Architecting the future: Patterns to build generative AI application

Praveen Jayakumar
Head of AI/ML Solution Architecture
AWS India

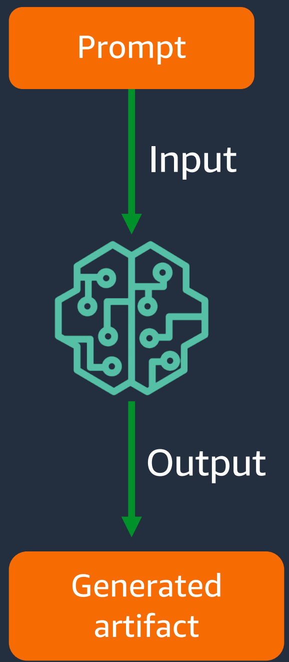
LLM customisation patterns

- Prompt engineering (In-context learning)
- Retrieval Augmented Generation (RAG)
- Fine-tuning
- Training your own LLM

Prompt engineering types

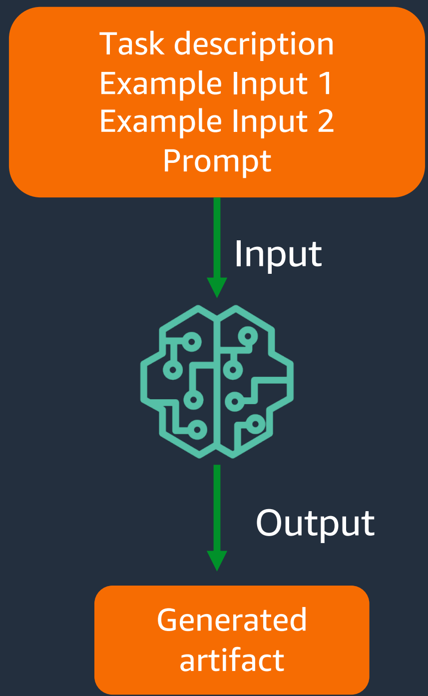
Zero shot prompts

- Direct request with sufficient context



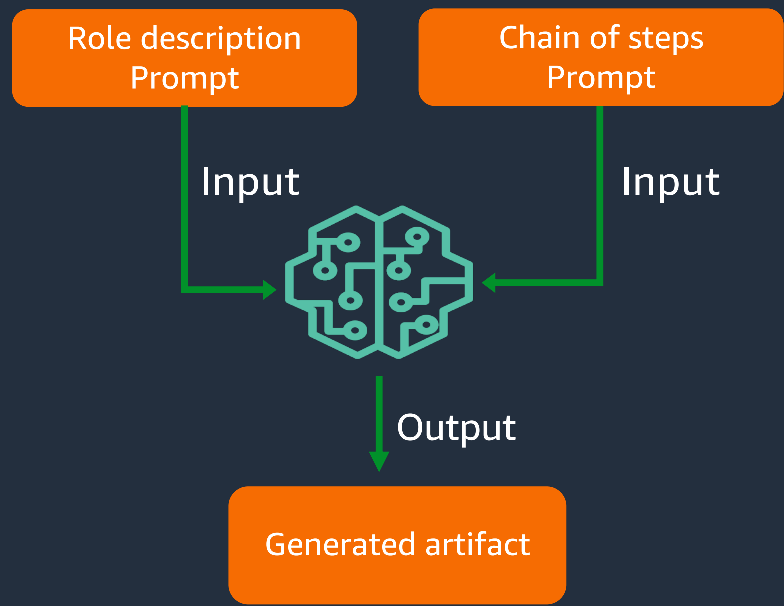
One shot or few shot prompts

- Provide one or more examples with a request



Role or Chain of Thought prompts

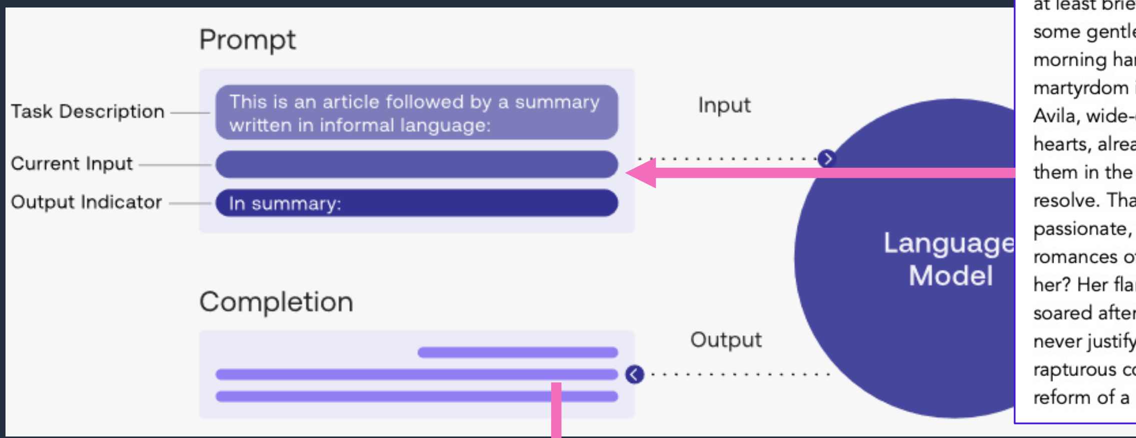
- Provide the model with a role or persona for the task
- Provide a chain of steps for the model to follow



Zero shot prompt: Prompting by instruction

Zero-shot prompting:

- Using LLM out of the box
- allows language models to perform tasks for which they have not been explicitly trained on

