Introduction to Agentic AI

-- AI Agent Memory Design

Instructor: Guangjing Wang

guangjingwang@usf.edu

This Lecture

- Short-term and Working Memory
  - Context Window
- Long-term Memory
  - Retrieval Augmented Generation
- Memory Design in Research

Short-term Memory: Context Window

- DeepSeek V3
  - The maximum sequence length is 4K during pre-training
  - Context Length Extension:
    - Stage 1 pre-training: maximum context length is extended to 32K
    - Stage 2 pre-training: maximum context length is extended to 128K
  - Context Length Extension Method: YaRN [1]

- Claude Opus 4.6: 1 million (1M) token context window.

[1] YaRN: Efficient Context Window Extension of Large Language Models https://arxiv.org/abs/2309.00071

Review: Position Encoding

- Attention mechanism is blind: process all tokens in parallel;
- Without position encodings, model only see a bag of words:
  - The dog bit the man
  - The man bit the dog
- The position encoding is as a ruler the model uses to measure the relative distance between words.
- Position encoding enables valuable supervision for **dependency modeling between elements** at different positions of the sequence.

Review: Position Encoding

- The vanilla Transformer uses Sinusoidal Function for **absolute position encoding**

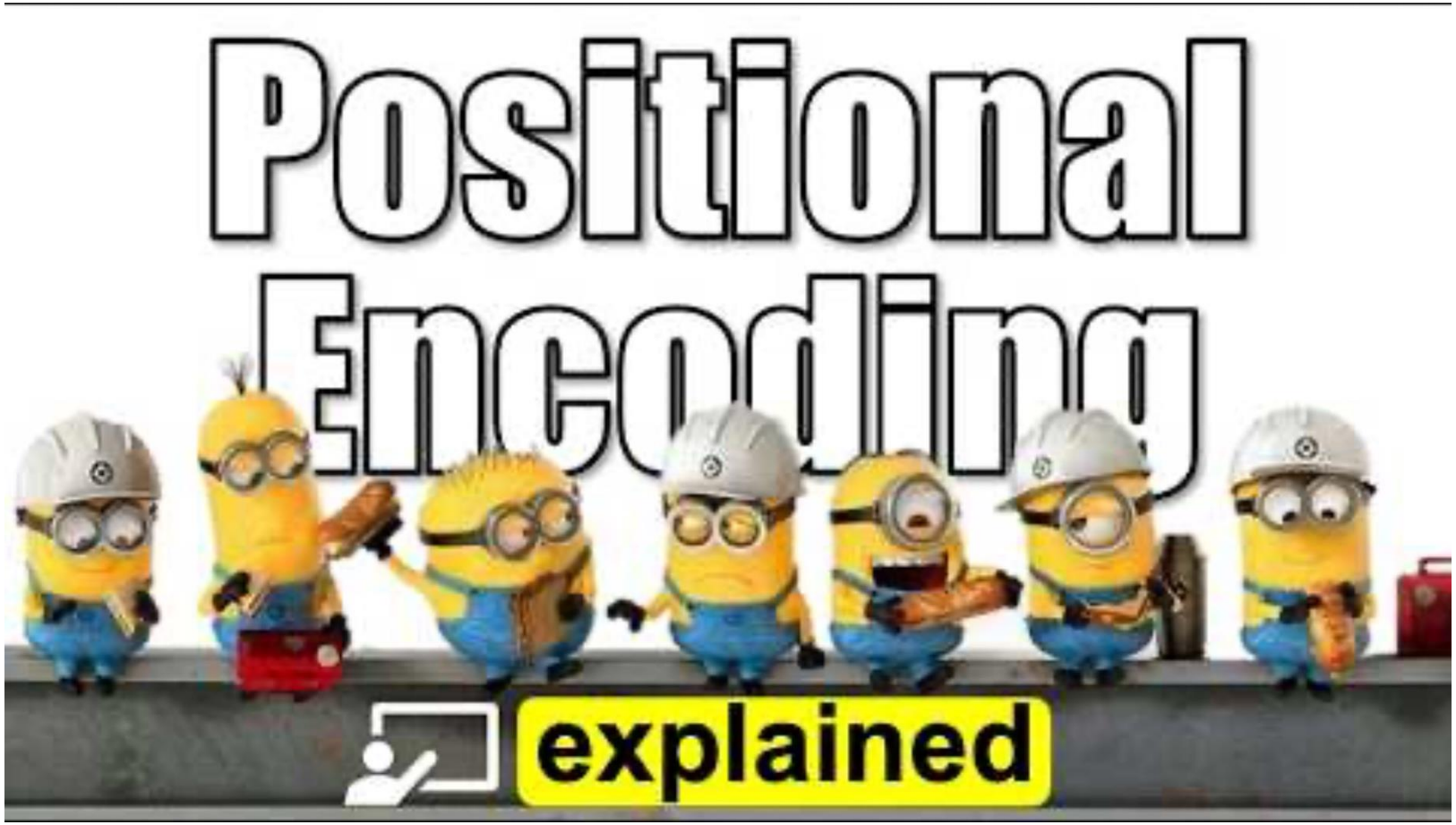
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

- DeepSeek V3 uses **relative position encoding** (RoPE [1]). Initially, the model is given a ruler that is exactly 4,000 units (tokens) long.

- However, when the model encounters a token at position 4,001, it has no learned representation for that location. The model has never optimized its weights for position indices that high.

[1] RoFormer: Enhanced Transformer with Rotary Position Embedding https://arxiv.org/abs/2104.09864
[2] A Length-Extrapolatable Transformer https://arxiv.org/abs/2212.10554
[3] Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation https://arxiv.org/abs/2108.12409

