GraphRAG and Knowledge Integration

Week 8: From Vector Search to Knowledge Graphs

PhD Course in Agentic Artificial Intelligence

12-Week Research-Level Course

Learning Objectives

Bloom's Taxonomy Levels Covered

- **Remember**: Define knowledge graph (entity-relationship structure), entities, relations, communities (clusters)
- **Understand**: Explain how GraphRAG enhances retrieval with structure
- **Apply**: Build a knowledge graph from unstructured text using LLMs
- **Analyze**: Compare vector-only vs graph-enhanced retrieval strategies
- **Evaluate**: Assess when GraphRAG provides value over standard RAG
- **Create**: Design a hybrid retrieval system combining vectors and graphs

---

By end of lecture, you will understand structured knowledge integration in agents.

Limitations of Vector-Only RAG

What Vector Search Does Well

- Semantic similarity matching ("find documents about X")
- Fast approximate nearest neighbor search
- Works with any text without preprocessing

Where Vector Search Struggles

- **Multi-hop reasoning**: "Who founded the company that acquired X?"
- **Global queries**: "What are the main themes across all documents?"
- **Relationship traversal**: "How are entities A and B connected?"
- **Aggregation**: "List all products mentioned with their features"

The Insight

- Structure enables reasoning that similarity alone cannot support

---

GraphRAG adds explicit structure to enable complex reasoning patterns.

What is a Knowledge Graph?

Definition

- A knowledge graph represents information as **entities** (nodes) connected by **relations** (edges)
- Triples: (subject, predicate, object) – e.g., (Apple, founded_by, Steve Jobs)

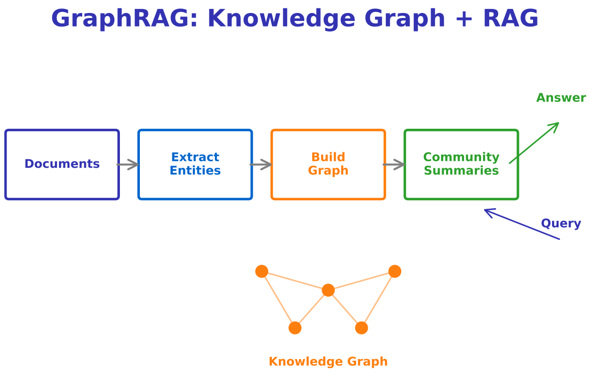
Key Components

- **Entities**: People, places, concepts, events (named objects)
- **Relations**: Connections between entities (typed edges)
- **Attributes**: Properties of entities (metadata)
- **Schema**: Optional type hierarchy and constraints

Examples

- Wikidata (90M+ entities), Google Knowledge Graph, enterprise KGs

Knowledge graphs make implicit relationships explicit and queryable.



GraphRAG builds structure from documents before retrieval.

Entity and Relationship Extraction

LLM-Based Extraction (Microsoft GraphRAG)

- Use LLM to identify entities and relationships from text chunks
- Prompt: "Extract all entities and their relationships from this text"
- Output: Structured triples (entity1, relation, entity2)

Entity Types Commonly Extracted

- People, organizations, locations, events, concepts
- Domain-specific: products, chemicals, diseases, etc.

Challenges

- Entity resolution ("Apple Inc." = "Apple" = "the company")
- Relation normalization ("works for" = "employed by")
- Extraction quality depends heavily on LLM capability

LLM-based NER (Named Entity Recognition) enables extraction without training.

Knowledge Graph Construction Trade-offs

Extraction Precision vs. Recall

- High precision: Miss valid entities/relations
- High recall: Include noise, false positives
- LLM extraction typically favors recall

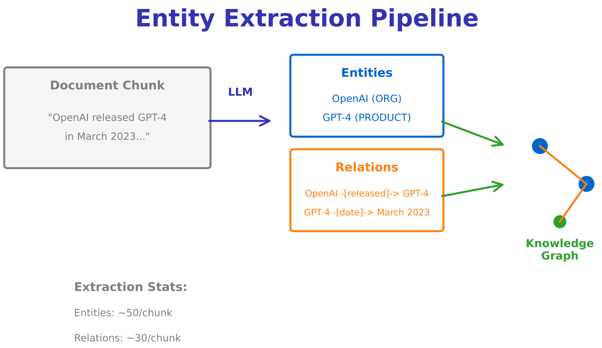
Schema Design Choices

- **Open schema**: Flexible but harder to query
- **Fixed schema**: Constrained but consistent
- **Hybrid**: Core schema + open extensions

Cost Breakdown (per 1M tokens)

- Entity extraction: \$1-3, Relation extraction: \$2-5
- Entity resolution: \$0.5-1, Community summarization: \$3-8
- Total GraphRAG indexing: ∼\$10/1M tokens vs. \$0.10 for basic RAG

GraphRAG indexing cost is 100x basic RAG – ensure query patterns justify it.



LLMs extract entities and relations to build the knowledge graph.

