Introduction:

- Retrieval-Augmented Generation (RAG) is a technique that enhances language model generation by incorporating external knowledge.
- This is typically done by retrieving relevant information from a large corpus of documents and using that information to inform the generation process.

Challenge:

- Clients often have vast proprietary documents.
- Extracting specific information is like finding a needle in a haystack.

2. GPT4-Turbo Introduction:

- OpenAI’s GPT4-Turbo can process large documents.

3. Efficiency Issue:

- “Lost In The Middle” phenomenon hampers efficiency.

- Model forgets content in the middle of its contextual window.

4. Alternative Approach — Retrieval-Augmented-Generation (RAG):

- Create an index for each document paragraph.
- Swiftly identify pertinent paragraphs.
- Feed selected paragraphs into a Large Language Model (LLM) like GPT4.

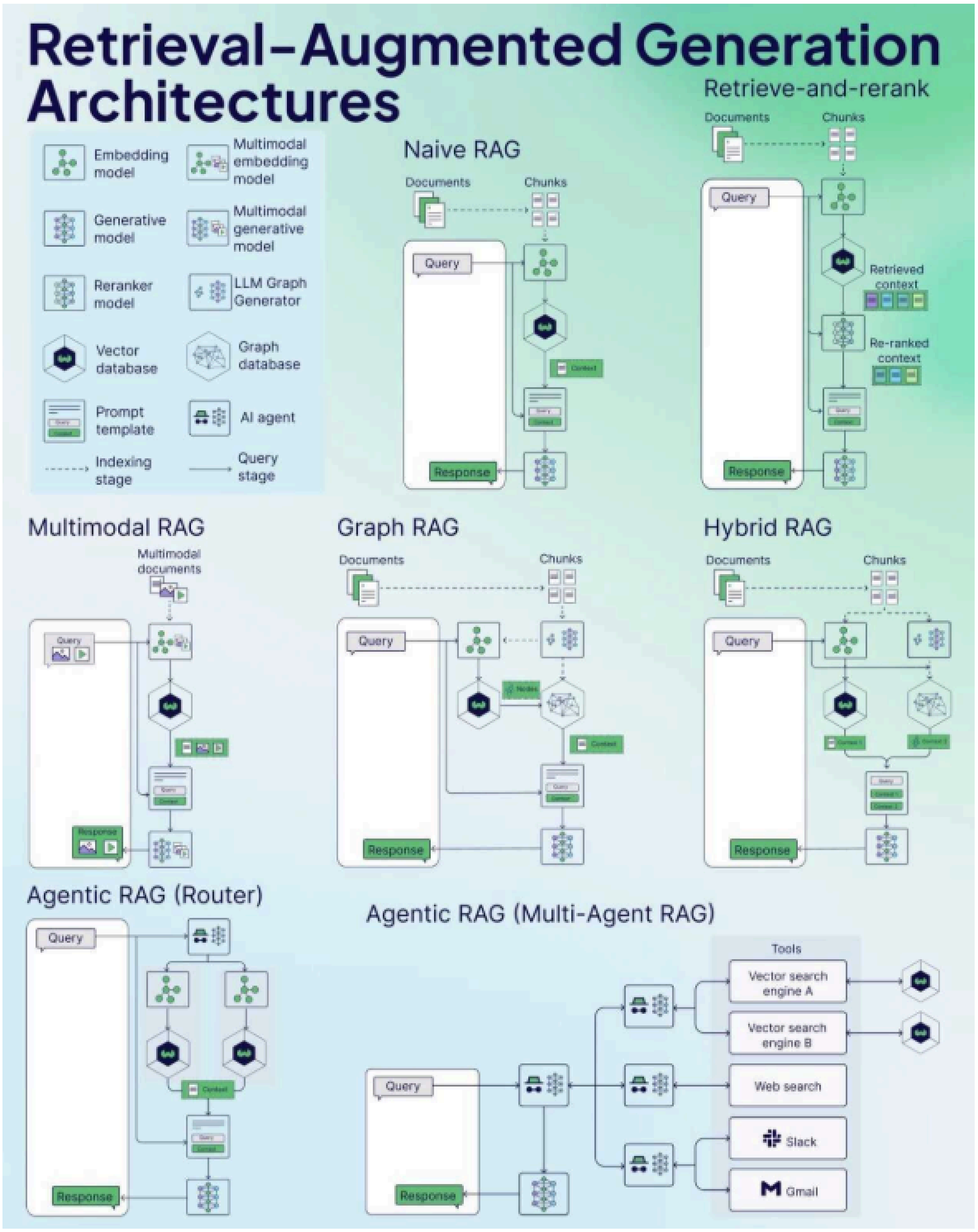
5. Advantages:

- Prevents information overload.
- Enhances result quality by providing only relevant paragraphs.

