南京师范大学
NANJING NORMAL UNIVERSITY

神经网络与深度学习期末大作业

LinearRAG论文阅读与复现实验

姓名：张熙浚

学号：19230424

学院：计算机与电子信息学院

专业：人工智能

2026 年 1 月 18 日

摘要

本文围绕 LinearRAG 的阅读、理解与复现实验展开，针对朴素 RAG 在多跳推理中的上下文丢失以及 GraphRAG 依赖关系抽取带来的高成本与噪声问题，系统梳理其动机与方法。LinearRAG 通过构建实体—句子—篇章的 Tri-Graph，在不进行关系抽取的前提下，采用语义传播激活实体并结合个性化 PageRank 进行全局重要性聚合，实现线性可扩展的检索。实验基于 HotpotQA、2WikiMultiHopQA、MuSiQue 与 Medical 数据集，生成模型使用阿里百炼 Qwen-Flash，嵌入模型为 all-mpnet-base-v2，索引阶段实体抽取采用 spaCy。结果表明，复现实验的索引耗时与原文趋势一致，准确率与论文报告基本吻合，说明 LinearRAG 在多跳场景下兼具较好的准确性与可控的构建成本。项目代码与材料链接：GitHub，百度网盘。

目录

1 课题背景

检索增强生成（RAG）通过引入外部知识库缓解大语言模型（LLM）的幻觉问题，已成为知识密集型问答与对话系统的主流范式。然而，面对大规模非结构化语料，朴素 RAG 通常依赖文本分块与相似度检索，容易切断文档间的逻辑联系，导致多跳推理时关键上下文丢失、逻辑链断裂，从而产生不完整或错误答案。

为提升跨文档推理能力，GraphRAG 通过结构化图连接文档与知识实体，代表性工作如 Microsoft GraphRAG 利用社区检测与层次聚类捕捉全局结构[1]；LightRAG 进一步采用双层索引在细粒度实体与粗粒度主题间取平衡[2]。但这些方法普遍依赖 LLM 进行显式关系抽取，索引成本高、速度慢，且容易引入错误三元组与噪声连接，造成“高召回、低相关”的检索困境[3]。

针对高成本与高噪声的双重痛点，LinearRAG 提出去关系化的索引范式，以实体为锚构建实体—句子—篇章的 Tri-Graph，通过语义传播与全局重要性聚合代替显式关系抽取，在保留原始语义的同时实现线性可扩展的检索效率，因此具备明确的研究价值与复现意义。

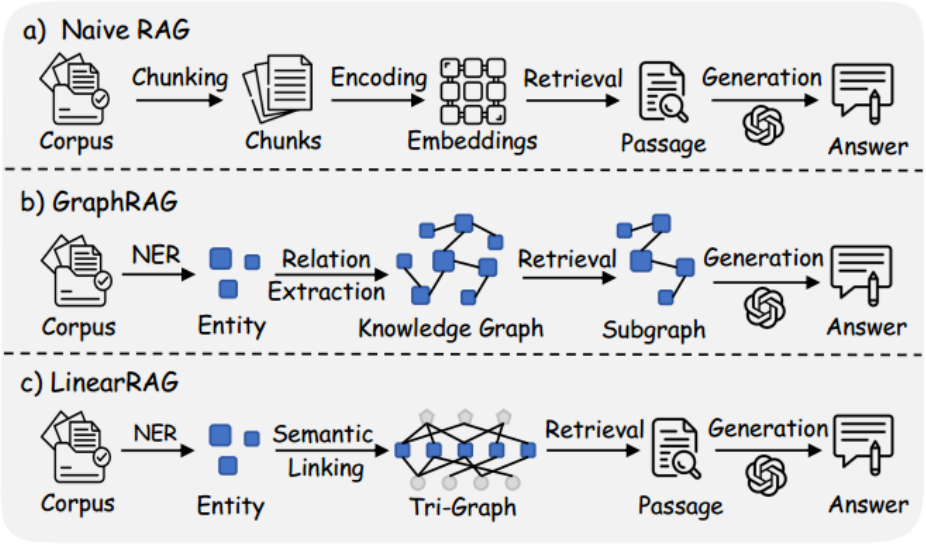


图 1: 多种 RAG 方式的对比

2 国内外研究现状

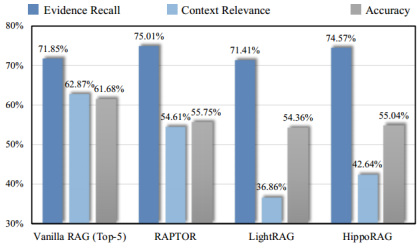
当前图检索范式在真实应用中暴露出两类核心问题：局部不准确与全局不一致[3]。其根源在于 GraphRAG 过度依赖 LLM 进行关系提取，使得知识图谱在三元组质量与整体结构两个层面出现系统性偏差，最终导致检索路径偏移与语义噪声积累。

