Artificial Intelligence and Robotics Research 人工智能与机器人研究, 2025, 14(6), 1410-1423
Published Online November 2025 in Hans. https://www.hanspub.org/journal/airr
https://doi.org/10.12677/airr.2025.146132

Hans汉斯

OptiCacheRAG：融合知识图谱与分布式缓存的多级自适应检索增强生成方法

秦　疆[1]，施水才[1,2]，王洪俊[1,2]，张亚豪[1,2]

[1]北京信息科技大学计算机学院，北京
[2]拓尔思信息技术股份有限公司，北京

收稿日期：2025年9月25日；录用日期：2025年11月6日；发布日期：2025年11月14日

摘　要

大模型在自然语言处理领域展现出强大的生成与推理能力，但幻觉现象、算力成本过高等问题严重制约其应用落地。检索增强生成(RAG)技术通过引入外部知识有效缓解了幻觉问题，而融合知识图谱的GraphRAG进一步解决了传统RAG平面知识表示的缺陷，实现了实体关联建模与多跳推理。然而，GraphRAG在知识图谱构建与问答过程中频繁调用大模型，导致算力消耗激增与响应延迟过长，难以形成“效果－成本”正反馈。为解决该问题，本文提出OptiCacheRAG模型，集成知识图谱驱动的文本索引范式、自适应多级检索框架与Redis分布式缓存机制。该模型首先通过问题分级机制，对低级问题采用轻量化检索、对全局性问题启用深度检索策略，动态匹配用户需求与检索资源；其次利用LRU算法维护分布式缓存，复用关联问题的实体知识以减少重复检索开销。实验从响应速度、检索精度与问答质量三个维度展开评估，结果表明，OptiCacheRAG在保障知识关联建模能力的同时，显著降低了算力成本与响应时间，相较于基线方法实现了性能与效率的协同优化。本文的核心贡献在于：1) 提出自适应多级检索结构以实现“需求－资源”精准匹配；2) 引入Redis分布式缓存以复用关联知识；3) 通过实证验证了模型在效率与精度上的综合优势。

关键词

大语言模型，知识图谱，检索增强生成，自适应多级检索，分布式缓存

OptiCacheRAG: Multi-Level Adaptive RAG with Knowledge Graph and Distributed Cache

Jiang Qin[1], Shuicai Shi[1,2], Hongjun Wang[1,2], Yahao Zhang[1,2]

[1]College of Computer Science, Beijing Information Science and Technology University, Beijing
[2]TRS, Beijing

文章引用: 秦疆, 施水才, 王洪俊, 张亚豪. OptiCacheRAG: 融合知识图谱与分布式缓存的多级自适应检索增强生成方法[J]. 人工智能与机器人研究, 2025, 14(6): 1410-1423. DOI: 10.12677/airr.2025.146132

Received: September 25, 2025; accepted: November 6, 2025; published: November 14, 2025

Abstract

Large language models (LLMs) show strong generation and reasoning abilities in NLP, yet hallucination and high computational costs impede their practical use. Retrieval-Augmented Generation (RAG) mitigates hallucinations via external knowledge, and GraphRAG (integrating knowledge graphs) resolves traditional RAG's flat knowledge representation issue, supporting entity relationship modeling and multi-hop reasoning. However, GraphRAG's frequent LLM invocations in knowledge graph construction and QA cause soaring computation and long latency, failing to form a positive "performance-cost" feedback loop. To address this, we propose OptiCacheRAG, integrating a knowledge graph-driven text indexing paradigm, adaptive multi-level retrieval, and Redis-based distributed caching. It classifies user questions: lightweight retrieval for low-level ones and deep retrieval for global ones, dynamically matching demand with resources. Additionally, it uses the LRU algorithm to maintain the cache, reusing entity knowledge from related questions to cut redundant overhead. Experiments on response speed, retrieval accuracy, and QA quality show that OptiCacheRAG retains knowledge modeling capability while reducing computational costs and latency significantly, realizing performance-efficiency synergy versus baselines. Core contributions: 1) An adaptive multi-level retrieval structure for accurate "demand-resource" matching; (2) Redis-based distributed caching for knowledge reuse; (3) Empirical validation of its efficiency and accuracy advantages.

Keywords

Large Language Models (LLMs), Knowledge Graph, Retrieval-Augmented Generation (RAG), Adaptive Multi-Level Retrieval, Distributed Cache

1. 引言

近年来，在自然语言处理领域出现了一项举世瞩目的新技术：大模型。大模型是由预训练语言模型发展而来的，其拥有超大的参数规模，通常在十亿乃至万亿级别，相对于传统的预训练语言模型，大模型在处理复杂任务时，所表现的生成能力和推理能力远超传统模型，并且有相关数据表明，大模型不仅能理解处理超大规模的文本数据，且还具备上下文学习能力和领域泛化能力，这使它们能轻松进行少样本的迁移学习[1] [2]。但是，在大模型的快速发展过程中，也出现了不少挑战与瓶颈，其中最典型的有大模型幻觉、大模型不可解释性、大模型知识实时性和大模型算力成本等问题。在这些问题中，最首要和最需要解决的问题无非是大模型幻觉和大模型算力成本这两个问题。有学者针对模型幻觉这个问题提出了检索增强生成(RAG)。这是一种结合了信息检索技术与语言生成模型的人工智能技术，该技术通过从外部知识库中检索相关信息，并将其作为提示(prompt)输入给大模型。通过该方法，能有效的缓解大模型幻觉问题，并且可以实时更新知识库，在一定程度上解决了大模型知识不能实时更新的问题。RAG 的出现对于大模型的快速发展至关重要，例如 Realm 提出检索增强语言模型预训练框架，通过动态检索外部知识提升模型泛化能力[3]；Atlas 则基于检索增强实现少样本学习，进一步拓展了 RAG 的应用场景[4]。然

