Slot 3: Advanced Graph RAG Approaches

ISWC 2024

ISWC Baltimore, USA
Fraunhofer FIT
RWTH AACHEN UNIVERSITY
TIB LEIBNIZ-INFORMATIONSZENTRUM TECHNIK UND NATURWISSENSCHAFTEN UNIVERSITÄTSBIBLIOTHEK
UPB

Part 1: Introduction

Graph RAG with Semantic Clustering

- This approach leverages clustering algorithms to group similar entities and concepts within the knowledge graph.
- This clustering enhances the retrieval process by allowing the system to identify related information even when there is no direct match between the user's query and the graph's entities.
- For instance, if a user asks about "electric vehicles," the system could retrieve information related to "Tesla," "Nissan Leaf," and "sustainable transportation" by recognizing their shared cluster membership within the knowledge graph

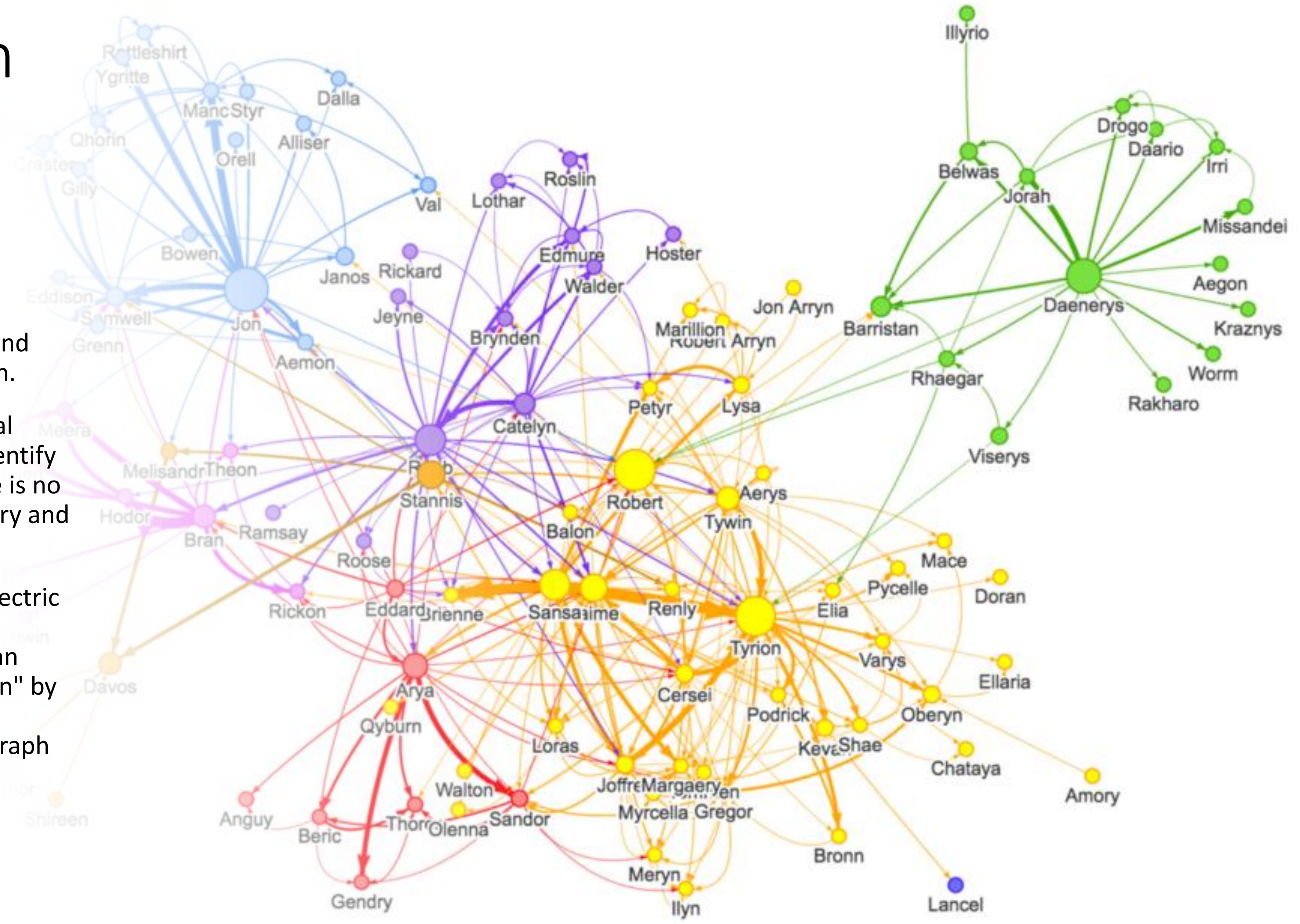


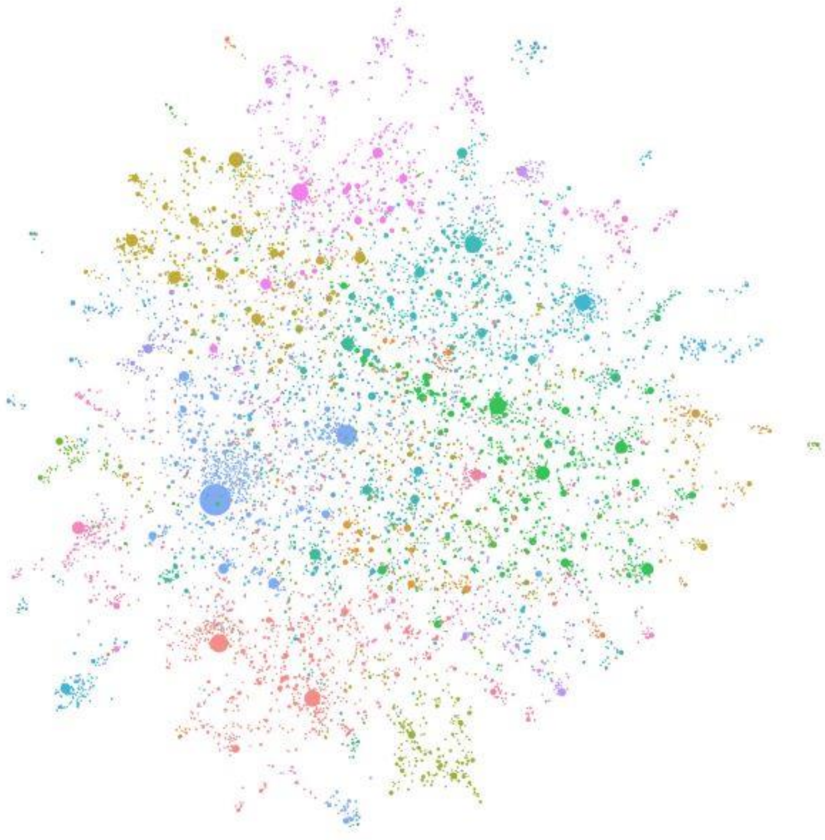
Image source: The [Graph of Thrones](#) visualization by William Lyon