2.1 局部不准确（Local Inaccuracy）

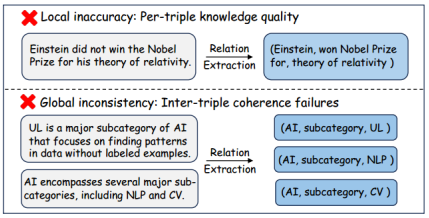
局部不准确指单条三元组层面的事实错误。关系提取模型在处理局部文本时容易忽略否定或语境，生成错误三元组。例如原文描述“爱因斯坦没有因为相对论获得诺贝尔奖”，却可能被错误提取为“(爱因斯坦, 获得了诺贝尔奖, 相对论)”，使图谱中混入虚假事实并直接误导后续检索与生成。

2.2 全局不一致（Global Inconsistency）

全局不一致指跨文档、跨三元组层面的结构冲突。由于关系提取通常在局部片段上独立完成，缺乏全局一致性校验，导致图谱结构碎片化、层次关系混乱。例如将 NLP、CV 与 UL 统一视为 AI 的并列子类，但逻辑上 UL 是技术手段而非平行子领域，造成层次不一致与连接断裂，使多跳推理难以形成正确逻辑链条。



(a) Retrieval and Generation Performance (%) of vanilla RAG *v.s.* typical GraphRAG baselines.



(b) Two types of errors in knowledge graphs brought by imperfect relation extraction methods.

图 2: 局部不准确与全局不一致的影响

上述两类缺陷共同造成 GraphRAG 的性能退化：虽然召回率可能提升，但错误事实与结构冲突带来的语义噪声会显著拉低上下文相关性与最终准确率，形成“高召回、低相关”的典型症状[3]。因此有必要引入一种弱化关系抽取依赖、保持语义传播优势的方案。LinearRAG 通过以实体为锚构建 Tri-Graph，并以语义传播与全局重要性聚合替代显式关系抽取，在降低噪声的同时保持多跳检索能力，成为解决上述问题的合理路径。

2.3 传统 GraphRAG 目前问题

语义表达受限于三元组结构。三元组难以完整承载复杂语义，关系被压缩后容易丢失语境细节。自然语言中的关系往往依赖上下文且具有组合性，例如“瑞秋不情愿地同意和菲比一起去跑步”这样的句子，很难被干净地简化为原子三元组而不损失关键语义；一旦忽略细微语义（如否定或语气），就会进一步加剧局部不准确问题。

成本与效率的 Token 陷阱。传统 GraphRAG 在索引阶段需要调用 LLM 进行大量关系抽取或摘要生成，导致 Token 消耗巨大、构建成本高，限制了大规模应用。以 LightRAG 为例，在 2Wiki 数据集上的索引阶段需要约 35.52M Token，耗时接近 5000 秒；相比之下，LinearRAG 通过轻量级实体提取实现零 Token 索引，并显著提升构建速度，从而在成本与效率上更具优势。