Community Detection for Global Queries

The Global Query Problem

- Local queries: "What did X say about Y?" – direct retrieval works
- Global queries: "What are the main themes?" – needs aggregation

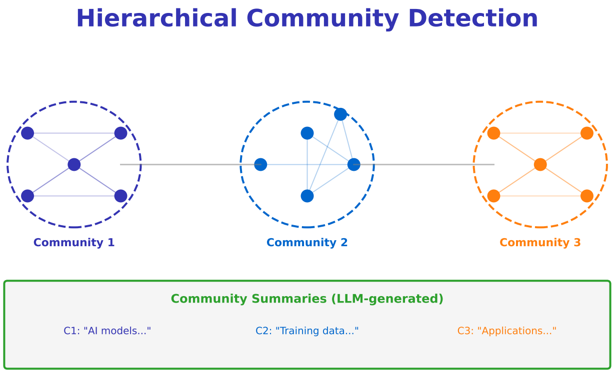
Hierarchical Summarization via Communities

- **Leiden algorithm** (graph clustering method): Cluster densely connected entities
- Generate summaries at each community level
- Build hierarchy: documents $\rightarrow$ entities $\rightarrow$ communities $\rightarrow$ global

Benefits

- Pre-computed summaries enable fast global queries
- Multiple granularity levels for different query types
- Captures both local detail and global themes

Community summaries enable answering "what is this corpus about?" efficiently.



Leiden algorithm clusters entities for hierarchical summarization.

Query Routing by Type

Local Search (Specific Queries)

- Query: "What did Einstein say about quantum mechanics?"
- Strategy: Entity lookup $\rightarrow$ traverse relations $\rightarrow$ retrieve text
- Uses: Entity embeddings + graph traversal

Global Search (Broad Queries)

- Query: "What are the main research themes in this corpus?"
- Strategy: Retrieve community summaries at appropriate level
- Uses: Pre-computed hierarchical summaries

Hybrid Search

- Combine vector similarity with graph structure
- Route based on query classification (local vs global)

Different query types benefit from different retrieval strategies.

Multi-hop Reasoning with Graphs

Multi-hop Query Types

- **Chain**: A $\rightarrow$ B $\rightarrow$ C ("Who founded the company that acquired X?")
- **Fan-out**: A $\rightarrow$ {B, C, D} ("All products from company X?")
- **Fan-in**: {A, B, C} $\rightarrow$ D ("What connects these entities?")

Graph Traversal Strategies

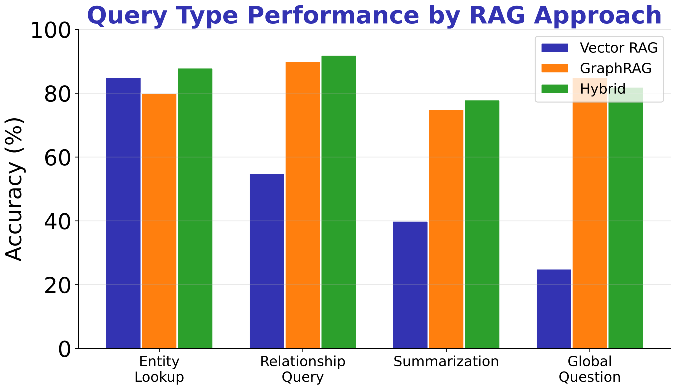
- BFS from query entities (breadth-first search)
- Embedding-guided traversal (prioritize relevant paths)
- LLM-guided exploration (let model decide next hop)

Challenges and Best Practices

- Path explosion with hop count – limit to 2-3 hops
- Noise accumulation – prune low-confidence edges
- Combine with vector retrieval for verification

Multi-hop reasoning trades coverage for precision – tune hop depth per query type.

Query Routing Architecture



Intelligent routing selects the optimal retrieval path.

Hybrid Vector-Graph Retrieval

Architecture

- Vector store for semantic similarity
- Graph store for structural queries
- Fusion layer to combine results

Fusion Strategies

- **Early fusion**: Combine before ranking
- **Late fusion**: Rank separately, then merge
- **Cascaded**: Vector retrieval, then graph expansion

Implementation Options

- Separate stores: Neo4j + Pinecone
- Integrated: Neo4j with vector indexes
- LlamaIndex: PropertyGraph with vector support

Hybrid retrieval captures both semantic and structural relevance.

Implementation Considerations

Graph Storage Options

- Neo4j (native graph DB), NetworkX (in-memory), Neptune (AWS)
- Hybrid: Vector store + graph DB for combined queries

Indexing Cost vs Query Benefit

- GraphRAG requires significant upfront processing
- Entity extraction: ~$1-5 per 1M tokens (LLM costs)
- Summarization: Additional passes over extracted entities
- Best ROI: Large, static corpora with complex query patterns

When to Use GraphRAG

- Multi-hop reasoning required
- Global/thematic queries common
- Corpus is relatively stable (infrequent updates)

GraphRAG trades indexing cost for query capability – choose based on use case.

Required Readings

This Week

- Edge et al. (2024). "From Local to Global: A GraphRAG Approach to Query-Focused Summarization." Microsoft Research. arXiv:2404.16130
- Pan et al. (2024). "Unifying Large Language Models and Knowledge Graphs: A Roadmap." arXiv:2306.08302

Supplementary

- Besta et al. (2024). "Graph of Thoughts: Solving Elaborate Problems with LLMs." arXiv:2308.09687
- Gutierrez et al. (2024). "HippoRAG: Neurobiologically Inspired Long-Term Memory." arXiv:2405.14831

Focus on Microsoft GraphRAG paper for implementation details.

Summary and Key Takeaways

Key Concepts

- **Knowledge Graph**: Entity-relationship structure enabling multi-hop reasoning
- **GraphRAG**: Combine KG with vector retrieval for comprehensive search
- **Communities**: Hierarchical clustering for global query support
- **Query Routing**: Match query type to optimal retrieval strategy

Design Principles

- Use structure when relationships matter
- Pre-compute summaries for global queries
- Route queries intelligently based on type

Next Week

- Hallucination Prevention and Verification

GraphRAG = Structure + Vectors for comprehensive retrieval.