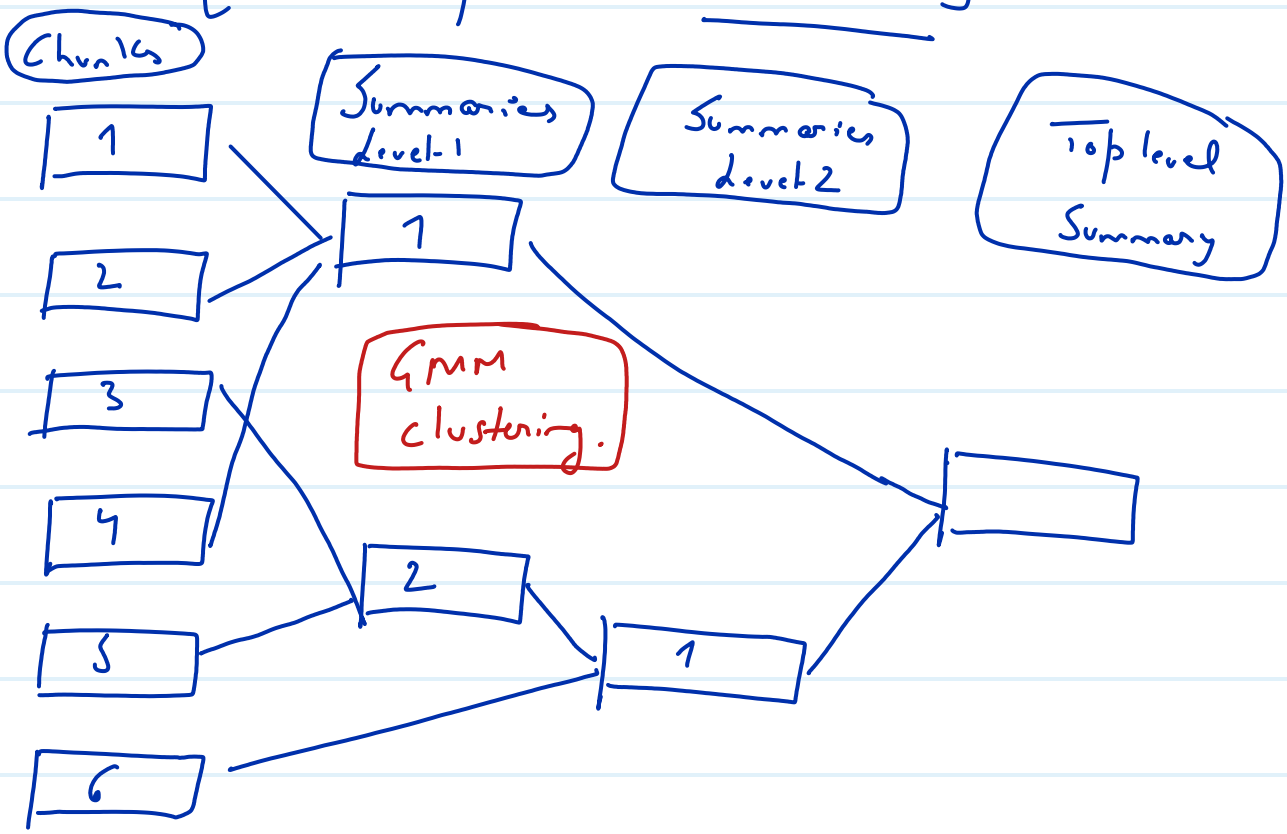
GRAPH RAG

Traditional RAG :-

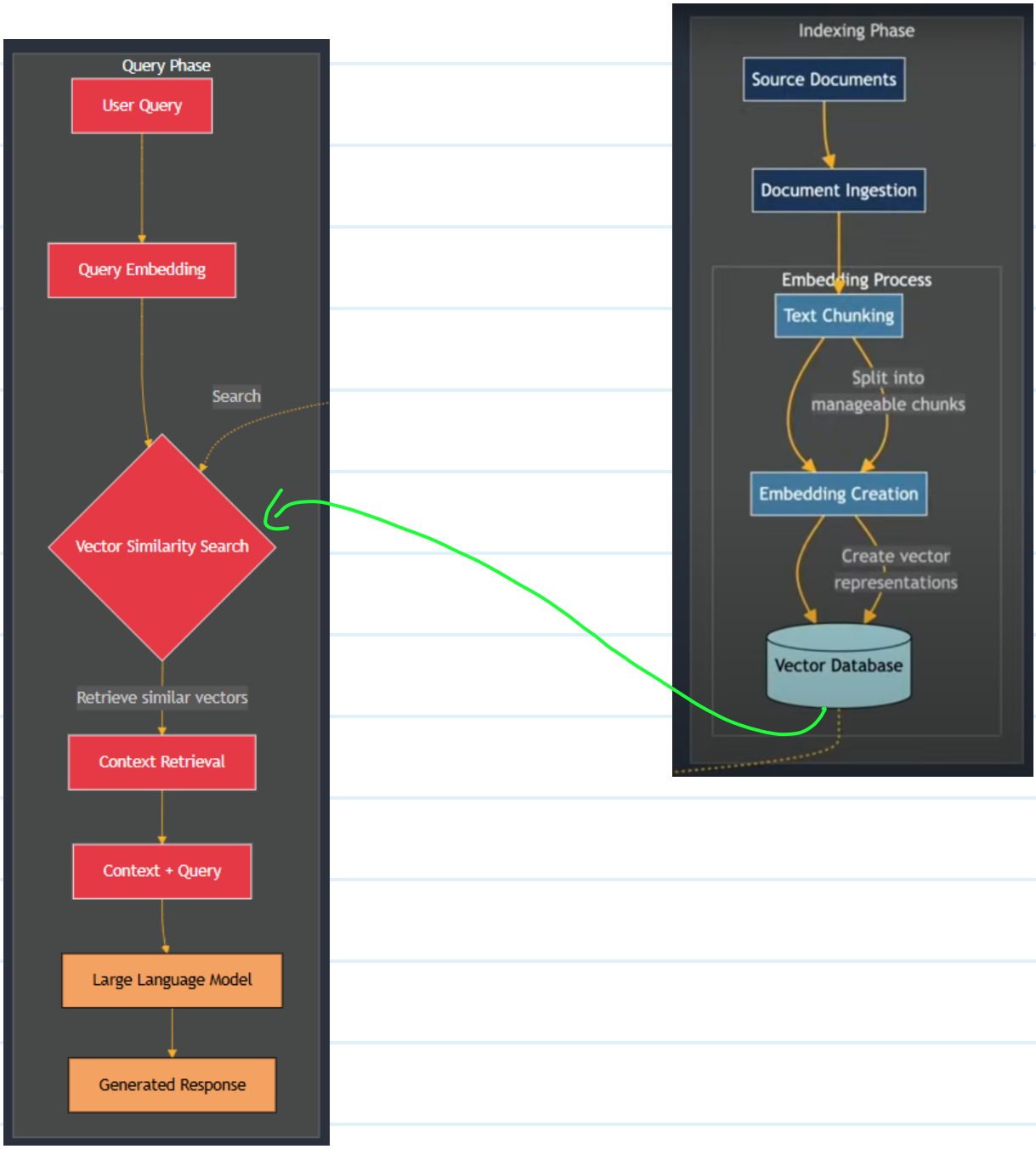
→ Relies on flat retrieval mechanism.
  → Embedding based vector store.
→ Retrieve documents based on Vector Similiarity.
→ Limited to the scope of individual text segments or embedding.

{ Recent Update → RAPTOR }



Helps the process capture information at varing levels of abstraction, enabling, the model to access both detailed & high-level contextual information.