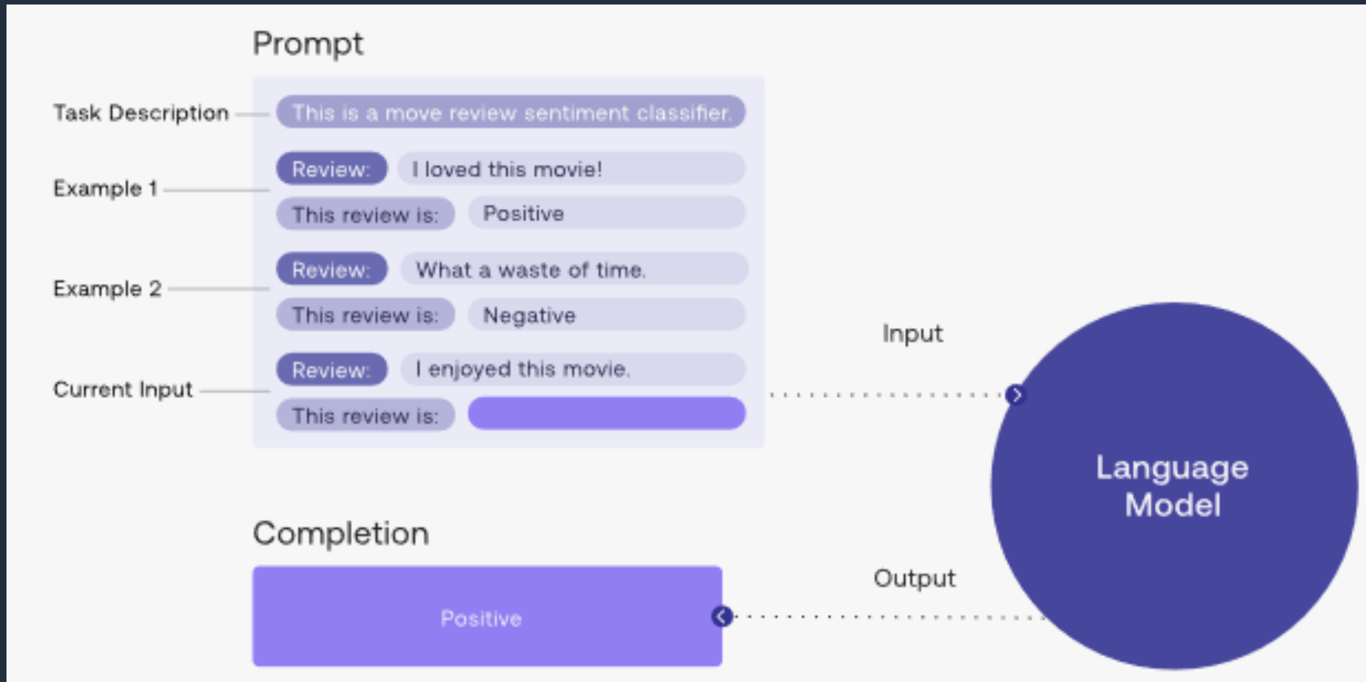


Who that cares much to know the history of man, and how the mysterious mixture behaves under the varying experiments of Time, has not dwelt, at least briefly, on the life of Saint Theresa, has not smiled with some gentleness at the thought of the little girl walking forth one morning hand-in-hand with her still smaller brother, to go and seek martyrdom in the country of the Moors? Out they toddled from rugged Avila, wide-eyed and helpless-looking as two fawns, but with human hearts, already beating to a national idea; until domestic reality met them in the shape of uncles, and turned them back from their great resolve. That child-pilgrimage was a fit beginning. Theresa's passionate, ideal nature demanded an epic life: what were many-volumed romances of chivalry and the social conquests of a brilliant girl to her? Her flame quickly burned up that light fuel; and, fed from within, soared after some illimitable satisfaction, some object which would never justify weariness, which would reconcile self-despair with the rapturous consciousness of life beyond self. She found her epos in the reform of a religious order.

Theresa was a young girl of Spanish origin who became a member of a religious order in Italy.

Prompt Engineering: N shots example

- A few-shot prompt normally includes n examples of (problem, solution) pairs known as "shots".
- Help to guide model performance.

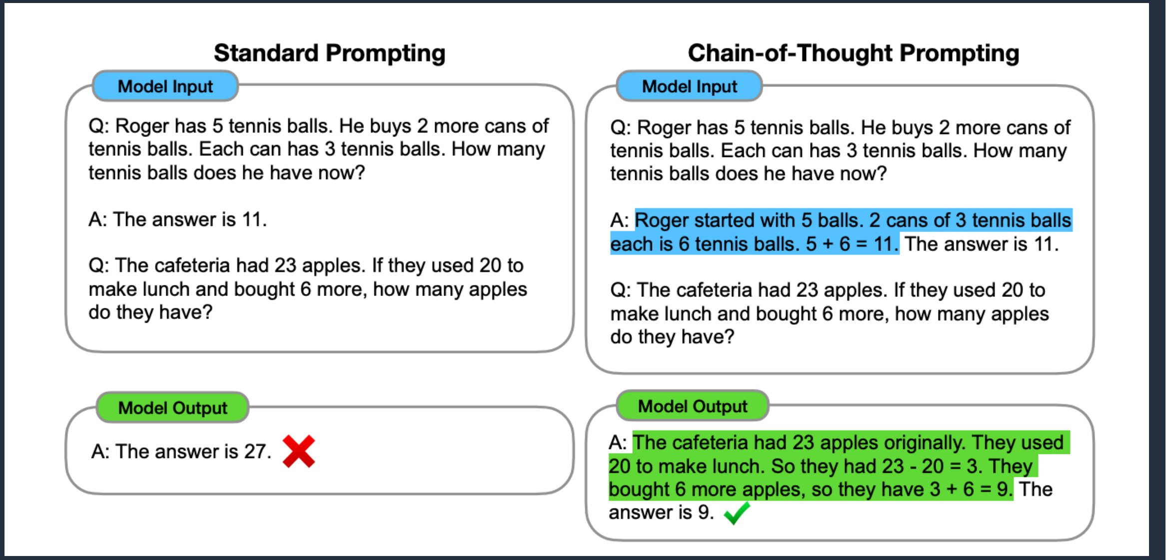


Prompt Engineering: Chain of thought prompting example

- Improves *reasoning* abilities in foundation models

- Addresses *multi-step problem-solving* challenges in arithmetic and *commonsense reasoning* task

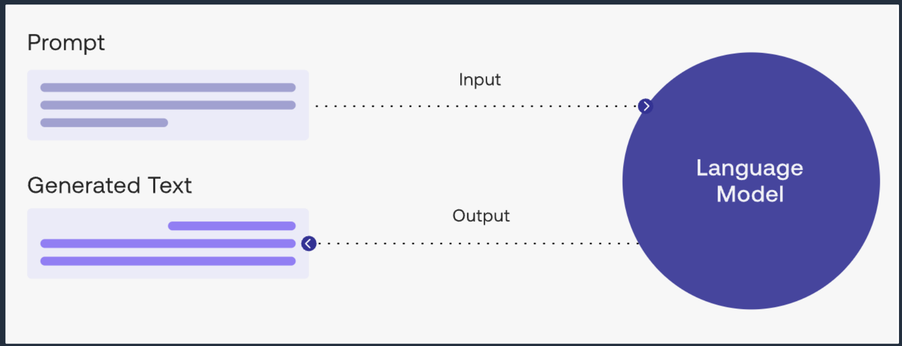
- Generates intermediate reasoning steps, mimicking *human train of thought*, before providing the final answer.



Source: https://arxiv.org/pdf/2201.11903.pdf

Prompting limitation

- Poor memory
- Limited context
- Accessing external knowledge sources to complete tasks.



What is Retrieval Augmented Generation?

Retrieval

Fetches the relevant content from the external knowledge base or data sources based on a user query

