Introduction to Graph RAG (Retrieval Augmented Generation)

Yogesh Haribhau Kulkarni

YHK

About Me

Yogesh Haribhau Kulkarni

Bio:

- 20+ years in CAD/Engineering software development
- Got Bachelors, Masters and Doctoral degrees in Mechanical Engineering (specialization: Geometric Modeling Algorithms).
- Currently doing Coaching in fields such as Data Science, Artificial Intelligence Machine-Deep Learning (ML/DL) and Natural Language Processing (NLP).
- Feel free to follow me at:
  - Github (github.com/yogeshhk)
  - LinkedIn (www.linkedin.com/in/yogeshkulkarni/)
  - Medium (yogeshharibhaukulkarni.medium.com)
  - Send email to yogeshkulkarni at yahoo dot com

Office Hours: Saturdays, 2 to 5pm (IST); Free-Open to all; email for appointment.

YHK

Retrieval Augmented Generation (RAG)

Quiz

- How to do domain adaptation? ie
- How to build custom LLM solution? meaning
- How can LLM based chatbot answer from my data?

Answers??

Need for Domain Adaptation

- Industrial settings prioritize cost, privacy, and reliability of solutions.
- LLMs are prone to hallucinations, factual errors, and looping.
- LLMs need to work on own/private/streaming/latest data
- Speed/latency/efficiency Concerns
- Solution: To build LLM from scratch?? needs??
- Extensive training data, high computational resources and $$$$

So, what to do?

YHK

Solutions

- Few Shots
- RAG
- Fine-tuning

YHK

Why RAG?

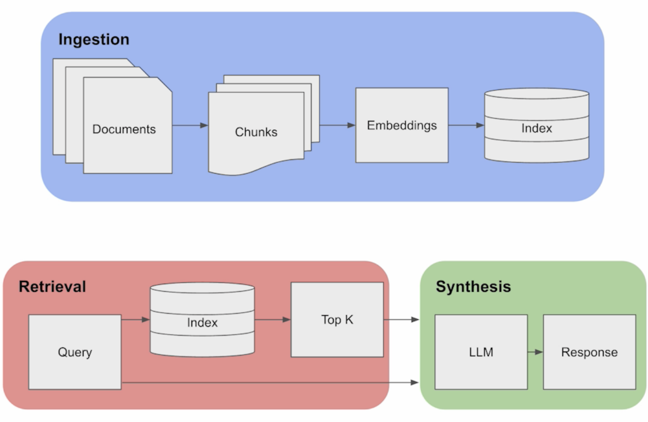
- **Unlimited Knowledge:**
  - RAG enables access external sources, surpassing limitations of its data.
  - Allows exploration of proprietary documents and internet searches
- **Easier to Update/Maintain:**
  - Offers a cost-effective way to update and maintain
  - Building a knowledge base minimizes ongoing maintenance financial burden.
- **Confidence in Responses:**
  - Enhances confidence by providing extra context for more accurate responses.
  - Practical boost to overall intelligence in generating responses.
- **Source Citation:**
  - RAG provides access to sources, improving transparency in LLM responses.
  - A step towards building trust in LLM systems.
- **Reduced Hallucinations:**
  - RAG-enabled LLMs exhibit reduced creative misfires.
  - Solid foundation of information keeps models focused and grounded.

Important Questions

- Does the use case require external data access?
- Does the use case require changing foundation model style?
- Does the use case require addressing hallucinations?
- Is labeled training data available?
- Is citing the source of information important?
- How critical is system latency?
- What are the cost implications?
- What are the scalability requirements?
- Do we have the necessary expertise?

Retrieval Augmented Generation (RAG) Workflow

RAG Components



(Ref: [Week 4] Retrieval Augmented Generation -Aishwarya Naresh Reganti)

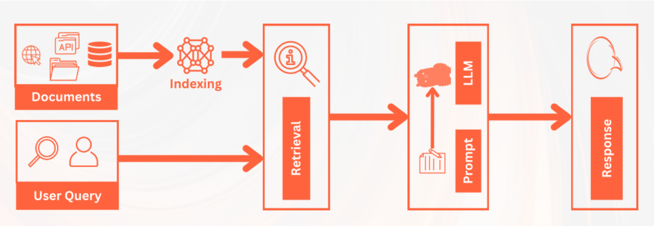
RAG Components

- Ingestion:
  - Documents undergo segmentation into chunks, and embeddings are generated from these chunks, subsequently stored in an index.
  - Chunks are essential for pinpointing the relevant information in response to a given query, resembling a standard retrieval approach.
- Retrieval:
  - Leveraging the index of embeddings, the system retrieves the top-k documents when a query is received, based on the similarity of embeddings.
- Synthesis:
  - Examining the chunks as contextual information, the LLM utilizes this knowledge to formulate accurate responses.

