Part-I: What is an Agent?

Short Answer: Text-to-Task

An LLM agent is an AI system that utilizes a LLM as its core computational engine to exhibit capabilities beyond text generation.

Exercise: Create an agent that can do some math

Do this exercise (18 mins) to understand what an agent does

Code: https://github.com/rohanbalkondekar/finetune_llama2

Data: rohanbalkondekar/generate_json

Part-II: RAG overview

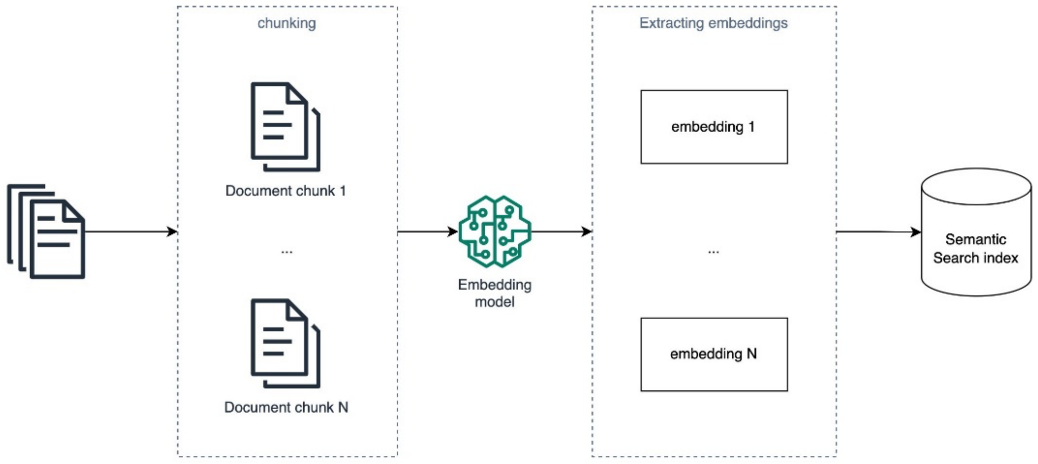
Lets review the RAG design pattern and discuss its limitations on analytical questions. Also cover a more versatile architecture that overcomes these limitations.

Overview of RAG

RAG solutions are inspired by representation learning and semantic search ideas that have been gradually adopted in ranking problems (for example, recommendation and search) and natural language processing (NLP) tasks since 2010.

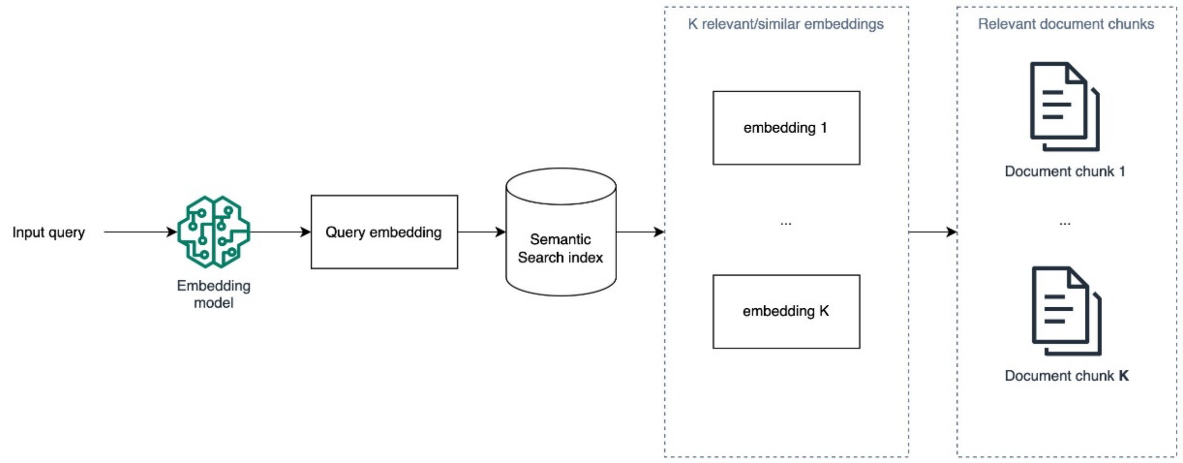
The popular approach used today is formed of three steps:

1. An offline batch processing job ingests documents from an input knowledge base, splits them into chunks, creates an embedding for each chunk to represent its semantics using a pre-trained embedding model, such as Amazon Titan embedding models, then uses these embeddings as input to create a semantic search index.