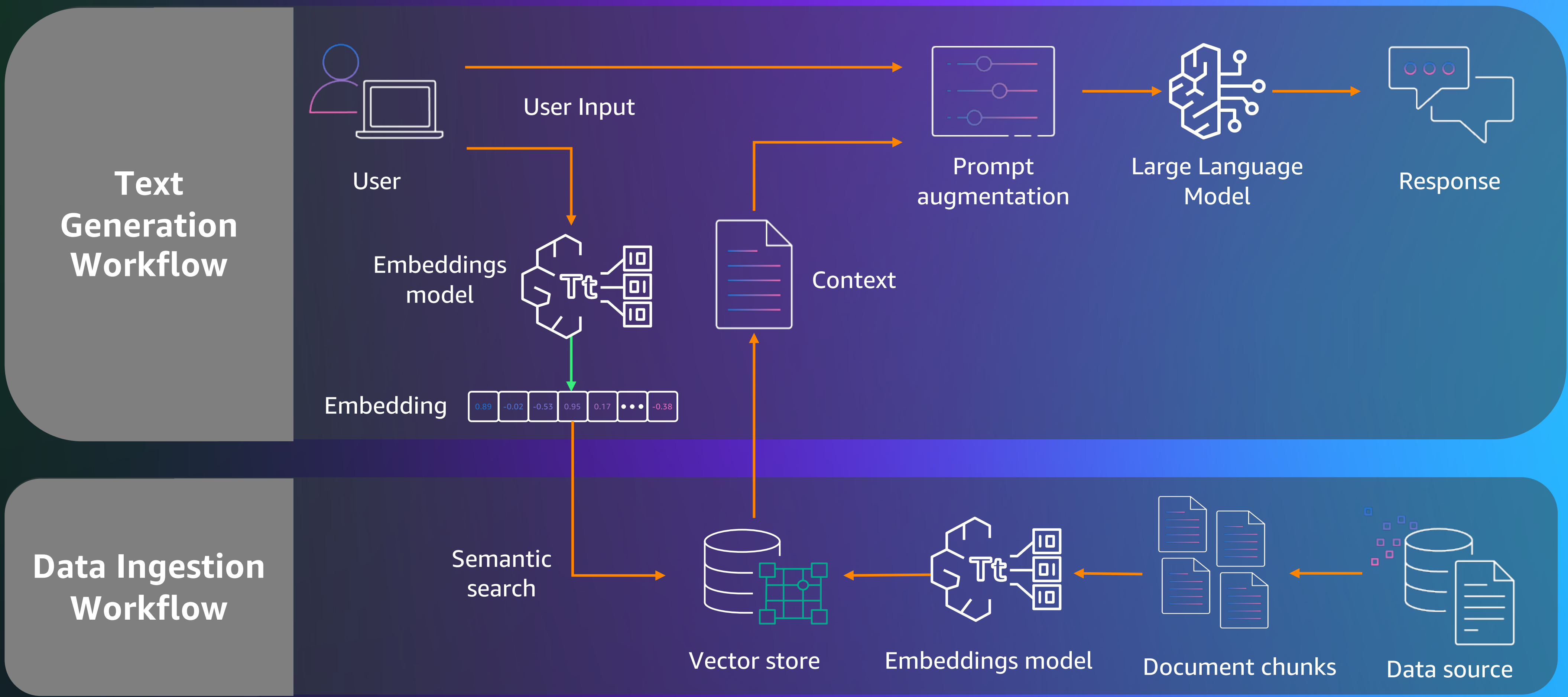
Augmentation

Adding the retrieved relevant context to the user prompt, which goes as an input to the foundation model

Generation

Response from the foundation model based on the augmented prompt.