Naive RAG Approach

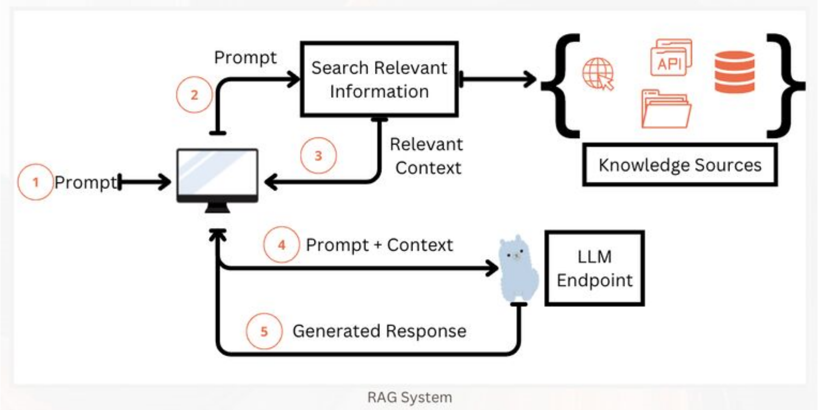
Simply retrieve texts and provide to language model as additional context during generation.

- Concept: Retrieve relevant documents from an external corpus before generation.
- Steps: 1. Input query/topic. 2. Retrieve documents. 3. Generate summary/response based on retrieved documents.
- Pros: Simple and effective for factual tasks.
- Cons: May lack fluency and originality due to heavy reliance on retrieved text.



(Ref: Progression of RAG Systems - Abhinav Kimothi)

RAG Architecture



(Ref: RAG Architecture -Abhinav Kimothi)

YHK

RAG Workflow

1. User writes a prompt or a query that is passed to an orchestrator
2. Orchestrator sends a search query to the retriever
3. Retriever fetches the relevant information from the knowledge sources and sends back
4. Orchestrator augments the prompt with the context and sends to the LLM
5. LLM responds with the generated text which is displayed to the user via the orchestrator

Two pipelines become important in setting up the RAG system. The first one being setting up the knowledge sources for efficient search and retrieval and the second one being the five steps of the generation.

Indexing Pipeline
Data for the knowledge is ingested from the source and indexed. This involves steps like splitting, creation of embeddings and storage of data.

Generation Pipeline
This involves the actual RAG process which takes the user query at run time and retrieves the relevant data from the index, then passes that to the model

(Ref: RAG Architecture -Abhinav Kimothi)

Challenges in Naive RAG

- Retrieval Quality
  - Low Precision leading to Hallucinations/Mid-air drops
  - Low Recall resulting in missing relevant info
  - Outdated information
- Augmentation
  - Redundancy and Repetition when multiple retrieved documents have similar information
  - Context Length challenges
- Generation Quality
  - Generations are not grounded in the context
  - Potential of toxicity and bias in the response
  - Excessive dependence on augmented context

(Ref: Progression of RAG Systems - Abhinav Kimothi)

YHK

RAG vs SFT (Supervised Fine - Tuning)

RAG & SFT: Complementary Techniques

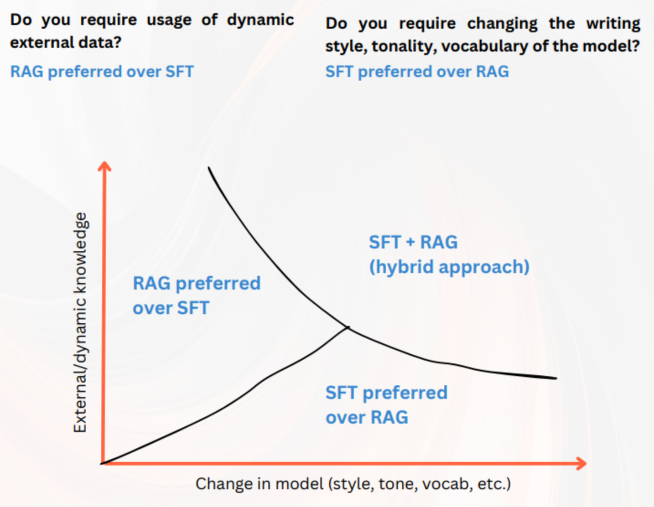
RAG Features

- Connects to dynamic external data sources.
- Reduces hallucinations.
- Increases transparency in source of information.
- Works well with very large foundation models.
- Does not impact style, tone, vocabulary.

SFT Features

- Changes style, vocabulary, tone of foundation model.
- Can reduce model size.
- Useful for deep domain expertise.
- May not address hallucinations.
- No improvement in transparency (black box models).