Positional Encoding and Interpolation



https://www.youtube.com/watch?v=SMBkImDWOyQ

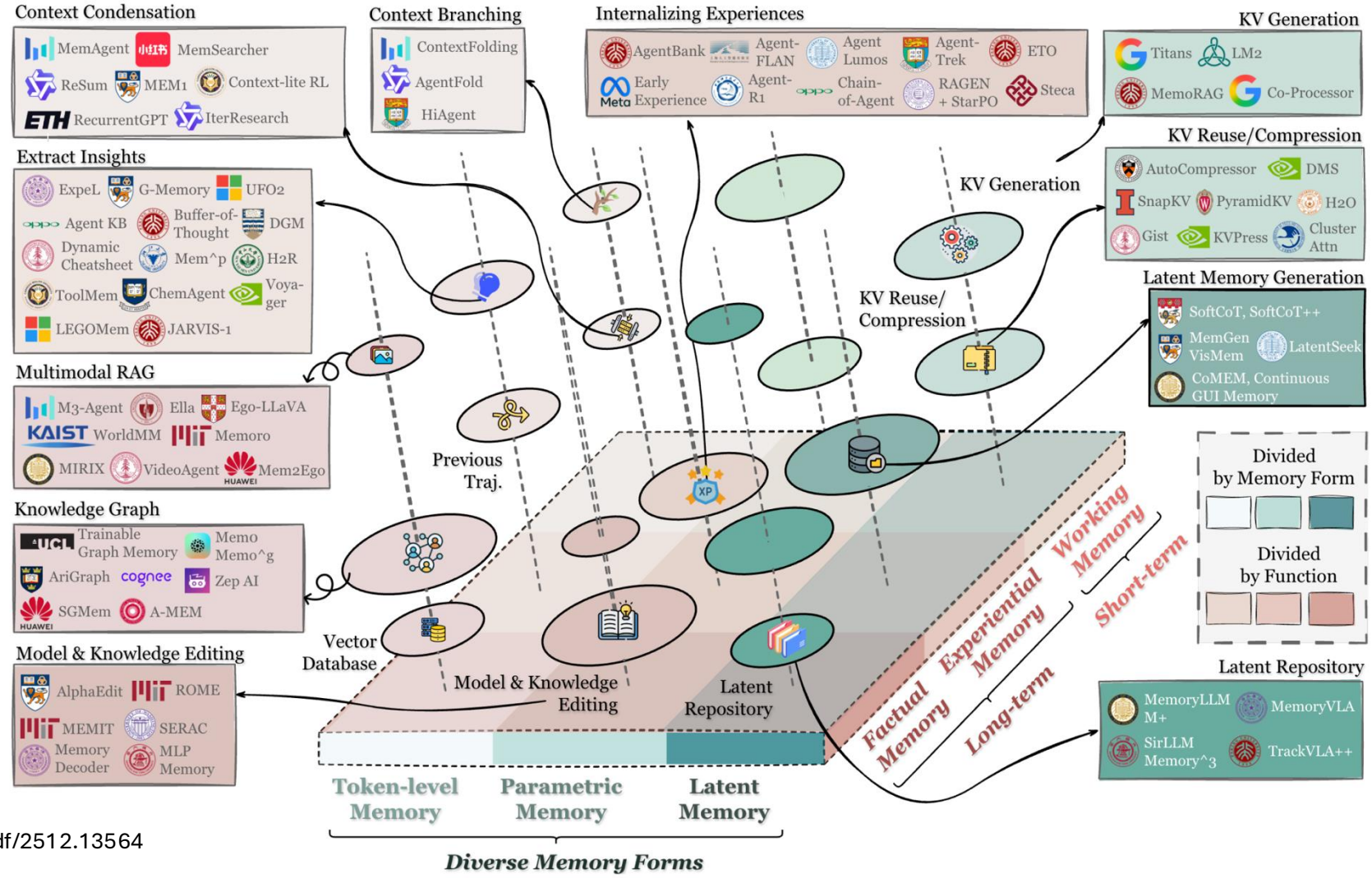
Extend the Context Window

- Extend the context window by adding more memory?
  - The limitation is **structural**, not just computational.
  - The attention mechanism relies on the query and key vectors interacting. This interaction includes the positional information. If the position encoding doesn't generalize, the attention mechanism cannot determine which previous tokens are relevant to the current one.

- Main idea of YARN [1]: Interpolation with multiple conditions
  - If you want to double the context window, you can tell the model that position 1 is position 0.5, and position 2 is position 1.0.
  - By slowing down the "rotation" or "frequency" of the position encoding, you map the new, longer sequence onto the old, familiar ruler.

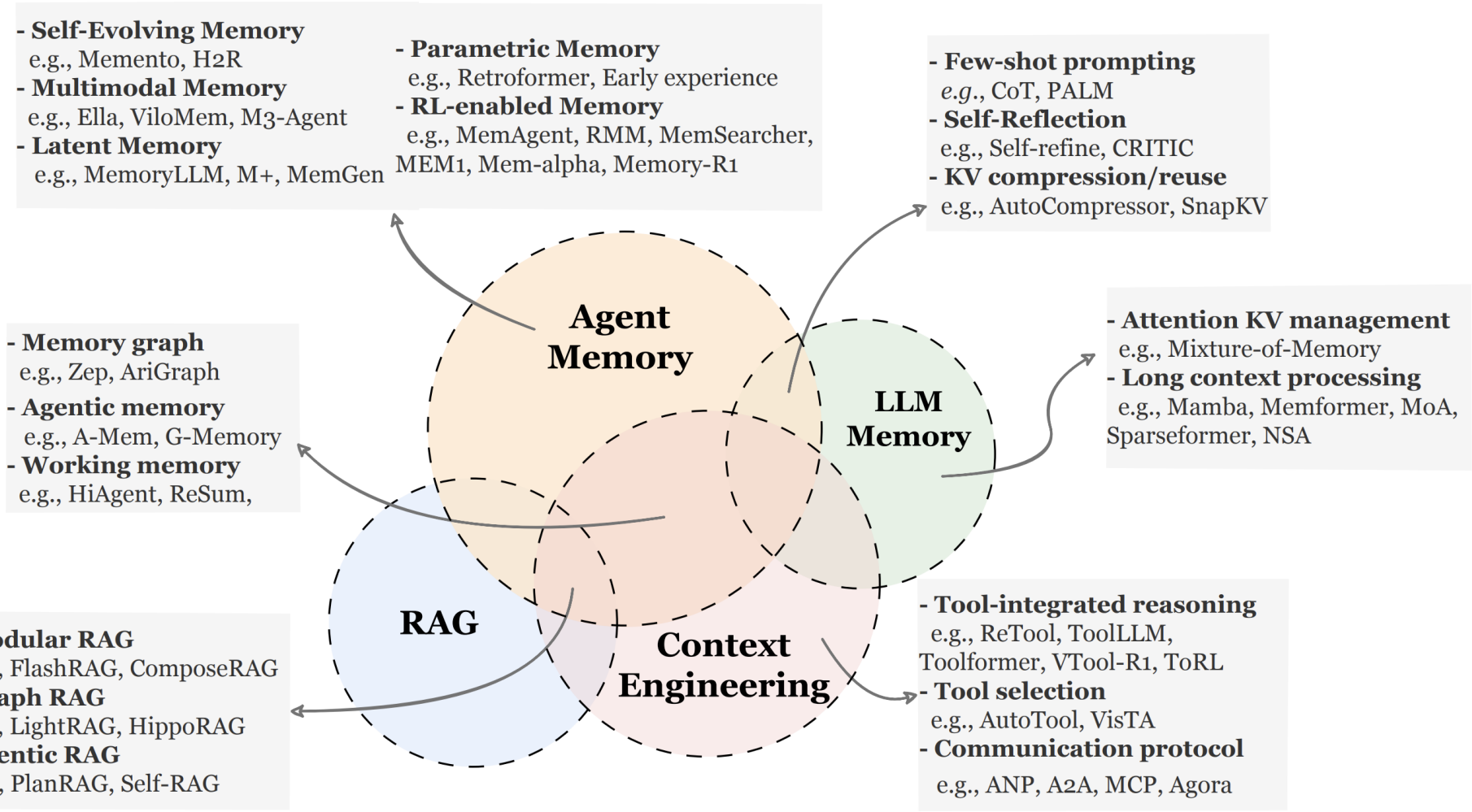
[1] YaRN: Efficient Context Window Extension of Large Language Models https://arxiv.org/abs/2309.00071