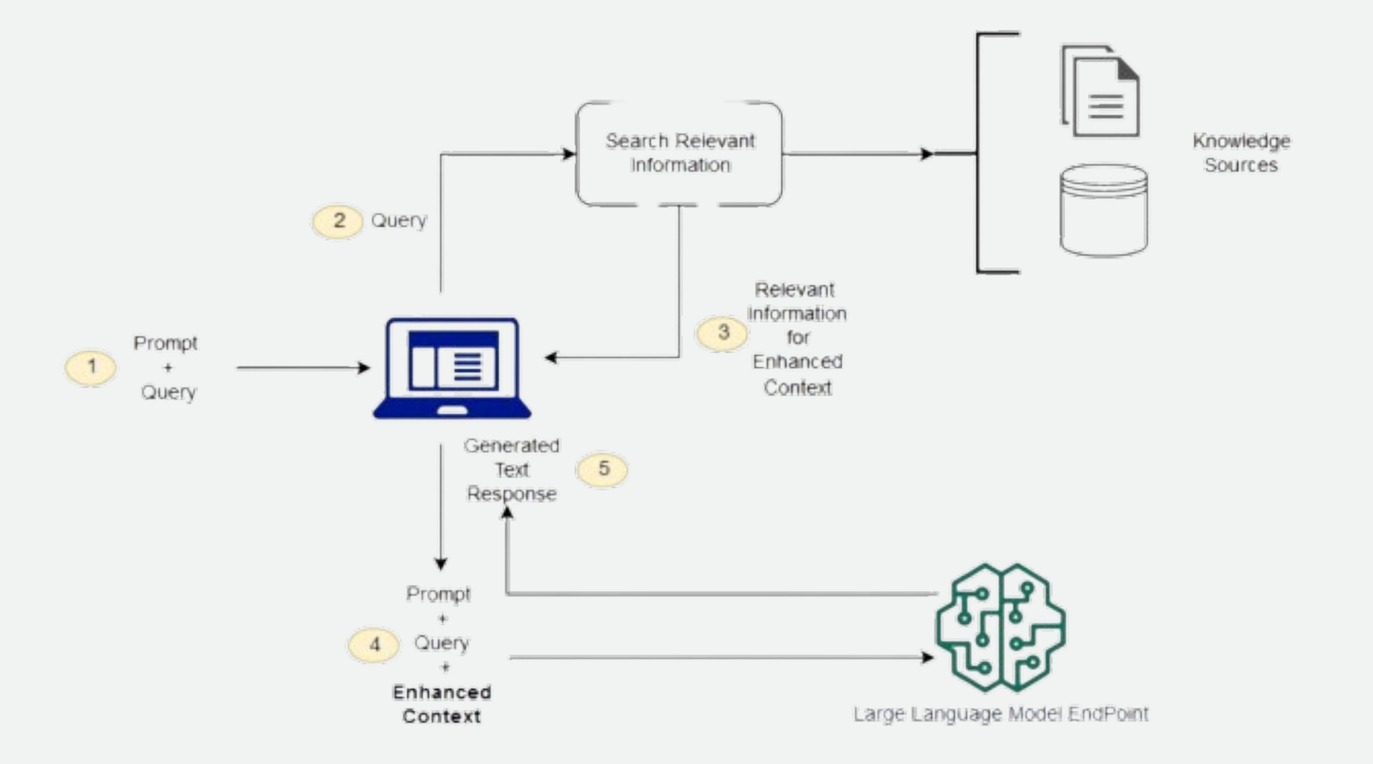
RAG in Action



RAG Patterns

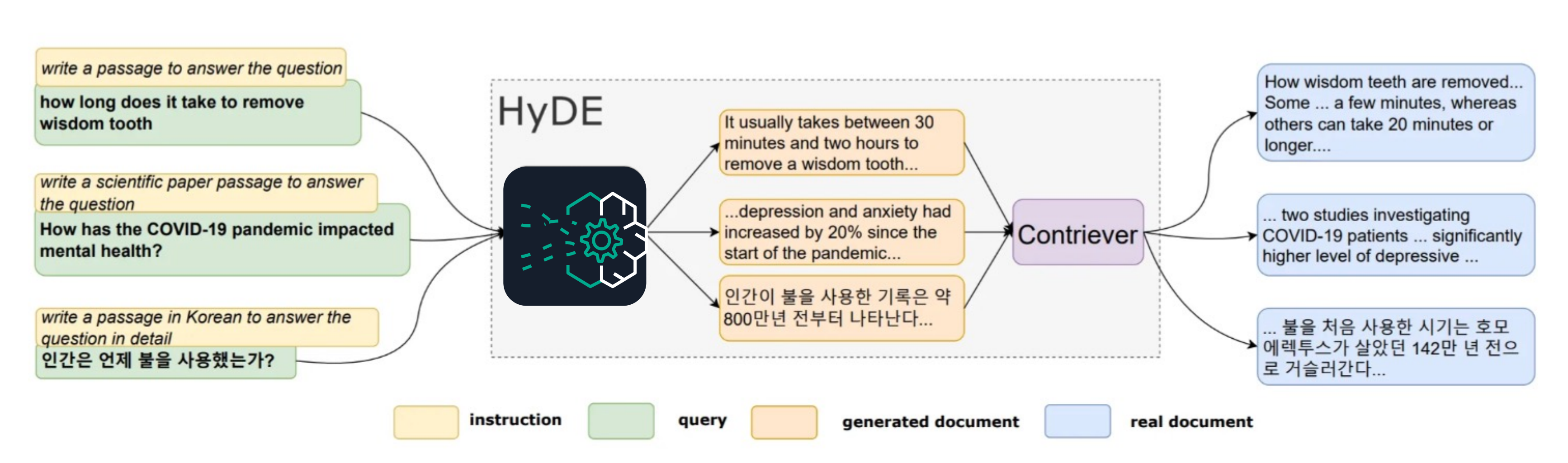
Simple RAG

1. Query
2. Retrieve Docs
3. Prompt + [Docs]
4. Generate Response



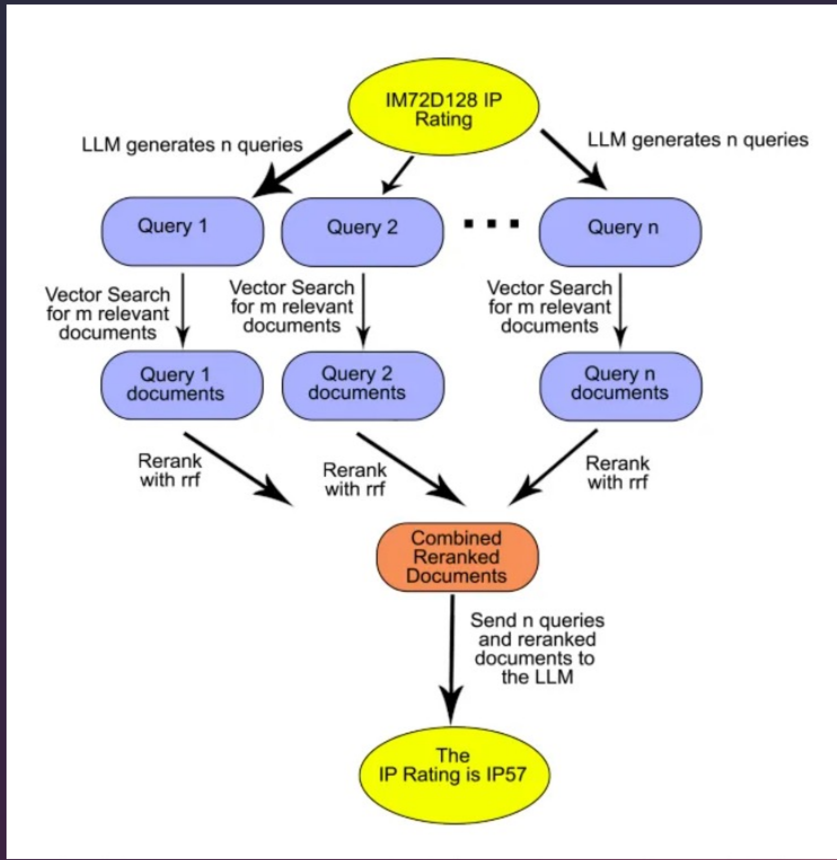
HyDE: Hypothetical Document Embedding

1. Query
2. Generate Hypothetical Answer
3. Embed Hypothetical Answer
4. Retrieve Docs with Embedding
5. [Docs]
6. Generate Response



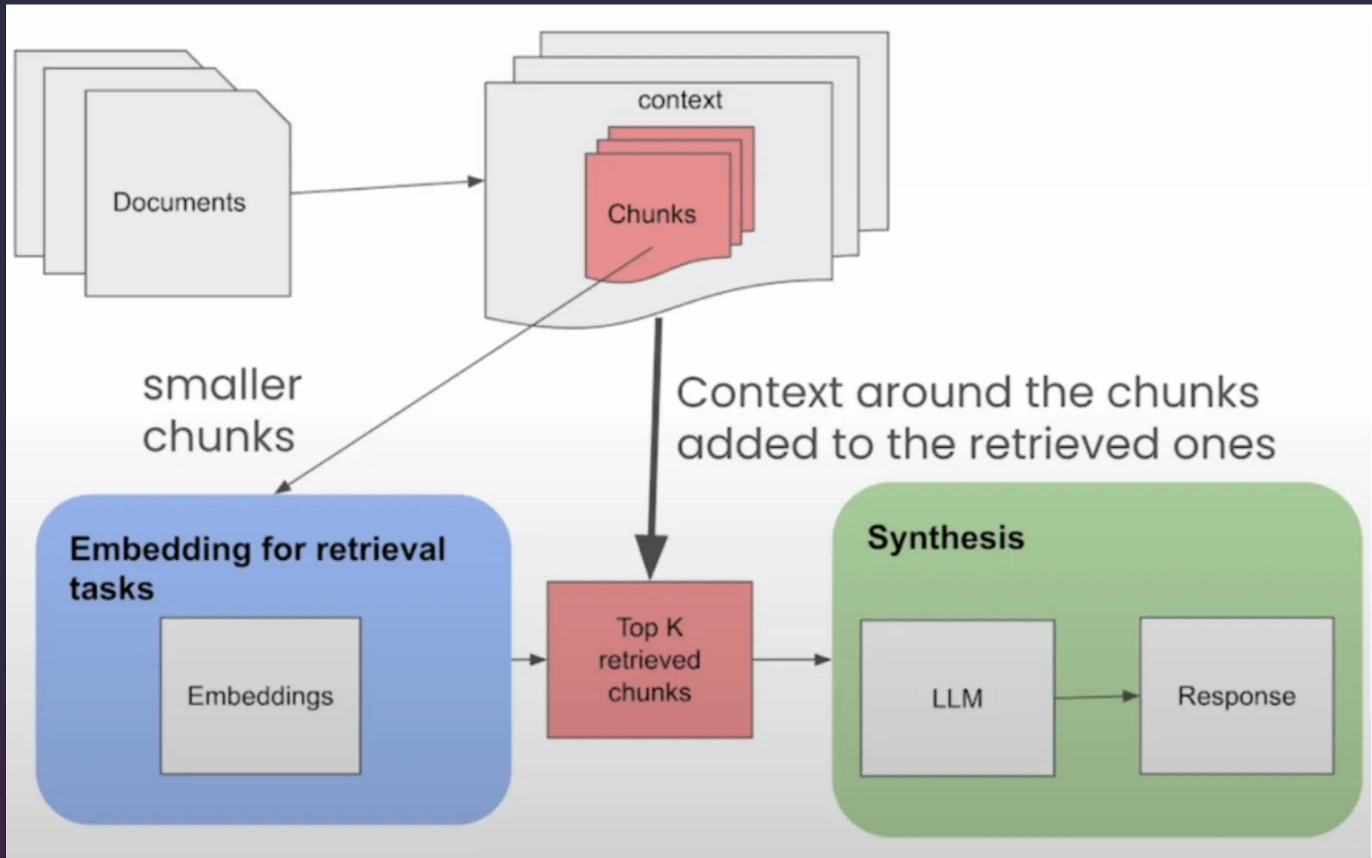
Multi Query RAG

1. Query
2. Generate Multiple Queries
3. [Retrieve Docs(query 1), Retrieve Docs(query 2), ...]
4. Rerank Docs – Reciprocal rank fusion
5. [Reranked Docs]
6. Generate Response