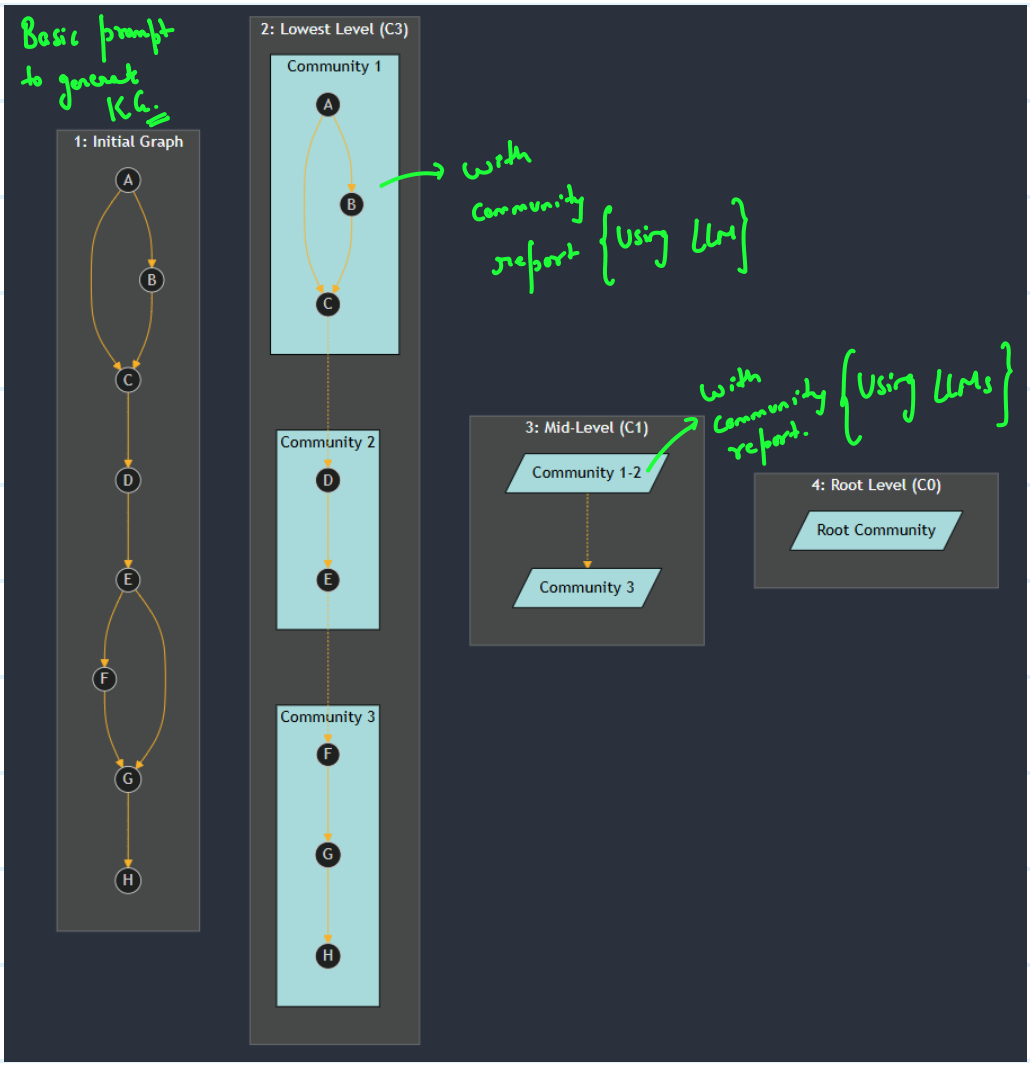
⇒ Leads to a more holistic understanding.