3 本文工作

3.1 文献阅读与问题梳理

系统阅读并理解 RAG 领域的代表性论文，梳理其发展脉络与关键问题。

3.2 源码理解与流程复现

阅读并理解 LinearRAG 源码，明确其核心模块与实现逻辑。

3.3 实验复现与评测验证

复现 LinearRAG 的实验流程与评测设置，以验证其方法有效性。

4 实验方案

4.1 LinearRAG 核心工作与流程

LinearRAG 的总体流程分为离线图构建与在线检索生成两个阶段，先构建 Tri-Graph，再通过两阶段检索输出 Top-K 篇章以支撑生成模型。

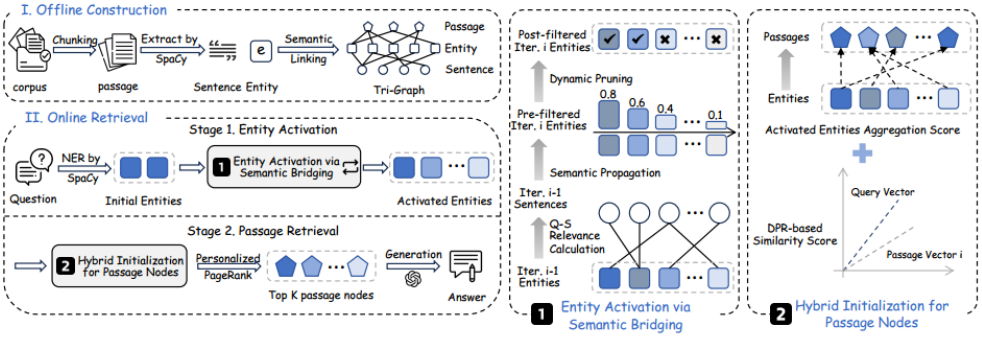


图 3: LinearRAG 的整体流程

离线图构建（Offline Construction）。首先将语料切分为篇章与句子节点；随后使用轻量级 NER 工具（如 spaCy）抽取实体节点；最后通过包含矩阵 C 连接实体与篇章、通过提及矩阵 M 连接实体与句子，形成无关系 Tri-Graph。该过程线性可扩展且不消耗 LLM Token。

在线检索与生成（Online Retrieval & Generation）。检索阶段包含两步：其一，在实体—句子子图上进行语义传播，激活查询未显式提及但具桥接作用的实体，并通过阈值 δ 动态剪枝；其二，在实体—篇章二部图上执行个性化 PageRank，完成全局重要性聚合并输出 Top-K 篇章。生成阶段将检索到的篇章与查询一并输入生成模型得到答案。

4.2 LinearRAG 优势与问题解决

LinearRAG 解决传统 GraphRAG 的三大痛点，形成可扩展、低成本且高相关的检索框架。

解决语义丢失：原始文本作为知识载体。LinearRAG 构建无关系图谱（Relation-free Tri-Graph），以实体、句子和篇章为三层节点，不再强行将文本压缩为三元组。通过实体对齐连接分散篇章，检索时直接提供原始篇章给大模型，完整保留语气、否定与上下文细节，避免关系抽取引发的语义扭曲。

解决高昂成本：零 Token 索引与线性扩展。LinearRAG 采用轻量级命名实体识别（如 spaCy）与语义链接构建图结构，不调用 LLM 进行关系抽取，索引阶段 Token 消耗为零。其算法复杂度随语料规模线性增长，索引构建时间较传统方法显著缩短，在大规模语料上具备更高的吞吐与可扩展性。

解决检索噪声：两阶段精准检索机制。第一阶段通过语义传播激活潜在桥接实体，并利用动态剪枝过滤无关路径，降低噪声传播；第二阶段采用个性化 PageRank 对篇章进行全局重要性聚合，确保 Top-K 篇章既相关又具有结构性价值。该机制在保持高证据召回的同时提升上下文相关性，克服“高召回、低相关”的瓶颈。

4.3 LinearRAG 创新点

4.3.1 零 Token 图构建

LinearRAG 以 Tri-Graph 作为层次化图结构，避免关系抽取导致的高成本与错误。图中包含三类节点：篇章节点 V_p（语料库段落集合 P）、句子节点 V_s（由篇章切分得到的句子集合 S）、实体节点 V_e（由轻量级 NER 识别的实体集合 E）。边仅基于包含或提及关系构建，通过两个邻接矩阵定义：

$$C = [C_{ij}]^{|V_p|\times|V_e|} \tag{1}$$

$$M = [M_{ij}]^{|V_s|\times|V_e|} \tag{2}$$

该构建过程仅依赖轻量工具，避免 LLM Token 消耗，并为后续检索提供稳定的结构支撑。

4.3.2 语义桥接的相关实体激活

第一阶段在实体—句子子图上激活与查询相关的中间实体，用以搭建多跳推理的语义桥接路径。

初始实体激活。从查询 q 中提取实体集合 E_q，并在实体库 V_e 中进行语义匹配，得到稀疏激活向量 a_q：

$$a_q = [a_{q,i}]^{|V_e|\times 1}, \quad a_{q,i} = \mathbb{1}_{i=\arg\max_{e_j\in V_e} sim(e_q,e_j)} \cdot sim(e_q, e_i) \tag{3}$$

