Agentic Knowledge Systems for Information Traversal/Management

Overview:

A novel approach to knowledge decomposition, management, organization, and retrieval that leverages intelligent agents to manage and traverse document knowledge graphs, moving beyond traditional chunk-based retrieval to topic-specialized expert systems.

Problem Statement

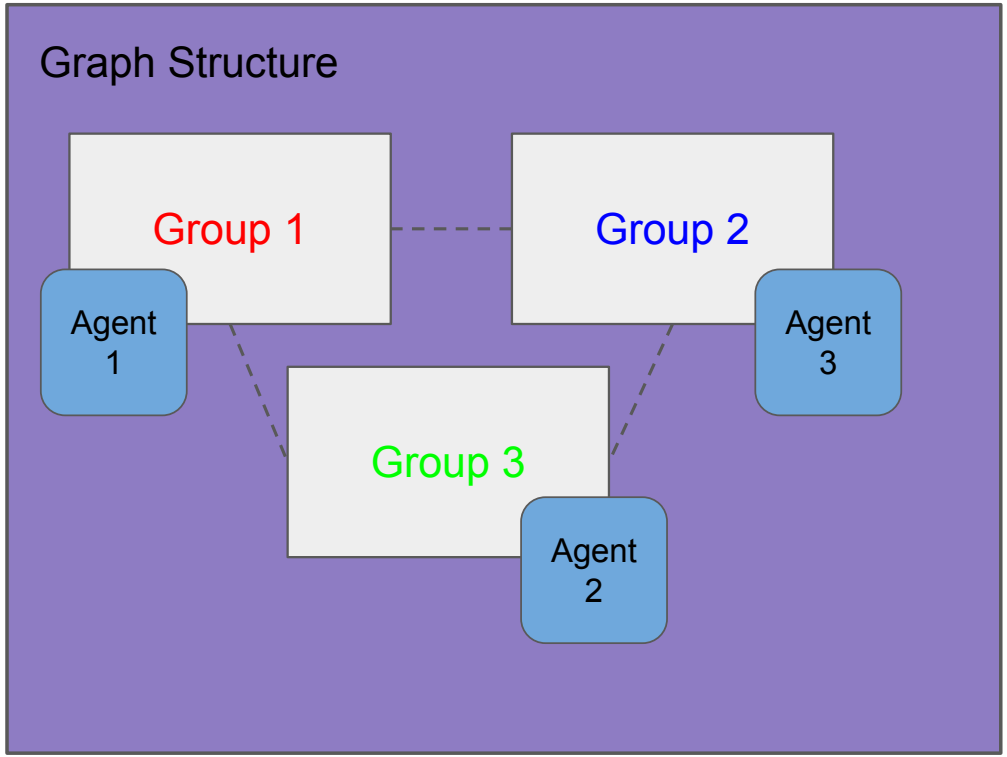
Traditional RAG pipelines suffer from:

- Isolated chunk retrieval without contextual relationships
- Fixed similarity matching that misses nuanced connections
- Lack of domain expertise in information gathering
- Inability to dynamically adapt to new document ingestion
- Requirement of linearly searching all documents/chunks

Key Innovation

Instead of static similarity search across document chunks, we propose agent-driven graph management & traversal where:

- **Intelligent Traversal** allows agents to navigate and gather information across graph partitions and prune irrelevant graph sources
- **Specialized Agents** become experts on specific topic clusters within the document graph (categorized + managed by the agents)
- **Dynamic Graph Formation** creates connections between related chunks using configurable similarity thresholds
- **Adaptive Learning** enables agents to decide what information to ingest and when to collaborate



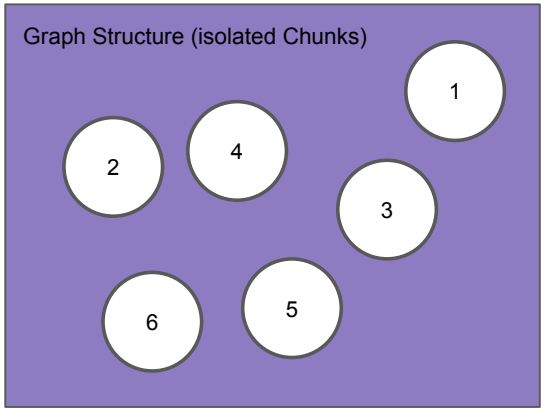
1. Each *group* is a collection of documents (or sub-documents) that are closely related
2. The relationships can be deep or shallow, with hierarchies and overlap
3. An agent (or collection of agents) are assigned to *moderate* over each group. They are limited in function
    a. They can *add* or *remove* elements from a group
    b. They can *modify* the relationship(s)
    c. Depending on the retrieval query, they *give* relevant information (nodes)
        i. They can also extract metadata
    d. They have an “org chart” so they can query for supplemental information

