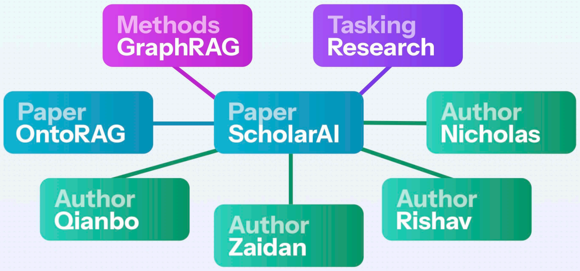
CS6101-5

ScholarAI

Simplifying Interdisciplinary AI **Research** Using **GraphRAG**

Dong **QIANBO**	/rollingpencil	/qianbodong
NICHOLAS *Cheng*	/df-cheng	/nicholas-cheng-
RISHAV *Ghosh*	/rishavghosh605	/rishavghosh605
ZAIDAN *Sani*	/zaidansani	/mzaidanbsani



Abstract

This project builds a graph- and ontology-augmented QA system over AI papers from arXiv by combining **GraphRAG** for entity/relation-aware retrieval with **OntoRAG** for schema- and rule-driven reasoning. We construct a heterogeneous knowledge graph of papers, authors, methods, datasets, align it with AI ontologies (subfields, method and dataset taxonomies), and **use ontological constraints to normalize terms, resolve synonyms, and support multi-hop inference.**

Implementation

System Overview

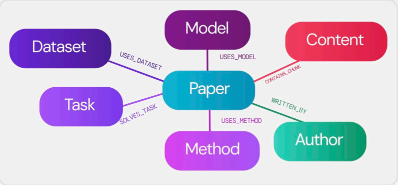
We build the datastore by randomly choosing 100 articles from ArXiV, searching for "RAG". Next, we ingest the articles into LLM extraction pipeline to get ontologies, which are used to create the knowledge graph. An agentic workflow with tools to query the system using both vector and GraphRAG methods is then responsible for handling the query formulation, expansion, and answer generation for user queries.

Automated Ontology Derivation

We define the ontologies of a research paper as the

- **tasks** solved
- **methods** used
- **models** used
- **authors** of the paper
- **datasets** used

We then utilise LLM extraction to extract these properties and build the knowledge graph to query in the RAG system.

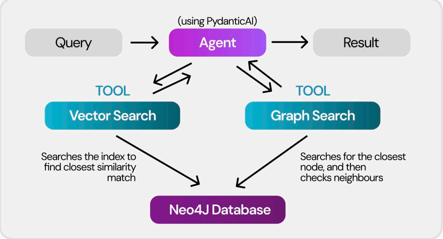


Additionally, within nodes of the same type, we add an edge if they are semantically similar enough, using cosine similarity.

Agentic Flow

We utilise an agent to handle retrieval query formation and determine when enough information is available to create an answer.
The agentic flow is responsible for

- determining the type of query being run (vector/graph)
- formulating the query being sent to datastore
- augmenting the context with retrieval results
- generating the answer


(using PydanticAI)
Query
Agent
Result
TOOL
Vector Search
TOOL
Graph Search
Searches the index to find closest similarity match
Searches for the closest node, and then checks neighbours
Neo4J Database

- Tools use LLMs to adapt natural language queries into Cypher queries for the Neo4J database
- Result returns a concise answer, as well as the retrieved context, and reasoning for the answer.

Further Avenues Of Development

Better Cypher Queries

We utilised the LLM to generate the queries, due to the large breadth of questions. While this generally worked, some of the queries were not as desired. Thus, designing query templates, with entity lookup by id/name (exact + fuzzy via trigram), k-hop neighborhood with length bounds, constrained paths and aggregations with index hints could retrieve better results.

Better Decision Policy

We utilised an agent to decide which tool to call within the current system. A more sophisticated decision policy, with explicit features such as presence of certain entities/relations in the query, the need for multi-hop reasoning, temporal or structural operators, which address when semantic recall or precise structural retrieval should be used.

Dataset

100 randomised RAG articles on ArXiV

Objective

We aim to explore the tradeoffs of applying the OntoRAG system into a new domain like academia, to discover new useful use cases for researchers and students.

Based on

OntoRAG: Enhancing Question-Answering through Automated Ontology Derivation from Unstructured Knowledge Bases

`(arXiv:2506.00664)`

Study

We run an ablation study, running 120 custom queries, evaluating with the following metrics:

Accuracy	Does the generated response contains factually accurate information? (Using LLM-as-a-judge, with an expected answer)
Groundedness	Is the response consistent with the context provided? (Measured using semantic similarity with the context)
Completeness	Does the response address all parts of the user query? (Measured using token count)
Relevance	How related is the generated response is with respect to the users query? (Measured using semantic similarity with query)

with the different systems:

Full System	The entire system, including all ontologies in the GraphRAG, as well as VectorRAG
GraphRAG only	Only queries that utilise the graph nodes without the vector indexes
VectorRAG only	Only queries that utilise the vector index to find similar nodes
w/o Similarity Links	The entire system, but without edges connecting similar nodes of the same type

The ablation configs were chosen in an attempt to distinguish when the ontologies would benefit the generation of the answer.

Experiment	Accuracy	Groundedness	Relevance	Completeness
Full System (Graph + Vector)	0.4130	0.7427	0.8674	0.7427
Graph only (Ontologies only)	0.4322	0.7413	0.8891	0.7413
VectorRAG only	0.4274	0.7784	0.9032	0.7784
w/o Similarity Links	0.4715	0.7436	0.8768	0.7436

Analysis of Results

Comparison between the different ablations

- Vector-only retrieval demonstrated the highest relevance, grounding, and completeness, effectively retrieving context that enables the LLM to generate coherent and comprehensive answers.
- Ontology-only retrieval maintained strong precision for structured and entity-based queries (e.g., authors, affiliations, dataset names), highlighting its utility for metadata-oriented tasks.
- Graph-RAG routing enabled per-question retrieval strategy selection, illustrating - that hybrid methods can dynamically leverage the strengths of each paradigm.
- Graph-similarity ablation (removing direct similarity edges) achieved the highest overall accuracy, suggesting that such edges may introduce ambiguous or noisy reasoning paths.

> When asked: "Wanlong Liu and Junying Chen are credited with equal contributions to RAG-Instruct. What are their respective institutional affiliations listed in the paper?", ontology-only **correctly retrieved precise affiliations:** "University of Electronic Science and Technology of China" and "The Chinese University of Hong Kong, Shenzhen"

Limitations of the systems

- Mathematical and formula-based content was frequently mishandled or misprinted, likely due to the fixed-length chunking strategy used during preprocessing.
- The full system provided broader coverage but at the cost of lower relevance, as compared to vector search's cosine-similarity precision and ontology-only evaluations. This is due to interconnected nodes for chunks were often retrieved which maybe irrelevant or noisy.

> When asked, "What specific problem regarding document utilization in RAG training does CL-RAG aim to solve?" The full system retrieved **broader but less relevant content** about "enhancing generalization and stability. Vector-only and ablated versions **provided more focused answers** about document quality variations.

- Ontology-only retrieval lacked local text context when compared to vector-only retrieval, leading to incomplete or underspecified explanations in technical or conceptual answers.

> When asked, "Describe the three difficulty levels used to categorize documents when applying Curriculum Learning (CL) for RALM training in CL-RAG." Ontology-only provided only surface-level categorization: "documents are categorized into multiple difficulty levels". It **missed detailed explanations about Easy/Common/Hard levels,** in contrast with vector-only which retrieved **full contextual descriptions.**