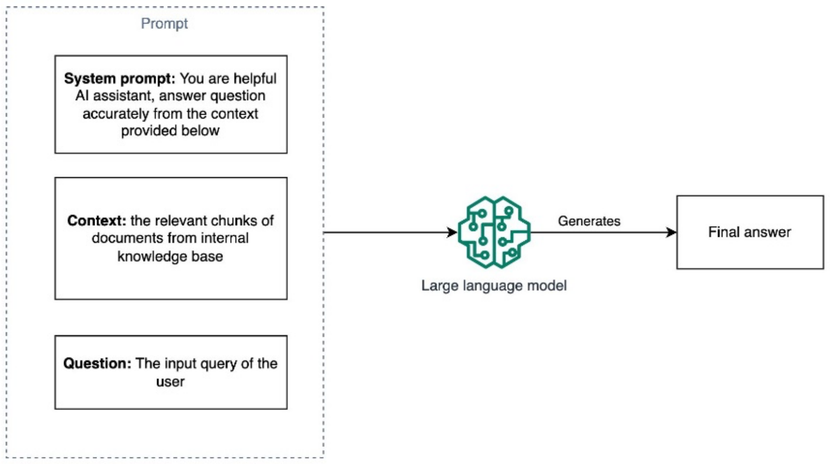


2. When answering a new question in real time, the input question is converted to an embedding, which is used to search for and extract the most similar chunks of documents using a similarity metric, such as cosine similarity, and an approximate nearest neighbors

algorithm. The search precision can also be improved with metadata filtering.



3. A prompt is constructed from the concatenation of a system message with a context that is formed of the relevant chunks of documents extracted in step 2, and the input question itself. This prompt is then presented to an LLM model to generate the final answer to the question from the context.


Prompt
System prompt: You are helpful AI assistant, answer question accurately from the context provided below
Context: the relevant chunks of documents from internal knowledge base
Question: The input query of the user
Large language model
Generates
Final answer

With the right underlying embedding model, capable of producing accurate semantic representations of the input document chunks and the input questions, and an efficient semantic search module, this solution is able to answer questions that require retrieving existent information in a database of documents. For example, if you have a service or a product, you could start by indexing its FAQ section or documentation and have an initial conversational AI tailored to your specific offering.

PART-III: RAG LIMITATIONS on Semantic Search (Multi Document)

1. Multi Document

Although RAG is an essential component in modern domain-specific AI assistants and a sensible starting point for building a conversational AI around a specialized knowledge base, it can't answer questions that require:

I. Scanning

II. Comparing

III. Reasoning across all documents in your knowledge base simultaneously, especially when the augmentation is based solely on semantic search.

