注意关注该领域的最新进展，因为大模型技术发展迅速，新的方法和工具不断涌现。

在准备面试时，建议将本章的知识点与实际项目经验相结合。不仅要能够清晰地解释技术原理，还要能够说明在实际项目中如何选型、如何调优、遇到了哪些问题以及如何解决。这种理论与实践结合的回答方式，能够给面试官留下深刻印象。此外，建议准备一些具体的数据和指标来支撑你的技术决策，例如选择某个 Embedding 模型后检索精度提升了多少、引入 Reranker 后 Top-3 命中率从 X 提升到 Y 等。

## 延伸思考

学习技术知识不能仅停留在表面，需要深入思考每项技术背后的设计动机和取舍。例如，为什么 RAG 系统要采用两阶段检索（粗排 + 精排）？这是因为精排模型（如 Cross-Encoder）虽然精度高，但计算成本大，无法对全库进行打分。通过粗排快速筛选候选集，再用精排做精细评估，既保证了精度又控制了延迟。这种'漏斗式'架构思想在搜索引擎、推荐系统等领域都有广泛应用。面试中如果能展示出这种跨领域的技术视野和工程直觉，会非常加分。同时，也要关注各种技术的局限性和适用边界。没有银弹，每种方案都有其最佳应用场景。能够根据具体需求选择合适的技术方案，是高级工程师的核心能力。

# 第十一章 多模态大模型

## 11.1 视觉语言模型 (VLM)

视觉语言模型（如 GPT-4o、Claude Vision、Gemini）能够同时理解文本和图像，是实现多模态 RAG 的基础。在本项目中，VLM 用于为 PDF 中提取的图片生成中文描述（Image Captioning），使图片内容也可以通过文字检索被发现。

## 11.2 多模态 RAG

多模态 RAG 将检索范围从纯文本扩展到图文混合内容。实现思路有两种：

1. 将图片转化为文本描述后统一做文本检索（本项目采用方案）

2. 使用 CLIP 等多模态 Embedding 模型同时编码文本和图片

方案 1 实现简单、兼容性好；方案 2 保留了更丰富的视觉信息但复杂度更高。

## 11.3 图片描述生成最佳实践

生成高质量 Image Caption 的关键：

1. 使用具体的 Prompt 指导描述重点（如「请详细描述图中的流程和数据」）

2. 为不同类型的图片使用不同的 Prompt（截图 vs 照片 vs 图表）

3. 输出中文描述以匹配中文检索需求

4. 限制描述长度，避免过长的 caption 干扰检索

## 11.4 OCR 与文档理解

对于扫描件 PDF，需要使用 OCR（光学字符识别）提取文字。现代 OCR 方案包括 Tesseract（开源）、Azure Document Intelligence（云服务）和 PaddleOCR（百度开源）。结合 Layout 分析模型可以进一步理解文档版面结构，正确识别表格、标题和段落。

## 本章小结

以上介绍了 多模态大模型的核心概念和关键技术要点。理解这些知识对于构建高质量的 RAG 系统至关重要。在实际面试中，面试官往往会从基础概念出发，逐步深入到实现细节和工程经验。建议读者在阅读本章内容的基础上，结合项目代码进行实践验证，加深对各项技术的理解。同时，注意关注该领域的最新进展，因为大模型技术发展迅速，新的方法和工具不断涌现。

在准备面试时，建议将本章的知识点与实际项目经验相结合。不仅要能够清晰地解释技术原理，还要能够说明在实际项目中如何选型、如何调优、遇到了哪些问题以及如何解决。这种理论与实践结合的回答方式，能够给面试官留下深刻印象。此外，建议准备一些具体的数据和指标来支撑你的技

术决策，例如选择某个 Embedding 模型后检索精度提升了多少、引入 Reranker 后 Top-3 命中率从 X 提升到 Y 等。

## 延伸思考

学习技术知识不能仅停留在表面，需要深入思考每项技术背后的设计动机和取舍。例如，为什么 RAG 系统要采用两阶段检索（粗排 + 精排）？这是因为精排模型（如 Cross-Encoder）虽然精度高，但计算成本大，无法对全库进行打分。通过粗排快速筛选候选集，再用精排做精细评估，既保证了精度又控制了延迟。这种'漏斗式'架构思想在搜索引擎、推荐系统等领域都有广泛应用。面试中如果能展示出这种跨领域的技术视野和工程直觉，会非常加分。同时，也要关注各种技术的局限性和适用边界。没有银弹，每种方案都有其最佳应用场景。能够根据具体需求选择合适的技术方案，是高级工程师的核心能力。