Initialization Phase

Documents Chunked

Chunker

Chunks



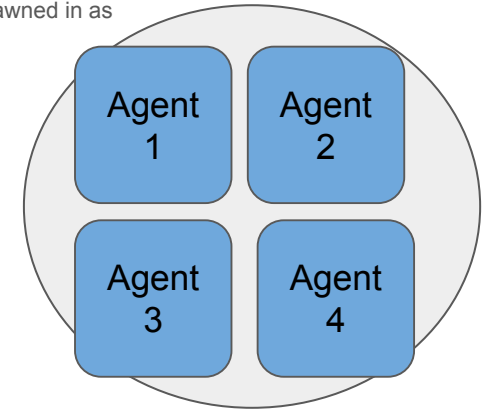
Instantiate Models:

- Parent Model always defined
- Agents can be dynamically instantiated or statically (defined k agents or spawned in as needed)

Config

Process ran

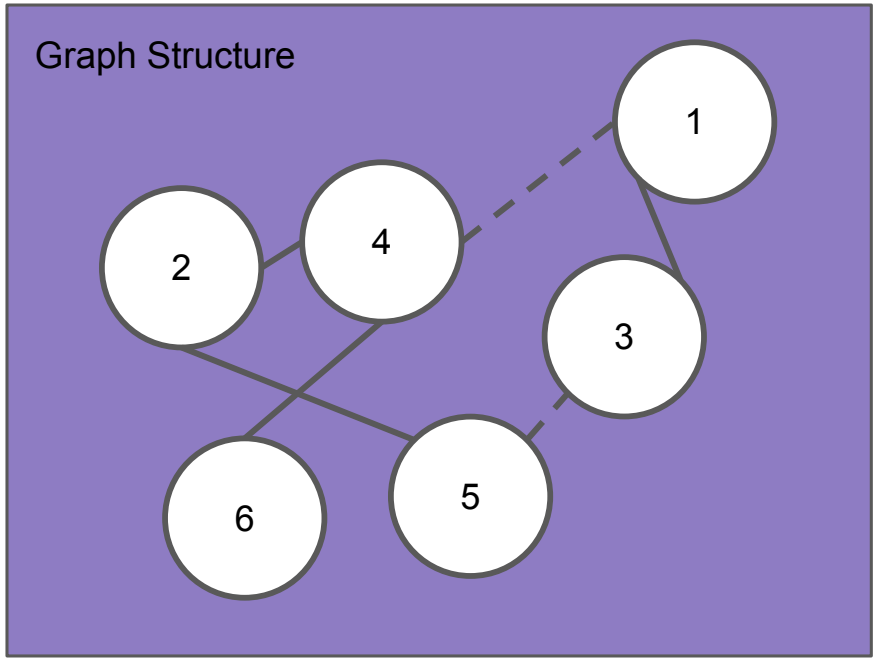
Parent LLM



Intercommunication between agents (knowledge *and* agent topology are both graph-based)

- Via MCP, or other *protocol*

Graph Edge Formation Phase
(cross-encoder, Cosine Sim, etc.)



α level decides what edges can be kept or not (calculate similarity between all chunks, use alpha to fine-tune recall vs precision)