A. First Check: Single document as knowledge base (complex data)

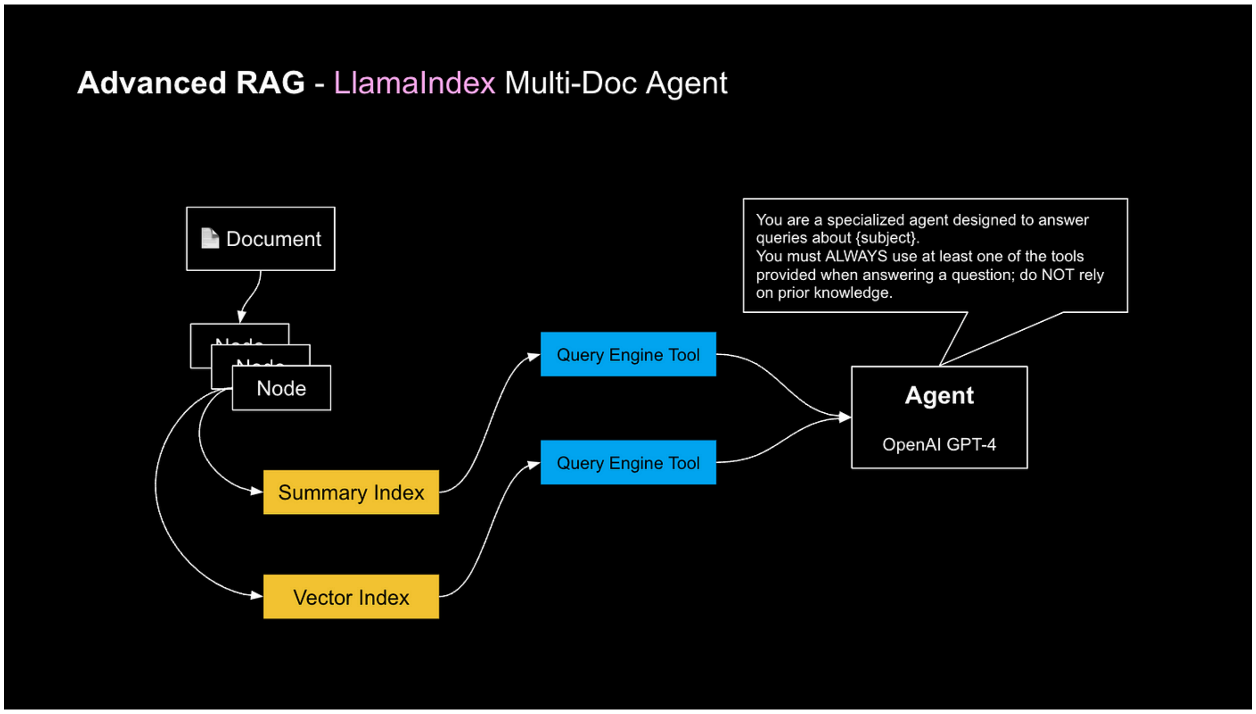
https://ai.gopubby.com/advanced-rag-semi-structured-data-with-langchain-ce46c8baa6cf

https://ai.gopubby.com/advanced-rag-multi-modal-rag-with-gpt4-vision-e4c11229682c

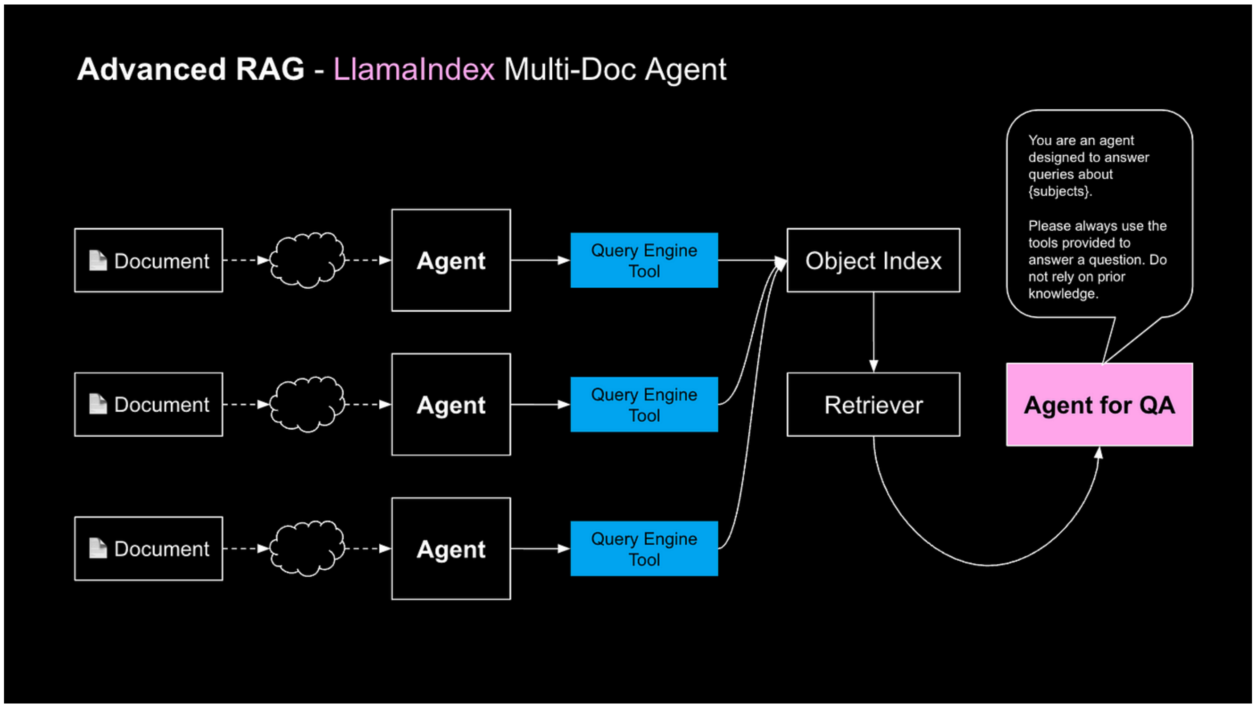
B. Advanced RAG – Multi Documents Agent with LlamaIndex