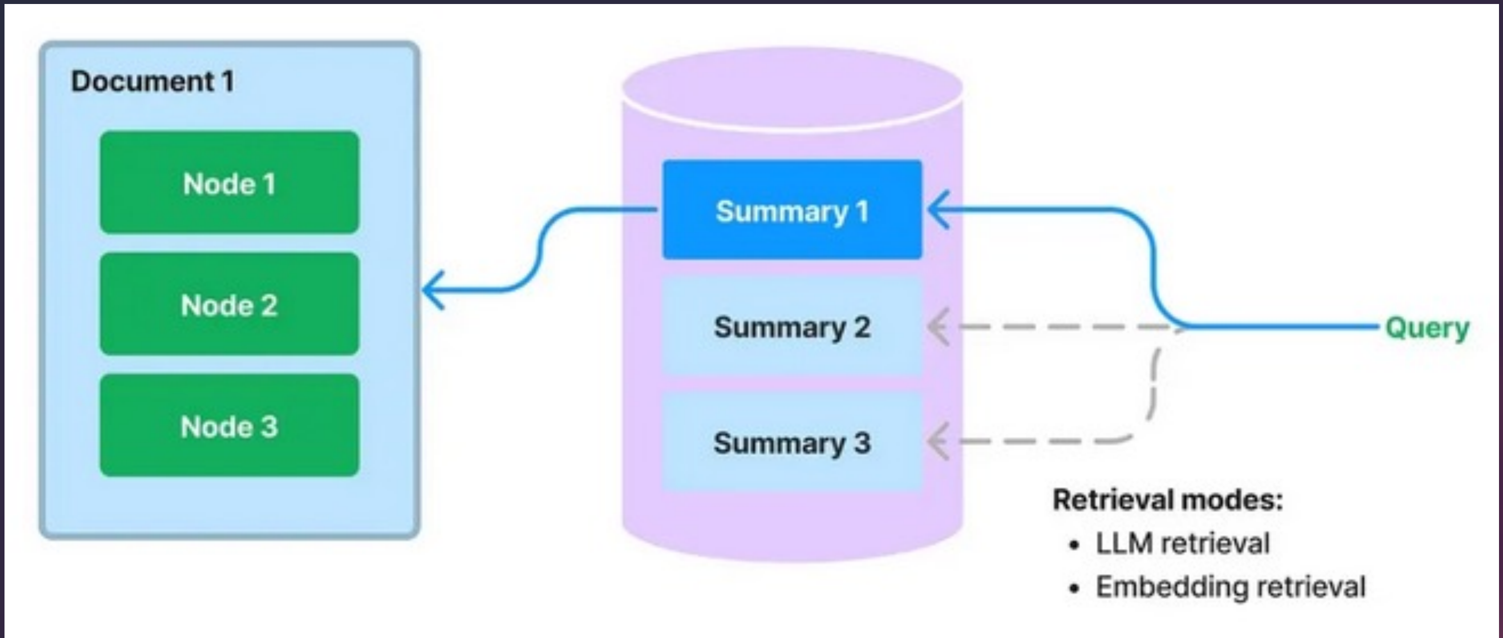
Sentence Window Retrieval

1. Query
2. Retrieve Sentence Windows
3. [Sentence Windows]
4. Add Context around Sentence Windows
5. Generate Response



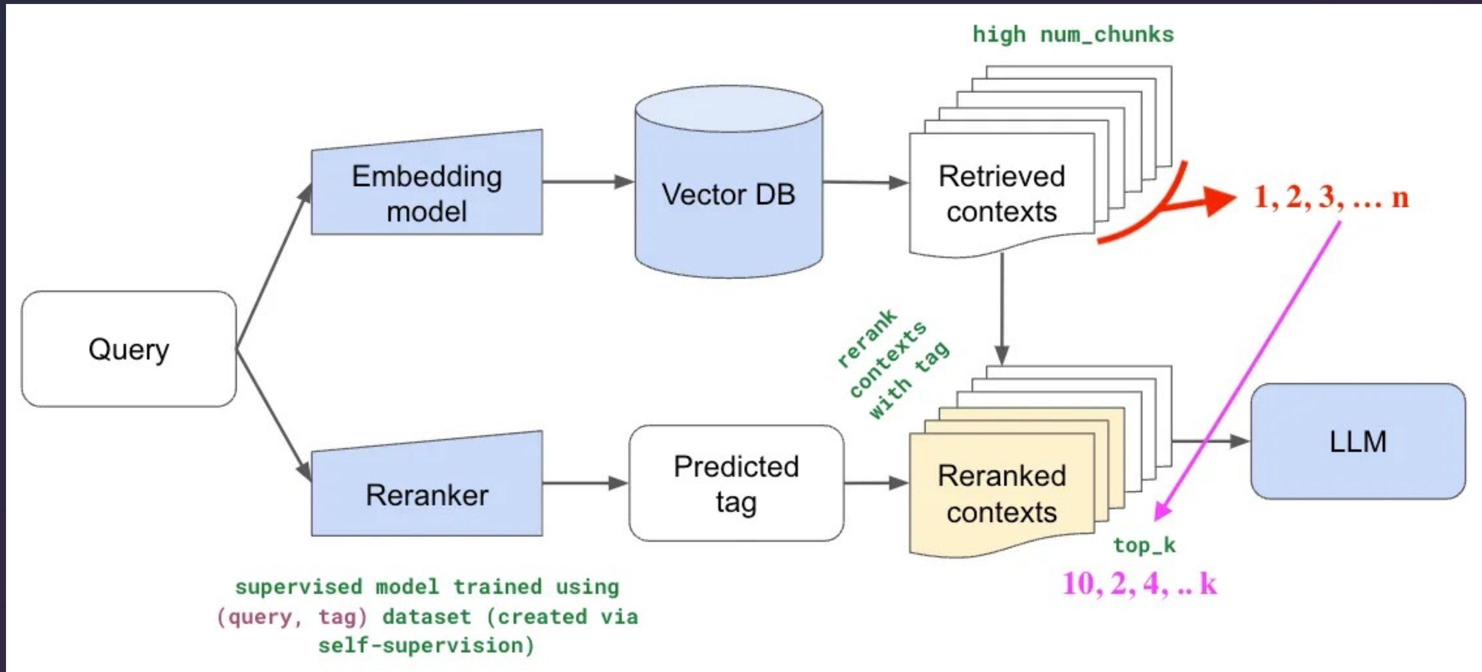
Document Summary Index

1. Query
2. Retrieve Document Summaries
3. [Full Documents]
4. Generate Response



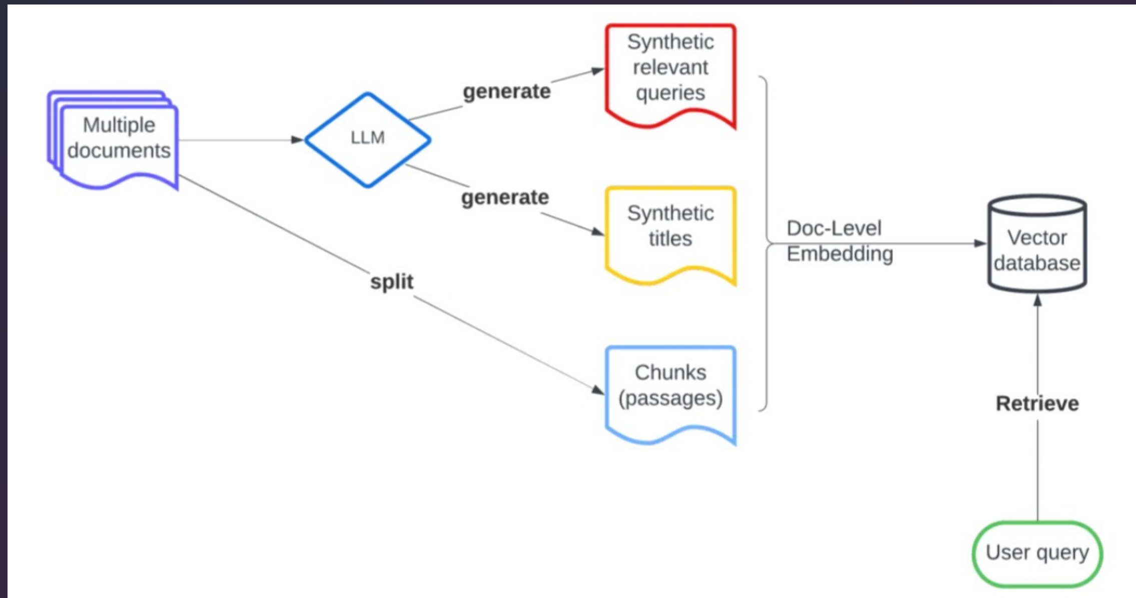
Reranker (MMR, Cohere, LLM)