而，现有的 RAG 系统也存在着很多阻碍其发展的限制[5] [6]。比如，大部分的 RAG 技术所使用的知识库，对于知识的表示都依赖于平面的数据表示，忽略了对于不同知识之间的相互联系、相互依赖的表示。仅仅是给模型单调的知识回答，一定程度上会限制模型的生成能力，使模型的回答缺乏层次感，缺乏对用户回答的思考，给出的答案更多的是停留在检索层面。例如，用户向模型提问：“什么手机适合听音乐？”，对于使用平面知识表示的 RAG 系统来说，系统就会通过寻找智能手机、音乐、等关键词汇来检索近似文本块，此时系统很有可能会返回一些仅仅与部分关键词相关的内容，但是这些推荐可能完全不符合用户的需求。由此可见，传统的 RAG 技术存在静态检索与用户的动态需求的矛盾，传统检索通常基于预定义的索引策略，难以动态适应用户查询的上下文或者隐含意图。且过度依赖检索结果，生成模型可能会机械化的复制检索内容，缺乏深层理解或逻辑整合，导致回答冗余或碎片化，与用户期待的答案大相径庭。对此，有学者提出了将有结构的知识库即知识图谱与 RAG 结合，从而出现了 GraphRAG 方法，该方法可以根据用户问题的一般性和要索引的源文本数量进行扩展。该方法使用 LLM (大语言模型)分两个阶段构建基于图的文本索引结构，首先从源文本中抽取出实体知识图谱，然后为所有密切相关的实体预先生成社区摘要。给定一个问题，每个社区摘要用于生成部分响应，最后对于用户的最终响应中再次总结所有部分的响应。该方法将文本数据转化为图结构，显式建模实体间的复杂关联，避免了传统 RAG 对关键词匹配的过度依赖，且通过图便利实现多跳检索(如“A 导致 B，B 影响 C”)，解决了传统 RAG 难以适用于复杂逻辑的问题，且根据子图关系链(如事件发展脉络)，生成结果可标注来源路径，从而可以实现追溯依据，增强了模型回答的可解释性。可以说，GraphRAG 的出现为大模型的发展注入了新的活力，不仅弥补了传统 RAG 的短板，更从底层逻辑上推动了大模型在知识处理、推理能力和应用场景上的突破。但是，GraphRAG 虽能弥补传统 RAG 的不足，但是该方法中在构建知识图谱和用户问答过程中，多次调用 LLM，会造成大量的算力消耗，且系统的响应时间会大大拉长，当用户提出问题后，往往要经过漫长的时间才会从模型得到答案，可以说，综合考虑 GraphRAG 投入的时间成本和算力成本，对于其效果提升来说，难以形成一种正反馈。

为了解决上述问题，我们提出了一种 OptiCacheRAG，这是一个集成了基于图形的文本索引范式和自适应多级检索框架的模型。该模型可以高效的捕捉实体之间复杂的相互依赖关系，从而能产生连贯的上下文丰富的响应，更重要的是模型使用的是自适应多级检索策略，我们注意到，对于用户所提出的问题，并不是所有的问题都需要全局性的知识结构，对于低级性的问题，只需要简单检索即可、对于全局性的问题则需要更加复杂的检索结构。对于自适应多级检索框架来说，就是要合理的对用户提出的问题进行分级，然后选择合适检索策略。该方法不仅能有效的减少系统反应用户问题的时间，还能够减少检索所需要的资源。同时，我们还注意到，当用户在询问一系列的问题时，往往这些问题都有相互关联的部分，很多问题会涉及到相同的实体知识，或者上一个问题的答案与下一个问题的答案密切相关，因此 OptiCacheRAG 中还集成了 Redis 作为分布式缓存，且采用 LRU 算法来自动更新缓存。采用次方法，对于用户所提出的问题集，可以减少更多的检索资源和算力成本，同时大大提高系统响应用户的效率。

综上所述，这项工作的关键贡献如下：

- 为了实现高效的 RAG 系统，我们提出了一种自适应多级检索结构，通过对用户问题的分类，来选择合适的检索方法。以此来实现全面高效的检索。
- 我们集成了 Redis 分布式缓存，用于缓存相关知识，在面对大量的问题集时，可以有效的节约算力成本和检索资源，同时加快系统的响应速度。
- 实验发现，我们进行了大量的实验来评估 OptiCacheRAG 与现有 RAG 模型的有效性。这些评估集中在几个关键维度上，包括响应速度、检索精度、问题回答质量等。结果表明，与基线方法相比，该方法有了显著改进。

2. 检索增强生成

大型语言模型(LLMs)在文本生成和自然语言理解方面展现出前所未有的能力，然而它们仍面临着诸如“幻觉”(即生成不准确或虚假信息)、知识滞后性(无法获取最新信息)以及缺乏特定领域专业知识等挑战。为了有效克服这些局限，检索增强生成(RAG)技术应运而生，并迅速成为提升 LLM 应用可靠性和准确性的关键范式[7]。RAG 的核心思想在于将 LLM 的强大生成能力与外部知识检索系统紧密结合，其典型工作流程始于知识库的构建与索引，将领域特定或最新知识整理后通过向量嵌入等技术进行高效索引。当用户提出查询时，系统随即进入检索阶段，利用查询内容从知识库中召回语义上最相关的文档片段或信息。随后，这些检索到的相关上下文信息被巧妙地增强到原始用户查询中，形成一个更为丰富和有针对性的提示。最终，LLM 在接收到这个增强后的提示后，结合其自身的语言理解和生成能力，生成一个基于事实、更为准确、相关且显著减少“幻觉”的答案。RAG 通过这种“先查证，再回答”的机制，不仅使 LLM 能够访问其训练数据之外的实时或特定领域知识，从而增强了回答的准确性和可靠性，也为 LLM 的输出提供了清晰可追溯的来源，极大地提高了系统的透明度和可信度[8]。然而，尽管 RAG 技术取得了显著进展，现有系统在处理复杂查询、优化检索效率以及适应多源异构知识方面仍面临挑战，例如，Kulkarni 等人虽尝试用强化学习优化领域聊天机器人的 RAG 性能，但未解决检索效率问题[9]；RankVicuna 聚焦零样本文档重排序，却未涉及知识图谱与缓存的融合[10]。本文所提出的 OptiCacheRAG 正旨在解决这些挑战，通过引入创新的问题分类、多级检索和分布式缓存机制，进一步提升 RAG 系统的综合性能。

3. OptiCacheRAG 架构

如图 1 所示，OptiCacheRAG 架构包含知识图谱构建与智能问答处理两大核心流程，为后续基于 LLM 的知识图谱构建及问答推理机制提供了端到端的系统框架。在知识图谱构建流程中，源文本经分块后由 LLM 完成实体关系抽取、融合，并通过向量化存储与图存储形成知识图谱；在智能问答流程中，用户问题经问题分类、实体关系检索、文本获取及缓存查询等模块构建上下文，最终由 LLM 生成回答，同时结合分布式缓存实现性能优化。