Memory in the Age of AI Agent



https://arxiv.org/pdf/2512.13564

Related Concepts



https://arxiv.org/pdf/2512.13564

Conceptual comparison

- LLM Memory: the architectural optimizations
- Context Engineering: the transient resource management
- RAG: static knowledge access
- Agent Memory: maintaining a persistent and self-evolving cognitive state that integrates factual knowledge and experience

Retrieval Augmented Generation (RAG)

1. **Ingestion & Indexing:** Unstructured data (PDFs, emails, docs) is ingested, broken into smaller "chunks" and converted into numerical representations called **vector embeddings.**
2. **Vector Database Storage:** The embeddings are stored in a specialized vector database, which allows for fast, efficient semantic search.
3. **Retrieval:** When a user asks a question, the system converts the query into a vector and searches the database for chunks of data that are semantically similar (conceptually, not just keyword matches).
4. **Augmentation:** The retrieved snippets of information are added to the user's original prompt (i.e., they are appended to the context).

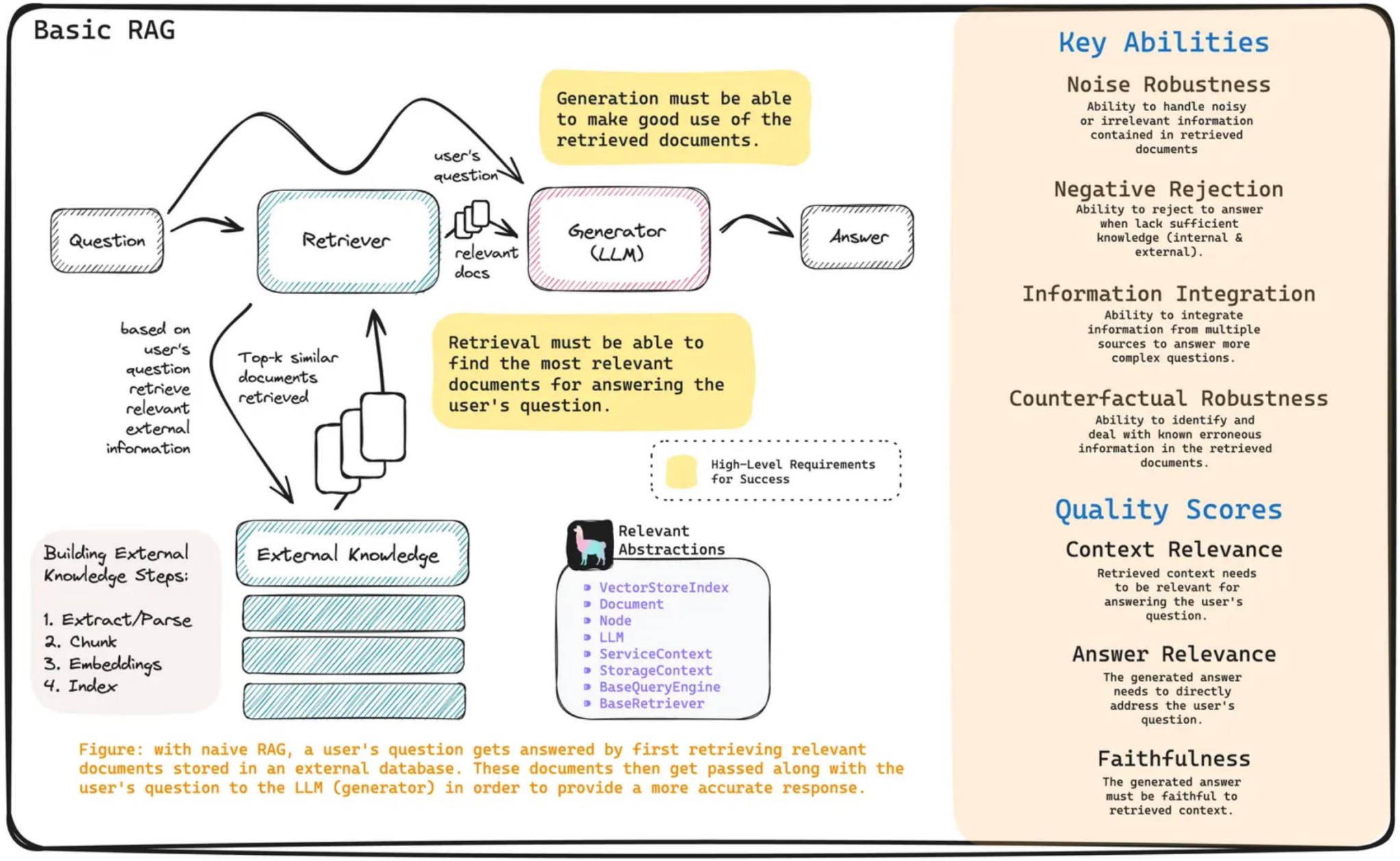
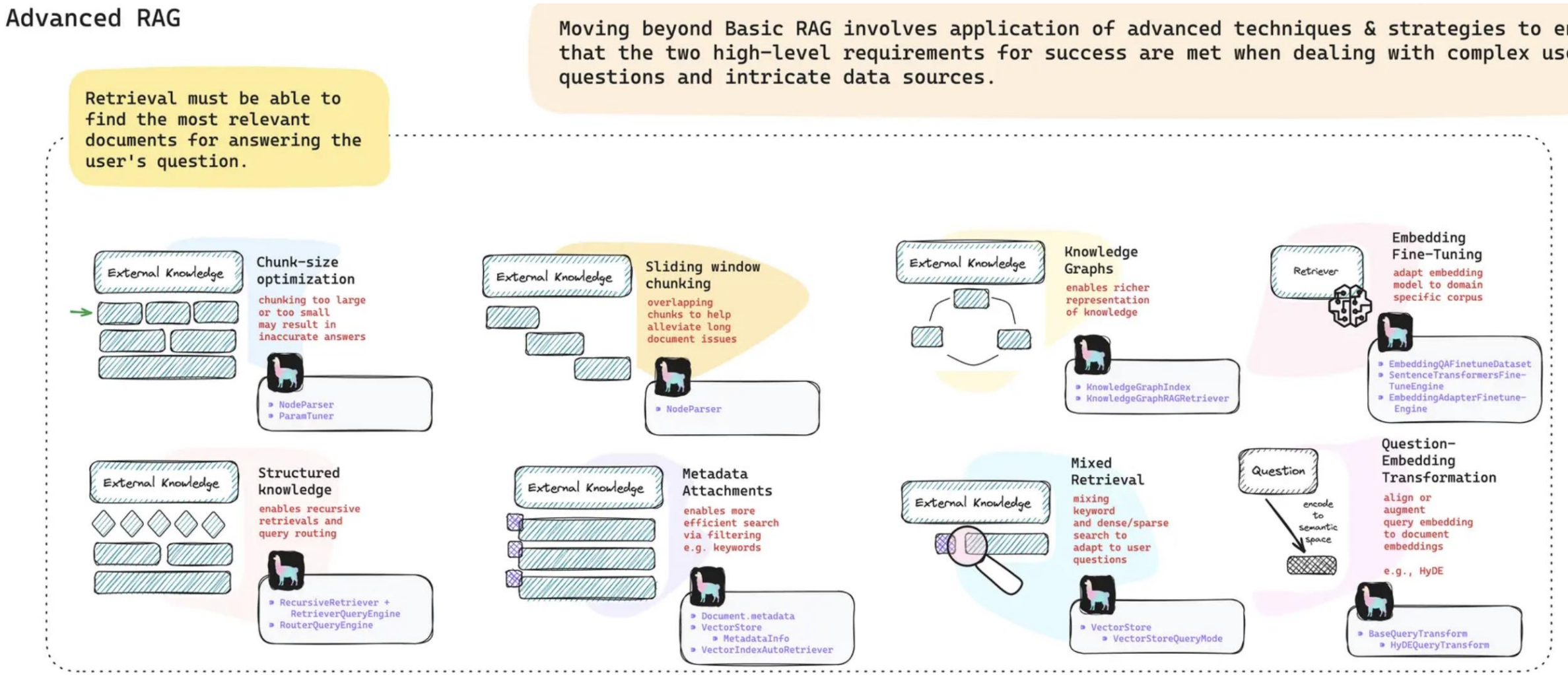