GraphRAG

From Microsoft

A Graph RAG Approach to Query-Focused Summarization

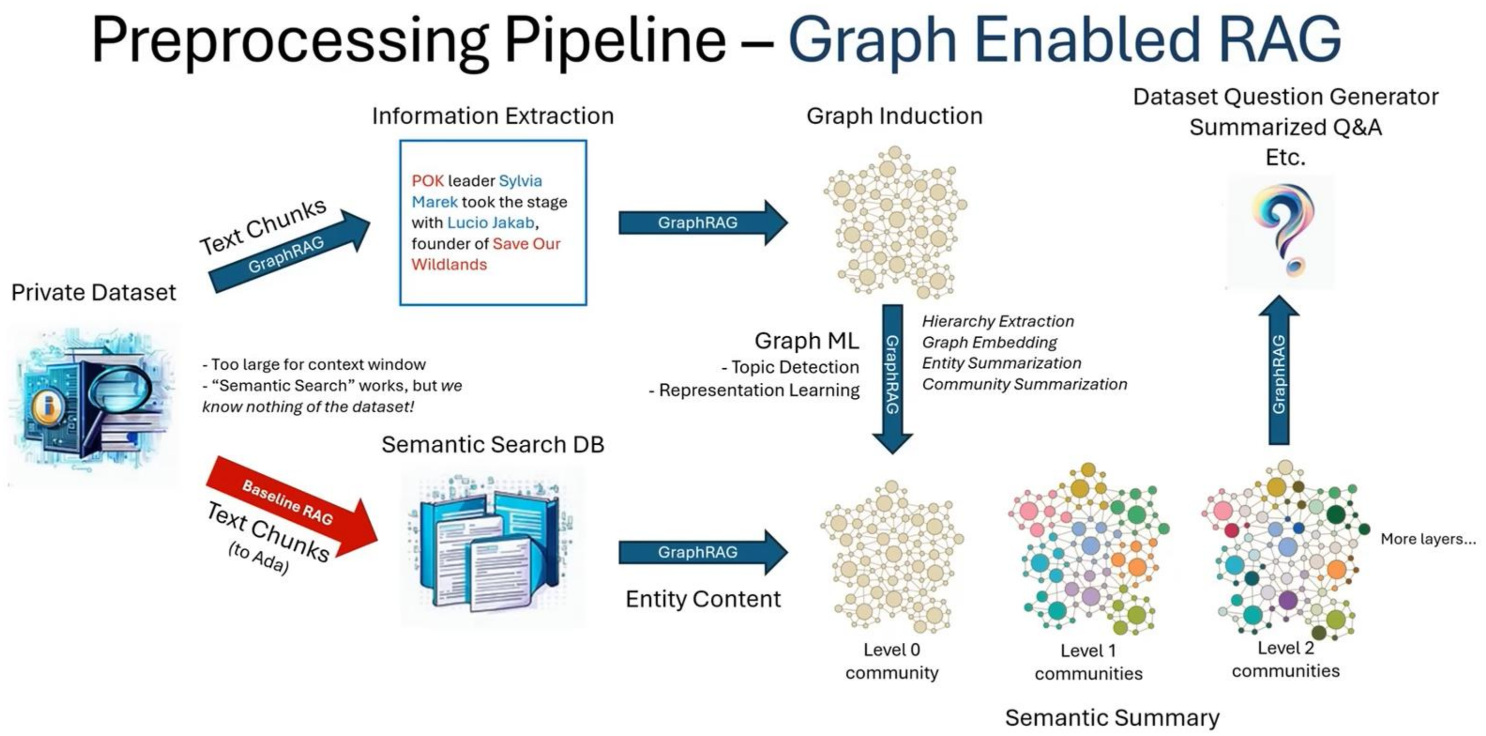
- Naive RAG Issue: Fails on global questions like "What are the main themes in the dataset?"
- Graph RAG Approach:
  - Extract knowledge graph from raw text
  - Build a community hierarchy
  - Generate community summaries
- Improved comprehensiveness and diversity of generated answers on large datasets



An LLM-generated knowledge graph built using GPT-4 Turbo, Microsoft, https://microsoft.github.io/graphrag/

A Graph RAG Approach to Query-Focused Summarization

- Execute the indexing pipeline to extract and construct the knowledge graph.
- Convert artifacts into RDF triples, mapping them according to the ontology.
- Perform semantic searches within the RDF graph to retrieve detailed entity information.
- Extract relevant subgraphs for focused and efficient querying.