Why is Retrieval-Augmented Generation important

- You can think of the LLM as an over-enthusiastic new employee who refuses to stay informed with current events but will always answer every question with absolute confidence.
- Unfortunately, such an attitude can negatively impact user trust and is not something you want your chatbots to emulate!

- RAG is one approach to solving some of these challenges. It redirects the LLM to retrieve relevant information from authortative, pre-determined knowledge sources.
- Organizations have greater control over the generated text output, and users gain insights into how the ML generates the response.


Retrieval-Augmented Generation Architectures
Embedding model
Multimodal embedding model
Generative model
Multimodal generative model
Reranker model
LLM Graph Generator
Vector database
Graph database
Prompt template
AI agent
Indexing stage
Query stage
Naive RAG
Documents
Chunks
Query
Context
Response
Retrieve-and-rerank
Retrieved context
Re-ranked context
Multimodal RAG
Multimodal documents
Graph RAG
Nodes
Hybrid RAG
Agentic RAG (Router)
Agentic RAG (Multi-Agent RAG)
Tools
Vector search engine A
Vector search engine B
Web search
Slack
Gmail

Why use RAG?

RAG offers several advantages augmenting traditional methods of text generation, especially when dealing with factual information or data-driven responses. Here are some key reasons why using RAG can be beneficial:

Access to fresh information

LLMs are limited to their pre-trained data. This leads to outdated and potentially inaccurate responses. RAG overcomes this by providing up-to-date information to LLMs.

Factual grounding

LLMs are powerful tools for generating creative and engaging text, but they can sometimes struggle with factual accuracy. This is because LLMs are trained on massive amounts of text data, which may contain inaccuracies or biases.

Providing “facts” to the LLM as part of the input prompt can mitigate “gen AI hallucinations.” The crux of this approach is ensuring that the most relevant facts are provided to the LLM, and that the LLM output is entirely grounded on those facts while also answering the user’s question and adhering to system instructions and safety constraints.

Using Gemini’s long context window (LCW) is a great way to provide source materials to the LLM. If you need to provide more information than fits into the

LCW, or if you need to scale up performance, you can use a RAG approach that will reduce the number of tokens, saving you time and cost.

Search with vector databases and relevancy re-rankers

RAGs usually retrieve facts via search, and modern search engines now leverage vector databases to efficiently retrieve relevant documents. Vector databases store documents as embeddings in a high-dimensional space, allowing for fast and accurate retrieval based on semantic similarity. Multi-modal embeddings can be used for images, audio and video, and more and these media embeddings can be retrieved alongside text embeddings or multi-language embeddings.

Advanced search engines like Vertex AI Search use semantic search and keyword search together (called hybrid search), and a re-ranker which scores search results to ensure the top returned results are the most relevant. Additionally searches perform better with a clear, focused query without misspellings; so prior to lookup, sophisticated search engines will transform a query and fix spelling mistakes.

Relevance, accuracy, and quality

The retrieval mechanism in RAG is critically important. You need the best semantic search on top of a curated knowledge base to ensure that the retrieved information is relevant to the input query or context. If your retrieved information is irrelevant, your generation could be grounded but off-topic or incorrect.

By fine-tuning or prompt-engineering the LLM to generate text entirely based on the retrieved knowledge, RAG helps to minimize contradictions and inconsistencies in the generated text. This significantly improves the quality of the generated text, and improves the user experience.

The Vertex Eval Service now scores LLM generated text and retrieved chunks on metrics like “coherence,” “fluency,” “groundedness,” "safety," “instruction_following,” “question_answering_quality,” and more. These metrics help you measure the grounded text you get from the LLM (for some metrics that is a comparison to a ground truth answer you have provided). Implementing these evaluations gives you a baseline measurement and you can optimize for RAG quality by configuring your search engine, curating your source data, improving source layout parsing or chunking strategies, or refining the user’s question prior to search. A RAG Ops, metrics driven approach like this will help you hill climb to high quality RAG and grounded generation.

RAGs, agents, and chatbots

RAG and grounding can be integrated into any LLM application or agent which needs access to fresh, private, or specialized data. By accessing external information, RAG-powered chatbots and conversational agents leverage external knowledge to provide more comprehensive, informative, and context-aware responses, improving the overall user experience.

Your data and your use case are what differentiate what you are building with gen AI. RAG and grounding bring your data to LLMs efficiently and scalably.