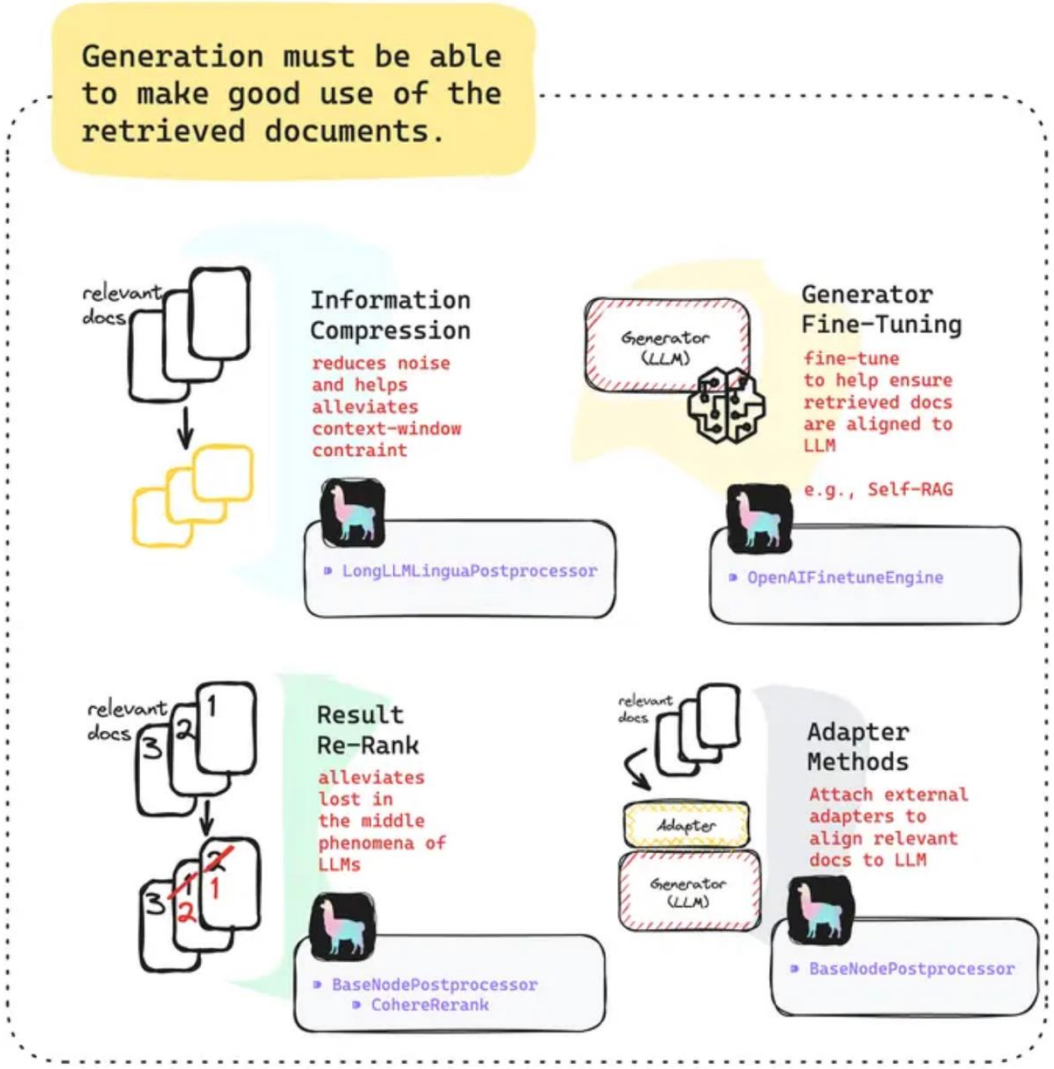
Figure: with naive RAG, a user's question gets answered by first retrieving relevant documents stored in an external database. These documents then get passed along with the user's question to the LLM (generator) in order to provide a more accurate response.