# 第十二章 安全与对齐

## 12.1 对齐技术

模型对齐（Alignment）旨在使 LLM 的行为符合人类价值观和意图。主要手段包括 RLHF、DPO、Constitutional AI 等。对齐的目标通常概括为 HHH 原则：Helpful（有帮助）、Harmless（无害）、Honest（诚实）。

## 12.2 RAG 安全注意事项

RAG 系统面临的安全风险包括：

1. 知识库投毒：恶意文档被摄入知识库，污染检索结果

2. Prompt 注入：用户通过查询注入恶意指令

3. 数据泄露：敏感文档内容通过检索结果暴露

4. 引用伪造：LLM 编造不存在的引用来源

## 12.3 防御措施

针对上述风险的防御方案：

1. 文档摄入时进行内容审查和分类

2. 查询输入过滤和 sanitization

3. 基于角色的访问控制（RBAC），不同用户只能检索授权文档

4. 输出检验：验证引用的 chunk_id 和 source 确实存在

5. 审计日志：记录所有查询和返回内容，便于追溯

## 12.4 红队测试

红队测试（Red Teaming）通过模拟攻击来发现系统安全漏洞。对 RAG 系统，红队测试应覆盖：恶意查询（尝试提取系统 Prompt）、越狱攻击（绕过安全限制）、数据提取攻击（尝试获取完整文档内容）等场景。建议定期进行红队评估。

## 本章小结

以上介绍了 安全与对齐的核心概念和关键技术要点。理解这些知识对于构建高质量的 RAG 系统至关重要。在实际面试中，面试官往往会从基础概念出发，逐步深入到实现细节和工程经验。建议读者在阅读本章内容的基础上，结合项目代码进行实践验证，加深对各项技术的理解。同时，注意关注该领域的最新进展，因为大模型技术发展迅速，新的方法和工具不断涌现。

在准备面试时，建议将本章的知识点与实际项目经验相结合。不仅要能够清晰地解释技术原理，还要能够说明在实际项目中如何选型、如何调优、遇到了哪些问题以及如何解决。这种理论与实践结合的回答方式，能够给面试官留下深刻印象。此外，建议准备一些具体的数据和指标来支撑你的技术决策，例如选择某个 Embedding 模型后检索精度提升了多少、引入 Reranker 后 Top-3 命中率从 X 提升到 Y 等。

## 延伸思考

学习技术知识不能仅停留在表面，需要深入思考每项技术背后的设计动机和取舍。例如，为什么 RAG 系统要采用两阶段检索（粗排 + 精排）？这是因为精排模型（如 Cross-Encoder）虽然精度高，但计算成本大，无法对全库进行打分。通过粗排快速筛选候选集，再用精排做精细评估，既保证了精度又控制了延迟。这种'漏斗式'架构思想在搜索引擎、推荐系统等领域都有广泛应用。面试中如果能展示出这种跨领域的技术视野和工程直觉，会非常加分。同时，也要关注各种技术的局限性和适用边界。没有银弹，每种方案都有其最佳应用场景。能够根据具体需求选择合适的技术方案，是高级工程师的核心能力。

# 第十三章 行业应用案例

## 13.1 企业知识库

企业内部知识库是 RAG 最典型的应用场景。核心需求包括：多格式文档支持、权限控制、实时更新和高可用性。典型架构为：文档管理系统 → 自动摄取 Pipeline → 向量数据库 → RAG API → 企业聊天机器人/工单系统集成。

## 13.2 智能客服

基于 RAG 的智能客服系统将产品文档、FAQ、历史工单等作为知识源，用户提问时检索相关内容并生成回答。相比纯 LLM 客服，RAG 客服的优势是回答更准确、可溯源、且可以通过更新知识库快速响应新产品/新政策变更。

## 13.3 代码助手

代码助手（如 GitHub Copilot）可以通过 RAG 技术接入项目代码库和文档，使 LLM 在生成代码时能够参考项目的现有实现和约定。MCP 协议为此提供了标准化接口，使 AI 代码助手能够动态发现和查询项目的知识库。

## 13.4 法律/医疗文档分析

在法律和医疗等专业领域，RAG 系统可以帮助专业人员快速检索和分析大量文档。这些场景对准确性要求极高，通常需要：领域微调的 Embedding 模型、严格的引用追溯机制、以及人工审核环节。RAG 在此类场景中作为「辅助工具」而非「替代决策者」。