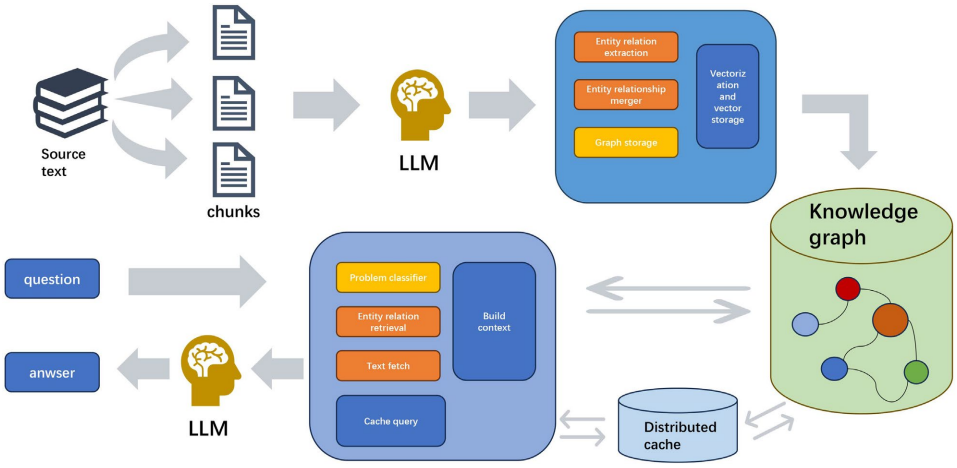

Figure 1. OptiCacheRAG architecture diagram
图 1. OptiCacheRAG 架构图

3.1. 基于 LLM 的知识图谱构建

利用 LLM 强大的语义分析能力，AdaptiveRAG 通过将长文本按规则切分成多个易于管理的小的文

本块。将这些小的文本块嵌入规定的模板，发送大模型。这种策略能够快速识别和访问相关的信息，无需耗费巨大的时间成本来分析整个文档。接下来，利用 LLM 来识别各种实体以及实体之间的关系(如时间、地点、人物、事件等)。通过这个流程所抽取的信息可以构建一个大致的知识图谱。然后再经过一系列的实体检测和知识冲突处理，可以得到一个全面的质量优良的知识图谱。我们将这个知识图谱生成模块正式表示如下：

$$\hat{G} = f_{final}(T) = G_{merge}\left(G_{dedupe}\left(\bigcup_{i=1}^{n} f_{profile}\left(f_{extract}\left(f_{chunk}\left(T_i\right)\right)\right)\right)\right)..$$

其中 $\hat{G}$ 表示生成的最终知识图谱。为了生成最终的知识图谱，我们对长文本 T 进行了一系列的操作，以将文本平面的知识表示构建成有结构的实体关系表示。我们基于 LLM 构建构建知识图谱的函数范式中使用的函数描述如下：

• 文档分块 $fchunk(T)$，该函数的输入是文档集 T = {T1, T2, …, Tn}，这些文档包含大量的文本信息，可能包括多个章节、段落或者不同格式的内容。该函数的输出为多个文本块 Ti，每个 Ti 是一个小的文本片段，可以是一个段落、一个句子或者是一个内容比较紧凑的单元，*fchunk* 函数的任务是将每个文档 T 进行合理的分割。对于处理大规模文档，直接对整个文档进行操作会带来计算上的巨大压力，尤其是在处理高维数据时，内存和计算效率会大打折扣。通过分块，可以有效降低计算的复杂度，提升处理效率。

• 实体和关系抽取 $fextract(Ti)$，该函数的输入是由 $fchunk(T)$ 得到的多个小文档，输出为每个文本块 *Ti* 对应的图 *Gi*，其中包含节点集合 *Vi* 和边集合 *Ei*。

• 节点 *Vi* 包含文本中识别出的所有实体，例如人名(如“Alice”)，地点(如“New York”)，时间(如“2025 年 2 月”)等等。

• 边 *Ei* 则表示这些实体之间的关系，例如“人名 Alice 与地点 New York 之间存在关系(居住、工作等)”。

• 键值对生成与图优化 $fprofile(Gi)$，在知识图谱构建过程中，生成有效的键值对对于高效查询与图优化来说至关重要。键值对生成的主要任务是为图中每个节点和边分配一个索引键(Key)和相关的值(Value)。这种方式使得图的查询变的更加高效，可以快速定位到相关实体以及它们之间的关系。该函数的输入主要是通过 $fextract(Ti)$ 得到的节点集合 V_i 和边集合 E_i。输出为每个节点 $v \in V_i$ 和边 $e \in E_i$ 生成的键值对集合 K_i，其中每个键 *k* 用于高效检索，而值 *v* 是与该节点或边相关的详细文本或数据片段(如相关背景知识或具体描述)。通过添加这些更有用的背景信息，系统在查询时能更高效地找到相关实体和关系，同时为用户提供与这些实体和关系相关的更多上下文信息。

$$K_i = P(V_i, E_i) = \left\{(k_1, v_1), (k_2, v_2), \cdots, (k_p, v_p)\right\}$$

• 去重和图合并 $Gdedupe(Gi)$，在利用 LLM 抽取实体和关系时，常常会抽取出相同实体(尽管这些实体的属性和关系可能略有不同)，但是在检索时都会造成数据冗余，去重不仅仅是去除重复的节点和边，还包括消除同一概念的不同表述。而图合并是将去重后的多个小图合并成为一个完整的图，把所有块中的信息集中起来，构建一个整体的知识图谱。该函数的输入是多个文本块 Ti 的图 Gi (这些图可能包含重复的实体和关系)。输出是去重后的图 *G*。

$$G = D\left(\bigcup_{i=1}^{n} G_i\right)$$

• 增量更新和图融合 $Gmerge(G')$，增量更新是处理新文档加入时的必要步骤。当新的文档 T’被引入时，不需要重建整个图，而是通过增量更新将新数据合并进现有的图中。这使得系统在面对动态变化的

数据时，能够实时更新而不损失性能(比如，当系统收到新的新闻报导时，其中涉及的人物或地点可能是之前图中未提及的，增量更新能够在原有图中添加新的节点和边)，该函数的输入是 G’和新的文档 T’，输出则是最终的知识图谱 $\hat{G}$ 。

3.2. 自适应多级检索结构

在传统的检索系统中，所有用户问题通常都会按照统一的策略进行处理，无论其复杂度和查询范围。然而，在实际应用中，用户提出的问题复杂度差异巨大，一些问题只需要简单的本地检索即可得到答案，而另一些问题则需要复杂的跨节点的全局检索。为此，我们提出了一种自适应多级检索结构，旨在通过智能化地选择合适的检索模式来提高检索的效率，节约计算资源，减少不必要的计算开销。

3.2.1. 问题分类与检索模式选择

自适应多级检索结构的核心是对用户的问题进行分类处理，根据问题的类型动态选择合适的查询模型。这种分类过程帮助系统识别问题的复杂度，并决定是否执行快速的本地查询还是更为复杂的全局查询。具体的检索模式包括：