https://www.genaiknowledge.info/post/agentic-rag-for-financial-memo-generation

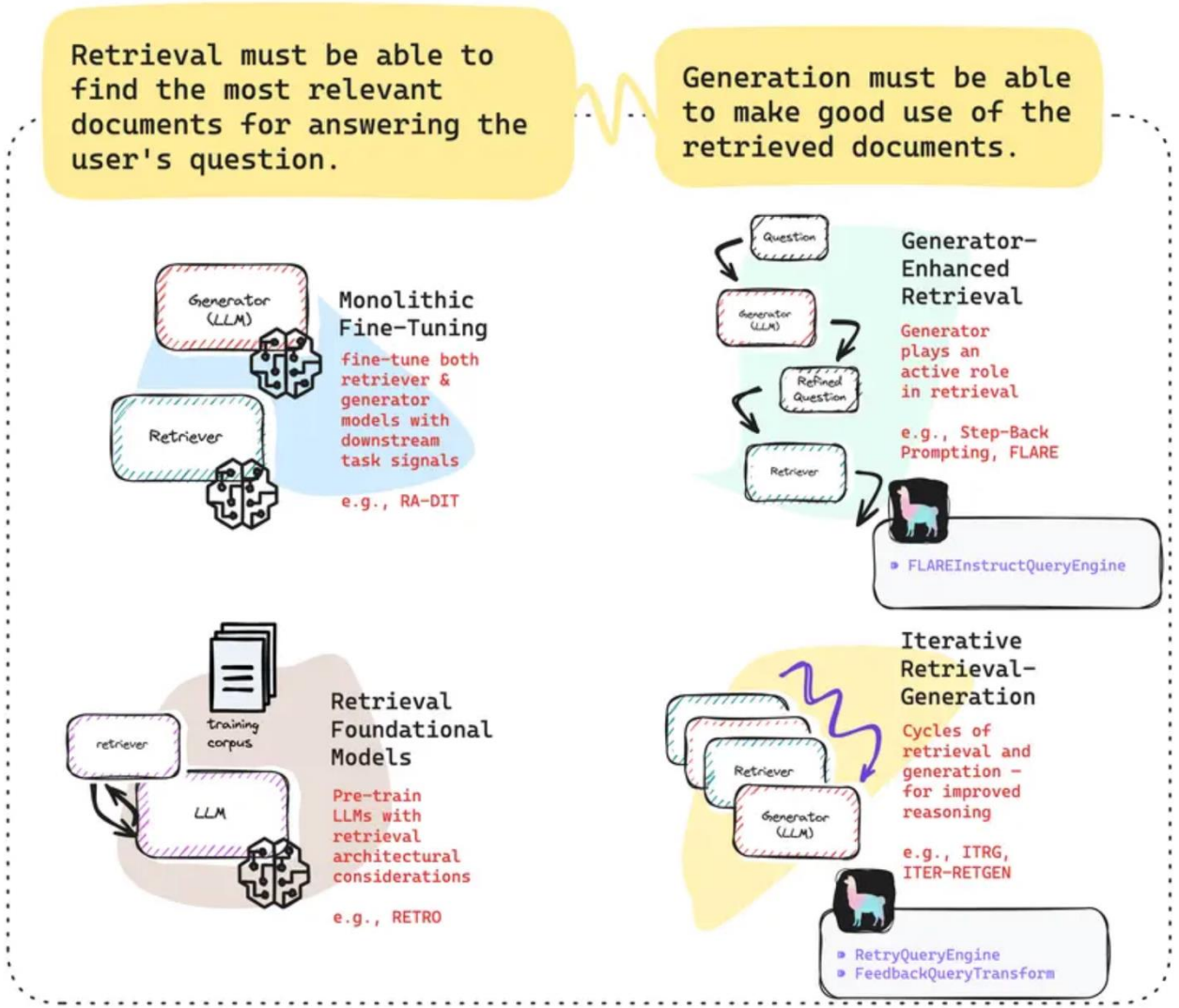
Advanced Techniques for Retrieval in RAG



Advanced Techniques for Generation in RAG



Techniques for Both Retrieval and Generation



A RAG Application Example: Background

- In the world of high-finance, the speed and quality of an investment memo can be the deciding factor in a multi-million-dollar decision.
- These memos, internal reports that determine whether to buy, hold, or sell a company's stock, are the lifeblood of investment strategy.
- Traditionally, a team of analysts spends weeks wading through a sea of data: annual/quarterly reports, competitor credit reports, investor relations transcripts, and historical internal opinions.
  - $ 20,000 per ppl per month * 5 ppl * 1 month = $100,000

Example: a Manual Memo Generation (Cont.)

1. **Data Collection:** Manually gathering dozens of disparate documents.
2. **Reading:** An analyst painstakingly reads through hundreds, if not thousands, of pages.
3. **Mental Synthesis:** The analyst begins to form initial ideas and a core thesis.
4. **Composition:** Following a predefined structure, the analyst composes the memo segment by segment.
5. **Evaluation:** The draft is circulated among peers and senior partners for critique.
6. **Downstream Tasks:** presentations (slides), internal briefings (web pages), and sent via email to stakeholders.

Example: RAG + Agents Solution (Cont.)

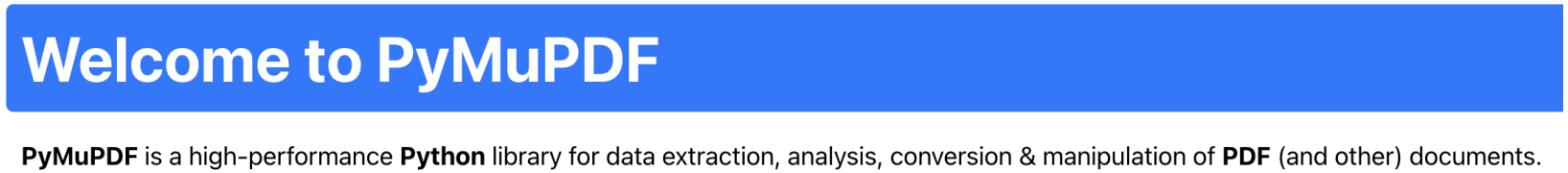
- Asking a generic LLM Chatbot to "write an investment memo for Company X" using a massive dump of documents would yield a generic, unreliable, and unactionable summary.

- Retrieval Augmented Generation (RAG) for the Core Analysis
  - Pre-define complex queries (acting as the analyst's research questions)
  - Transform all the relevant financial documents into chunked vectors

- Agentic Workflows for Post-Generation Tasks:
  - LLM-powered agents to handle the downstream tasks: slides, emails

Example: Engineering Procedures (Cont.)

- Extracting Information from PDF: complex layouts with columns, headers, footers, tables, and images.



- Breaking down documents:
  - Fixed-size Slicing: e.g., chunk size 1024 tokens.
  - Semantic chunking: find natural semantic breaks in the text [1].