GraphRAG: LLM-Derived Knowledge Graphs for RAG, YouTube, uploaded by Alex Chao, May 4, 2024. Available at: https://youtu.be/r09tJfON6kE

Part 2: G-Indexing

Creating a Knowledge Graph with LLMs for RAG systems

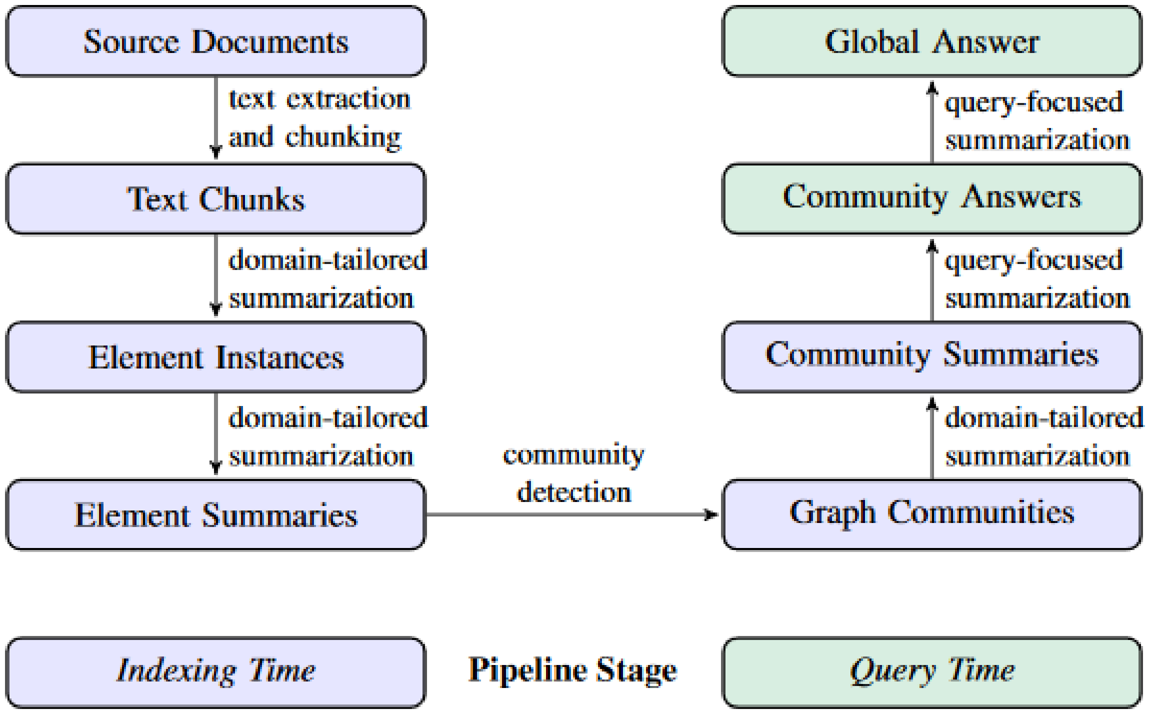
Indexing Pipeline

Knowledge Graph Construction:

- **LLM-Based:** Automatically identify entities and relationships.
- **Iterative Gleaning:** Multi-round processing ensures completeness.

Community Summarization:

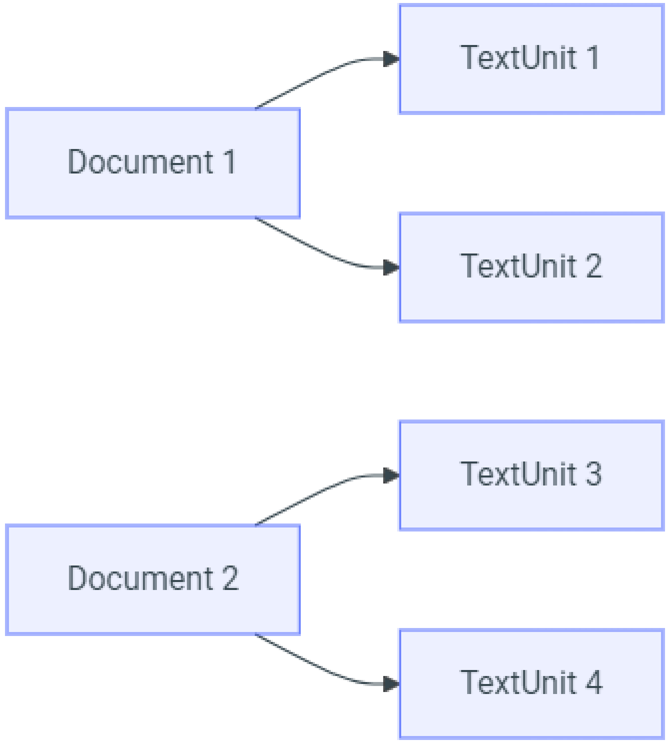
- Detect communities in the graph (e.g., using Leiden algorithm).
- Summarize each community for answering questions.

