ChatGPT

基于大语言模型的混合检索系统整体架构

近年来，多种混合检索系统被提出，用于结合稀疏检索（如BM25）与密集检索（如DPR、向量检索）的优势。常见方案是**双通路检索**：先分别用稀疏方法和密集向量方法检索候选文档，然后合并排序。但这种方式需要维护两个索引，系统复杂度高，并且可能漏检在任一路径Top-K之外的相关项 [1]。另有研究尝试构建**统一索引**，例如采用图结构近似最近邻索引（如HNSW）对拼接后的稀疏+密集向量进行检索。该方法通过对齐稀疏与密集向量的距离分布、两阶段计算（先计算密集距离再计算稀疏距离）等技巧来提高检索效率和准确度 [2]。此外，一些检索增强生成（RAG）系统将检索模块与生成模块紧密耦合，形成混合型RAG框架：它们将检索和生成视为协同推理过程，通过生成子查询、中间反馈等机制联合优化检索结果和生成输出 [3] [4]。总体而言，混合检索系统的架构趋于模块化和多阶段化，典型流程包括：查询预处理/重写 → 初始检索（稀疏和/或密集）→ 候选排序（或再检索）→ 生成模型综合回答。

利用LLM提升检索质量的方法

- **查询重写（Query Rewriting）**：针对原始用户查询中的歧义、噪声或多意图，使用大语言模型生成更精准的检索查询是提升召回的有效手段。研究表明，通过LLM进行多样化查询重写，可检索出更多相关文档。例如，DMQR-RAG框架利用LLM生成多种信息量不同的查询变体，以覆盖更广的相关项，从而提升了检索召回率 [5] [6]。在FreshQA等数据集上，DMQR-RAG相比最佳基线的P@5指标提升了约14.5% [6]。此外，Chan等人（2024）和其他工作也指出，LLM重写可明显改善复杂多跳或复合意图查询的检索质量 [5]。
- **文档重排序（Reranking）**：将LLM用于对候选文档重新评分排序，可以进一步提高检索结果的精确度。例如，LlamaIndex等框架采用先用向量召回Top-K文档，再用LLM对这些候选进行排序的二阶段流水线 [7] [8]。LLM4Ranking框架指出，LLM凭借其丰富的语义知识和推理能力，在零样本或少样本条件下就能对文档进行有效排序，其多项实验表明基于LLM的重排方法（如RankGPT）已优于传统神经重排序模型 [9]。FIRST等工作进一步设计了高效的LLM列表式重排序策略，利用LLM生成顺序标识符并结合学习排序的损失函数，加速推理同时保持排序质量 [10]。总体来看，LLM重排序方案在提高检索精度和鲁棒性方面展现出优势，尤其适用于对准确性要求较高的场景。
- **语义过滤（Semantic Filtering）**：LLM还可用于对检索结果进行精细筛选，剔除与查询弱相关的内容。例如，ChunkRAG方法将检索结果分割为多个语义一致的段（chunk），并通过LLM判断每个段与查询的相关性，动态过滤掉不相关或弱相关的信息 [11]。此类方法通过细粒度评价减少冗余和误导性内容，从而提升生成回答的准确性和一致性 [12] [11]。类似思路也可应用于“对查询响应的内容过滤”或“事实校验”等环节，避免低质量或无关证据进入生成模型。
- **其他增强策略**：此外，还有工作利用LLM生成假设文档嵌入（HyDE）来扩充检索召回；或基于LLM的返回反馈对检索模型迭代训练（如利用LLM评分指导DPR的Relevance Feedback）。虽然此类方法文献尚稀，但总体趋势是借助LLM的语义理解能力，提升传统检索流程（预检索/重检索/生成）的准确性与鲁棒性。

各方法的实验效果、适用场景与挑战

大量实验证明，混合检索与LLM增强的策略在问答和信息检索任务上普遍带来显著改进。已有研究表明，结合稀疏和密集检索后，检索准确度和召回率明显高于单一方法 [13]。例如，Hybrid RAG系统在多跳问答（HotpotQA、2Wiki、MuSiQue等）和开放域QA（NQ、MSMARCO）上均取得优异效果，其中一些混合策略在医学或图谱问答任务上相较于纯检索基线也获得10%以上的准确率提升 [13] [14]。在查询重写方面，DMQR-RAG在AmbigNQ、HotpotQA、FreshQA等数据集上对比基线都有提升，其中FreshQA的检索精度（P@5）提高了14.46% [6]。更进一步地，RQ-RAG等研究在PopQA上实现了288%的性能提升、Self-CRAG在PopQA上提

升了320% [15] [16]，说明改进检索质量对复杂推理任务效果尤为重要。LLM重排序的实验也表明，直接使用LLM对候选文档排序，可以在无标注条件下获得超越传统神经网络的方法 [9]。

适用场景：上述方法主要应用于面向复杂QA、知识检索和对话系统等场景。混合检索广泛用于开放域问答、生物医学问答、法律检索等领域，因其兼顾关键词精确度和语义泛化能力而备受重视 [13]。LLM驱动的查询重写和重排序技术，则特别适用于**多跳推理**、**长尾查询**或**查询含糊不清**的场景；在构建智能助手、企业知识库问答时，这些技术可以有效提升检索召回和答案质量。