Important Use Case Considerations



(Ref: Generative AI with Large Language Model - Abhinav Kimothi)

Other Considerations

- **Latency:** RAG introduces inherent latency due to search and retrieval.
- **Scalability:** RAG pipelines are modular and scalable; SFT requires retraining.
- **Cost:** Both methods require upfront investment; costs vary.
- **Expertise:** RAG pipelines are moderately simple with frameworks; SFT needs deep understanding and training data creation.

Applications

Applications and Use Cases

- Code generation in software development
- Creative writing and storytelling
- Educational material generation
- Personalized product descriptions
- many many more . . .

Open Source Resources and Tools

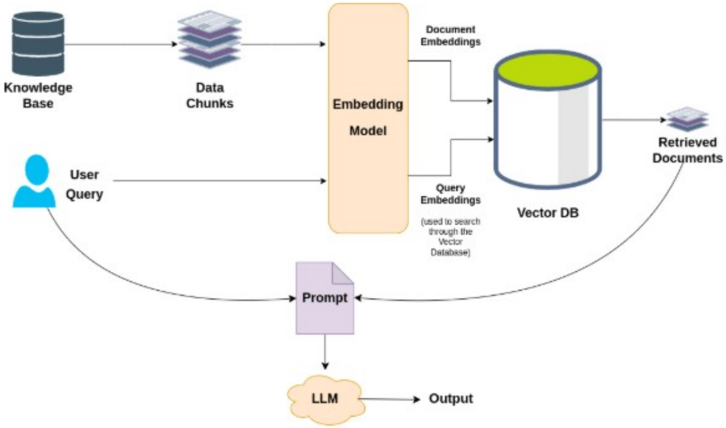
- LangChain, LlamaIndex like frameworks
- Transformers library, Hugging Face model hub
- Datasets and evaluation benchmarks

Introduction to Graph RAG

Why Graph RAG?

- Language models struggle with factual accuracy and real-world knowledge.
- Retrieval-Augmented Generation (RAG) improves accuracy using external text data.
- Traditional RAG has limitations in context understanding and scalability.
- GraphRAG leverages knowledge graphs for better retrieval and response generation.

Overview of Traditional RAG



(Ref: GraphRAG: The Practical Guide for Cost-Effective Document Analysis with Knowledge Graphs -Jaykumaran)

Limitations of Traditional RAG

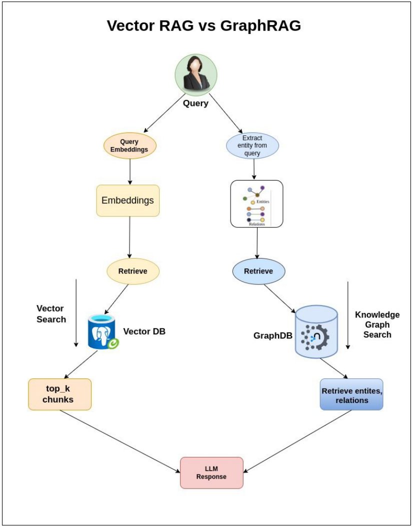
- Answer has to be in the chunk fully, cannot be across the chunks.
- Cannot have multiple chunks compose the final answer.
- Documents are treated as isolated entities.
- Lacks deep semantic understanding as multi-hop answers
- Slower retrieval with increasing data volume.
- Lacks global context over the entire data corpus.
- Inefficient for complex reasoning across multiple documents.
- Hard to trace the source of retrieved information.

What is GraphRAG?

R

AG on Graph: combines traditional RAG techniques with graph-based knowledge representations to leverage structural relationships between entities.

Vector RAG vs GraphRAG



(Ref: GraphRAG: The Practical Guide for Cost-Effective Document Analysis with Knowledge Graphs -Jaykumaran)

Traditional RAG vs GraphRAG

Feature	**Traditional RAG**	**GraphRAG**
Data Representation	Flat Vectors	Knowledge Graph
Query Scope	Local context	Global Reasoning
Scalability	Low	High
Citation Transparency	Low	High (Traceable sources)
Response Coherence	Fragmented	Relevant and Context-Rich

YHK

How GraphRAG Solves the Problem

- GraphRAG combines graph structures with vector search.
- Traverses multi-hop connections to infer relationships.
- Offers more reliable responses than vector-only RAG.