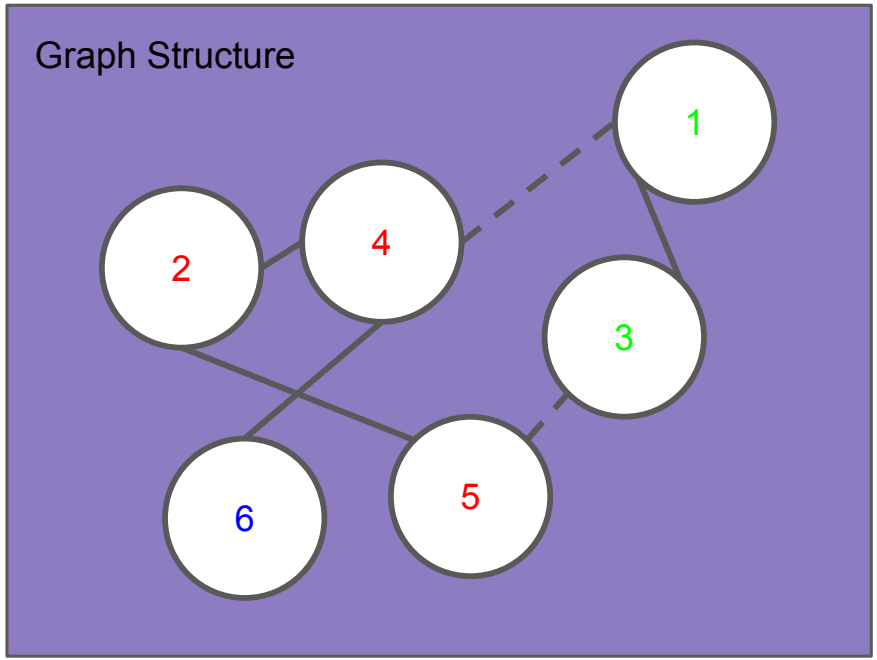
Graph Partition Phase

Parent LLM

"You are a topic identifier expert … Look at the following chunks and **assign them a topic label labeled 1 - k** based on similarities between isolated chunks"

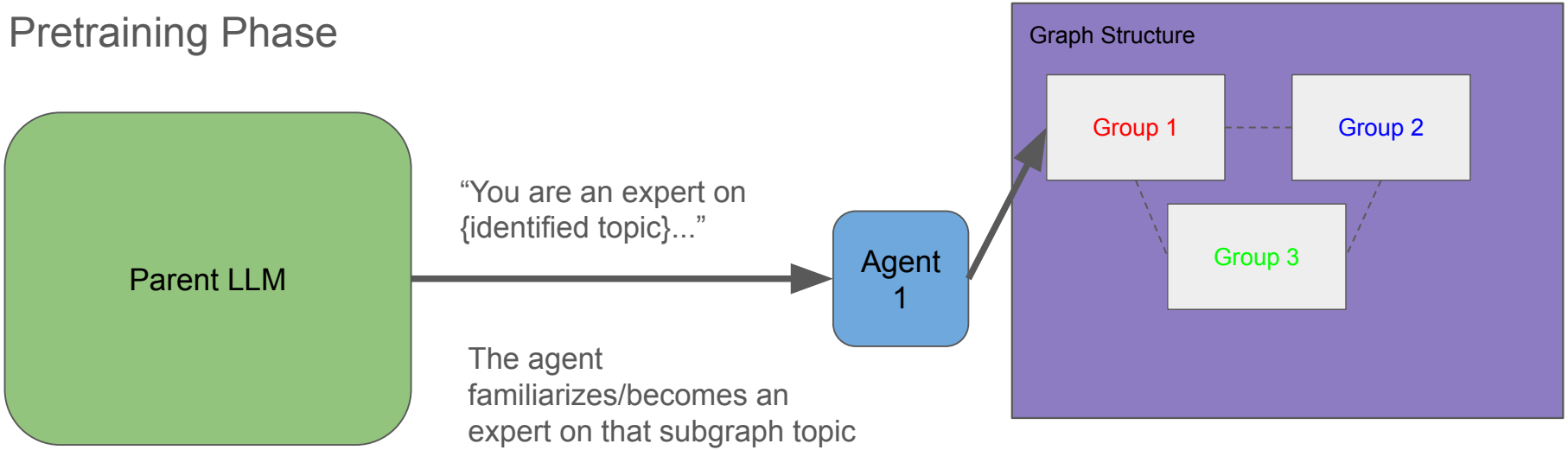
"You are a topic identifier expert … Look at the following chunks and **decide how many topics are being discussed**. Then label them based on similarities between isolated chunks"

← (Static vs Dynamic)



Alternate: LDA, K-means, Density scan for topics/topic extraction

Pretraining Phase



For each agent:

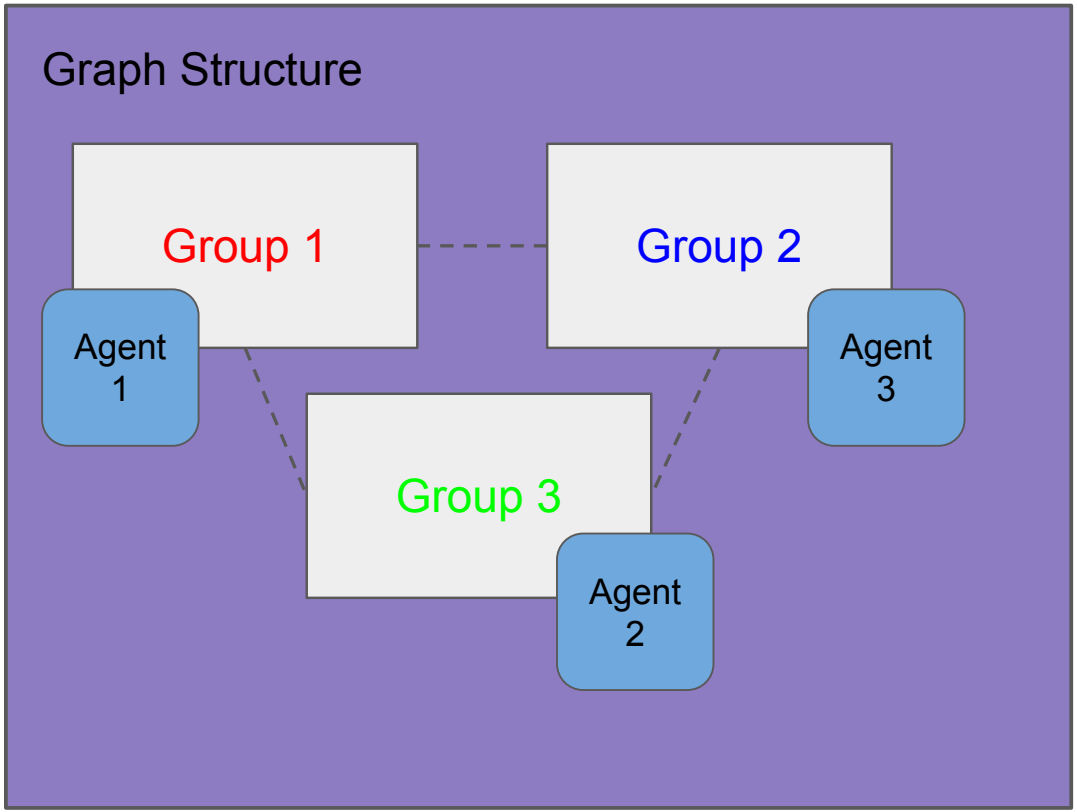
```
agent.assign(subgraph)
agent.prompt(parent.prompt_agent)

# fine tune on subgraph data
agent.train(subgraph)
```

Questions for us:

- How is fine-tuning done?

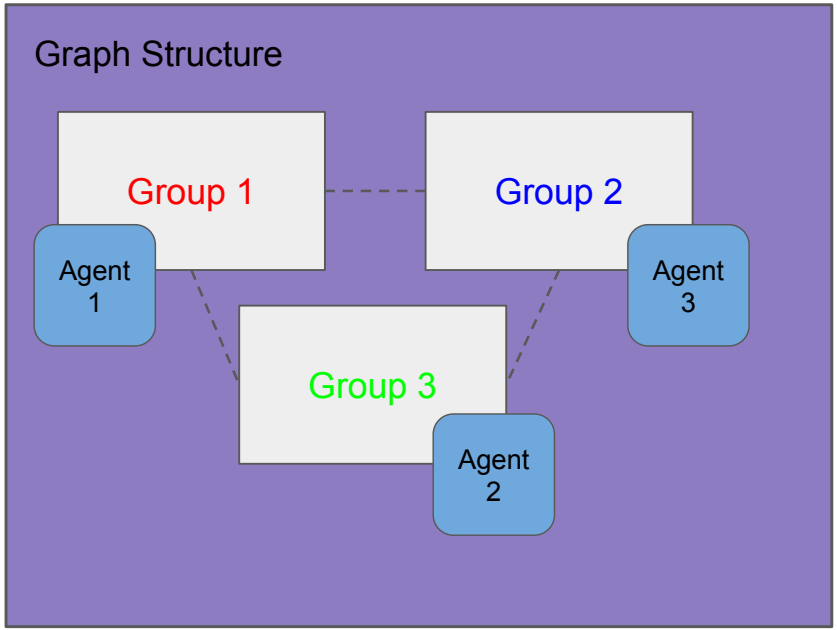
Pretraining Complete State



Inference State - (Suppose Ingested documents are math textbooks with overlapping topics)

