ChatGPT

GraphRAG и LightRAG: обзор подходов

- **GraphRAG** – структура «узлы–рёбра»: вместо простого поиска по векторам строится **граф знаний** из сущностей и отношений в корпусе. Затем граф разбивается на сообщества (кластеризуется, например, алгоритмом Leiden [1]), для которых генерируются краткие обобщения. При запросе каждая «общественная» сводка используется для частичного ответа, а затем эти ответы объединяются в итоговый ответ [2]. Такой подход демонстрирует **значительный рост полноты и разнообразия** ответов по сравнению с наивным RAG [2]. Как отмечают Han et al., графы «по своей природе кодируют огромный объём разнородной и реляционной информации», что делает их «золотым ресурсом» для RAG [3]. GraphRAG позволяет найти скрытые связи между разрозненной информацией: он «использует знания в виде графа… чтобы захватить не только больше точек данных, но и их взаимосвязи», тем самым получая более точные результаты [4]. Это особенно важно для многопереходного (multi-hop) вывода: векторный RAG часто возвращает изолированные фрагменты, а GraphRAG – связанные подграфы, что повышает качество объяснений и обоснованность ответов [4] [5].
- **LightRAG** – упрощённый и быстрый RAG с графом. LightRAG также строит граф знаний, но акцентирует внимание на эффективности. Он реализует *двухуровневый* поиск: при разметке текста извлекаются сущности и отношения, после чего используются как **низкоуровневые ключи** (точный поиск по сущностям/отношениям) и **высокоуровневые ключи** (широкие темы и соседние узлы) [6] [7]. Интеграция графовой структуры с векторными представлениями позволяет LightRAG «эффективно извлекать связанные сущности и их отношения», заметно **сокращая время ответа** при сохранении релевантности контекста [8]. По сравнению со стандартным RAG, LightRAG обеспечивает более высокую точность извлечения и снижает задержку примерно на 20–30 мс [9]. Таким образом, LightRAG выгоден там, где критична скорость и обновляемость индекса (он поддерживает инкрементальные обновления графа без полной перестройки) [8] [9].
- **Сравнение GraphRAG vs LightRAG:** блог Tanya Aravind подчёркивает, что GraphRAG даёт **лучшее понимание отношений** и обеспечивает глубину анализа (примерно +10% точности на задачах отношения) за счёт обхода графа, тогда как LightRAG жертвует частью точности ради скорости и простоты [9]. Решение о выборе зависит от задачи: GraphRAG подходит для сложных взаимосвязанных доменов, LightRAG – для сценариев, где важна скорость и ресурсная экономия [10] [11].

Графовая архитектура RAG и оптимизация поиска

- **Построение и запрос графа:** GraphRAG обычно строит графовую индексацию с помощью LLM (например, GPT-4) для извлечения сущностей и отношений из текста [12]. Затем для каждого сообщества в графе генерируются лексические сводки. При запросе GraphRAG формирует материал для LLM через три основных режима: глобальный поиск (используя сводки сообществ), локальный поиск (расширение по соседям сущности) и «DRIFT»-поиск (специальные шаблоны) [13]. Обход графа может осуществляться через формальные запросы (например, Cypher: `(node)-[REL]->(node)`) [14] или с помощью графовых

маршрутизирующих рекурсивных алгоритмов. Например, в модели **KG2RAG** после получения топ-K семантически релевантных фрагментов они извлекают подграф знаний и применяют обход (BFS) по смежным сущностям, расширяя выборку другими связанными фрагментами [15] [16]. Этот двойной этап («топ-K семантика + расширение по графу») обеспечивает большую **разнообразность и связанность** найденной информации [15] [16]. Аналогично, подход **SemRAG** сначала выполняет семантический «chunking» (поиск по embedding) для топ-K сообществ, а затем строит из извлечённой информации иерархический граф, который захватывает отношения между сущностями, улучшая точность и релевантность [17] [18].

- **Оптимизация обхода графа:** современные работы предлагают ускорить графовый RAG несколькими способами. Например, **LeanRAG** строит иерархический KG, группируя сходные сущности, а затем осуществляет «снизу-вверх» поэтапный поиск – начиная с наиболее релевантных узлов и последовательно расширяя контекст по релевантным путям – что значительно сокращает избыточность поиска (~46% меньше дубликатов) [19]. **ReMindRAG** использует LLM для динамического управления обходом (подходы node exploration/exploitation) и даже «воспроизведение памяти» для ребалансировки при обходе, что улучшает эффективность и качество при меньших затратах [20]. Кроме того, вместо перебора всех фрагментов можно сохранять граф в специализированной СУБД (например, Neo4j) – современные графовые БД имеют встроенные алгоритмы обхода (BFS/DFS) и поиска компонент связности. Так, Neo4j позволяет хранить данные в виде узлов/рёбер, поддерживает запросы Cypher/SPARQL и автоматически вычисляет метрики (например, компоненты связности, кластеризацию) [21]. Это обеспечивает лёгкое склеивание разрозненных источников (напр., микросервисы, задачи и т.д.) в единый граф [22] [21].