1. Query
2. Retrieve Docs
3. Rerank Docs
4. [Reranked Docs]
5. Generate Response



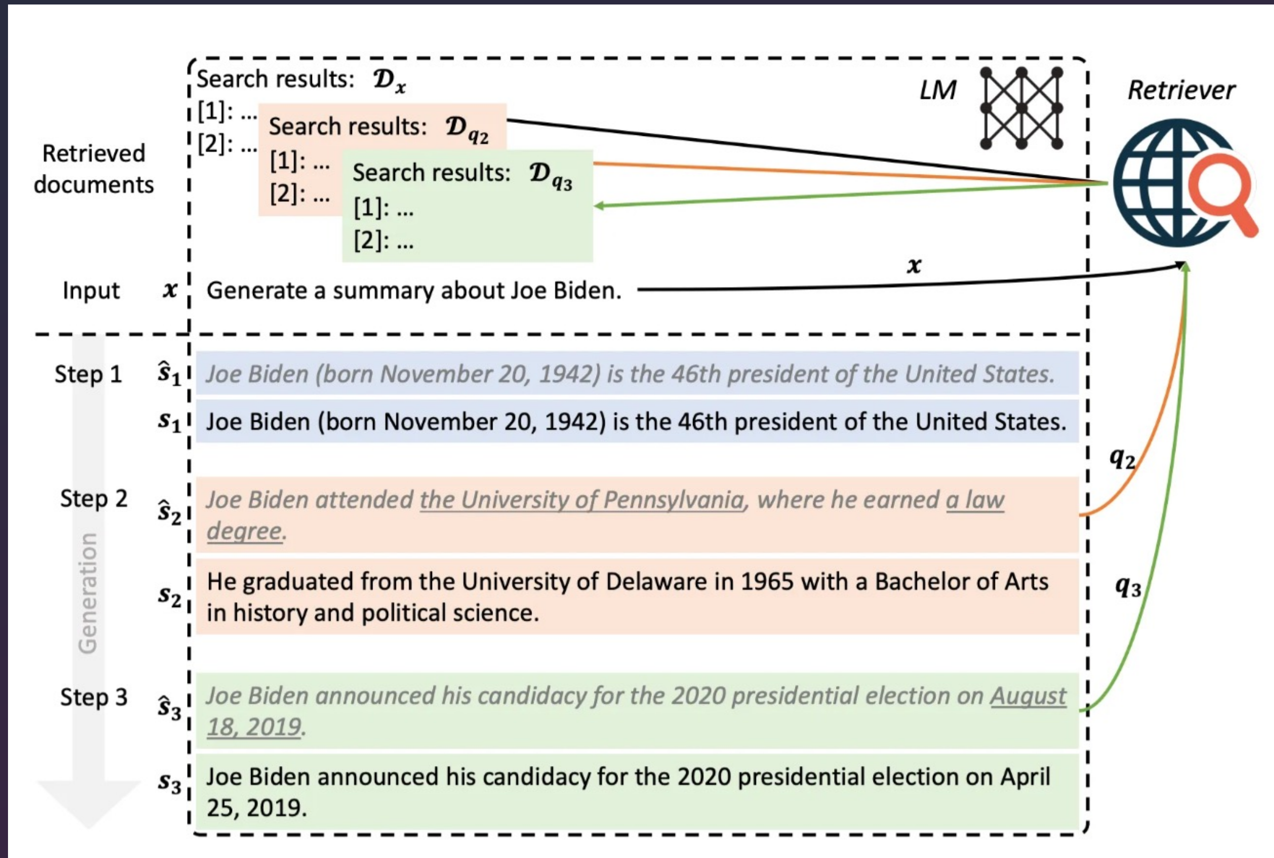
LLM-Augmented Retrieval

1. Documents
2. Split
3. [Chunks]
4. LLM
5. Generate [Synthetic Titles]
6. Generate [Relevant Queries]
7. Combine ([Chunks, Synthetic Titles, Relevant Queries])
8. Doc-Level Embedding
9. Vector Database
10. Retrieve ([Relevant Docs])
11. User Query



FLARE - Forward Looking Active RAG

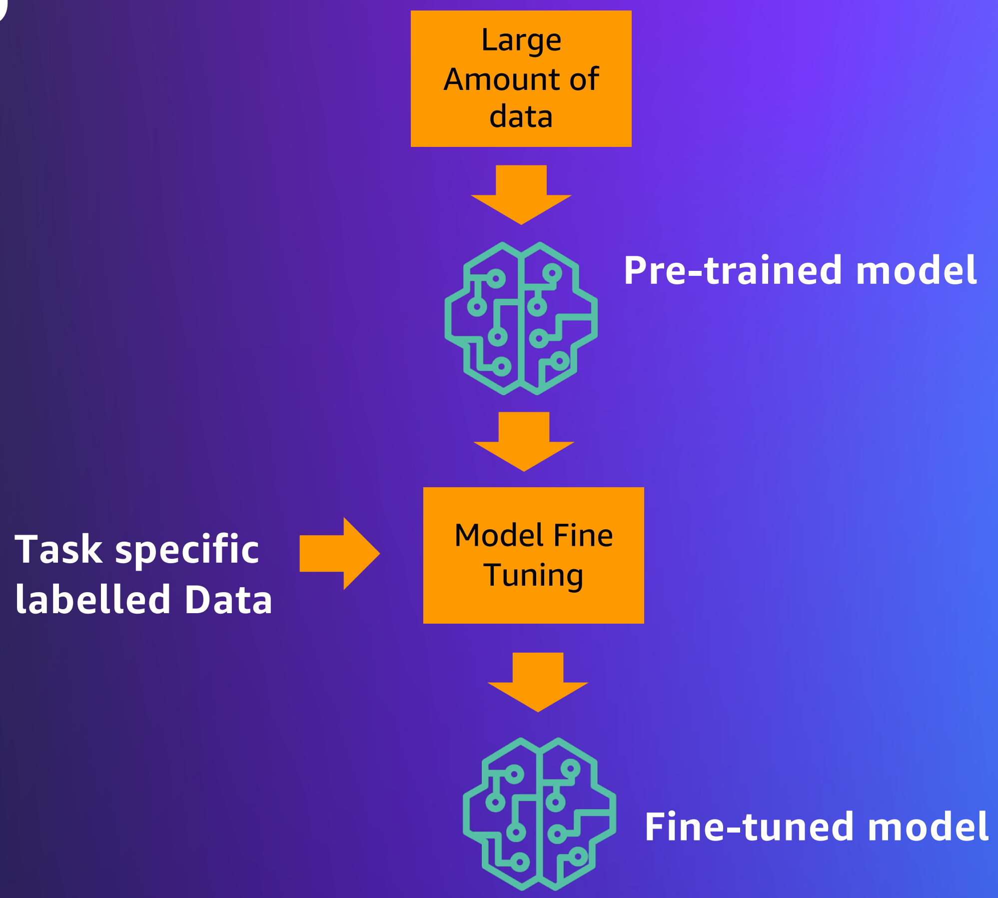
1. Documents
2. Split
3. [Chunks]
4. LLM
5. Generate [Temporary next sentences]
6. Combine Search results
7. Retrieve [Relevant Docs]
8. User Query



Key challenges with Retrieval Augmented Generation (RAG)

- Managing multiple data sources
- Creating embeddings for large volume of data
- Incremental updates to vector database
- Increased Complexity => adding retriever component to generation model
- Limited Creativity => constrained by the retrieved information
- Context window limited by LLMs

Fine-tuning


Large Amount of data
Pre-trained model
Task specific labelled Data
Model Fine Tuning
Fine-tuned model

Fine-tuning

Traditional fine-tuning

- Expensive and resource intensive for LLMs scaling to multibillion parameters.
- Overfitting and Knowledge degradation: also known as "catastrophic forgetting"

PEFT fine-tuning

- Training only a subset of parameters => Faster, cheaper, less computational power
- Set of newly added parameters or select some existing model parameters.
- Most of existing weights do not change :
  - Reduces overfitting and catastrophic forgetting
- Fast improving techniques such as LORA and others