Sample implementation:

https://github.com/sugarforever/Advanced-RAG/blob/main/03_llama_index_multi_doc_agent.ipynb



You are a specialized agent designed to answer queries about {subject}.
You must ALWAYS use at least one of the tools provided when answering a question; do NOT rely on prior knowledge.


Advanced RAG - LlamaIndex Multi-Doc Agent
Document
Agent
Query Engine Tool
Document
Agent
Query Engine Tool
Document
Agent
Query Engine Tool
Object Index
Retriever
You are an agent designed to answer queries about {subjects}. Please always use the tools provided to answer a question. Do not rely on prior knowledge.
Agent for QA

- VectorStoreIndex
- SummaryIndex
- ObjectIndex
- QueryEngineTool
- OpenAIAgent
- FnRetrieverOpenAIAgent

Full Explanation

https://blog.gopenai.com/advanced-rag-multi-documents-agent-with-llamaindex-43b604f84909

Additional Resources from Official Documentation

https://docs.llamaindex.ai/en/stable/examples/agent/multi_document_agents.html

In this guide, you learn towards setting up an agent that can effectively answer different types of questions over a larger set of documents.

These questions include the following

- QA over a specific doc
- QA comparing different docs
- Summaries over a specific odc
- Comparing summaries between different docs

We do this with the following architecture:

- Setup a “document agent” over each Document: each doc agent can do QA/summarization within its doc

- Setup a top-level agent over this set of document agents. Do tool retrieval and then do CoT over the set of tools to answer a question.

If you want additional features:

https://docs.llamaindex.ai/en/stable/examples/agent/multi_document_agents-v1.html