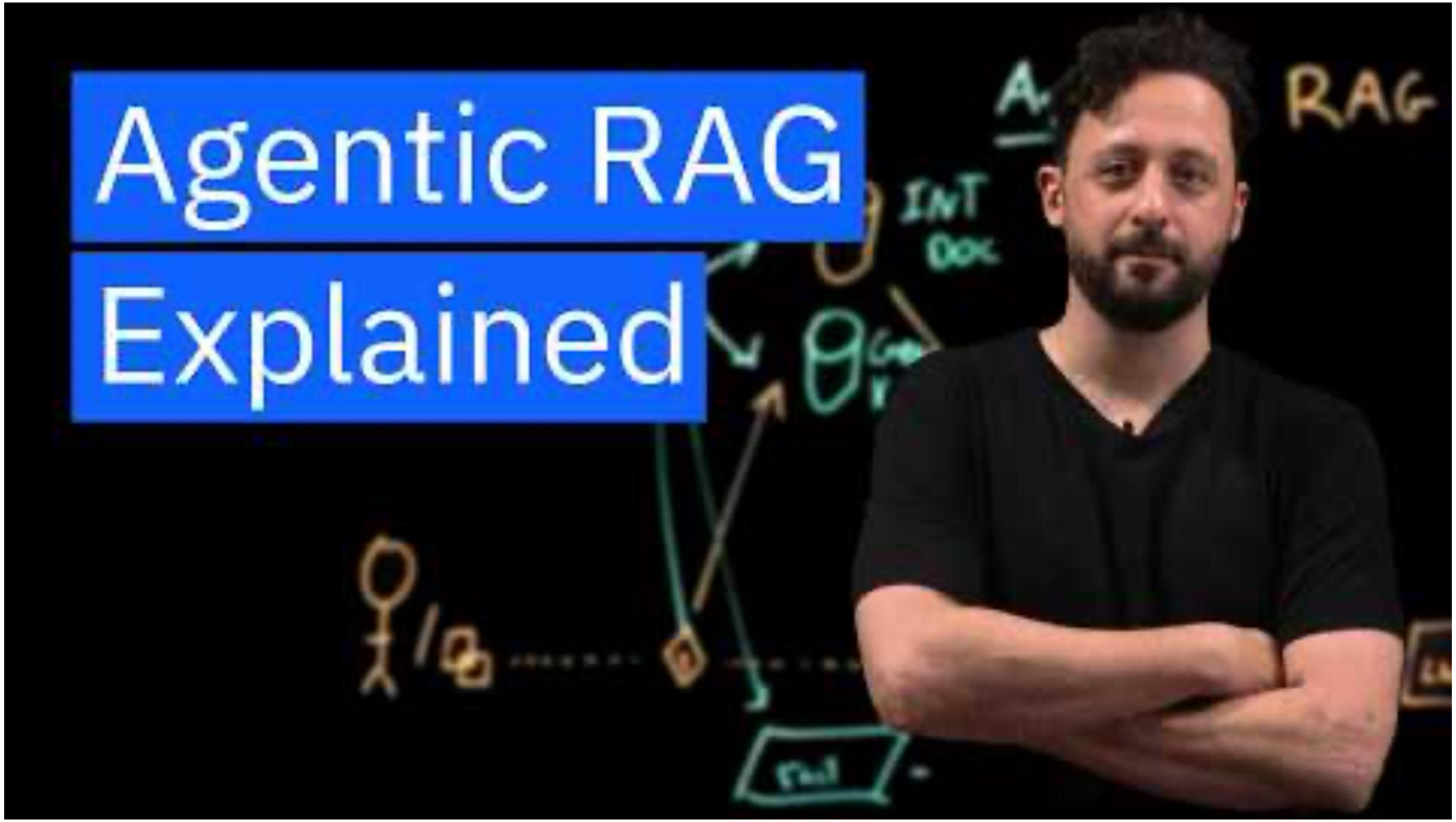
- Metadata Extraction: enrich each chunk with structured metadata
  - Instructor Library: Extract structured data from any LLM with type safety [2]

[1] https://developers.llamaindex.ai/python/examples/node_parsers/semantic_chunking/
[2] https://python.useinstructor.com/

Example: Engineering Procedures (Cont.)

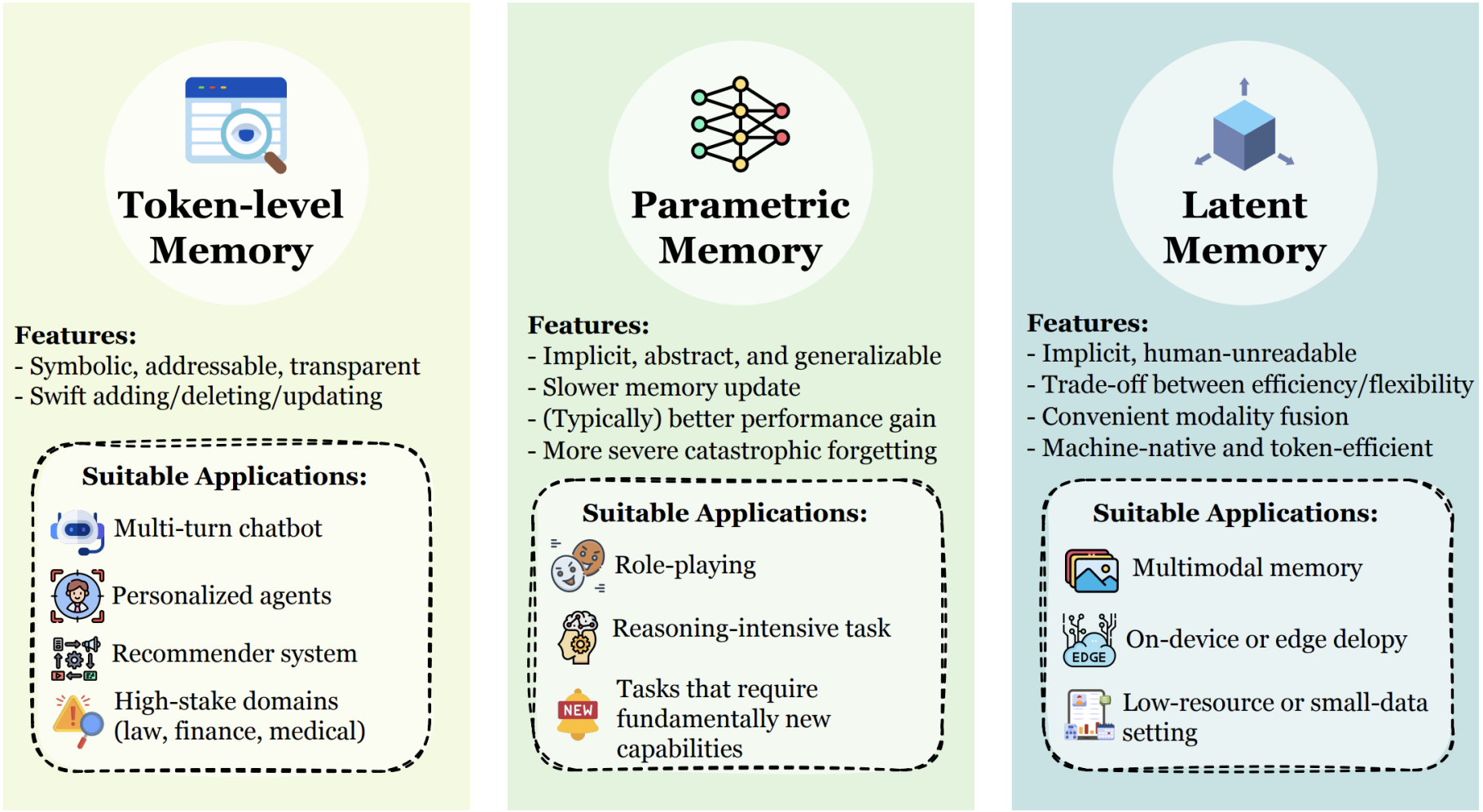
- Special Format Processing:
  - Table: (1) convert table to a clean format like Markdown; (2) use an LLM to generate a human-readable summary.
  - Images: OCR; Multimodal LLM to interpret a chart and meaning.
- Embedding and Reranking:
  - Converting the enriched chunks into numerical vectors using a powerful open-source model like BAAI/bge-large-en-v1.5;
  - A reranker model (BAAI/bge-reranker-large) takes the top results from the initial search and re-orders them based on a more nuanced understanding of the query.
- Storage:
  - ChromaDB Vector Database;
  - MangoDB: track the original files, processing status, user information, and references to their chunk.