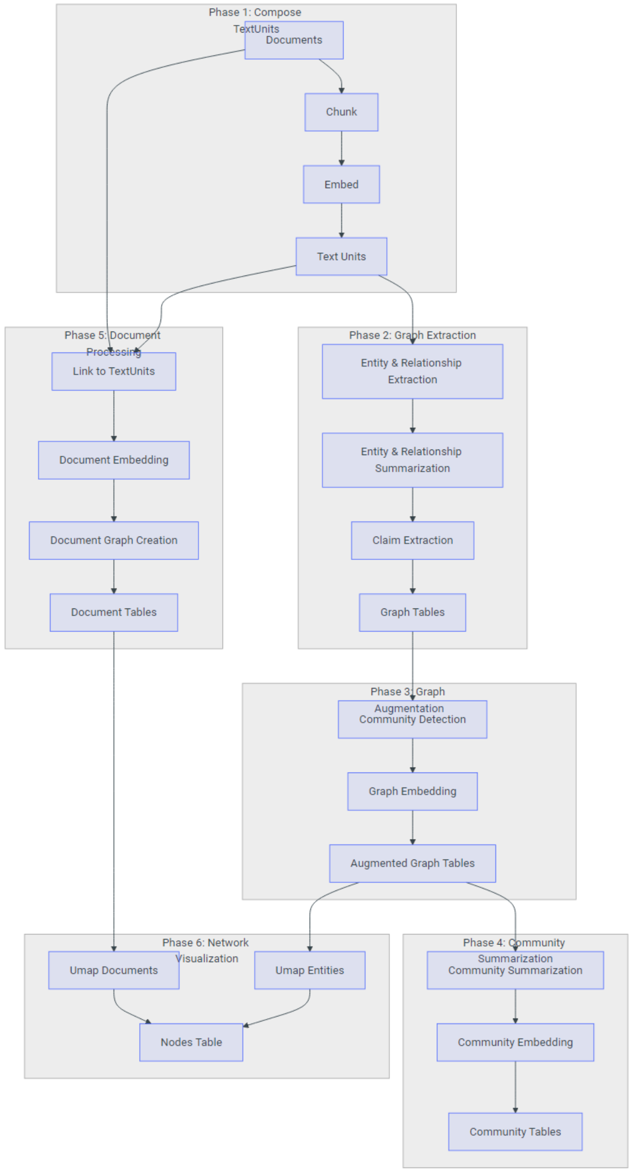


Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., & Larson, J. (2024). From local to global: A graph RAG approach to query-focused summarization. arXiv. https://arxiv.org/abs/2404.16130