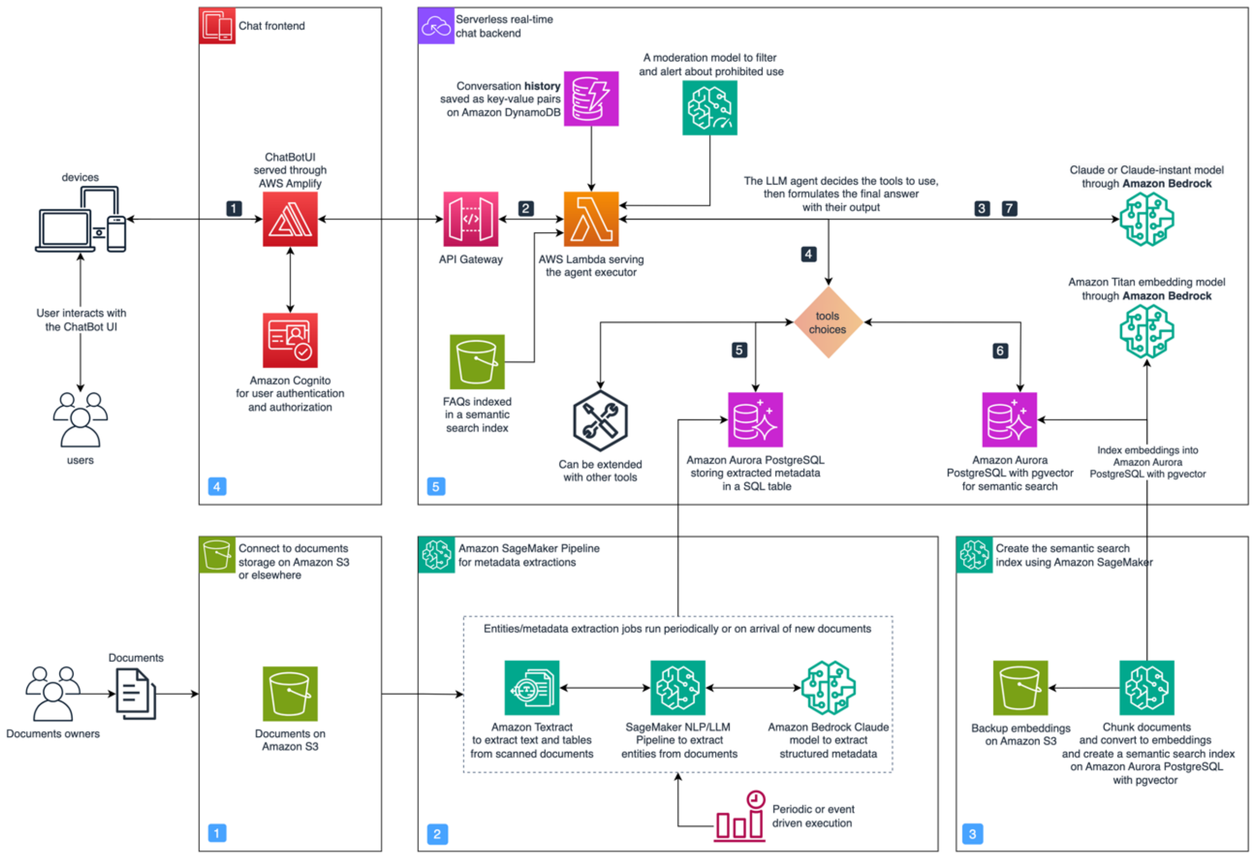
- Reranking during document (tool) retrieval
- Query planning tool that the agent can use to plan

We do this with the following architecture:

- setup a “document agent” over each Document: each doc agent can do QA/summarization within its doc
- setup a top-level agent over this set of document agents. Do tool retrieval and then do CoT over the set of tools to answer a question.

PART-IV: RAG LIMITATIONS on Semantic Search (STRUCTURED DATA)

Entity extraction, SQL querying, and agents with Amazon Bedrock


Chat frontend
Serverless real-time chat backend
devices
ChatBotUI served through AWS Amplify
1
2
API Gateway
Conversation history saved as key-value pairs on Amazon DynamoDB
A moderation model to filter and alert about prohibited use
AWS Lambda serving the agent executor
The LLM agent decides the tools to use, then formulates the final answer with their output
3 7
Claude or Claude-instant model through Amazon Bedrock
4
tools choices
Amazon Titan embedding model through Amazon Bedrock
User interacts with the ChatBot UI
Amazon Cognito for user authentication and authorization
users
FAQs indexed in a semantic search index
5
6
Can be extended with other tools
Amazon Aurora PostgreSQL storing extracted metadata in a SQL table
Amazon Aurora PostgreSQL with pgvector for semantic search
Index embeddings into Amazon Aurora PostgreSQL with pgvector
4
5
Connect to documents storage on Amazon S3 or elsewhere
Amazon SageMaker Pipeline for metadata extractions
Create the semantic search index using Amazon SageMaker
Entities/metadata extraction jobs run periodically or on arrival of new documents
Documents
Documents owners
Documents on Amazon S3
Amazon Textract to extract text and tables from scanned documents
SageMaker NLP/LLM Pipeline to extract entities from documents
Amazon Bedrock Claude model to extract structured metadata
Backup embeddings on Amazon S3
Chunk documents and convert to embeddings and create a semantic search index on Amazon Aurora PostgreSQL with pgvector
Periodic or event driven execution
1
2
3

Full Stack Example with Public Data