## 本章小结

以上介绍了 行业应用案例的核心概念和关键技术要点。理解这些知识对于构建高质量的 RAG 系统至关重要。在实际面试中，面试官往往会从基础概念出发，逐步深入到实现细节和工程经验。建议读者在阅读本章内容的基础上，结合项目代码进行实践验证，加深对各项技术的理解。同时，注意关注该领域的最新进展，因为大模型技术发展迅速，新的方法和工具不断涌现。

在准备面试时，建议将本章的知识点与实际项目经验相结合。不仅要能够清晰地解释技术原理，还要能够说明在实际项目中如何选型、如何调优、遇到了哪些问题以及如何解决。这种理论与实践结合的回答方式，能够给面试官留下深刻印象。此外，建议准备一些具体的数据和指标来支撑你的技术决策，例如选择某个 Embedding 模型后检索精度提升了多少、引入 Reranker 后 Top-3 命中率从 X 提升到 Y 等。

## 延伸思考

学习技术知识不能仅停留在表面，需要深入思考每项技术背后的设计动机和取舍。例如，为什么 RAG 系统要采用两阶段检索（粗排 + 精排）？这是因为精排模型（如 Cross-Encoder）虽然精度高，但计算成本大，无法对全库进行打分。通过粗排快速筛选候选集，再用精排做精细评估，既保证了精度又控制了延迟。这种'漏斗式'架构思想在搜索引擎、推荐系统等领域都有广泛应用。面试中

如果能展示出这种跨领域的技术视野和工程直觉，会非常加分。同时，也要关注各种技术的局限性和适用边界。没有银弹，每种方案都有其最佳应用场景。能够根据具体需求选择合适的技术方案，是高级工程师的核心能力。

# 第十四章 面试高频问题精选

## 14.1 RAG 相关问题

Q1: 解释 RAG 的工作原理，以及它如何解决 LLM 的幻觉问题?

Q2: 混合检索（Hybrid Search）的原理和优势是什么?

Q3: Cross-Encoder 和 Bi-Encoder 的区别？在 RAG 中各自的角色?

Q4: 你是如何设计文档分块策略的？考虑了哪些因素?

Q5: RAG vs 微调（Fine-tuning），分别在什么场景下使用?

## 14.2 Embedding 相关问题

Q1: 余弦相似度和内积（Dot Product）有什么区别？什么时候用哪个?

Q2: 什么是 ANN（近似最近邻）？常见的 ANN 算法有哪些?

Q3: 如何评估 Embedding 模型的质量？有哪些 Benchmark?

Q4: Embedding 维度越高越好吗？维度对检索效果和性能有什么影响?

Q5: 如何处理「领域特定词汇」的 Embedding 效果不好的问题?

## 14.3 系统设计问题

Q1: 设计一个支持百万级文档的 RAG 系统，你会怎么做?

Q2: 如何实现 RAG 系统的增量更新（新增/删除/修改文档）?

Q3: 如何监控 RAG 系统的线上质量？有哪些关键指标?

Q4: 如何处理 RAG 中的多语言问题?

Q5: 你的 RAG 项目中遇到了什么技术挑战？如何解决的?

## 14.4 Agent 相关问题

Q1: 什么是 Agent？和普通的 LLM 对话有什么区别?

Q2: 解释 ReAct 模式的工作流程。

Q3: 工具调用（Function Calling）是如何实现的?

Q4: 多 Agent 协作有哪些常见模式?

Q5: 如何评估 Agent 的能力和可靠性?

## 本章小结

以上介绍了 面试高频问题精选的核心概念和关键技术要点。理解这些知识对于构建高质量的 RAG 系统至关重要。在实际面试中，面试官往往会从基础概念出发，逐步深入到实现细节和工程经验。建议读者在阅读本章内容的基础上，结合项目代码进行实践验证，加深对各项技术的理解。同时，注意关注该领域的最新进展，因为大模型技术发展迅速，新的方法和工具不断涌现。

在准备面试时，建议将本章的知识点与实际项目经验相结合。不仅要能够清晰地解释技术原理，还要能够说明在实际项目中如何选型、如何调优、遇到了哪些问题以及如何解决。这种理论与实践结合的回答方式，能够给面试官留下深刻印象。此外，建议准备一些具体的数据和指标来支撑你的技术决策，例如选择某个 Embedding 模型后检索精度提升了多少、引入 Reranker 后 Top-3 命中率从 X 提升到 Y 等。