Query = “What is a fixed point iteration method?”

```
For each agent:
        If agent.expert_on_topic(query):
                Data = agent.gather_info(query)
                agent.generate(data)
```



Inference State - (Suppose Ingested documents are math textbooks with overlapping topics)

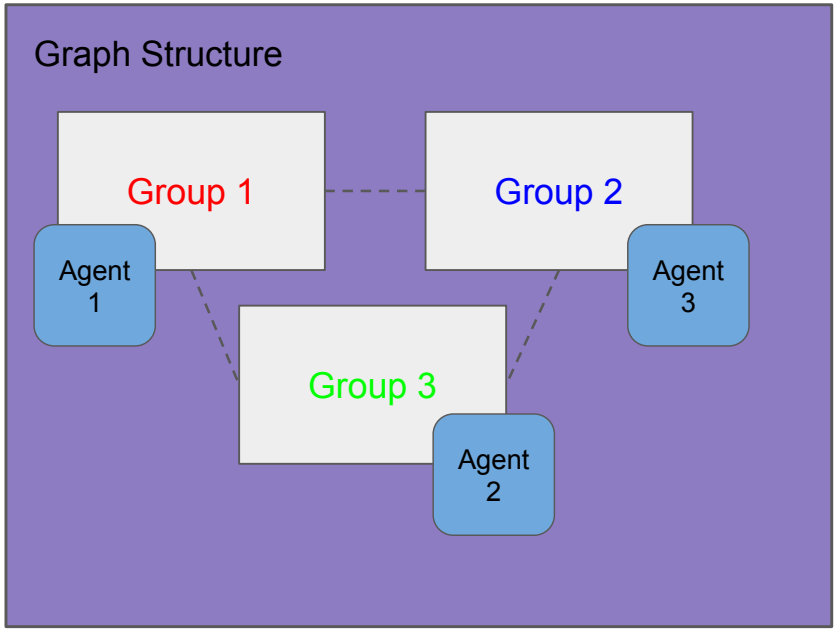
```
Def Gather_Info():
        Stack = []

        # collect agents that contain possible info
        For each n in subgraph[agent_id].neighbors:
                If n.contains_info(query):
                        stack.add(n)

        While stack:
                query_agent(stack[-1])
                stack.pop()
```

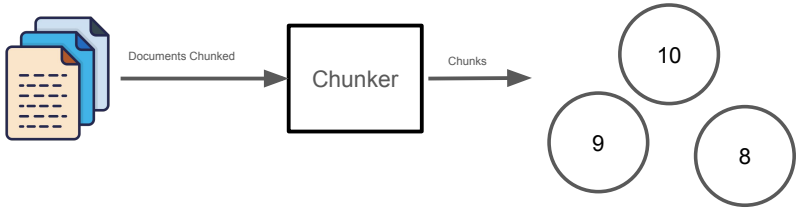
Questions:

- How do we transition from inference agents for info to graph traversal between groups?
- How can agents control traversal?