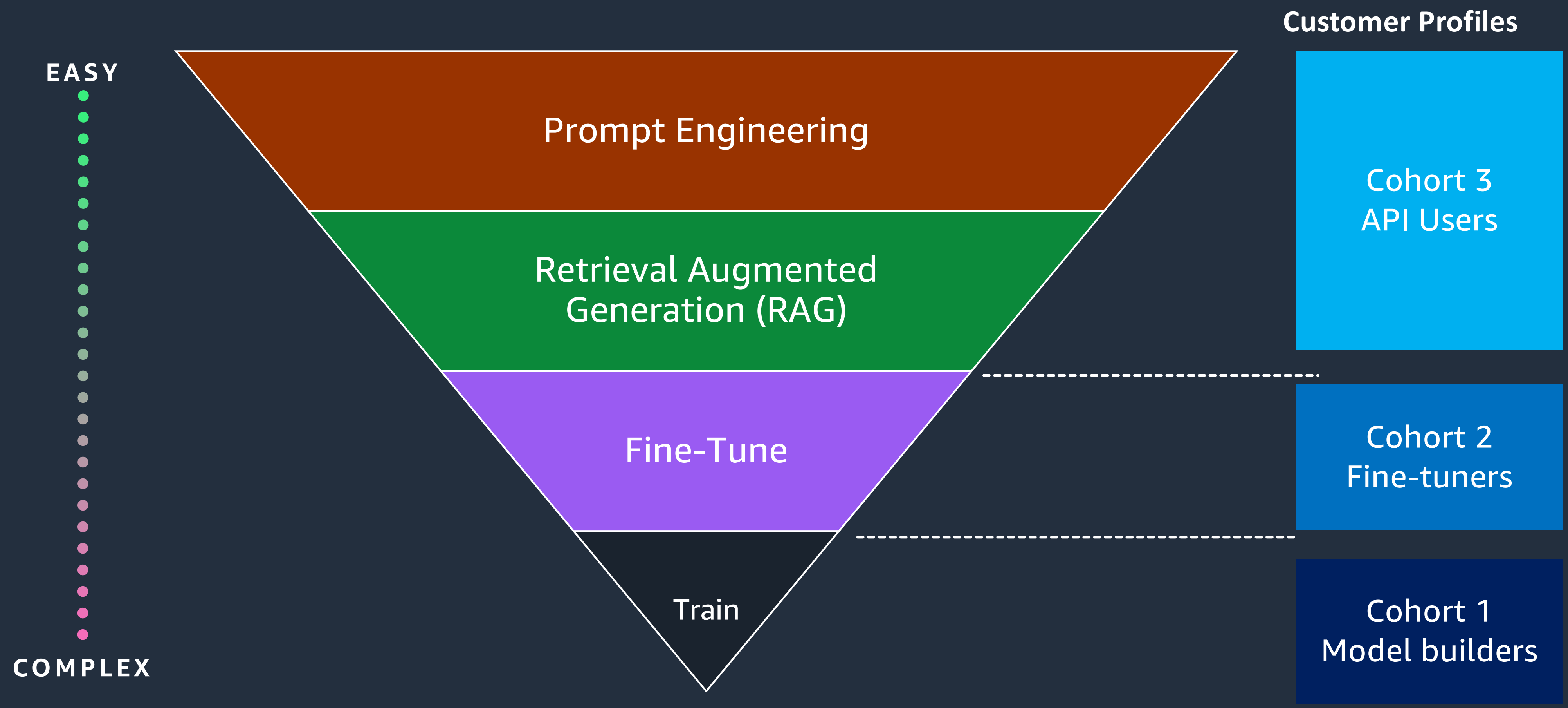
Limitation of fine-tuning

- Deep knowledge
- Time consuming and resources
- Need enough data to fine-tune
- Data could be constantly changing
- Some LLM model are not available for fine-tuning

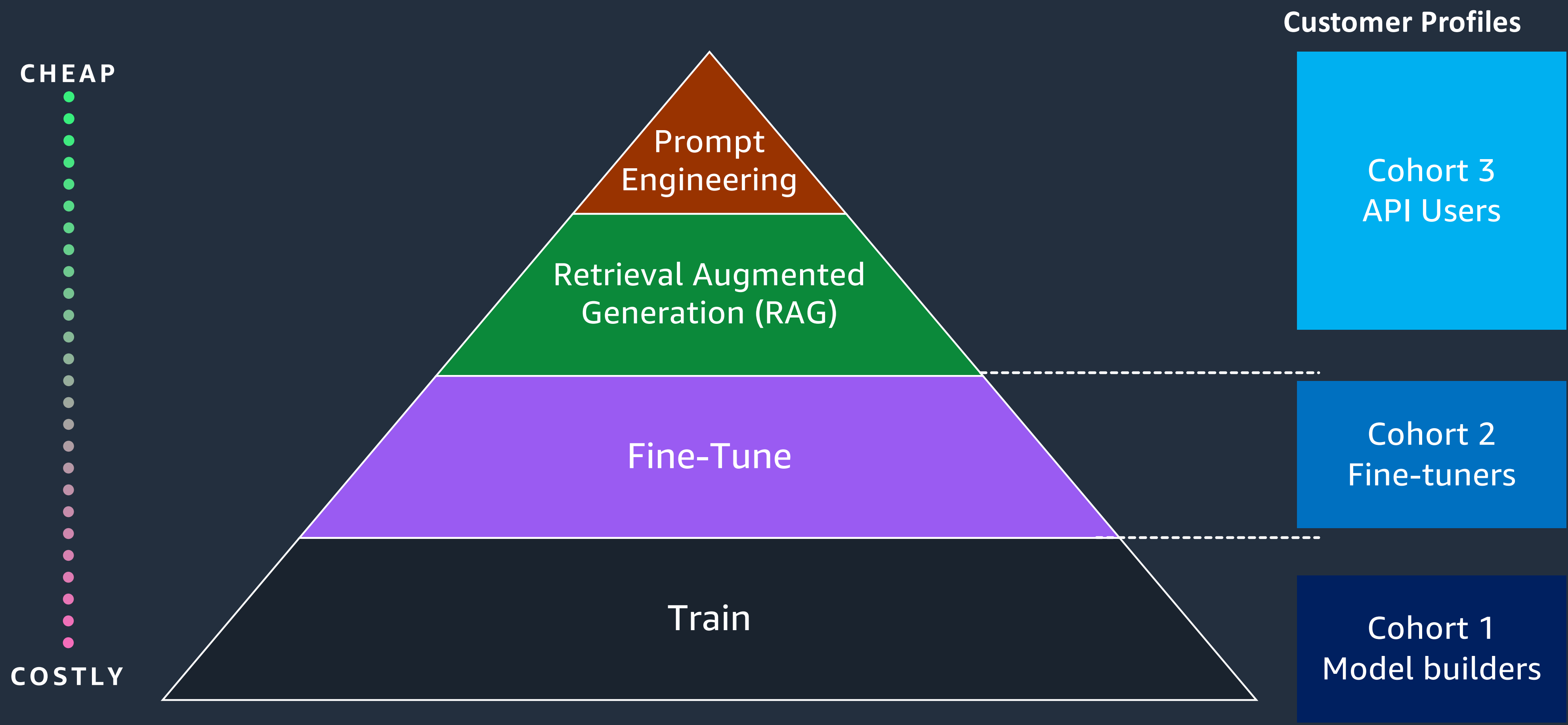
When should customers fine-tune

Problem	Example	Likelihood of success with fine-tuning	Likelihood of success with prompting (+ RAG)
Make the model follow a specific format or tone.	Use this specific JSON schema or talk like my customer service reps.	VERY HIGH	HIGH
Teach the model a new skill.	Learn how to call APIs, fill out proprietary documents, or classify customer support tickets.	HIGH	MEDIUM
Teach the model a new skill, and hope it learns similar skills.	Teach the model to summarize contract documents and hope it learns how to write better contract documents.	LOW	MEDIUM
Teach the model new knowledge and expect it to use that knowledge for general tasks.	Learn my company's acronyms or know more facts music.	VERY LOW	MEDIUM