Agentic RAG



https://www.youtube.com/watch?v=0z9_MhcYvcY

Memory Paradigms for LLM Agents



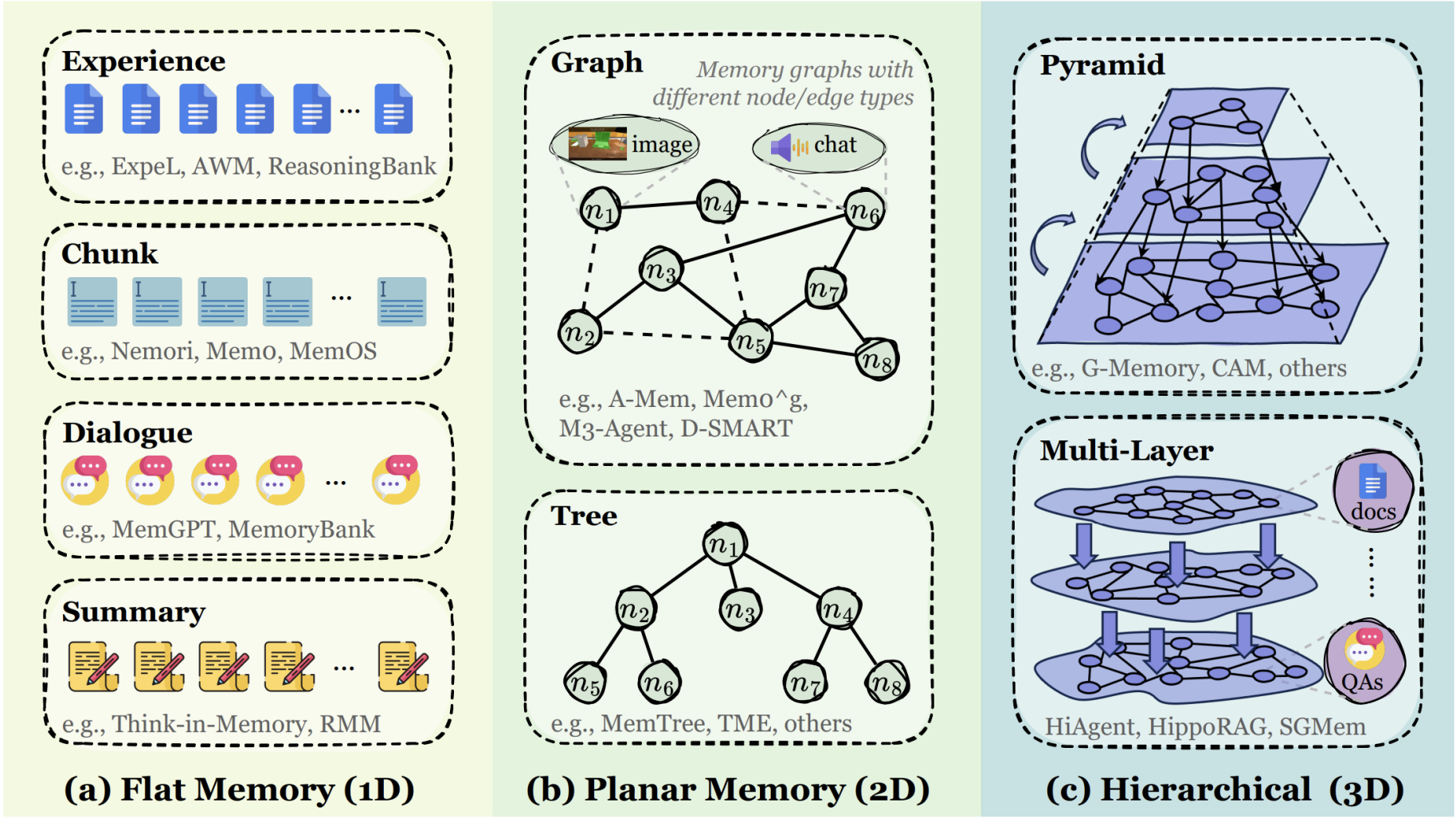
Token-level Memory (1)

- Token-level memory stores information as persistent, discrete units that are externally accessible and inspectable.

- The token here is a broad representational notion: beyond text tokens, it includes visual tokens, audio frames—any discrete element that can be written, retrieved, reorganized, and revised outside model parameters.

- Topology of Token-level Memory
  - 1D: sequences or bags of units
  - 2D: graphs, trees, tables
  - 3D: Hierarchical layers

Token-level Memory (2)



Parametric Memory

- Internal Parametric Memory:
  - Memory encoded within the original parameters of the model (e.g., weights, biases).
  - These methods directly adjust the base model to incorporate new knowledge or behavior.