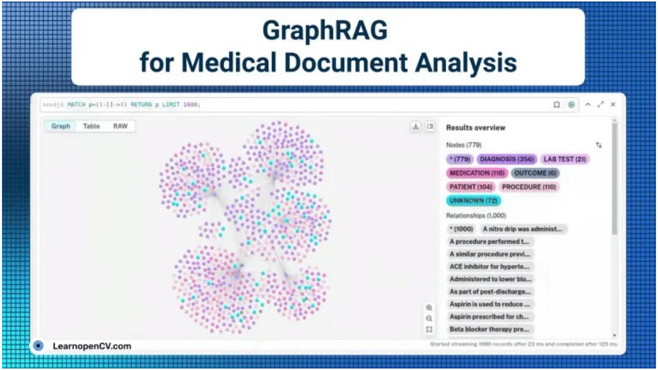
Graph-based RAG helps agentic AI make human-like decisions. – May Habib, CEO of Writer.com

Different Approaches

- English Query is converted to GraphQL or Cypher query then calls GraphDB to fetch the context/answers. Uses few-shots or specialized parsers or fine-tuned LLMs for the conversion (Neo4j way).
- Knowledge Graph is indexed, then English query then sent to index (which fetches, relevant triplets, also node's data), , KG may be created by LLM Graph Transformer and put in NetworkX object, you can populate this externally as well. (Langchain/Llamaindex way)
- From documents, populates the Knowledge Graph first, indexes it, then brings the relevant context from the index (Microsoft way)
- Contextual Subgraph/Path-based Retrieval Approach: Extracts reasoning paths or connected subgraphs for the input query (mostly academic research)

Example: Medical Query Challenge

- Query: How does Medication A in Patient Record 1 affect Condition B in Patient Record 2?
- LLM needs to infer relationships across multiple records.
- Standard RAG struggles with such complex dependencies.
- Scaling this to millions of patient records is infeasible.



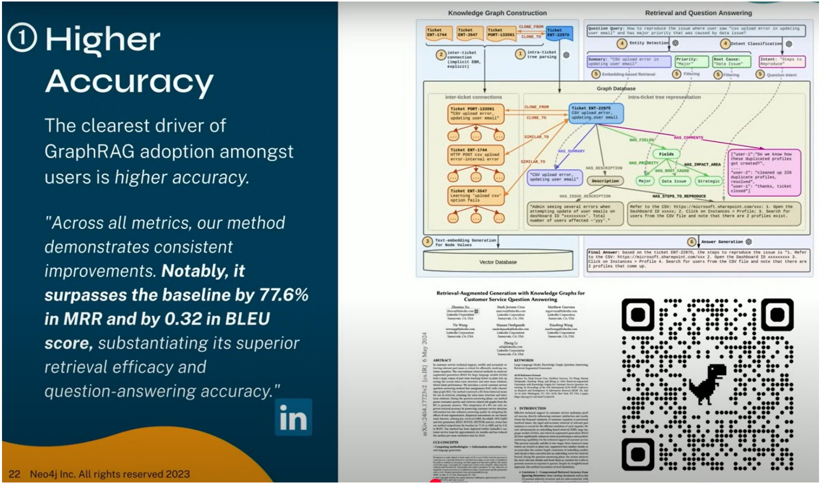
(Ref: GraphRAG: The Practical Guide for Cost-Effective Document Analysis with Knowledge Graphs -Jaykumaran)

YHK

Advantages of GraphRAG

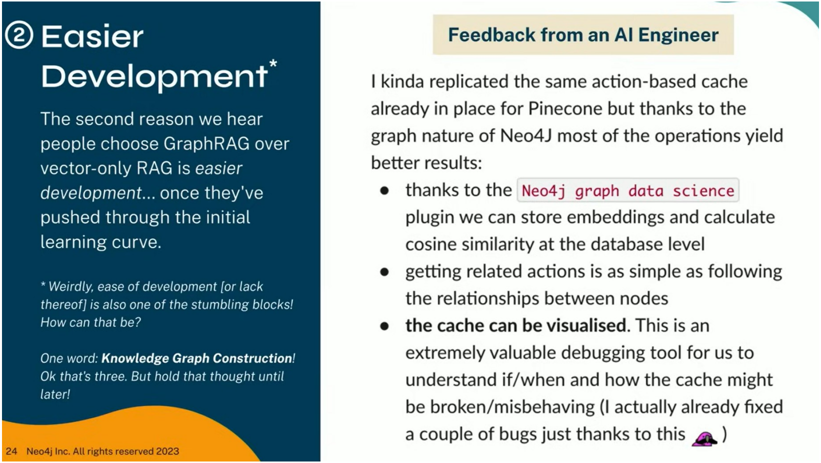
- Structured knowledge retrieval, so far less hallucinations, thus more accuracy.
- Context-aware and efficient retrieval.
- Handles complex queries effectively.
- Provides explainability and transparency.