• Local Query：本地查询通常指在本地存储(如文本文件、文档合集等)中查找相关信息。当用户的问题比较具体，查询目标明确时，本地查询能够快速返回结果

• Global Query：全局查询适用于涉及多个节点或跨系统、跨领域的复杂问题。此类查询需要从全局知识库或者多个子系统中提取信息，适用于需要整合不同数据源信息的情况。

• Hybrid Query：混合查询结合了本地查询和全局查询两种模式。当问题的部分内容可以通过本地数据解决，而其他部分需要全局的信息检索时，混合查询会先从本地进行查询，接着再从全局数据源获取补充信息。

• Naïve Query：最简单的匹配查询，依赖基础的匹配算法，直接搜索关键词或文本片段，处理简单的问题。

在自适应多级检索结构中，当用户提出问题时，首先会进行问题的分类，以确定 该问题最适合哪种检索模式。分类的目标是根据问题的类型和复杂度选择合适的查询策略，从而在不牺牲准确性的情况下提高查询效率，节约计算资源的浪费。

3.2.2. Redis 分布式缓存优化

为进一步提升系统响应效率并降低计算资源消耗，自适应多级检索结构引入 Redis 分布式缓存作为关键支撑组件。当用户发起问题查询时，系统将优先检索 Redis 缓存，判断其中是否已存储该问题的相似答案、用户问答历史内容摘要、相关实体及对应向量，此类信息的缓存存储能够为后续相关问题的匹配提供基础，既无需重复调取底层数据进行计算，也可减少冗余查询操作。考虑到用户在实际使用场景中常提出系列化相关问题，且这类问题的答案、关联实体及向量特征往往存在较强关联性，Redis 缓存机制可直接基于已存储的历史摘要、实体与向量实现快速匹配，有效规避重复计算与冗余查询带来的资源消耗，进一步提升系统响应速度。

为了确保缓存的高效利用，我们在 Redis 缓存中采用了最久未使用淘汰算法(LRU)。该算法根据缓存中项的使用频率和时间，自动淘汰最久未被使用的条目，从而确保频繁查询的结果能够持续保持在缓存中，而不常用的数据则会被移除。这有助于优化存储并提高缓存命中的概率，避免缓存过载。

• 缓存命中：如果缓存中已经有类似问题的答案，系统会直接返回缓存中的结果，避免了重复的查询过程，从而大大节省了检索资源和时间

• 缓存未命中：如果缓存中没有相关的答案，系统将根据问题分类结果，启动响应的检索模式进行

查询。此时，检索的结果会被存储到 Redis 缓存中。

通过引入 LRU 淘汰策略，我们能够在处理大量请求时保持缓存的高效性，同时避免内存占用过多，确保缓存能够适应动态变化的查询需求

3.3. 检索增强型答案生成

当系统完成信息检索以后，在检索增强答案生成中，我们利用检索到的信息块，通过集成和查询相关的实体和关系信息，为答案生成提供基础。这些数据(包括提取的相关实体、关系的描述、以及从原始文档中提取的文本片段)将会嵌入到相关模块发送给 LLM，以此来提供给大模型回答问题的逻辑以及问题的关键信息，生成针对用户问题的答案。通过这种方式，模型能够根据查询的上下文和目标，生成符合需求且标准的答案，确保与用户的意图高度一致。这一过程优化了答案生成，提升了效率和准确性。在答案生成质量评估上，我们参考 ROUGE 工具的文本相似度计算逻辑，确保生成内容与标准答案的一致性[11]。

4. 评估

我们主要通过以下的研究问题，来对 OptiCache-RAG 的有效性来进行评估。(RQ1) OptiCache-RAG 与传统的一些 RAG 框架对比，响应速度如何？(RQ2) OptiCache-RAG 与传统的一些 RAG 框架相比，检索质量如何？(RQ3)问题分类器与 Cache 缓存机制对于 OptiCache-RAG 速度上的增益如何？

4.1. 实验设置

数据集，我们为了评估 optiCacheRAG 在综合知识密集型问答任务上的表现，我们所采用的数据集包含了四个不同领域的专业知识，主要包括军事、社会科学、历史、农业四种领域知识。涵盖了不同类型的挑战。同时我们根据高中语文阅读理解，生成了抽象问题问答集合，以此来测试 OptiCacheRAG 的抽象问题回答能力。以下是我们使用的五种数据集的详细介绍：

• 军事：该数据集主要侧重于开源军事装备和军事事件的考察，侧重于具体细节参数的检索能力。

• 社会科学：该数据集主要由社会科学相关概念一些相关社会事件组成，侧重于对概念相关的知识检索能力。

• 历史：该数据集主要由中国历史时间构成，主要的侧重方向是对于模型幻觉方面的考察。

• 农业：该数据集主要集中于农业实践相关知识的考查，主要的侧重方向是对知识检索顺序的考察。

• 抽象问题集：该数据集主要由阅读理解短文构成，主要考察的是对于语义的分析与关键信息检索的能力。

问题生成，我们分别对四个领域的专业知识进行分割，生成了多个不同的知识社区集群，然后设计 prompt，要求模型扮演不同的角色理解专业知识，并根据相关的文本片段进行提问，并要给出标准答案和相关的知识片段，以便后期进行人工核验答案准确性。

我们的基线分别选取了传统的 RAG 框架 LocalRAG 和最新的 GraphRAG，LocalRAG 是传统的针对文本进行分割，且在向量和文本的基础上进行检索。而 GraphRAG 与 OptiCacheRAG 思想近似，都是利用 LLM 生成知识图谱，再利用知识图谱提高 LLM 的逻辑推理能力，GraphRAG 主要在知识图谱构建时，采用 Leiden 社区检测算法构造多个社区，在检索时根据这些高级社区入手，从而保证了知识检索的全面性和逻辑性。

实验细节，在我们的实验中，主要使用 nano 向量数据库来存储向量，这种轻量化的向量数据库方便部署和使用，且对于所使用的模型，我们分别在三个不同的开源大模型上进行实验，分别是 deepseek-v3、

Qwen-72B、GLM4-32B 三种模型上进行问答测试，来比较附加 RAG 后模型的回答效果。对于实验结果的测评，我们采用了多种不同的评判方式，如 Hu 等人使用多个领域的中文数据集训练微调的 Bert 语义相似度模型来判断模型给出的答案和实际的标准答案是否相似，以此来判断正确还是错误。我们还使用了 BGE 这种向量模型来计算模型的回答与标准答案在向量空间的相似度。在最后的抽象数据集的测试中，除了上述两种方法外，我们还增设了专家打分机制、来判断模型的回答是否更符合人类的偏好。除了在模型回答效果上的测试外，我们在实验测试过程中还会记录模型的响应时间。以此来判断模型的回答效率。