查询—句子相关性分布。计算查询与句集 S 中各句子的相似度，得到相关性向量 σ_q：

$$\sigma_q = [\sigma_{q,i}]^{|S|\times 1}, \quad \sigma_{q,i} = sim(q, s_i) \tag{4}$$

语义传播。通过实体—句子—实体路径进行传播，公式为：

$$a_q^t = \mathrm{MAX}(M^T(\sigma_q \odot (M a_q^{t-1})), a_q^{t-1}) \tag{5}$$

传播过程相当于先从实体流向句子，再依据句子相关性进行加权过滤，最后回流到实体，从而激活未被显式提及的桥接实体。

动态剪枝。为抑制语义漂移与规模膨胀，在每轮迭代后用阈值 δ 过滤低分实体；当新一轮无实体超过阈值时自动停止。实践中 δ 过小会引入噪声，过大则可能丢失关键中间实体，需在检索质量与效率间权衡。

4.3.3 全局重要性聚合的篇章检索

第二阶段在实体—篇章二部图上对篇章进行全局重要性聚合，从而获得最终的 Top-K 检索结果。

混合初始化。实体节点 $v_i \in V_e$ 的初始分数直接取第一阶段激活分数 $a_{q,i}$。篇章节点 $v \in V_p$ 的初始分数采用混合评分机制，结合查询—篇章相似度与已激活实体贡献：

$$I(v_i \mid v_i \in V_e) = a_{q,i} \tag{6}$$

$$I(v \mid v \in V_p) = \left(\lambda \cdot sim(q, v) + \ln\left(1 + \sum_{e_i \in E_a} \frac{a_{q,i} \cdot \ln(1 + N_{ei})}{L_{ei}}\right)\right) \cdot W_p \tag{7}$$

个性化 PageRank 聚合。在实体—篇章子图上执行 PPR 迭代，获得全局重要性分布：

$$I(v_i) = (1 - d) + d \cdot \sum_{v_j \in B(v_i)} \frac{I(v_j)}{deg(v_j)} \tag{8}$$

其中 d 为阻尼因子，$B(v_i)$ 为指向 v_i 的邻居集合，$deg(v_j)$ 为节点 v_j 的出度。

排序输出。迭代收敛后，按篇章节点的重要性分数 $I(v \mid v \in V_p)$ 降序排序，取 Top-K 作为最终检索上下文，并与查询一起输入生成模型。

5 数据集介绍

本文使用四个数据集评估 LinearRAG，分为通用多跳问答基准与特定领域数据集两类。从每个数据集的验证集中抽取 1,000 个问题进行评测。

通用多跳问答基准。包括 HotpotQA、2WikiMultiHopQA。HotpotQA 约 97,000 个问答实例，需要从大量干扰文档中定位 2 个金标支持段落，考验跨文档推断与证据选择能力。2WikiMultiHopQA 约 192,000 个问题，通常需要在 2–4 篇文章间建立推理链，强调结构化跨文档推理与信息流连贯性。MuSiQue 约 25,000 个问答对，要求 2–4 步连续推理，重点检验长链条逻辑推理与上下文一致性。

特定领域数据集。Medical 来自 GraphRAG-Bench，基于 NCCN 临床指南构建，包含 4,076 个高专业问题，涵盖治疗方案、药物相互作用与诊断标准等内容。该数据集按难度划分为事实检索、复杂推理、上下文总结与创意生成四类任务，用于评估模型在专业领域下的检索质量与推理能力。

数据结构与组织。数据集包含文本分块与问题标注两类核心文件，字段结构如下：`chunks.json` 按列表存储的语料分块文本，单条记录为一个字符串，常以 `index: text` 形式保存，用于构建检索语料库。`questions.json` 按列表存储的问题对象，核心字段包括 `id`、`question_type`、`question`、`answer`、`source`，以及证据链字段 `evidence` 与 `evidence_relations`（均为三元组列表，形如 [head, relation, tail]），用于多跳证据评估与关系一致性分析。