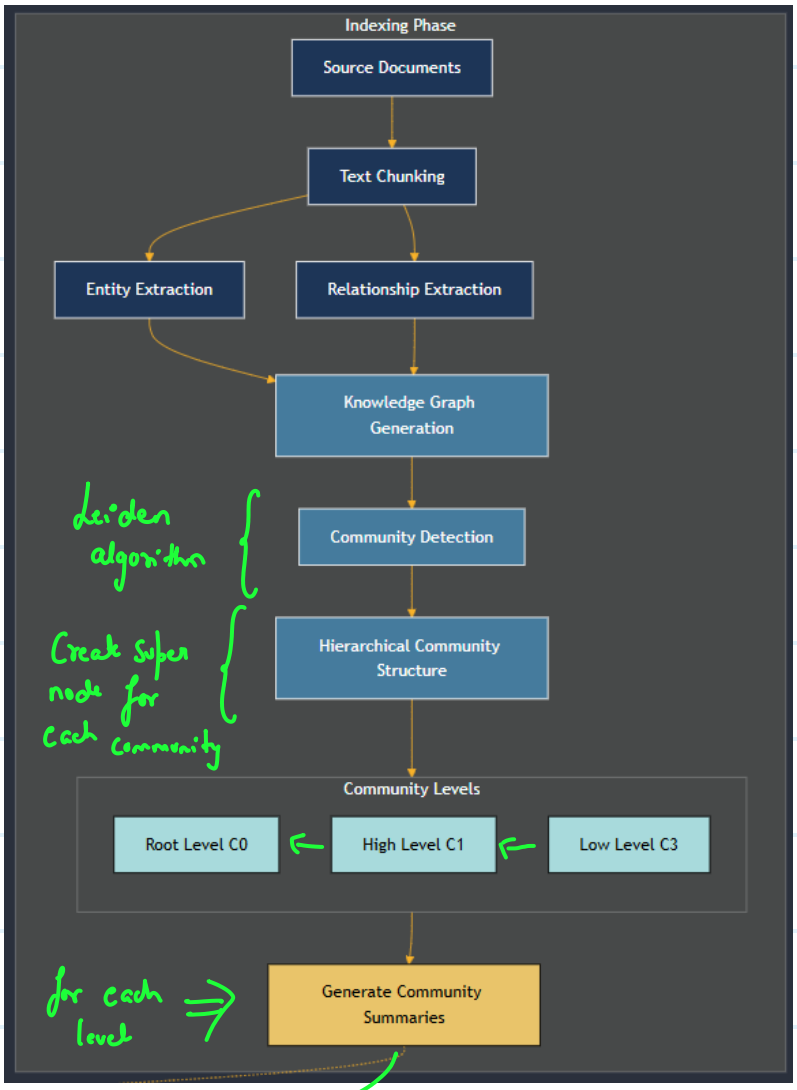


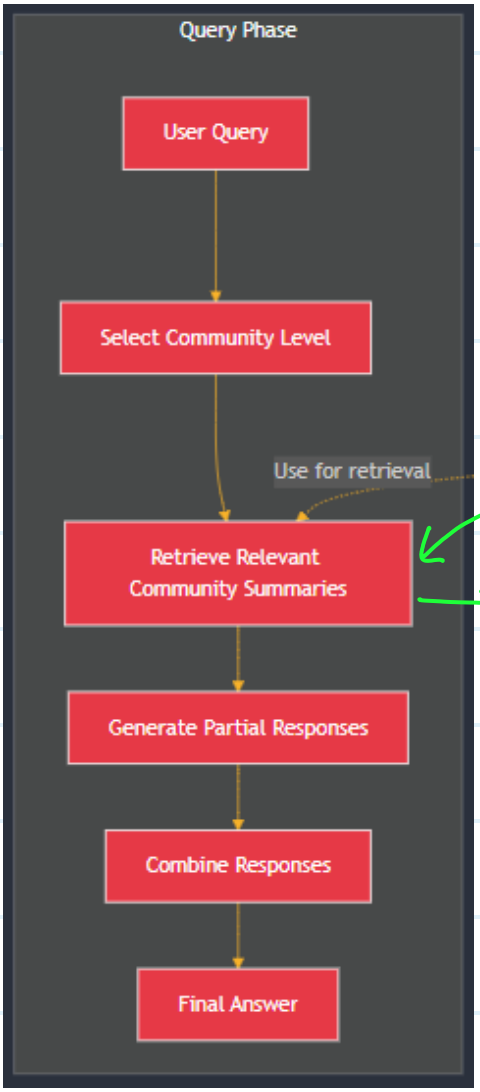
GRAPH RAG → [SOTA]

→ Uses LLMs to generate KG, to facilitate query focused summarization over extensive text - corpora.

→ Uses Leiden Algo ⇒ to detect communities within the graph, grouping closely related entities.

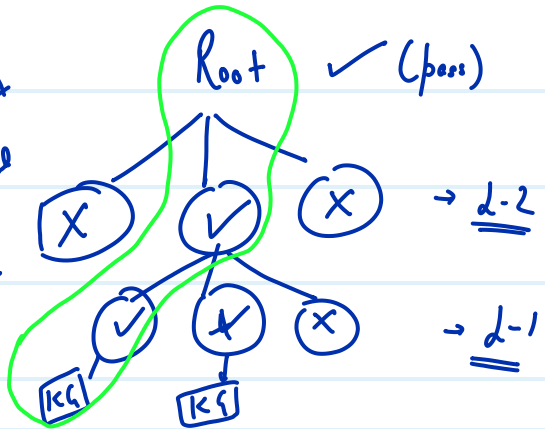






based on Community report. ⇒ query is compared with each community report ⇒ pass / fail.