4.2. 比较

我们对所提出的 OptiCacheRAG 系统进行了全面评估。我们将 OptiCacheRAG 部署在三个不同的基准大型语言模型(LLM)上，包括 DeepseekV3、GLM 和 Qwen，并在垂直领域数据集上与基线 RAG 方法(LocalRAG)以及类似的图谱增强 RAG 系统(GraphRAG)进行了深入的对比实验。评估维度涵盖了答案的标准答案相似度、答案错误率以及平均响应时间等关键指标。表 1 汇总了这些在垂直领域数据集上的详细评估结果。

Table 1. Performance comparison of various RAG frameworks
表 1. 各 RAG 框架性能对比

RAG Framework	Model	Standard answer similarity	Error rate of answers	Average response time (second)
GraphRAG	DeepseekV3	65.56%	16.23%	60.687 s
	GLM	58.42%	16.79%	38.840 s
	Qwen	66.35%	15.73%	46.528 s
OptiCacheRAG	DeepseekV3	71.09%	12.23%	10.968 s
	GLM	69.62%	13.03%	10.834 s
	Qwen	72.17%	13.10%	12.426 s
LocalRAG	DeepseekV3	52.41%	18.65%	21.094 s
	GLM	46.42%	17.54%	42.897 s
	Qwen	55.29%	18.47%	25.430 s

基于表 1 的对比结果，我们观察到以下关键发现：

首先，在处理复杂且未知垂直领域问题时，基于知识图谱检索增强系统的结构(包括 GraphRAG 和 OptiCacheRAG)在答案质量方面显著优于传统的标准分块检索增强系统(LocalRAG)。具体而言，无论是标准答案相似度还是答案错误率，图谱增强系统均表现出显著优势。例如，OptiCacheRAG-DeepseekV3 的标准答案相似度达到了 71.09%，显著高于 LocalRAG-DeepseekV3 的 52.41%；同时，OptiCacheRAG 的答案错误率也远低于 LocalRAG，这表明知识图谱所提供的结构化知识能够更精准地定位和提取相关信息，有效避免了传统分块检索中可能出现的低质量信息或不相关上下文干扰，在文档检索相关研究中，Parade 等模型也强调了精准检索对答案质量的重要性。

其次，针对相同结构的图谱检索增强系统，OptiCacheRAG 在检索效率和系统响应速度上表现出显著优势，明显快于 GraphRAG。从表 1 中“平均响应时间”一列可以看出，OptiCacheRAG 在所有基准模型下的响应时间(例如，OptiCacheRAG-DeepseekV3 为 10.968 s)均远低于 GraphRAG (例如，GraphRAG-DeepseekV3 为 60.687s)。这种效率的显著提升主要源于 GraphRAG 在图谱构建以及回答问题的检索过程中，需要频繁、多次地访问和调用 LLM，从而导致巨大的 Token 消耗，造成计算资源的浪费和时间延迟。OptiCacheRAG 则通过对问题的合理分类以及对相关问题和内容的分布式缓存机制，能有效减少这种重复的 LLM 调用，从而极大节约计算资源并加快响应速度。

此外，OptiCacheRAG 在对比基线 LocalRAG 时也展现出明显优势。由于知识图谱的结构性，OptiCacheRAG 能够有效且快速地锁定相关知识区域并进行精准检索，而传统的分块 RAG 则需要进行耗时的全文检索，这对于处理包含大量 Token 的数据集而言是极其费时的。从表 1 的标准答案相似度和答案错误率数据来看，OptiCacheRAG 均明显优于传统的 LocalRAG，进一步证实了其在知识利用和答案生成方面的显著优势。

除了上述垂直领域数据集的测评对比以外，我们还增设了抽象问题集的对比分析，以此来证明 OptiCacheRAG 在语义理解方面的有效性。表 2 汇总了各 RAG 框架在抽象问题集上关于全面性(comprehensive)、准确率(accuracy)、整体性(overall)以及平均响应时间的详细评估结果。

Table 2. Comparison of answer performance among various RAG frameworks
表 2. 各 RAG 框架回答效果对比

RAG Framework	Model	comprehensive	accuracy	overall	Average response time (second)
GraphRAG	DeepseekV3	51.86%	70.08%	54.21%	36.85 s
	GLM	62.73%	69.41%	55.83%	24.49 s
	Qwen	60.94%	69.16%	52.15%	32.97 s
OptiCacheRAG	DeepseekV3	65.33%	75.52%	65.40%	11.72 s
	GLM	69.71%	77.18%	60.06%	12.41 s
	Qwen	68.83%	72.41%	66.56%	15.52 s
LocalRAG	DeepseekV3	37.41%	50.69%	36.03%	30.42 s
	GLM	36.52%	58.20%	37.55%	26.15 s
	Qwen	39.63%	49.25%	35.30%	24.28 s

为了更直观地展示这些多维度性能，以及更好的展示各个框架的综合性能，图 2 (LocalRAG)、图 3 (OptiCacheRAG)和图 4 (GraphRAG)分别呈现了对应系统在抽象问题集上的性能雷达图，其中雷达图的面积大小直观反映了系统的综合性能优劣。