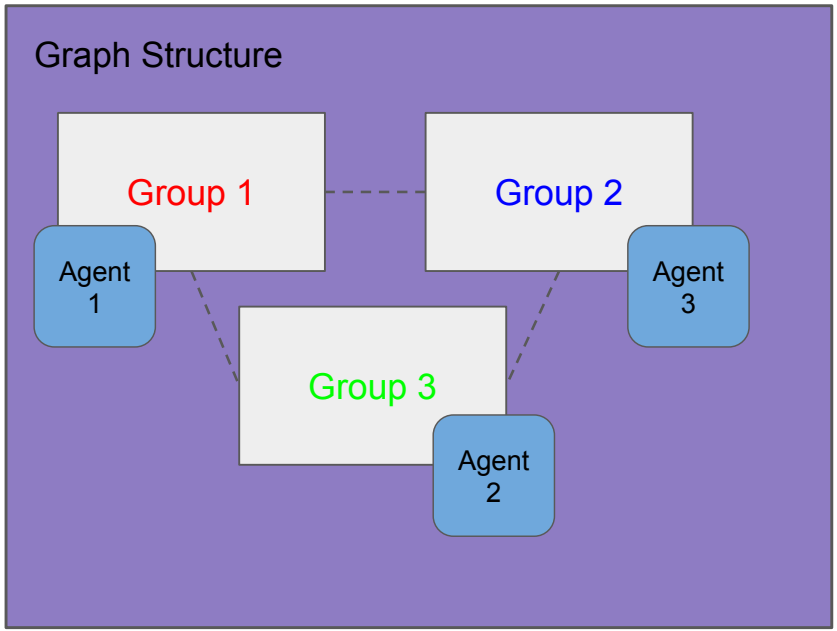


- The most simplest is a linear non-relational database that contains all of our domain knowledge and the agent does a lookup over all entries and chooses the *most relevant* ones
- Beyond this, text needs to be broken down into *units of information*
    - Some will have hierarchical groups, others may be related to edges. Each subgraph can be dynamically defined *given the information present in the subgraph*

Document Ingestion Phase



For each agent:
    For each chunk:
        If agent.can_add(chunk) and
          chunk not taken:
            agent.ingest(chunk)
            subgraph[agent_id].reformat()

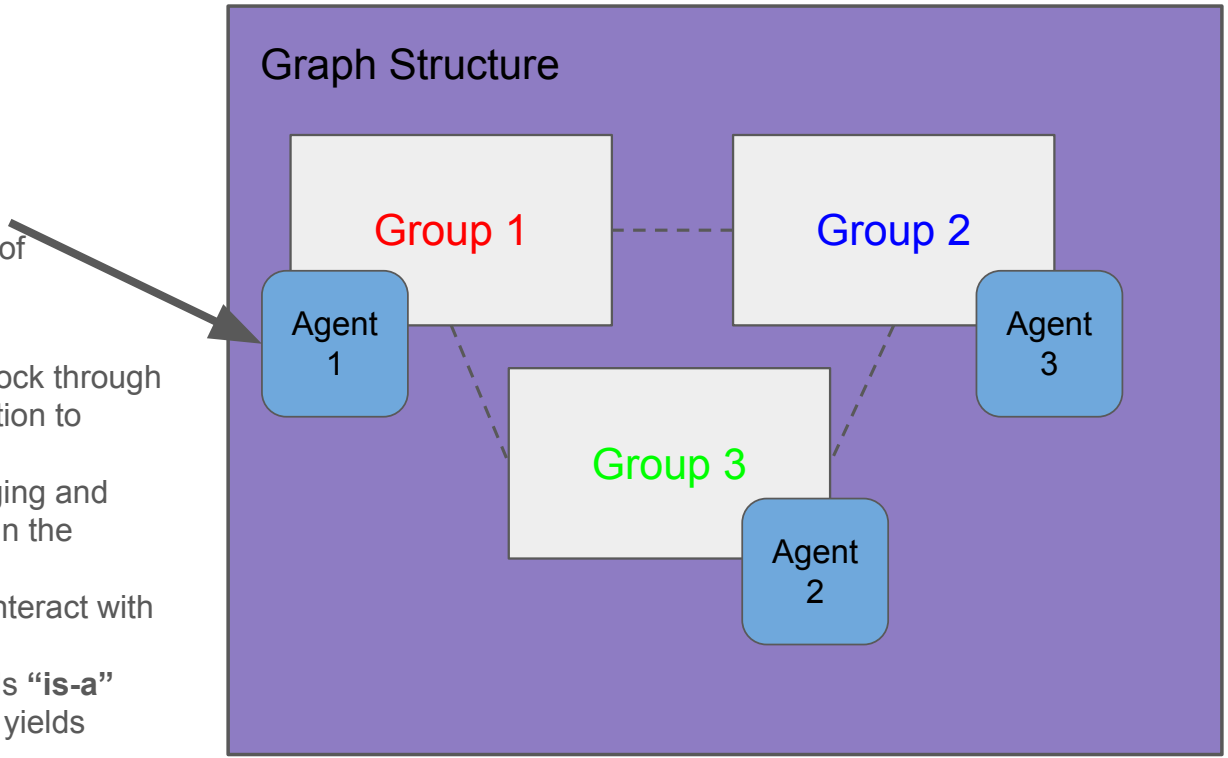


Document Ingestion Phase - can_add()

"Does your graph contain content related to the chunk below and could it enrich your understanding of the topic?: {chunk}"

What other capabilities can we unlock through letting agents dictate what information to learn?