各数据集的分块数量如下：2WikiMultiHop 为 658，HotpotQA 为 1311，Medical 为 225，MuSiQue 为 1354。

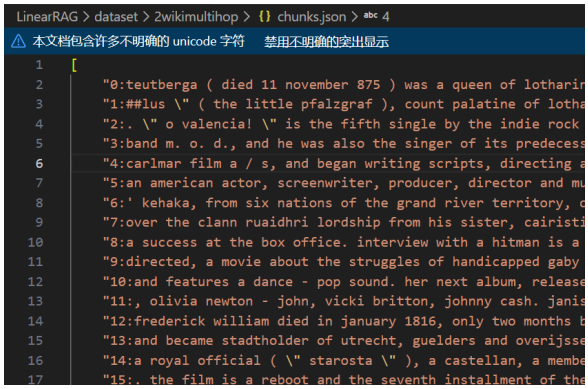


图 4: `chunks.json` 的数据结构示例

图 5: `questions.json` 的数据结构示例

6 实验结果展示与分析

本实验的生成模型采用阿里百炼的 Qwen-Flash，向量嵌入模型为 all-mpnet-base-v2。索引阶段的实体抽取使用轻量级 spaCy 工具包（通常基于 BERT 的模型）完成，保证构建成本可控并保持语义覆盖。

6.1 模型选择与硬件配置

本实验的生成模型采用阿里百炼的 Qwen-Flash，向量嵌入模型为 all-mpnet-base-v2。索引阶段的实体抽取使用轻量级 spaCy 工具包（通常基于 BERT 的模型）完成，保证构建成本可控并保持语义覆盖。显卡型号为 NVIDIA GeForce RTX 4080，总显存 12282 MiB；索引阶段显存占用为 5476 MiB / 12282 MiB，主要用于运行向量嵌入模型与 spaCy 实体识别。

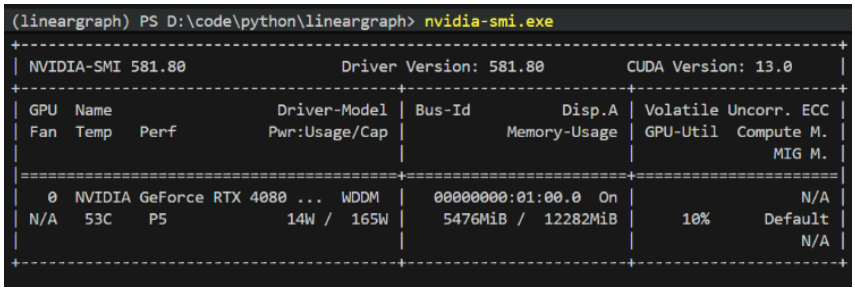


图 6: 索引建立阶段的显存占用情况

6.2 索引建立阶段

索引构建阶段耗时对比。以下耗时来自本次运行日志，单位为秒（s），对比不同数据集在各阶段的耗时。

阶段	2WikiMultiHop	HotpotQA	Medical	MuSiQue
段落向量生成	6.269	10.475	2.170	10.425
spaCy 实体识别	276.937	560.879	99.898	584.481
句子向量生成	14.294	28.193	1.566	29.430
实体向量生成	6.739	11.582	0.120	11.315
总索引耗时	304.899	612.284	103.773	636.908

表 1: 索引构建阶段耗时对比（单位：秒）

从表中可以看出，索引构建的主要瓶颈集中在 spaCy 实体识别阶段，其他向量生成与汇总阶段耗时明显更低。整体耗时与分块规模呈正相关：Medical 分块最少、总耗时最低，而 MuSiQue 与 HotpotQA 分块更多，总耗时显著升高，说明索引成本主要随语料规模线性增长。

由于 LinearRAG 不需要像传统 GraphRAG 那样进行三元组抽取，也不依赖大模型完成实体与关系抽取，因此索引建立阶段的耗时显著低于传统图检索方案。该结果与 LinearRAG 的线性可扩展设计一致，表明在较大规模语料下仍能保持可控的构建时间。

索引产物与文件结构。索引建立完成后生成以下文件：

- `passage_embedding.parquet`：段落向量库缓存，包含 `hash_id / text / embedding`，用于密集检索与段落相似度计算。