Indexing Dataflow

- Phase 1: Compose TextUnits
    - Transform input documents into *TextUnits*

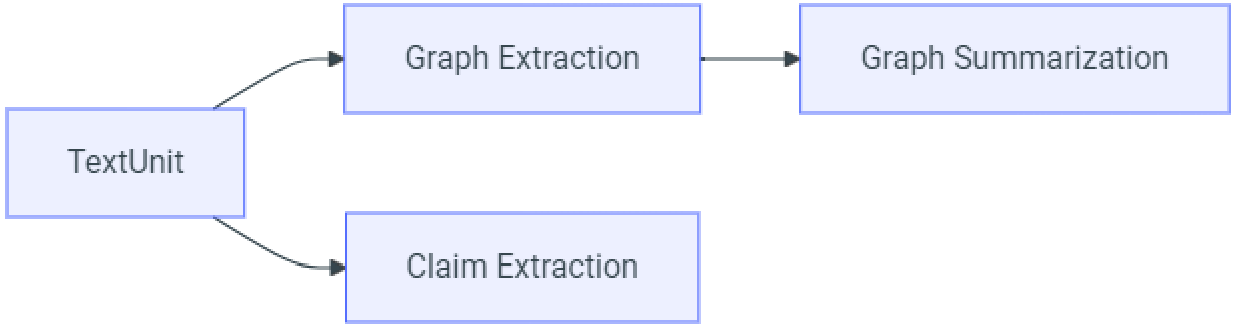


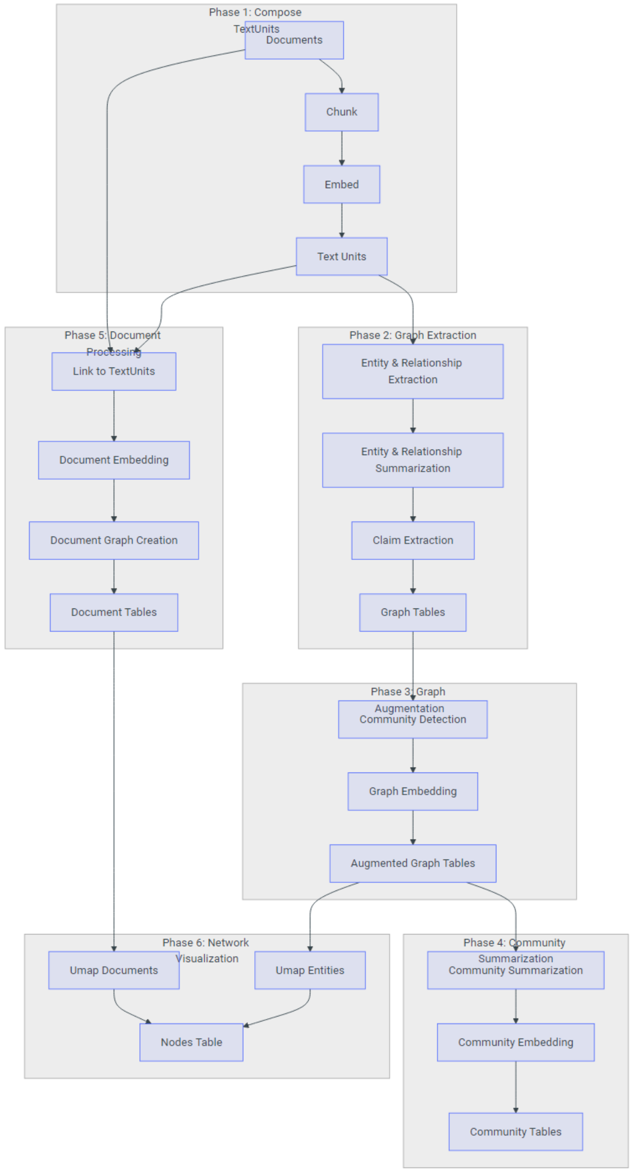


The Default Configuration Workflow transforms text documents into the GraphRAG Knowledge Model, Microsoft, https://microsoft.github.io/graphrag/

Indexing Dataflow

- Phase 2: Graph Extraction
  - Entity & Relationship Extraction
    - A list of entities with a name, type, and description
    - A list of relationships with a source, target, and description
  - Entity & Relationship Summarization
    - Summarize these lists into a single description per entity and relationship

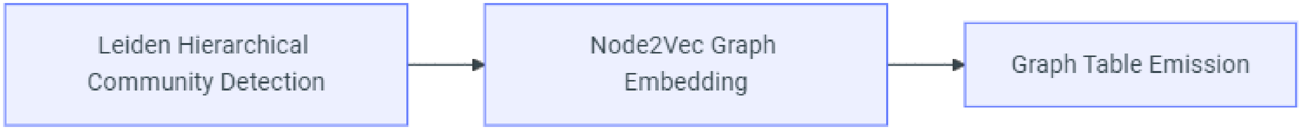


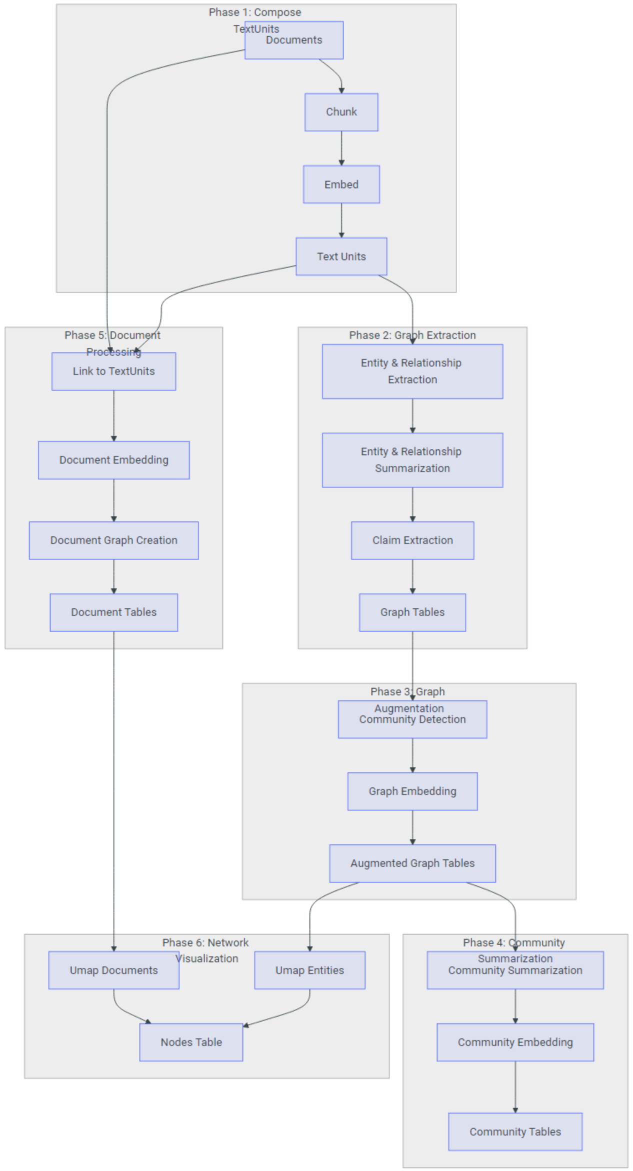


The Default Configuration Workflow transforms text documents into the GraphRAG Knowledge Model, Microsoft, https://microsoft.github.io/graphrag/

Indexing Dataflow

- Phase 3: Graph Augmentation
  - **Community Detection**: Generate a hierarchy of entity communities using the Hierarchical Leiden Algorithm
  - **Graph Embedding**: Generate a vector representation of our graph using the Node2Vec algorithm