Комбинация семантического и графового поиска

- **Гибридные стратегии:** объединение семантических и графовых методов – ключевая идея современных RAG. Стандартный RAG возвращает топ-K документов по косинусной близости эмбеддингов [23] [24], тогда как GraphRAG начинает с семантики, а затем «проходит» по графу отношений. Многие решения именно так и работают: сначала выполняют *семантический поиск* топ-K (по embedding), а затем «расширяют» результаты по связям. Например, **KG2RAG** после семантического поиска извлекает релевантный подграф и выполняет обход (BFS), чтобы добавить к результатам смежные элементы [15] [16]. SemRAG аналогично использует топ-K «семантических» сообществ, а затем структурирует их в графовые иерархии [17]. Такие комбинированные схемы обеспечивают покрытие как ближайших семантически подходящих фрагментов, так и связанных отношений, что улучшает полноту извлечения.
- **Эффективность подхода:** эксперименты подтверждают выгоду графовых RAG. GraphRAG-подходы демонстрируют более **точные и объяснимые** ответы по сравнению с обычным RAG [25] [5]. Например, в HotpotQA KG2RAG превзошёл классический RAG и свои аналоги по качеству ответов и релевантности извлечений [26]. LightRAG доказывает свою эффективность: он «значительно улучшает точность поиска и сокращает время ответа» [8] [9]. Одновременно графовые обходы могут быть более затратными по времени: как замечено, GraphRAG обеспечивает «наивысшую реляционную точность, но с удвоенным временем поиска» по сравнению с плоским RAG [27]. Поэтому при внедрении важно искать компромиссы – например, **LightRAG** убирает необходимость полной перестройки индекса и использует оптимизированные структуры, чтобы снизить накладные расходы [8].

Выводы и перспективы

- Использование **графовых структур** в RAG позволяет улавливать связи между сущностями и отвечать на комплексные вопросы, для которых векторный RAG неэффективен [3] [5]. Согласно последним исследованиям, гибридные схемы («топ-K семантика + топ-K отношения» [15]) дают наилучший эффект: сначала извлекается наиболее релевантная по смыслу информация, затем она расширяется по графу.
- Для оптимизации обхода графов предлагаются специализированные алгоритмы и системы. Например, **графовые СУБД** (Neo4j, JanusGraph и др.) уже включают алгоритмы поиска компонент и путей, что ускоряет агрегацию данных; также применяются алгоритмы кластеризации (Leiden, Louvain) для предварительного разбиения графа [1].
- Актуальны разработки, строящие **интеллектуальный графовый движок** для RAG: LLM-наставленные обходы (ReMindRAG [20]), иерархическое агрегационное моделирование (LeanRAG [19]) и др. Такие методы позволяют «уменьшить избыточность» и «оптимизировать стоимость», сохраняя высокое качество генерации ответов.
- В целом, графовые RAG-решения находятся в активной стадии исследований. Изучите последние работы и блоги (см. перечисленные источники) по GraphRAG и LightRAG, чтобы выбрать подход, пригодный для вашей задачи. В частности, применительно к облачным системам полезно обратить внимание на то, как выделять сущности и связи из имеющихся данных и как эффективно хранить их в графовой БД для быстрого обхода и обновления.

Источники: современные исследования и обзоры по RAG с графами [3] [2] [8] [4] [15], а также технические блоги по GraphRAG и LightRAG [25] [5] [9], подробно описывающие архитектуры и эффективность упомянутых подходов. Каждая цитата выше ссылается на эти источники.

---

[1] [12] [13] Welcome - GraphRAG
https://microsoft.github.io/graphrag/

[2] [2404.16130] From Local to Global: A Graph RAG Approach to Query-Focused Summarization
https://ar5iv.labs.arxiv.org/html/2404.16130

[3] [2501.00309] Retrieval-Augmented Generation with Graphs (GraphRAG)
https://arxiv.org/abs/2501.00309

[4] [14] [25] What Is GraphRAG?
https://neo4j.com/blog/genai/what-is-graphrag/

[5] [21] [22] [23] [24] Using a Knowledge Graph to implement a RAG application
https://neo4j.com/blog/developer/rag-tutorial/

[6] [8] [2410.05779] LightRAG: Simple and Fast Retrieval-Augmented Generation
https://ar5iv.labs.arxiv.org/html/2410.05779

[7] [9] [10] [11] [27] Understanding GraphRAG vs. LightRAG: A Comparative Analysis for Enhanced Knowledge Retrieval - Maarga Systems
https://www.maargasystems.com/2025/05/12/understanding-graphrag-vs-lightrag-a-comparative-analysis-for-enhanced-knowledge-retrieval/

[15] [16] [26] Knowledge Graph-Guided Retrieval Augmented Generation
https://arxiv.org/html/2502.06864v1

17 18 SemRAG: Semantic Knowledge-Augmented RAG for Improved Question-Answering Citation: Kezhen Zhong. SEMRAG. Pages.... DOI:000000/11111.
https://arxiv.org/html/2507.21110v1

19 LeanRAG: Knowledge-Graph-Based Generation with Semantic Aggregation and Hierarchical Retrieval
https://arxiv.org/html/2508.10391v1

20 [2510.13193] ReMindRAG: Low-Cost LLM-Guided Knowledge Graph Traversal for Efficient RAG
https://arxiv.org/abs/2510.13193