- `sentence_embedding.parquet`：句子向量库缓存，来源于 NER 过程中抽取的句子，用于实体—句子传播与相似度计算。
- `entity_embedding.parquet`：实体向量库缓存，包含所有识别到的实体，用于从问题实体出发的图检索与权重传播。
- `ner_results.json`：NER 结果缓存，包含 `passage_hash_id_to_entities` 与 `sentence_to_entities`，用于避免重复 NER 计算。
- `LinearRAG.graphml`：构建完成的图结构文件（igraph GraphML），节点包含 passage/entity，边包含 passage-entity 关联与相邻 passage 连接。

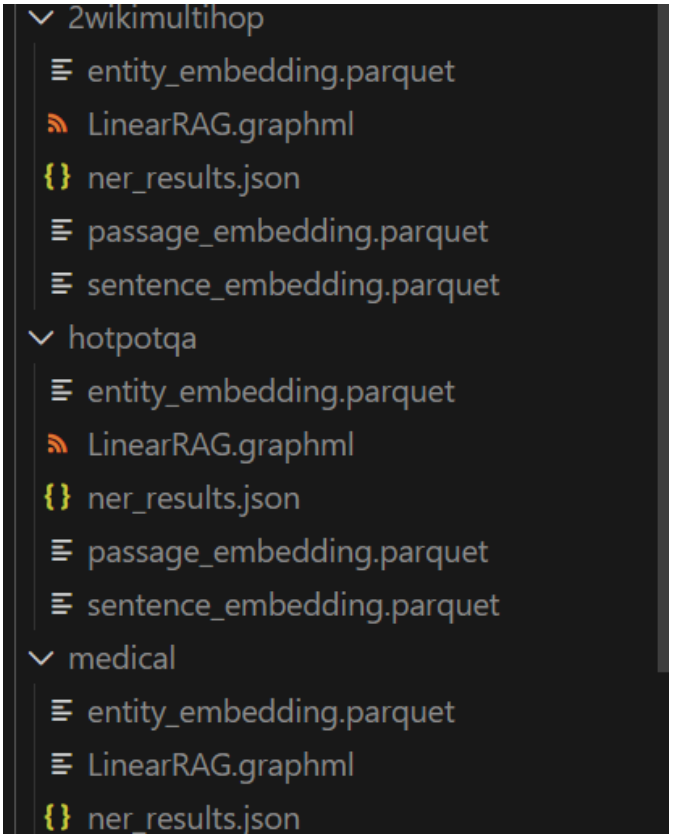


图 7: 数据集索引结构示意

6.3 检索与生成阶段

检索设置说明。当前使用 LinearRAG 默认配置，检索方式为 BFS 迭代图检索，关键参数如下：

- `use_vectorized_retrieval=False`：使用 BFS 迭代检索（非矩阵化向量检索）。
- `retrieval_top_k=5`：每个问题返回 5 个候选 passage。
- `max_iterations=3`：图传播最大迭代轮数。
- `top_k_sentence=1`：每轮仅保留相似度最高的 1 个句子参与传播。

- `iteration_threshold=0.5`：低于阈值的传播会被剪枝。

检索与生成阶段耗时对比如下（单位：秒）：

数据集	检索耗时	生成耗时
2WikiMultiHop	119.000	113.000
HotpotQA	150.854	111.680
Medical	84.431	419.185
MuSiQue	143.480	180.018