Advantages of GraphRAG



(Ref: GraphRAG: The Marriage of Knowledge Graphs and RAG: Emil Eifrem)

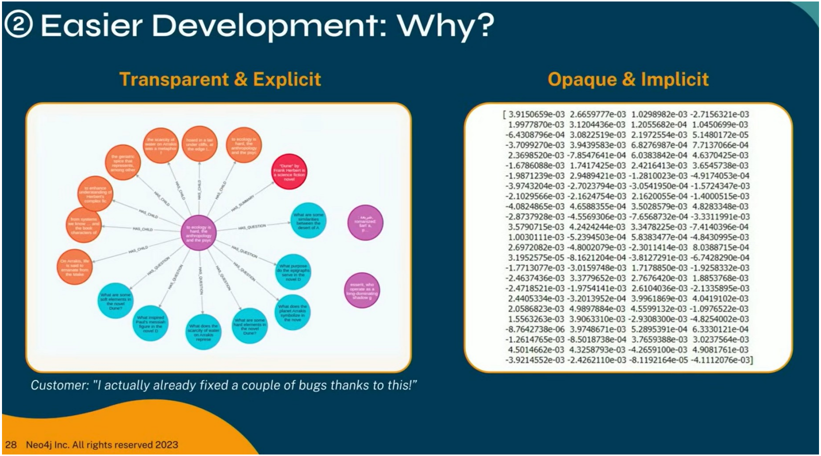
Advantages of GraphRAG



(Ref: GraphRAG: The Marriage of Knowledge Graphs and RAG: Emil Eifrem)

YHK

Advantages of GraphRAG



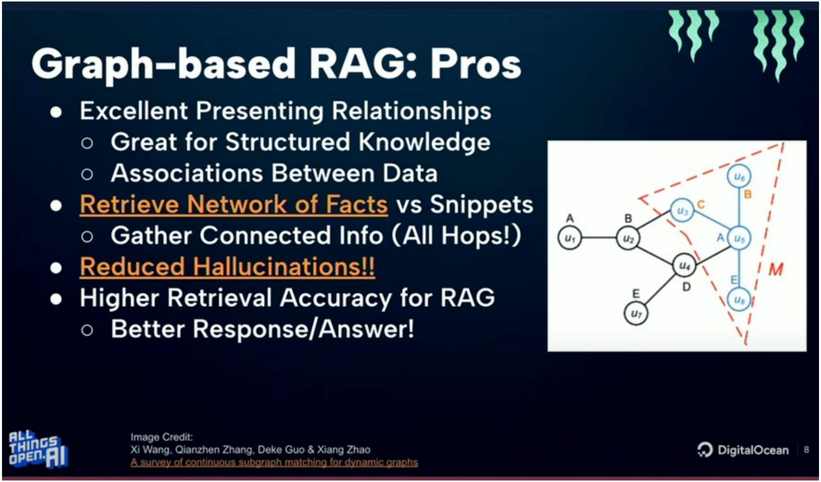
(Ref: GraphRAG: The Marriage of Knowledge Graphs and RAG: Emil Eifrem)

YHK

Advantages of GraphRAG

(Ref: GraphRAG: The Marriage of Knowledge Graphs and RAG: Emil Eifrem)

Advantages of GraphRAG



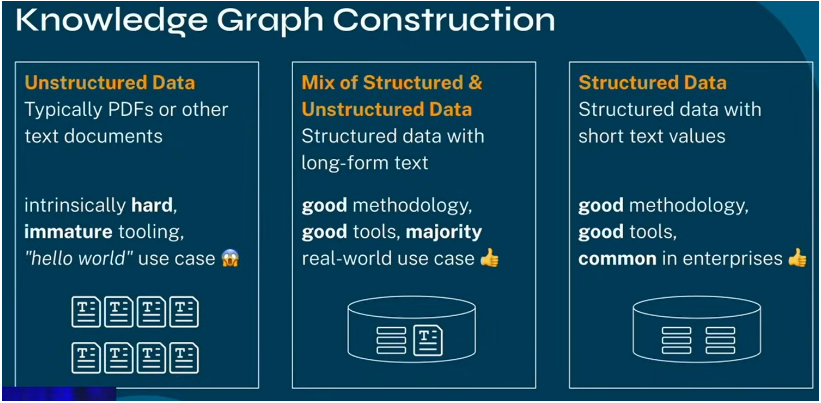
(Ref: Leveraging Knowledge Graphs for RAG: A Smarter Approach to Contextual AI Applications)