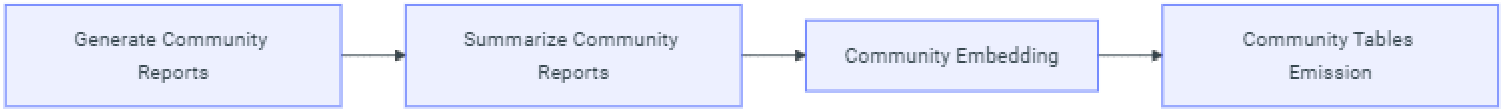


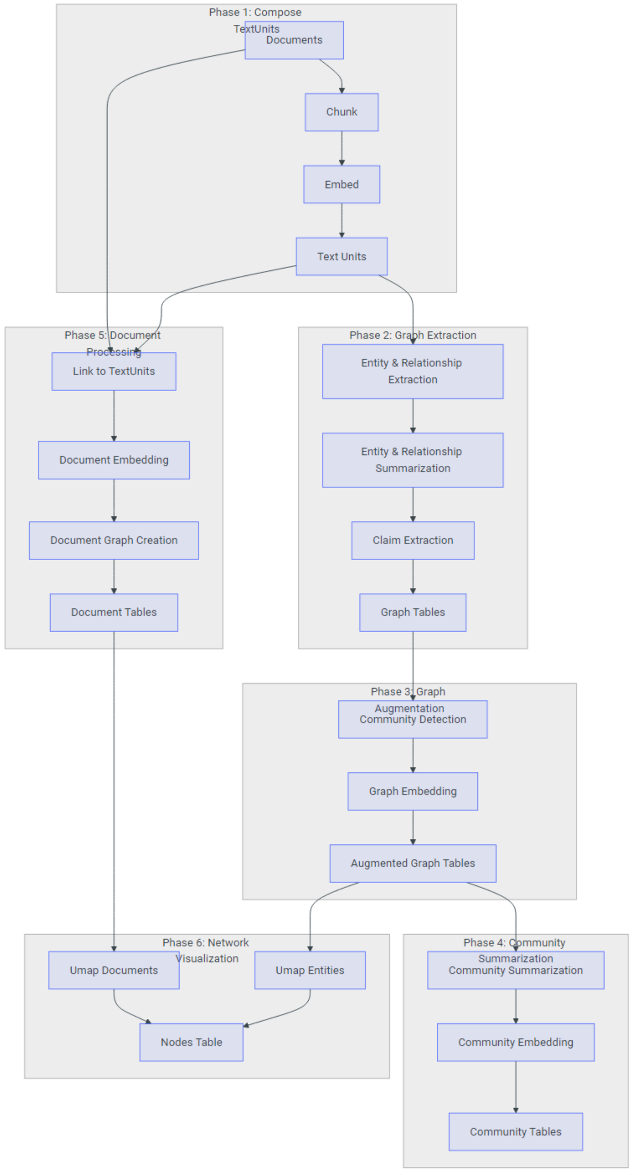


The Default Configuration Workflow transforms text documents into the GraphRAG Knowledge Model, Microsoft, https://microsoft.github.io/graphrag/

Indexing Dataflow

- Phase 4: Community Summarization
  - **Generate Community Reports**: Generate a summary of each community using the LLM
  - **Summarize Community Reports**: Each community report is then summarized via the LLM for shorthand use
  - **Community Embedding**: Generate a vector representation of communities

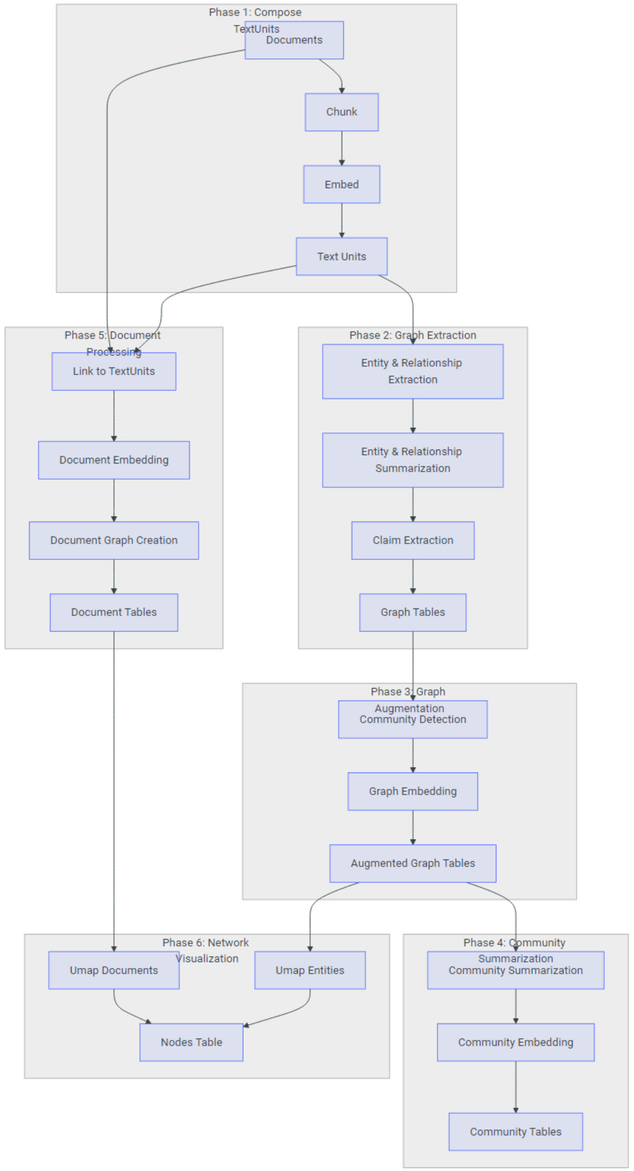




The Default Configuration Workflow transforms text documents into the GraphRAG Knowledge Model, Microsoft, https://microsoft.github.io/graphrag/

Indexing Dataflow

- Phase 5: Document Processing
  - **Link to TextUnits**: Link each document to the text-units that were created in the first phase
  - **Document Embedding**: Generate a document embedding by averaging token-weighted, non-overlapping chunks to capture document relationships