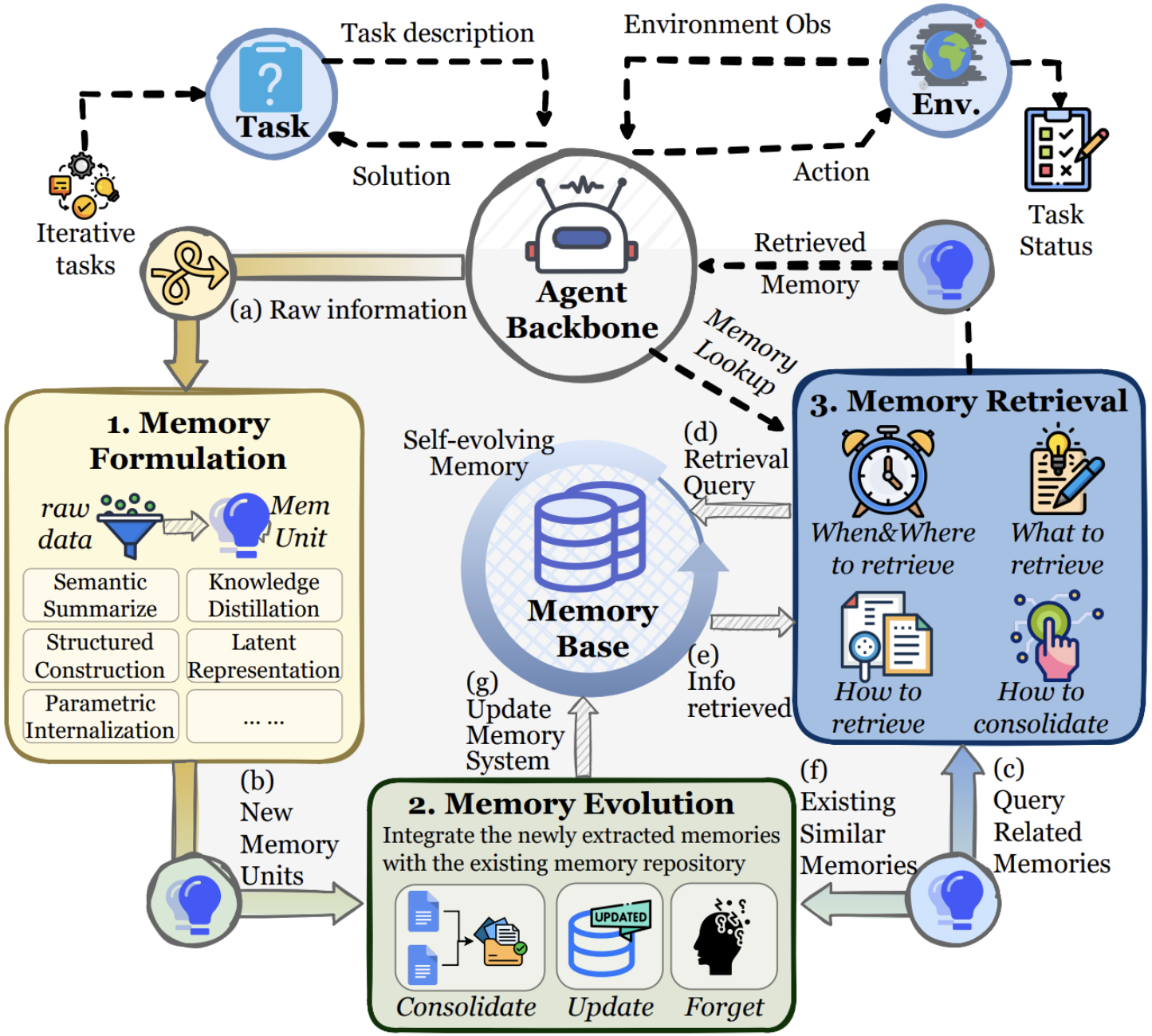
- External Parametric Memory:
  - Memory stored in additional or auxiliary parameter sets, such as adapters, LoRA modules, or lightweight proxy models.
  - These methods introduce new parameters to carry memory without modifying the original model weights.

Latent Memory

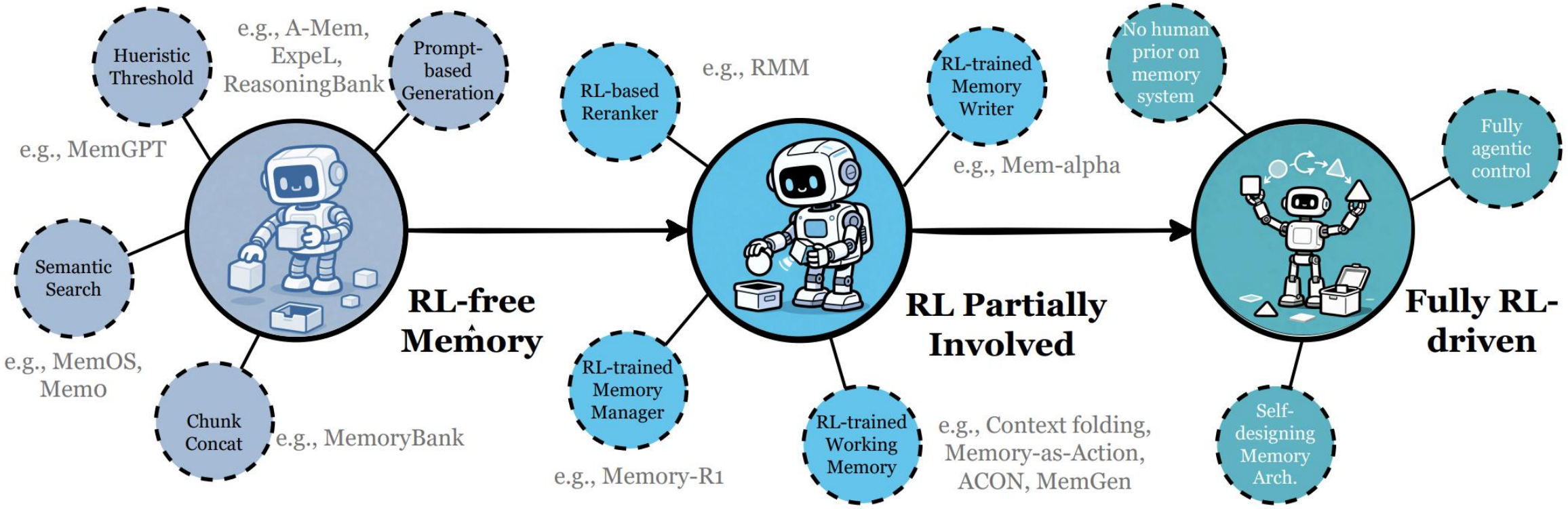
- Latent memory refers to memory that is carried implicitly in the **latent representations** (e.g., KV cache, activations, hidden states, latent embeddings), rather than being stored as explicit, **human-readable tokens** or dedicated **parameter sets**.
- Generate: latent memory is produced by an independent model or a module and then supplied to the agent as reusable internal representations.
- Reuse: latent memory is directly carried over from prior computation, most prominently KV-cache reuse (within or across turns), as well as recurrent or stateful controllers that propagate hidden states.
- Transform: existing latent state is transformed into new representations(e.g., distillation, pooling, or compression), so the agent can retain essentials while reducing latency and context footprint.

Dynamics of Agent Memory

Memory Evolution dynamically integrates new memories into the existing repository through **consolidation, updating, and forgetting mechanisms** to ensure the knowledge base remains coherent and efficient



Future: RL-enabled Agent Memory System



References

- YaRN: Efficient Context Window Extension of Large Language Models
  - https://arxiv.org/abs/2309.00071

- Survey: Memory in the Age of AI Agents
  - https://arxiv.org/abs/2512.13564

- GenAI Tech Blog
  - https://www.genaiknowledge.info/