All the selected nodes taken into account to generate final answer. based on rel. score.



Knowledge Graph Formation

→ Basic prompt → [Graph RAG] → MS

↓

[Github Link]

→ Community Creation [Using Leiden Algo]

2) Langchain KG creation

Steps

① Prompt → to extract → nodes, relationships & properties

② Structured assembly

→ head → "ABC"

→ head-type → "Person"

→ relation → "Conducted_Research_on"

→ tail → "Physics"

→ tail-type → "Concept"

→ Store it in a JSON format

→ Standardisation

→ Nodes → Titled case

→ Relationship → " " replaces with "_"

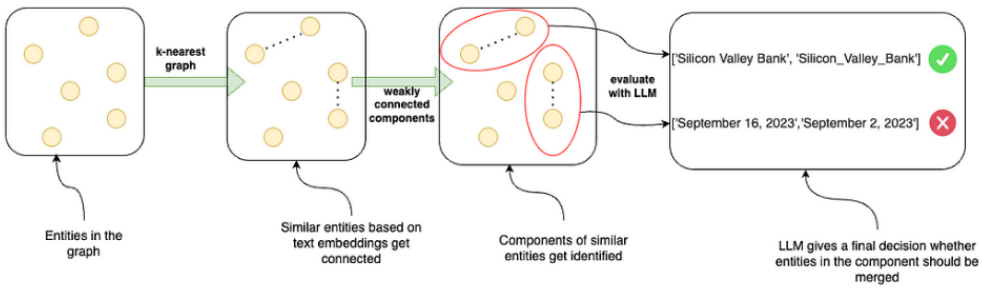
→ Store output in Graph documents.

* No deduplication ?? → in both MS Graph RAG & Langchain !?

Reason :-

> *Overall, entity resolution enhances the efficiency of data retrieval and integration, providing a cohesive view of information across different sources. It ultimately enables more effective question-answering based on a reliable and complete knowledge graph.*

Unfortunately, the authors of the GraphRAG paper did not include any entity resolution code in their repo, although they mention it in their paper. One reason for leaving this code out could be that it is tough to implement a robust and well-performing entity resolution for any given domain. You can implement custom heuristics for different nodes when dealing with pre-defined types of nodes (when they aren't predefined, they aren't consistent enough, like company, organization, business, etc.). However, if the node labels or types aren't known in advance, as in our case, this becomes an even harder problem. Nonetheless, we will implement a version of entity resolution in our project here, combining text embeddings and graph algorithms with word distance and LLMs.



Potential Research Areas

① Identify a specific research area in the Graph RAG pipeline to improve it's performance.

→ Possible areas :-

ⓐ MS Graph RAG relies only on Pure prompt engineering to extract nodes & relationship.

ⓑ Another algo better than Leiden for community creation ??

ⓒ Better retrieval strategy ??

② Core of Graph RAG is KG

ⓐ → Creation of KG from unstructured data using LLM still has a lot of research potential.

→ MS graph RAG relies only on pure prompt engineering.

→ Langchain & LlamaIndex, even Neo4j prefers to have specific pre-defined functions for consistent KG creation.

(b) No robust method for deduplication of nodes in MS Graph RAG, Langchain & LlamaIndex.

(c) Domain Specific pipeline for KG creation.

Example :- KG for Biomedical
[Need Domain Expertise]

Fact :- Creation of KG cost us a lot of money.

$$\text{Quality of KG} \propto \frac{1}{\text{cost}}$$

[proprietary models → expensive (pay as you go), open source models → free (Quality ↓↓)]

Current Variants of KG → RAG

① MS Graph RAG → based only on Graph, Communities
→ require good quality of LLM
→ Cost (↑↑)

② KG Graph + Vector DB ⇒ Supported by Langchain, Neo4j communities,
⇒ helpful in making GenAI apps

Hybrid Search → (a) Results from KG
(b) ⟶ Vector DB

No paper that uses RAPTOR & KG.
even with opensource LLM, ⇒ Better results.

Hypothesis } Framework would be helpful in building GenAI applications.