- *WHEN* will they start managing and interacting with information in the pipeline?
- *WHAT* will require them to interact with each other?
- intra-group *information* yields **"is-a"** relationship and inter-group yields **"has-a"** relationship



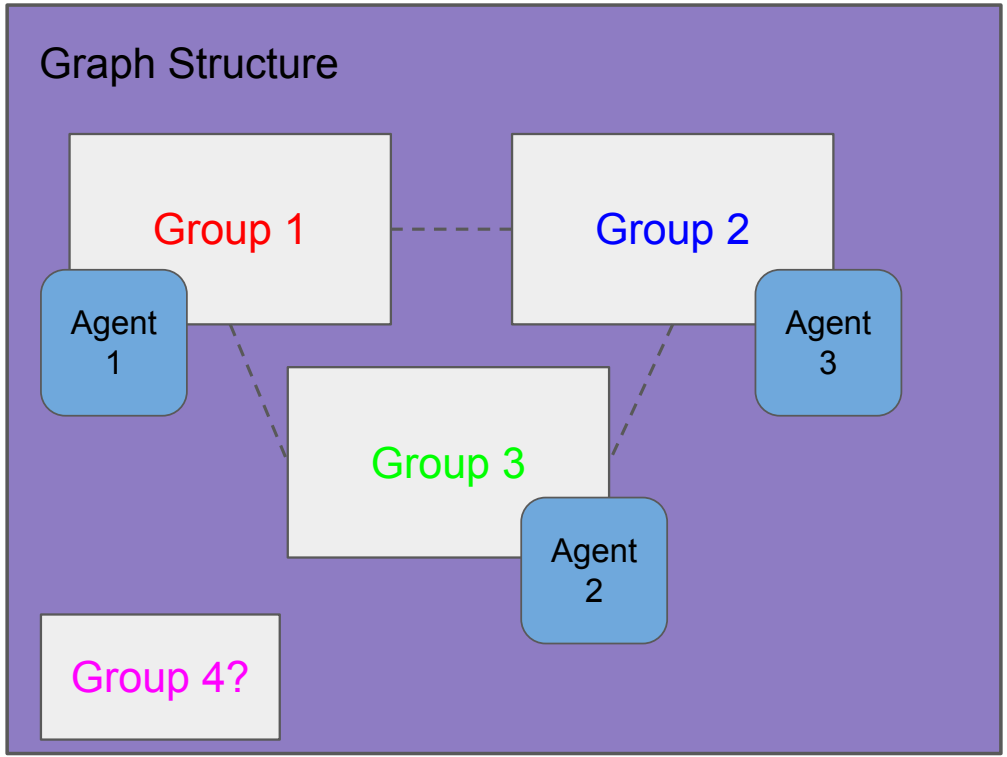
Model Drift/Graph Drift

Detect when graphs are becoming noisy with irrelevant information

Refactor graphs and agents once this happens (restart from graph partition)

Since agents are experts NOW, we can allow agents to interact with each other to what information they should take and are experts on (or if they need a new agent)

- A top-level agent that derives the topics and/or ontologies and distributes it - this causes the agents to spawn and take ownership and pull the relevant information to create the subgraph



Some questions:

1. What information about the network does the agent need to have
    a. Instead of doing a node walk, we rely on the agents to give information
        i. ie how does it know to *activate* other subgraphs to retrieve information?
2. The scale problem of having disparate information
    a. If I have many unrelated documents, then i will suffer from having lots of agents that may not directly interact. How do I address this
        i. I can't do a linear search every time
3. We are looking at this as “i have a question, give me some resources and a response”
    a. But what if someone just wants to see some information (e.g. a network diagram of the nodes, etc.) - apart from a viz, what else can we do?
        i. Why not show the whole graph (eg Obsidian) and highlight the nodes and/or subgraphs that have the relevant information? The composite sum of these units of information = knowledge (to answer the query).