The Default Configuration Workflow transforms text documents into the GraphRAG Knowledge Model, Microsoft, https://microsoft.github.io/graphrag/

Running the Indexing pipeline

```
python -m graphrag.index --root ./ragtest
```

```
Loading csv files from  ./input
loading 1 csv files
Total number of unfiltered csv rows:  4748
Final # of rows loaded: 4748
⠏ Executing Pipeline...
├── Loading Input (csv) - 1 files loaded (0 filtered) ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00 0:00:00
├── Workflow: create_base_text_units
└── Workflow: create_base_extracted_entities
    └── verb: entity_extract ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━  99% 0:01:20 12:22:38
```

Indexing pipeline execution in GraphRAG, Microsoft, https://microsoft.github.io/graphrag/

Knowledge Graph Visualization

- Artifacts from the Indexing Pipeline
  - The outputs of the indexing pipeline are a set of Parquet files, which serve as the knowledge base for the subsequent retrieval stage.

- For a quick overview of the graph structure, visit GraphRAG-Visualizer.

The resulting knowledge graph visualization from GraphRAG, shown in the GraphRAG Visualizer.

Part 3: RDF Adaptions

Adapting GraphRAG Artifacts to RDF

RDF Adaptions

- We follow the steps below to transform GraphRAG artifacts into RDF.

Data Ingestion: Read Parquet files into Pandas Data Frames.

RDF Graph Initialization: Set up the RDF graph using rdflib.

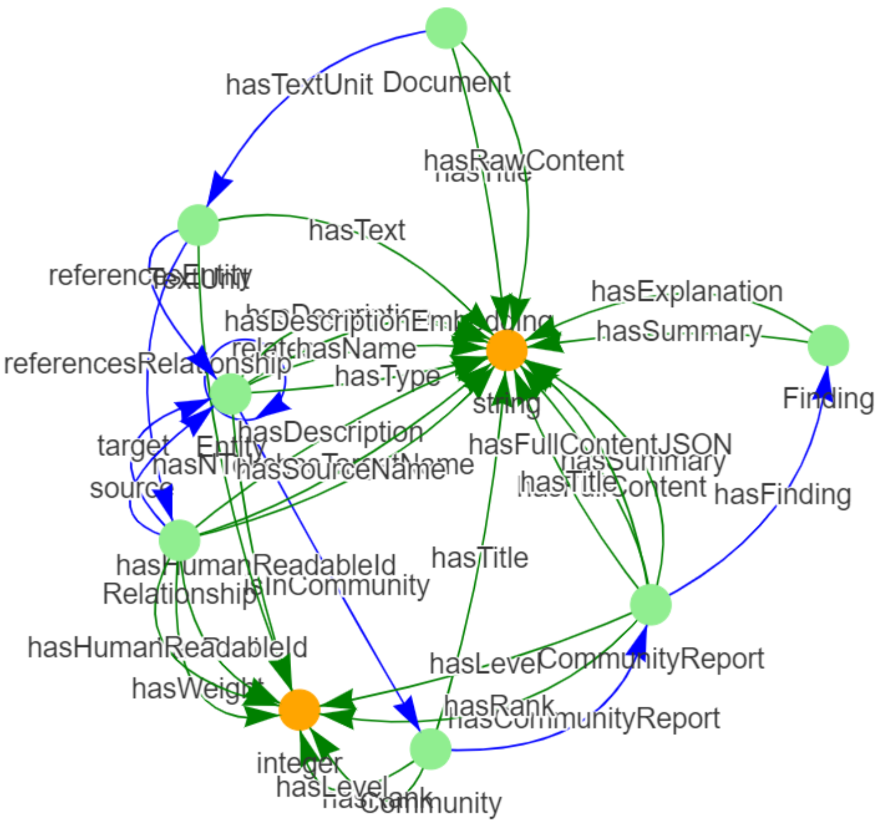
Ontology Definition: Define classes and properties in RDF.

Mapping Nodes and Relationships to RDF: Convert each node and its relationships into RDF triples.

Serializing the RDF Graph: Export the RDF graph in turtle format.

Ontology Visualization

Source	Relationship	Target
Entity	RELATES	Entity
Entity	IN_COMMUNITY	Community
Document	HAS_TEXTUNIT	TextUnit
Community	HAS_COMMUNITYREPORT	CommunityReport
CommunityReport	HAS_FINDING	Finding
TextUnit	REFERENCES_ENTITY	Entity