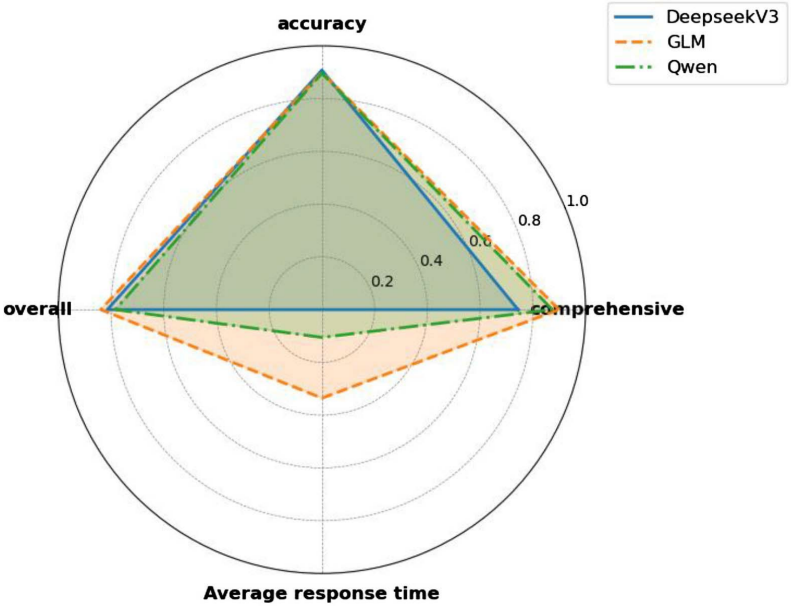

Figure 2. Localrag
图 2. Localrag

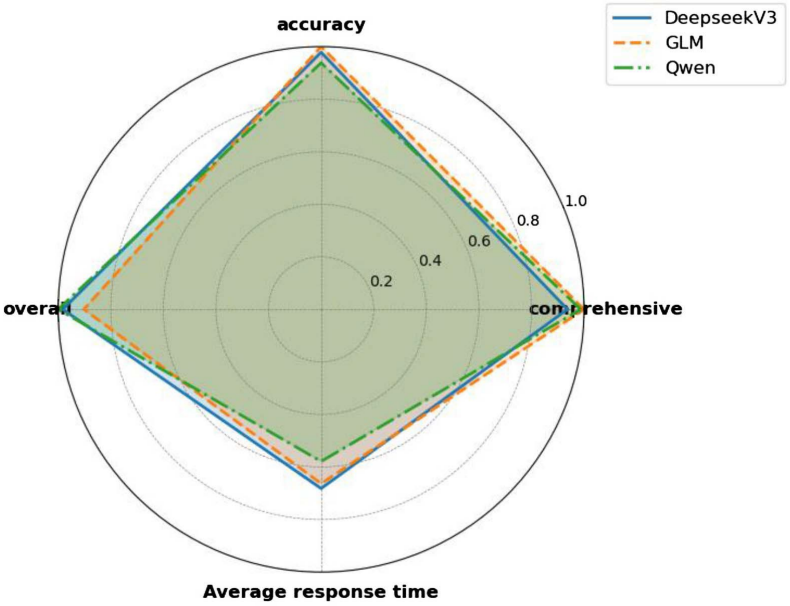

Figure 3. optiCacheRAG
图 3. optiCacheRAG

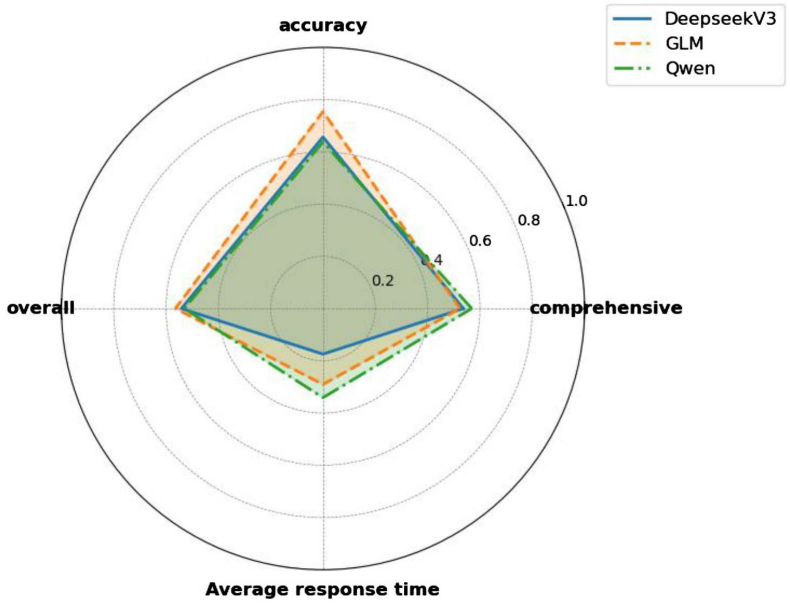

Figure 4. GraphRAG
图 4. GraphRAG

从表 2 和雷达图(图 2、图 3、图 4)可以看出，在对于语义理解方面，这种图结构(GraphRAG 和 OptiCacheRAG)的 RAG 系统显著优于传统的分块 RAG (LocalRAG)。具体而言，图 2 所示的 LocalRAG 在所有基准模型下，其雷达图所覆盖的面积最小，表明其在处理抽象问题时的准确率、全面性和整体性均表现较弱。这主要归功于知识图谱的结构性，它能有效地根据主体的关系找到与主体相关的众多实体，并以一种结构化的格式发送给 LLM，从而使得 LLM 的参考知识更具有逻辑性，在相关研究中，Self-RAG 等模型也通过引入检索和反思机制来提升语义理解能力[12]。

而在 GraphRAG 与 OptiCacheRAG 的对比中，图 3 (OptiCacheRAG)的雷达图在平均响应时间维度上明显更靠近中心(数值更小)，表明其响应速度快于图 4 (GraphRAG)。这进一步证实了 OptiCacheRAG 采用更简洁的问题分类器和分布式缓存后所能带来的显著效率提升。此外，从图 3 和图 4 的对比中也可以看出，OptiCacheRAG 在准确率、全面性和整体性方面普遍优于 GraphRAG，其雷达图在这些维度上通常覆盖了更大的面积，显示了其在回答抽象问题时更优异的综合表现和全面性。这也体现了 OptiCacheRAG 中多级检索结构的有效性，即在问题分类后能够选择更合适的检索方式，从而提升问题回答效果。

在下一小节中，我们将会用消融实验来证明问题分类、多级检索以及分布式缓存等核心组件的有效性。

4.3. 消融研究

为了验证我们提出的 RAG 框架各核心模块的有效性及其对系统整体性能的贡献，我们进行了一系列消融实验。这些实验以我们最终提出的完整架构 OptiCacheRAG 为基准，并系统地通过移除其特定组件来构建不同的消融变体。具体来说，“Low + Cache”框架代表移除了问题分类器和多级检索模块的基础配置；“Classifier + Multi-level”框架代表移除了分布式缓存模块的配置；而“High + Cache”框架则代表移除了高级检索模块的配置。实验结果详细记录于表 3 中，涵盖了 DeepseekV3、GLM 和 Qwen 三种主流大型语言模型在不同 RAG 框架下的综合得分、准确率、整体得分以及平均响应时间。

Table 3. Ablation comparison
表 3. 消融对比

RAG Framework	Model	comprehensive	accuracy	overall	Average response time (second)
Low + Cache	DeepseekV3	53.28%	66.42%	40.96%	18.25 s
	GLM	56.33%	63.21%	42.18%	16.36 s
	Qwen	49.64%	63.56%	43.60%	16.57 s
Classifier + Multi-level	DeepseekV3	64.23	73.80%	64.71%	14.82 s
	GLM	67.95%	76.52%	59.86%	14.53 s
	Qwen	68.25%	73.38%	65.32%	15.94 s
High + Cache	DeepseekV3	60.54%	60.31%	60.23%	17.68 s
	GLM	66.87%	62.82%	55.73%	16.15 s
	Qwen	62.41%	63.57%	58.91%	17.44 s
OptiCacheRAG	DeepseekV3	65.33%	75.52%	65.40%	11.72 s
	GLM	69.71%	77.18%	60.06%	12.41 s
	Qwen	68.83%	72.41%	66.56%	15.52 s