YHK

Challenges of GraphRAG

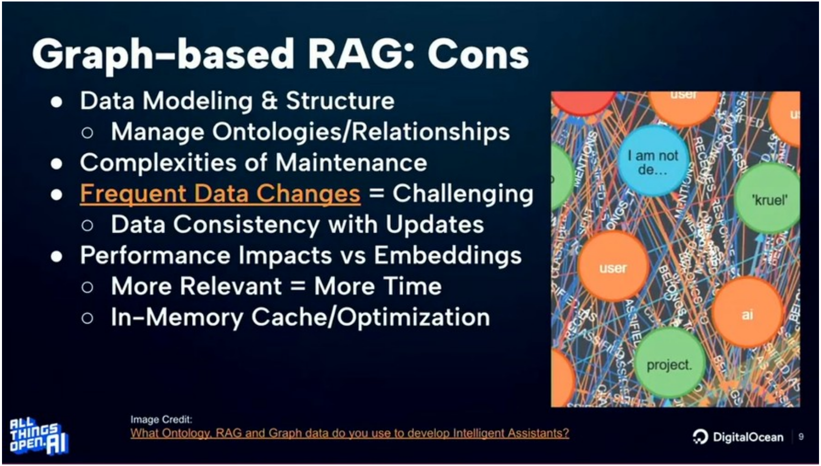
- **Complex Knowledge Graph Construction:** Requires sophisticated NLP techniques.
- **Data Dependency:** Performance relies on input data quality.
- **Scalability Issues:** Large graphs require significant computational resources.

Challenges of GraphRAG



(Ref: GraphRAG: The Marriage of Knowledge Graphs and RAG: Emil Eifrem)

Challenges of GraphRAG

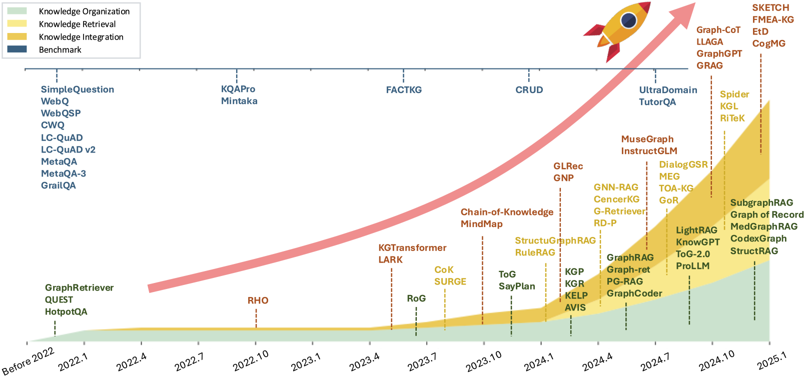


(Ref: Leveraging Knowledge Graphs for RAG: A Smarter Approach to Contextual AI Applications)

Applications of GraphRAG

- **Healthcare:** Assists in diagnoses and treatment decisions.
- **Banking:** Detects fraudulent transactions using knowledge graphs.
- **Customer Service :** quickly answer customer questions from thousands of pages of policy documentation
- **Recommendations:** understand customer behavior and preferences better, to provide personalized services.
- **Supply Chain:** product recall and associated quality control checking, internal documentation search

Trend of GraphRAG Research



(Ref: Awesome-GraphRAG (GraphRAG Survey))

Conclusion

- GraphRAG enhances traditional RAG models using structured knowledge.
- Improves accuracy, context-awareness, and efficiency.
- Useful in various domains like healthcare and banking.
- A promising approach for future AI-powered knowledge retrieval.

YHK

Implementation using Microsoft Graph RAG

Source: Microsoft Research (https://microsoft.github.io/graphrag/) and GitHub repository (https://github.com/microsoft/graphrag)

Introduction to Microsoft's GraphRAG

- GraphRAG is Microsoft's framework for RAG (Retrieval-Augmented Generation) on graphs
- Combines knowledge graphs with LLMs for enhanced context-aware responses
- Addresses limitations of traditional RAG by leveraging graph relationships
- Provides more structured, accurate, and explainable answers
- Open-sourced by Microsoft; code is available on GitHub.
- Works with both proprietary (e.g., GPT-4) and local models (e.g., LLaMA3).
- Aim: overcome limitations of traditional RAG systems.

Traditional RAG Overview

- Converts documents into vector embeddings via chunking.
- Stores chunks in a vector database.
- At query time: compute query embedding, retrieve similar chunks.
- Combines query + retrieved chunks to generate final answer.

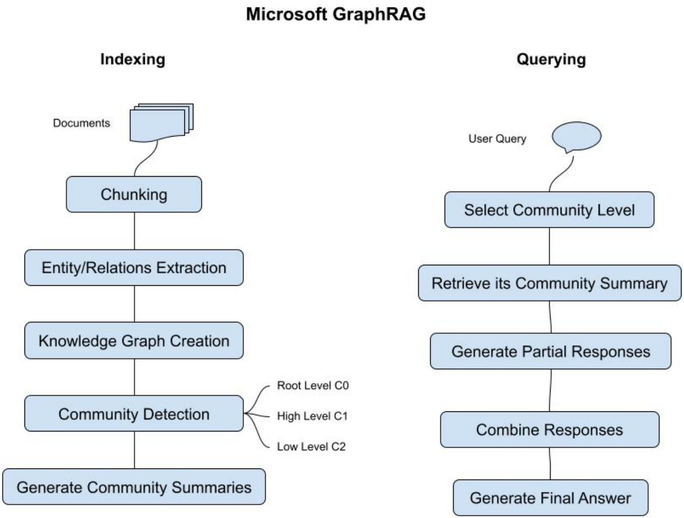
Limitations of Traditional RAG

- Lacks holistic understanding of entire document set.
- Retrieval efficiency drops as data size increases.
- Complex integration of external knowledge sources.