表 2: 检索与生成阶段耗时对比（单位：秒）

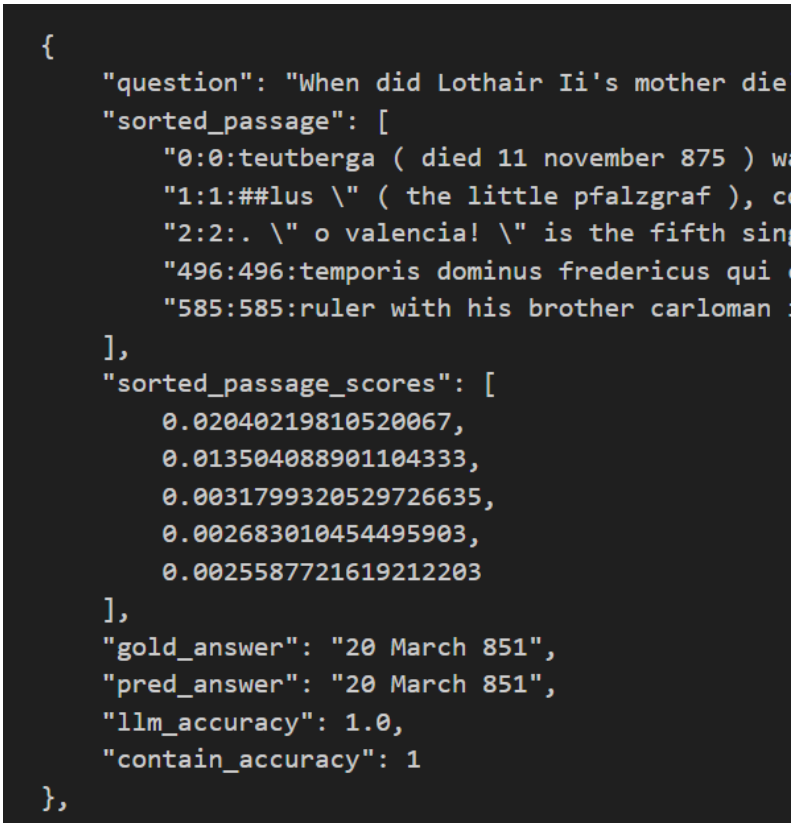


图 8: 检索与生成结果单个样本示例

字段说明：`question` 为输入问题文本；`sorted_passage` 为按相关性排序的检索段落列表；`sorted_passage_scores` 为对应的相关性分数；`pred_answer` 为模型生成答案；`gold_answer` 为数据集标准答案；`llm_accuracy` 为 LLM 判定正确性（1/0）；`contain_accuracy` 为基于字符串包含关系的快速判定（1/0）。

6.4 结果评估

本节使用两种指标评估生成结果：

- **LLM Accuracy**：将 `pred_answer` 与 `gold_answer` 交给 LLM 判断，仅允许返回 `correct`/`incorrect`。返回 `correct` 记为 1，否则记为 0，最后取平均值。
- **Contain Accuracy**：对答案进行归一化（小写、去标点、去冠词、压缩空白），判断 `gold_answer` 是否被 `pred_answer` 包含；包含记 1，否则记 0，最后取平均值。

各数据集的评估结果如下：

数据集	LLM Accuracy	Contain Accuracy
2WikiMultiHop	0.6530	0.7380
HotpotQA	0.6340	0.6550
Medical	0.6722	–
MuSiQue	0.3270	0.3040

表 3: 各数据集准确率对比

注：Medical 数据集的标准答案多为较长的描述性文本，Contain-Acc 依赖字符串包含匹配，难以合理衡量此类答案的质量，因此未记录该指标。

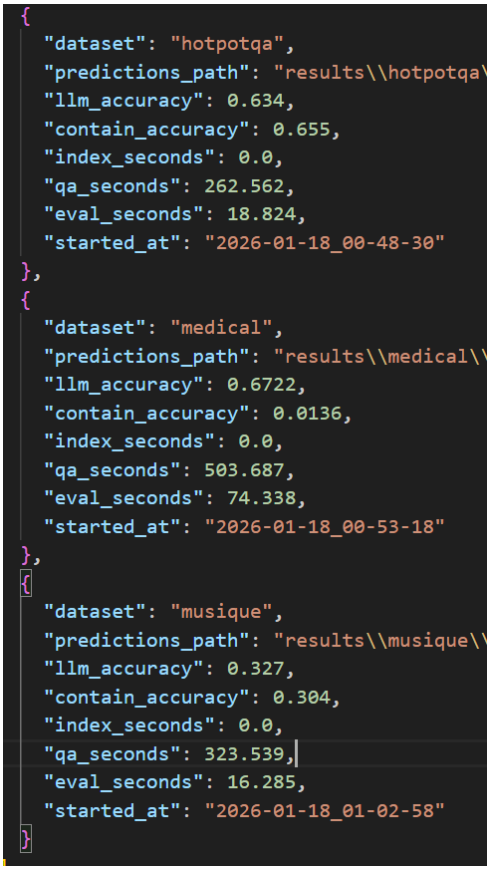


图 9: 评估结果

6.5 与论文实验结果对比

Method	Time (s)		Token Consumption ($\times 10^6$)		Accuracy
	Indexing	Retrieval (Avg.)	Prompt	Completion	
G-retriever	2745.94	11.487	6.05	2.26	36.85
RAPTOR	1323.57	0.062	0.81	0.03	46.10
E^2GraphRAG	534.60	**0.053**	0.78	0.08	46.20
LightRAG	4933.22	10.963	35.52	51.16	47.10
HippoRAG	936.00	1.461	3.05	0.98	63.00
GFM-RAG	1202.77	1.211	3.05	0.98	63.20
HippoRAG2	1147.01	1.694	4.98	1.22	58.85
LinearRAG (Ours)	**249.78**	0.093	**0**	**0**	**66.95**