为了更直观、清晰地展示这些消融变体与完整架构之间的性能差异以及各个模块引入后的性能演进趋势，我们进一步将表 3 中的数据可视化为图 5。图 5 是一个包含四个子图的折线图，分别展示了各模型在不同 RAG 框架下各项性能指标的动态变化。通过这些折线图，我们可以清晰地观察到每个关键模块(如问题分类器、多级检索、分布式缓存和高级检索)对系统整体性能的独立贡献和协同效应。

首先，我们评估了问题分类器和多级检索模块的有效性。我们通过比较图 5 中“Low + Cache”与“OptiCacheRAG”这两个框架下各模型性能的趋势变化来体现其贡献。如图 5 (综合得分、准确率和整体得分子图)所示，从最基础的“Low + Cache”框架向“OptiCacheRAG”的演进(这代表了问题分类器和多级检索等关键组件的引入)，在各项评分上均展现出显著的提升。例如，在综合得分子图中，DeepseekV3 模型从“Low + Cache”的 53.28%提升至“OptiCacheRAG”的 65.33%；准确率子图中，DeepseekV3 也从

66.42%提升至 75.52%。这些明显的增长趋势充分表明，我们提出的问题分类器能够有效地识别用户意图，并引导系统进行更精准的多级检索，从而显著提高了答案的综合质量和准确性。值得注意的是，尽管更复杂的检索流程通常会增加开销，但从图 5(平均响应时间子图)的趋势来看，从“Low + Cache”到“OptiCacheRAG”的整体优化，甚至使平均响应时间略有缩短(例如 DeepseekV3 从 18.25s 降至 11.72s)。这一结果有力地证明了精准的分类和检索有效减少了不必要的匹配和计算，反而优化了整体效率。

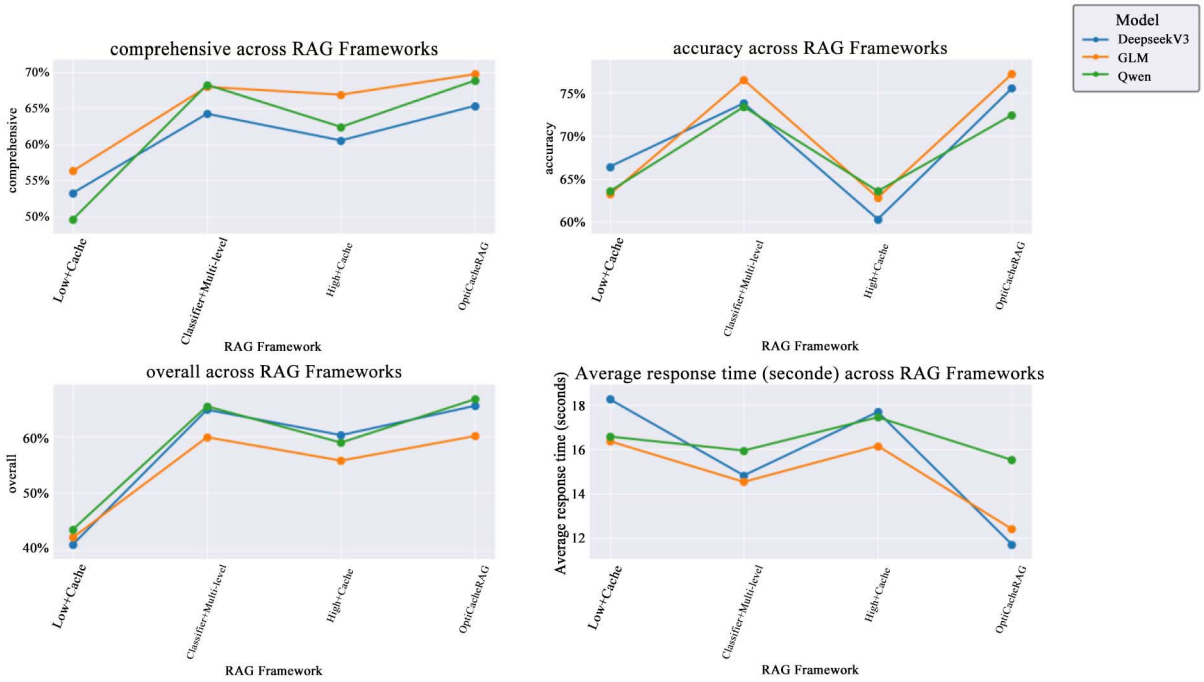

Figure 5. Performance comparison chart
图 5. 性能比对图

接着，我们探究了分布式缓存模块的贡献。我们通过比较图 5 (平均响应时间子图)中“Classifier + Multi-level”与“OptiCacheRAG”这两个框架下模型的平均响应时间趋势来评估其重要性。可以观察到，在引入分布式缓存(即从“Classifier + Multi-level”到“OptiCacheRAG”的转变)后，平均响应时间指标呈现显著下降。例如，以 DeepseekV3 为例，其响应时间从“Classifier + Multi-level”的 14.82 s 大幅降至“OptiCacheRAG”的 11.72 s。这种效率的显著提升验证了分布式缓存能够有效存储和复用常见查询的检索结果，从而显著减少了重复计算和 LLM 的推理时间，极大地提升了系统的实时响应能力。同时，如表 3 所示，缓存的使用在提升效率的同时，并未对回答的综合得分和准确率造成负面影响，这表明我们的缓存策略在保证性能的同时，也兼顾了结果的准确性。

最后，我们考察了高级检索模块的贡献。我们通过比较图 5(综合得分和准确率子图)中“High + Cache”与“OptiCacheRAG”这两个框架下模型的性能趋势来评估其效果。尽管图 5 显示，单纯的“High + Cache”框架在准确率和综合得分上(如表 3 所示，DeepseekV3 的准确率为 60.31%，GLM 为 62.82%，Qwen 为 63.57%)可能低于“Classifier + Multi-level”框架，但从“High + Cache”到“OptiCacheRAG”的转变(这代表了高级检索及其与其他组件的协同作用)使得准确率和综合得分均有显著恢复和提升。这表明高级检索在捕获更深层次、更复杂的语义信息方面具有不可替代的作用。它能够弥补单一低级检索在处理抽象或复杂问题时的不足，为模型提供更全面、更丰富的上下文信息，从而在某些特定场景下提升回答的质

量和准确性，尤其是在处理需要更高级推理能力的问题时，并且在最终的 OptiCacheRAG 框架中，其价值得到了充分体现。

综上所述，本次消融实验通过对系统不同模块的拆分与重组，并以完整的 OptiCacheRAG 框架作为性能基准，如图 5 所示，充分验证了我们所提出的问题分类器、多级检索以及分布式缓存模块的有效性。实验结果直观地表明，每个模块都对系统的性能提升做出了重要贡献，并且它们的协同作用使得 OptiCacheRAG 框架在回答质量和响应速度上均达到了最佳表现。这为我们后续的系统优化和扩展提供了坚实的实验依据。