GraphRAG: High-Level Flow

- Two phases: indexing and querying.
- Entities and relationships extracted during indexing.
- Builds a knowledge graph, then forms communities of related entities.
- Summaries are generated at different levels of detail.

GraphRAG: High-Level Flow



Indexing Phase in GraphRAG

- Chunk documents and extract entities and relationships.
- Build a knowledge graph from this information.
- Detect communities of closely related entities.
- Generate summaries at multiple community levels.

Query Phase in GraphRAG

- Select community level based on query detail needed.
- Retrieve relevant summaries from selected community.
- Combine responses from multiple communities if needed.
- Final answer is composed from merged summaries.

Setting Up GraphRAG Locally

- Create and activate a Conda environment.
- Install the package:

```
conda create -n GraphRag python=3.10
conda activate GraphRag
or
pip install graph-rag

```

Running the Indexing Process

- Organize input data in a folder.
- Initialize configurations and set environment variables.
- Create index
- Requires API key and settings in 'settings.yml'.
- Configure model (e.g., GPT-4o) and embedding model in 'settings.yml'.
- Chunk size, overlap, and base API path are customizable.
- Prompt templates control how entities and summaries are extracted.

```
python -m graph_rag.index

```

Querying with GraphRAG

- Run queries
- Specify method: ‘global‘ for high-level themes, ‘local‘ for specific details.
- Responses cite sources and use graph structure to enhance relevance.

```
python -m graph_rag.query
```

YHK

Conclusions

GraphRAG: Evolution of Knowledge Graphs

- Earlier, KGs were built using statistical and NLP techniques.
- GraphRAG scales efficiently by using LLMs for entity extraction.
- Entities in KG are nodes, linked by edges encoding relationships.

Key Features of GraphRAG

- Maintains hierarchical communities preserving local and global insights.
- Ensures source traceability down to the node level for citations.
- Aggregates information from multiple sources, reducing bias.
- Focuses on meaningful nodes, filtering out irrelevant information.

Advantages of Knowledge Graphs

- Connects information from multiple sources for deeper insights.
- Boosts retrieval accuracy and enables multi-hop reasoning.
- Enhances LLM responses by integrating structured relationships.

GraphRAG Cost Considerations

- GraphRAG can be expensive—570 LLM requests vs. 25 embeddings.
- Over 1 million tokens processed cost around $7 (using GPT-4).
- Cost increases with corpus size and LLM choice.

Alternatives and Ecosystem

- Other GraphRAG implementations:
- **LlamaIndex** - Graph Query Engine.
- **Neo4j** - Native GraphRAG integration.
- Each offers different trade-offs in performance and cost.

Case Studies & Applications

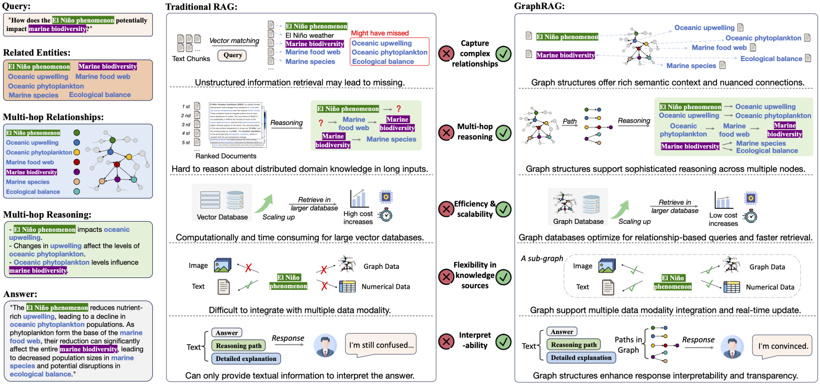
- Enterprise knowledge management
- Scientific literature analysis
- Customer support and troubleshooting
- Regulatory compliance and legal research
- Healthcare and biomedical information systems
- Financial analysis and risk assessment

YHK

Conclusions

Challenges in Standard RAG

- **Lack of Explainability:** Hard to trace the source of retrieved information.
- **Local Window:** Limited chunk-level context leads to incomplete responses.
- **Scalability Issues:** Struggles with large-scale medical and legal corpora.
- **Loss of Structural Relationships:** Ignores hierarchical relationships in data.