挑战和问题：尽管进展显著，混合检索系统仍面临多项挑战。首先，系统复杂度和计算成本较高——串联多个大模型或索引结构会带来较大延时和资源开销。其次，如何有效评估和对比各组件仍缺乏标准化基准；现有RAG系统的评测多依赖特定QA数据集，真实用户场景下的检索需求和指标可能更复杂。再者，用户意图建模不足：当前大多静态地处理单次查询，忽略了用户上下文和长期交互；在多轮对话或连续查询场景中，如何维护上下文一致性、避免答案前后矛盾是难点 [17] [18]。此外，检索结果的可信度和可解释性问题也亟需解决：系统需明示证据来源和相关性理由，避免用户对黑箱LLM输出失去信任 [18]。随着实际应用需求增长，这些领域将成为未来研究的重点。

已公开的大模型驱动检索系统

- **LlamaIndex**（原名GPT Index）：由Jerry Liu等开发，用于快速构建基于LLM的知识检索应用。LlamaIndex提供文档分段、索引构建和查询接口等功能，支持多种知识库（文件、数据库、API等）接入。其引擎结合向量检索与LLM生成能力，可对自然语言查询进行解析并返回相关文档片段。目前LlamaIndex也支持混合检索方案，例如利用Qdrant的向量存储开启Hybrid Search，将向量检索与关键词检索结合，以提升检索精确度和召回率 [19]。
- **LangChain Retriever**：LangChain是一个通用的LLM应用框架，其`Retriever`组件提供了统一接口来调用不同类型的检索工具（如向量数据库、搜索API、图数据库等）。开发者只需实现简单的查询-返回文档列表逻辑，就可以将常见引擎（BM25、ElasticSearch、Pinecone、Wikipedia API等）无缝接入流程 [20]。LangChain通过模块化和链式调用的方式，方便构建检索增强的对话机器人或问答系统，支持用户灵活选用混合检索策略（例如并行调用向量检索和关键词检索）。
- **DSPy**：DSPy是IBM推出的一个工具包，旨在以编程方式优化LLM系统的提示和检索流程 [21]。在DSPy中，用户通过Python定义检索模块和生成模块的管道，DSPy会自动搜索和优化提示（Prompt）。该框架借鉴进化算法思想，让LLM自行生成多种提示变体并根据评估指标选优 [22]。DSPy支持典型的RAG流程：它简化了设置检索增强生成任务的过程，可用于多跳问答等场景，并通过自动调优减少手工Prompt工程成本。总体而言，DSPy并非一个检索引擎，而是一个流程编排工具，但它展示了将LLM与检索紧耦合、自动寻优的思路，为混合检索系统的快速开发提供了实践支持。

研究空白与未来挑战

综合现有文献，混合检索系统领域还有若干未解问题值得探索：一是**检索策略的动态自适应**。当前多数系统使用静态检索配置，难以根据查询难度或上下文动态调整稀疏/密集检索的权重和深度。未来工作需要研究联合优化检索-生成流程，根据任务实时调节检索行为（如利用强化学习信号或不确定性估计来导航检索策略） [4]。二是**多轮交互与子目标分解**。在多轮对话或多步推理场景下，检索系统需记忆先前轮次信息并分解子目标，保证跨轮次实体和主题一致性。目前相关研究仍有限 [17]。三是**用户意图和个性化建模**。如何将用户的历史查询、偏好或任务背景融入检索模块，使检索结果更贴近用户真实意图，是未来提升用户体验的关键。此外，可解释性与信任度也是重要方向：系统需要提供检索依据的透明说明，并通过添加可信度度量（如来源可靠性评分、事实显著性评估等）来增强用户对生成结果的信任 [18]。最后，检索效率和可扩展性依然是工程挑战，尤其是在大规模文档库和低延迟要求下，需要新的索引结构与近似搜索算法来加速混合检索过程。总之，用户意图建模、动态融合策略和多轮交互等方向蕴含创新潜力，将推动LLM驱动的混合检索系统迈向更智能和实用的阶段 [4] [17] [18]。

参考文献： 本文观点和数据主要参考了近年来相关领域的研究和开源框架资料 [1] [5] [7] [9] [11] [15] [6] [20] [19] [21] [4] [17] [18] 等。

---

[1] [2] [13] Efficient and Effective Retrieval of Dense-Sparse Hybrid Vectors using Graph-based Approximate Nearest Neighbor Search
https://arxiv.org/html/2410.20381v1

[3] [4] [15] [16] [17] [18] Retrieval-Augmented Generation: A Comprehensive Survey of Architectures, Enhancements, and Robustness Frontiers
https://arxiv.org/html/2506.00054v1

[5] [6] DMQR-RAG: Diverse Multi-Query Rewriting for Retrieval-Augmented Generation
https://arxiv.org/html/2411.13154v1

[7] [8] Using LLM’s for Retrieval and Reranking — LlamaIndex - Build Knowledge Assistants over your Enterprise Data
https://www.llamaindex.ai/blog/using-llms-for-retrieval-and-reranking-23cf2d3a14b6

[9] LLM4Ranking: An Easy-to-use Framework of Utilizing Large Language Models for Document Reranking
https://arxiv.org/html/2504.07439v1

[10] [2406.15657] FIRST: Faster Improved Listwise Reranking with Single Token Decoding
https://arxiv.org/abs/2406.15657

[11] [12] ChunkRAG: A Novel LLM-Chunk Filtering Method for RAG Systems
https://arxiv.org/html/2410.19572v5

[14] Large Language Models for Information Retrieval: A Survey
https://arxiv.org/html/2308.07107v4

[19] Memory and Hybrid Search in RAG using LlamaIndex
https://www.analyticsvidhya.com/blog/2024/09/memory-and-hybrid-search-in-rag-using-llamaindex/

[20] Retrievers | LangChain
https://python.langchain.com/docs/concepts/retrievers/

[21] [22] What is DSPy? | IBM
https://www.ibm.com/think/topics/dspy