## 延伸思考

学习技术知识不能仅停留在表面，需要深入思考每项技术背后的设计动机和取舍。例如，为什么 RAG 系统要采用两阶段检索（粗排 + 精排）？这是因为精排模型（如 Cross-Encoder）虽然精度高，但计算成本大，无法对全库进行打分。通过粗排快速筛选候选集，再用精排做精细评估，既保证了精度又控制了延迟。这种'漏斗式'架构思想在搜索引擎、推荐系统等领域都有广泛应用。面试中如果能展示出这种跨领域的技术视野和工程直觉，会非常加分。同时，也要关注各种技术的局限性和适用边界。没有银弹，每种方案都有其最佳应用场景。能够根据具体需求选择合适的技术方案，是高级工程师的核心能力。

# 第十五章 项目实战经验总结

## 15.1 项目架构回顾

本项目采用 Modular RAG 架构，核心模块包括：文档解析（MarkItDown）、智能分块（RecursiveCharacterTextSplitter + LLM Refiner）、双路编码（Azure Embedding + BM25/jieba）、混合检索（RRF 融合）、可选重排序（Cross-Encoder / LLM Reranker）、以及 MCP Server 对外接口。全链路采用配置驱动的工厂模式，支持一键切换 Provider。

## 15.2 技术亮点与面试话术

简历和面试中可以重点突出：

1. 全链路可观测性：Streamlit Dashboard + 结构化 Trace + 评估面板

2. 混合检索策略：Dense + Sparse + RRF 融合 + 可选 Reranking

3. 多模态支持：PDF 图片提取 + Vision LLM Captioning

4. 可插拔架构：工厂模式 + 配置驱动，零代码切换 Provider

5. 数据完整性：SHA256 幂等摄取 + 跨存储级联删除

## 15.3 踩坑记录

开发过程中遇到的典型问题和解决方案：

1. 中文 BM25 分词：需使用 jieba 替代默认英文分词器

2. PDF 表格解析：MarkItDown 对复杂表格支持有限，可能丢失格式

3. 向量维度不匹配：切换 Embedding 模型后需要重新摄取所有文档

4. Trace 文件过大：需要定期清理或引入日志轮转

5. Windows 编码问题：控制台输出需要显式设置 UTF-8 编码

## 15.4 下一步计划

可以继续迭代的方向：

- 引入 GraphRAG，结合知识图谱提升多跳推理能力

- 添加 Streaming Response 流式生成

- 接入更多向量数据库（FAISS、Milvus）

- 实现自动化 CI/CD 测试管线

- 支持更多文档格式（HTML、代码文件、音视频转写）

## 本章小结

以上介绍了 项目实战经验总结的核心概念和关键技术要点。理解这些知识对于构建高质量的 RAG 系统至关重要。在实际面试中，面试官往往会从基础概念出发，逐步深入到实现细节和工程经验。建议读者在阅读本章内容的基础上，结合项目代码进行实践验证，加深对各项技术的理解。同时，注意关注该领域的最新进展，因为大模型技术发展迅速，新的方法和工具不断涌现。

在准备面试时，建议将本章的知识点与实际项目经验相结合。不仅要能够清晰地解释技术原理，还要能够说明在实际项目中如何选型、如何调优、遇到了哪些问题以及如何解决。这种理论与实践结合的回答方式，能够给面试官留下深刻印象。此外，建议准备一些具体的数据和指标来支撑你的技术决策，例如选择某个 Embedding 模型后检索精度提升了多少、引入 Reranker 后 Top-3 命中率从 X 提升到 Y 等。

## 延伸思考

学习技术知识不能仅停留在表面，需要深入思考每项技术背后的设计动机和取舍。例如，为什么 RAG 系统要采用两阶段检索（粗排 + 精排）？这是因为精排模型（如 Cross-Encoder）虽然精度高，但计算成本大，无法对全库进行打分。通过粗排快速筛选候选集，再用精排做精细评估，既保证了精度又控制了延迟。这种'漏斗式'架构思想在搜索引擎、推荐系统等领域都有广泛应用。面试中如果能展示出这种跨领域的技术视野和工程直觉，会非常加分。同时，也要关注各种技术的局限性和适用边界。没有银弹，每种方案都有其最佳应用场景。能够根据具体需求选择合适的技术方案，是高级工程师的核心能力。