图 10: 2WikiMultiHopQA 上的运行时间与 Token 消耗对比

该图展示研究团队对不同 GraphRAG 模型在 2WikiMultiHopQA 数据集上的运行时间与 Token 消耗对比。可以看出，本文复现的构建索引时长与其趋势一致，表明复现实验与原文设置具有一致性。

Method	HotpotQA		2Wiki		MuSiQue		Medical
	Contain-Acc.	GPT-Acc.	Contain-Acc.	GPT-Acc.	Contain-Acc.	GPT-Acc.	GPT-Acc.
Direct Zero-shot LLM Inference							
llama-8B	31.10	27.30	33.60	16.20	7.40	8.10	27.31
llama-13B	24.20	16.80	21.90	10.50	3.30	4.40	28.86
GPT-3.5-turbo	33.40	43.20	28.70	31.00	10.30	21.90	45.60
GPT-4o-mini	38.90	40.20	36.30	31.40	13.60	15.80	42.10
Vanilla Retrieval-Augmented-Generation							
Retrieval (Top-1)	46.30	49.10	36.60	31.70	17.80	21.10	48.01
Retrieval (Top-3)	53.00	56.00	44.90	39.70	25.10	27.50	59.07
Retrieval (Top-5)	55.70	58.60	48.60	43.00	26.10	29.60	61.68
Graph-based Retrieval-Augmented-Generation Methods							
KGP	61.50	60.90	31.60	30.00	25.60	30.10	54.22
G-retriever	42.20	40.60	46.60	27.10	14.40	15.50	50.36
RAPTOR	55.90	58.30	50.10	42.10	23.30	27.40	55.75
E^2GraphRAG	61.00	63.90	54.30	38.10	23.80	26.20	58.00
LightRAG	60.30	59.50	55.20	39.00	27.40	28.60	54.36
HippoRAG	57.00	59.30	66.10	59.90	29.30	24.10	55.04
GFM-RAG	62.70	65.60	66.80	59.60	29.90	34.60	56.07
HippoRAG2	62.90	64.30	62.70	55.00	31.00	35.00	60.77
LinearRAG (Ours)	**64.30**	**66.50**	**70.20**	**63.70**	**33.90**	**37.00**	**63.72**

图 11: LinearRAG 与基准模型在四个数据集上的性能对比

LinearRAG 原文结果表明其在多项数据集上优于基准模型。对比本文复现实验的准确率，可以看出整体趋势与原文基本吻合，说明复现结果具有较好一致性。

7 总结

本文围绕 LinearRAG 的阅读、实现理解与复现实验展开，系统梳理了 RAG 与 GraphRAG 的关键问题，并在源码层面明确了 LinearRAG 的核心流程与创新机制。实验部分完成了索引构建、检索生成与评估分析，记录了不同数据集的运行耗时、显存占用与准确率指标，验证了线性可扩展的索引成本与与原文一致的性能趋势。整体结果表明，LinearRAG 在避免三元组抽取与大模型关系提取的前提下，仍能保持稳定的多跳检索能力与可控的构建开销。项目代码与材料已整理于 GitHub 与 百度网盘。

8 参考文献

参考文献

[1] Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., ... & Larson, J. (2024). **From local to global: A graph rag approach to query-focused summarization**. *arXiv preprint arXiv:2404.16130*.

[2] Guo, Z., Xia, L., Yu, Y., Ao, T., & Huang, C. (2024). **LightRAG: Simple and Fast Retrieval-Augmented Generation**. *arXiv preprint arXiv:2410.05779*.

[3] Xiang, Z., Wu, C., Zhang, Q., Chen, S., Hong, Z., Huang, X., & Su, J. (2025). **When to Use Graphs in RAG: A Comprehensive Analysis for Graph Retrieval-Augmented Generation**. *arXiv preprint arXiv:2506.05690*.