5. 讨论

评价方法的局限性，尽管 OptiCacheRAG 展现出了显著性能，但是本研究仍然存在局限性。首先是数据集和模型泛化性能有限，当前评估主要是基于特定垂直领域的数据集和抽象问题集，并使用了有限数量的开源 LLM(如 DeepseekV3、GLM、Qwen)。这可能限制了结论在更广泛的领域和不同的 LLM 上的普适性。缓存策略的进一步优化空间，虽然分布式缓存有效提升了效率，但其缓存淘汰机制、缓存命中率优化以及动态缓存策略仍有待深入研究与完善。检索策略的适应性，多级检索策略虽然有效，但在面对极端复杂或领域外的问题时，其自适应性和鲁棒性仍需进一步提升，例如探索更智能的检索路径选择或融合不同类型的知识源。实际部署的复杂性，本研究主要关注技术可行性与性能验证，在实际生产环境中的大规模部署、运维成本、实时数据更新以及安全性方面的挑战，仍需未来的工作进行深入探讨和解决。基于上述局限，未来的工作将聚焦于拓展框架在多领域数据集上的泛化能力，探索更先进的自适应策略，并着力解决实际部署中的工程挑战。在 Autoregressive Search Engines 等研究中，也探讨了更灵活的检索与生成结合方式，为我们未来优化检索策略提供了参考[13]。

6. 结语

本文提出并深入探讨了 OptiCacheRAG，一个旨在优化大型语言模型(LLM)问答系统中检索增强生成(RAG)流程的创新框架。OptiCacheRAG 核心融合了问题分类器、多级检索机制和分布式缓存等关键组件，旨在克服传统 RAG 方法在复杂垂直领域问题处理和响应效率上的瓶颈。

通过在 DeepseekV3、GLM 和 Qwen 三种主流 LLM 上进行的全面评估，我们证实了 OptiCacheRAG 的卓越性能。与传统的基于分块检索的 LocalRAG 相比，OptiCacheRAG 凭借其知识图谱增强的特性，在答案准确率和相似度方面均表现出显著优势。更值得强调的是，在与同为图谱增强型 RAG 系统的 GraphRAG 对比中，OptiCacheRAG 在保证甚至超越回答质量的同时，实现了平均响应时间的大幅优化，证明了其在效率上的巨大突破。此外，在抽象问题集上的多维度评估也进一步验证了 OptiCacheRAG 在语义理解和综合表现上的优越性。

尤为重要的是，我们的消融实验清晰地揭示了 OptiCacheRAG 核心组件的有效性及其协同作用。问题分类器和多级检索确保了信息检索的精准性，有效提升了回答质量，并出乎意料地优化了整体响应效率。分布式缓存则直接且显著地加速了系统响应，有效管理了 LLM 调用开销。这些组件的有机结合，共同构建了一个在效率与准确性之间取得卓越平衡的 RAG 系统。

展望未来，我们将致力于拓展 OptiCacheRAG 在更广泛领域和大规模场景下的泛化能力，并持续优化其内部组件策略，以应对真实世界应用中的复杂挑战。我们相信，OptiCacheRAG 为构建高效、精确且可扩展的 LLM 问答系统提供了新的思路和有价值的贡献。

参考文献

[1] Hurst, A., Lerer, A., Goucher, A.P., Perelman, A., Ramesh, A., *et al*. (2024) GPT-4o System Card.

https://arxiv.org/abs/2410.21276

[2] Achiam, J., Adler, S., Agarwal, S., Ahmad, L., *et al*. (2024) GPT-4 Technical Report. https://arxiv.org/abs/2303.08774

[3] Guu, K., Lee, K., Tung, Z., Pasupat, P. and Chang, M.-W. (2020) Realm: Retrieval-Augmented Language Model Pre-training. 2020 *International Conference on Machine Learning*, Vienna, 12-18 July 2020, 3929-3938.

[4] Izacard, G., Lewis, P., Lomeli, M., Hosseini, L., *et al*. (2023) Atlas: Few-Shot Learning with Retrieval Augmented Language Models. *Journal of Machine Learning Research*, **24**, 1-43.

[5] Drozdov, A., Zhuang, H., Dai, Z., Qin, Z., Rahimi, R., Wang, X., *et al*. (2023). PaRaDe: Passage Ranking Using Demonstrations with LLMs. *Findings of the Association for Computational Linguistics*: *EMNLP* 2023, Singapore, December 2023, 14242-14252. https://doi.org/10.18653/v1/2023.findings-emnlp.950

[6] Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., *et al*. (2020) Dense Passage Retrieval for Open-Domain Question Answering. *Proceedings of the* 2020 *Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), Online, 16-20 November 2020, 6769-6781. https://doi.org/10.18653/v1/2020.emnlp-main.550

[7] Bevilacqua, M., Ottaviano, G., Lewis, P.S.H., *et al*. (2022) Autoregressive Search Engines: Generating Substrings as Document Identifiers. *Advances in Neural Information Processing Systems*, **35**, 31668-31683.

[8] Asai, A., Wu, Z., Wang, Y., Sil, A. and Hajishirzi, H. (2023) Self-Rag: Learning to Retrieve, Generate, and Critique through Self-Reflection. https://arxiv.org/abs/2310.11511

[9] Kulkarni, M., Tangarajan, P., Kim, K. and Trivedi, A. (2024) Reinforcement Learning for Optimizing Rag for Domain Chatbots. https://arxiv.org/abs/2401.06800

[10] Pradeep, R., Sharifymoghaddam, S. and Lin, J. (2023) RankVicuna: Zero-Shot Listwise Document Reranking with Open-Source Large Language Models. https://arxiv.org/abs/2309.15088

[11] Lin, C.Y. (2004) ROUGE: A Package for Automatic Evaluation of Summaries. In: *Text Summarization Branches out*, Association for Computational Linguistics, 74-81.

[12] Khandelwal, U., Levy, O., Jurafsky, D., Zettlemoyer, L. and Lewis, M. (2020) Generalization through Memorization: Nearest Neighbor Language Models. 2020 *International Conference on Learning Representations* (*ICLR*), Addis Ababa, 26-30 April 2020. https://arxiv.org/abs/1911.00172

[13] Liu, Z., Ping, W., Roy, R., *et al*. (2024) ChatQA: Surpassing GPT-4 on Conversational QA and RAG. https://arxiv.org/abs/2401.10225