Query:
"How does the El Niño phenomenon potentially impact marine biodiversity?"
Related Entities:
El Niño phenomenon
Marine biodiversity
Oceanic upwelling
Marine food web
Oceanic phytoplankton
Marine species
Ecological balance
Multi-hop Relationships:
El Niño phenomenon
Oceanic upwelling
Oceanic phytoplankton
Marine food web
Marine biodiversity
Marine species
Ecological balance
Multi-hop Reasoning:
- El Niño phenomenon impacts oceanic upwelling.
- Changes in upwelling affect the levels of oceanic phytoplankton.
- Oceanic phytoplankton levels influence marine biodiversity.
Answer:
"The El Niño phenomenon reduces nutrient-rich upwelling, leading to a decline in oceanic phytoplankton populations. As phytoplankton form the base of the marine food web, their reduction can significantly affect the entire marine biodiversity, leading to decreased population sizes in marine species and potential disruptions in ecological balance."
Traditional RAG:
Text Chunks
Vector matching
Query
El Niño phenomenon
El Niño weather
Marine biodiversity
Marine food web
Marine species
Might have missed
Oceanic upwelling
Oceanic phytoplankton
Ecological balance
Unstructured information retrieval may lead to missing.
1 st
2 nd
3 rd
4 st
5 st
Reasoning
Ranked Documents
Hard to reason about distributed domain knowledge in long inputs.
Vector Database
Scaling up
Retrieve in larger database
High cost increases
Computationally and time consuming for large vector databases.
Image
Text
El Niño phenomenon
Graph Data
Numerical Data
Difficult to integrate with multiple data modality.
Text
Answer
Reasoning path
Detailed explanation
Response
I'm still confused...
Can only provide textual information to interpret the answer.
Capture complex relationships
Multi-hop reasoning
Efficiency & scalability
Flexibility in knowledge sources
Interpret -ability
GraphRAG:
El Niño phenomenon
Marine biodiversity
Oceanic upwelling
Oceanic phytoplankton
Marine food web
Ecological balance
Marine species
Graph structures offer rich semantic context and nuanced connections.
Path
Reasoning
El Niño phenomenon → Oceanic upwelling
Oceanic upwelling → Oceanic phytoplankton
Oceanic phytoplankton → Marine food web → Marine biodiversity
Marine biodiversity → Marine species
Ecological balance
Graph structures support sophisticated reasoning across multiple nodes.
Graph Database
Scaling up
Retrieve in larger database
Low cost increases
Graph databases optimize for relationship-based queries and faster retrieval.
A sub-graph
Image
Text
El Niño phenomenon
Graph Data
Numerical Data
Graph support multiple data modality integration and real-time update.
Text
Answer
Reasoning path
Detailed explanation
Paths in Graph
Response
I'm convinced.
Graph structures enhance response interpretability and transparency.

Key Features of GraphRAG

- **Structured Representation:** Uses knowledge graphs.
- **Contextual Retrieval:** Understands semantic relationships.
- **Efficient Processing:** Reduces computational cost.
- **Multi-Faceted Queries:** Synthesizes data from multiple sources.
- **Explainability:** More transparent than black-box models.
- **Continuous Learning:** Expands knowledge over time.

Different Approaches

Approach	Description	Key Differentiators
Neo4j	LLM extracts entities, builds KG, translates queries to Cypher	Domain-specific entities + structured queries
LlamaIndex	Hierarchical graph + community detection + multi-level summaries	Layered context resolution
Microsoft	KG construction + Leiden clustering + community summaries	Dataset-wide insights
Hybrid	Vector search + graph traversal	Semantic + relational fusion

YHK

References

Slides primarily borrowed from ...

- End-to-End Implementation of GraphRAG(Knowledge Graphs + Retrieval Augmented Generation): Unlocking LLM Discovery on Narrative Private Data - Vinod Kumar G R
- GraphRAG tutorials by Data Science in your pocket
- GraphRAG: Enhancing LLMs with Knowledge Graphs - Vijayakumar Ramdoss
- Awesome-GraphRAG (GraphRAG Survey) https://github.com/DEEP-PolyU/Awesome-GraphRAG
- Build your hybrid-Graph for RAG & GraphRAG applications using the power of NL - Irina Adamchic
- The GenAI Stack - Andreas Kollegger - Neo4j
- GraphRAG: The Practical Guide for Cost-Effective Document Analysis with Knowledge Graphs JaykumaranJaykumaran

YHK

Thanks . . .

- Search **"Yogesh Haribhau Kulkarni"** on Google and follow me on LinkedIn and Medium
- Office Hours: Saturdays, 2 to 5pm (IST); Free-Open to all; email for appointment.
- Email: yogeshkulkarni at yahoo dot com

(https://www.linkedin.com/in/yogeshkulkarni/, QR by Hugging Face QR-code-AI-art-generator, with prompt as "Follow me")