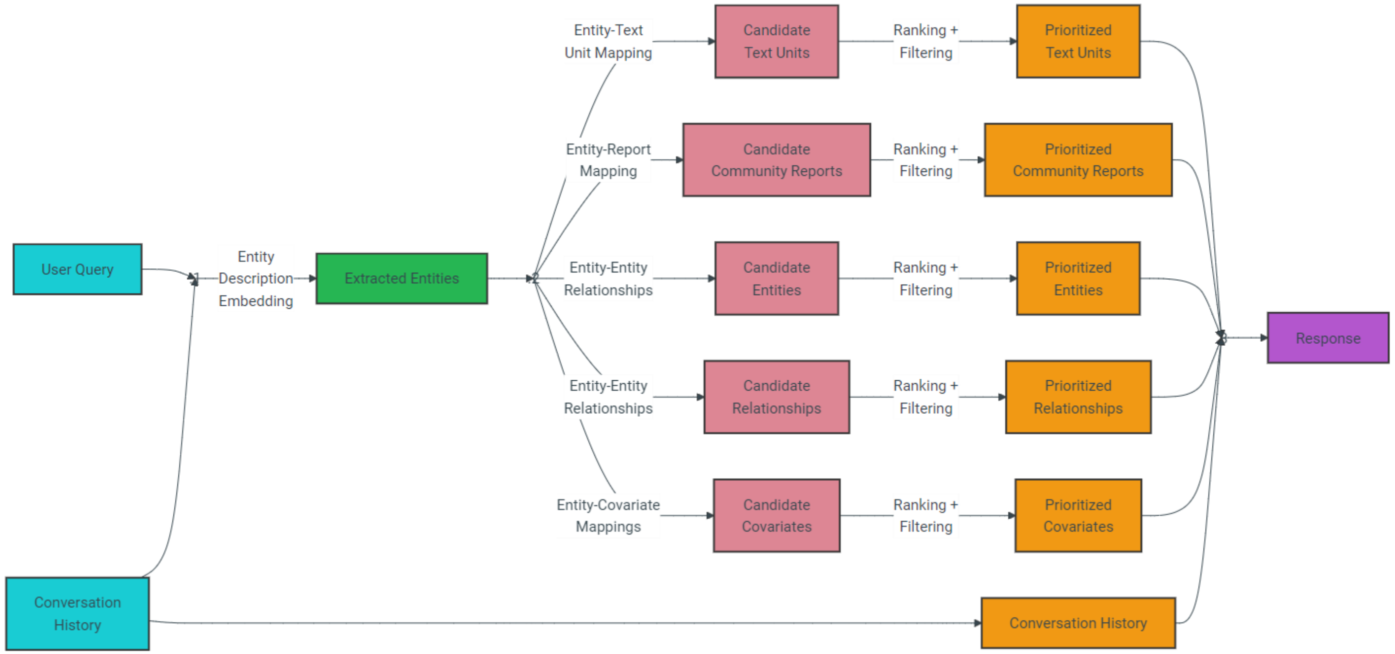
Visualization of the RDF Graph Ontology

Part 4: G-Retrieval & G-Generation

Using SPARQL with Knowledge Graphs for RAG

Local Search

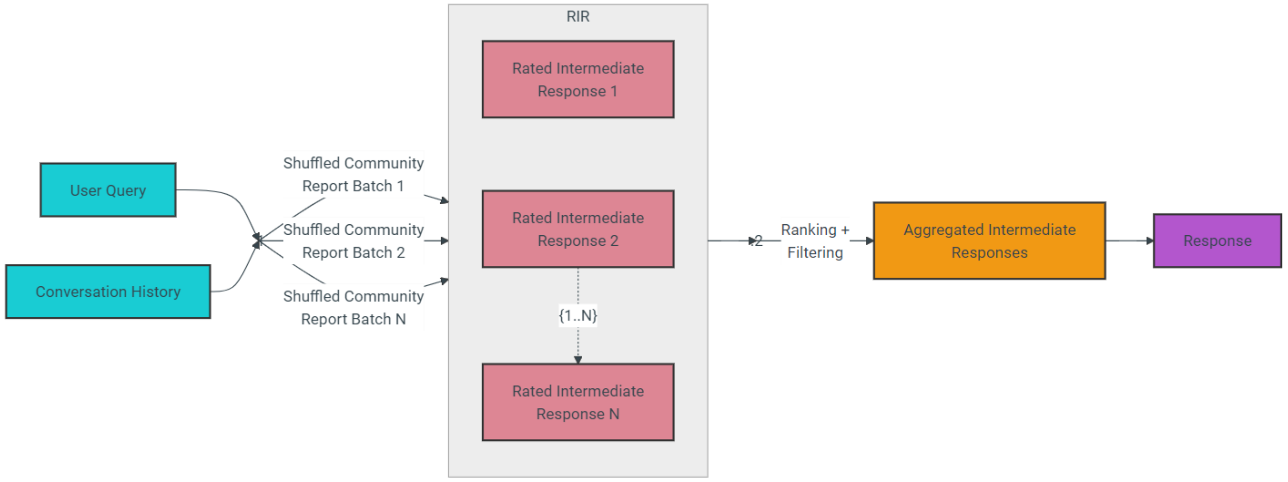
- **Entity-based Reasoning**
- **Local Search Method**: Augments LLM context by combining structured knowledge graph data with unstructured input documents.
- **Entity Extraction Process**:
  - **Identify Entities**: Extracts entities from the knowledge graph related to the query.
  - **Access Points**: Uses these entities to retrieve connected details, relationships, and relevant text from input documents.
- **Ideal for Entity-Specific Queries**: Suitable for questions about specific entities, such as "What are the healing properties of chamomile?"



Local Search Dataflow, Microsoft, https://microsoft.github.io/graphrag/query/local_search/

Global Search

- **Whole Dataset Reasoning**
- **Global Search Method**: Aggregates information from community reports in the knowledge graph.
- **Map-Reduce Process**:
  - **Map** Step: Breaks down reports into smaller chunks to generate intermediate responses.
  - **Reduce** Step: Combines key points to form a final summary.
- **Ideal for Overview Queries**: Useful for questions like "What are the top 5 themes?" or queries requiring a broad dataset overview.



Global Search Dataflow, Microsoft, https://microsoft.github.io/graphrag/query/global_search/

Hands-On 1

Adapting Graph RAG Artifacts to RDF Knowledge Graph

Hands-On 2

Using SPARQL with a Knowledge Graph for RAG