LLM customisation comparison: Skills required



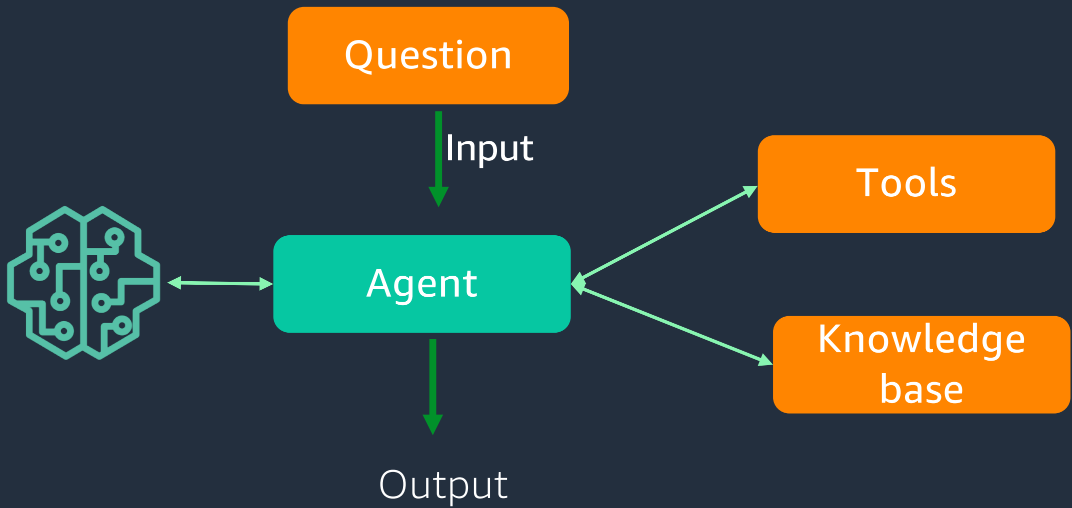
LLM customisation comparison: Cost



I think AI agentic workflows will drive massive AI progress this year — perhaps even more than the next generation of foundation models. This is an important trend, and I urge everyone who works in AI to pay attention to it. –
- Andrew Ng

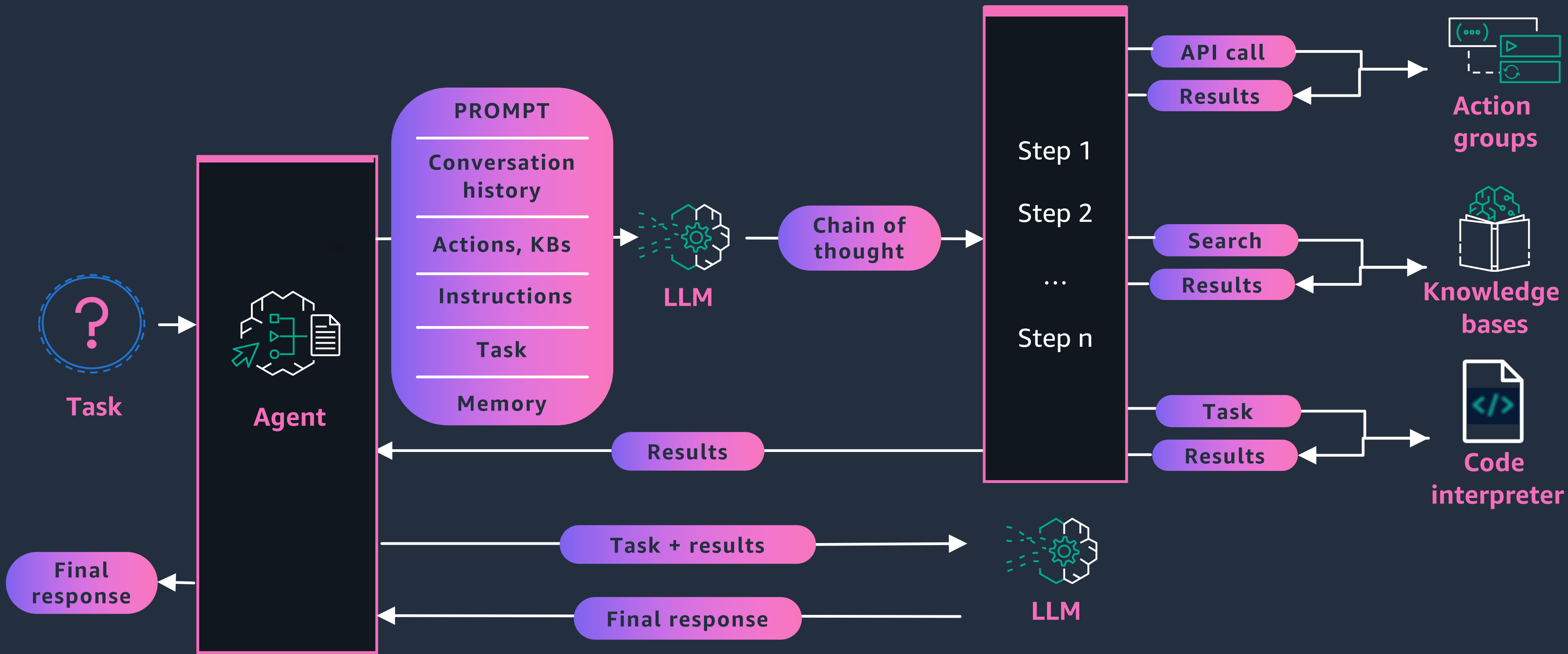
Prompt engineering types: Reasoning & Acting (ReAct)

- ReAct is technique which enable LLMs to do reasoning and take task specific actions.
- Combines chain of thought reasoning with action planning.



Reasoning & Acting (ReAct)

Agent orchestration



Agent breaks task into subtasks, determines the right sequence, and executes actions and knowledge searches on the fly

Agents use turn-by-turn dialog within a session to provide a natural experience

Jane books a trip using an agent

Hi, my name is Jane

How can I help?

Plan a trip to NYC

Sure. What dates?

Depart July 9, return July 11

Leaving from where?

Boston

Flying, or train?

…

Thanks

Have a great day

Travel advisor agent

Jane uses the agent for ~10 minutes, ~20 turns

Math calculations using code interpretation, no files

Mortgage agent

What is my estimated monthly payment for a $500K 30-year fixed loan?

Agent orchestration

1. Find the current interest rate in knowledge base
2. Generate code for calculating payments
3. Execute code to estimate payment amount
4. Use results in final response

Rates lookup

Rates knowledge base

Generated calculation

Code interpreter

The current rate for that loan type is 8%. Your payment would be $3,668.82.

How Computer Use works

1. Provide Claude with computer use tools and a user prompt

- Add Anthropic-defined computer use tools to your API request.
  - `{ "type": "computer_20241022", "name": "computer" }`
  - `{ "type": "text_editor_20241022", "name": "str_replace_editor" }`
  - `{ "type": "bash_20241022", "name": "bash" }`
- Include a user prompt that require these tools, e.g., “Go to Amazon.com and add diapers to my cart”

2. Claude decides to use a tool

- Claude loads the stored computer use tool definitions and assesses if any tools can help with the user’s query.
- If yes, Claude constructs a properly formatted tool use request.
- The API response has a `stop_reason` of `tool_use`, signaling Claude’s intent.

How Computer Use works

3. Extract tool input, evaluate the tool on a computer, and return results

- On the client side, extract the tool name and input from Claude's request

Use a dedicated virtual machine or container with minimal privileges to prevent direct system attacks or accidents

4. Claude continues calling computer use tools until it's completed the task

- Claude analyzes the tool results to determine if more tool use is needed or the task has been completed.

Summary

- Prompt Types
- Retrieval Augmented Generation
- Fine Tuning
- Agents
- Computer Use

Thank you!

Praveen Jayakumar
Head of AI/ML Solution Architecture
AWS India