

Large AI **To Everywhere**



Large AI & The Next Wave

Agentic AI

2025. 03. | 서강대학교 인공지능학과 | 장두성 | duseong.chang@gmail.com

1. Agentic AI?
 - 1) Beyond LLM
 - 2) LLM / RAG / AI Agent
2. RAG
3. AI Agent
4. AI Agent platforms
5. Use Cases

Agentic AI Beyond LLM

LLM으로 모든 걸 해결할 수 있다고 하는데, 뭐가 부족한거지?

- LLM은 **신경망 내부**에 **미리 습득한** 방대한 지식을 바탕으로 다양한 질문에 **자신있게 응답** 가능
- Hallucination
 - LLM은 응답하는 방법만을 주로 배움. 모른다는 걸 이해못함.
 - 자신있게 응답을 하지만, 실제 일부 정보가 잘못 조합되었을 가능성 존재
- Domain Knowledge
 - LLM 내부에 미리 습득한 지식을 기반으로 응답 추론
 - 최신 뉴스나 다양한 세부 전문 분야에 대한 지식은 미리 습득하지 못함
- Real World 연계 부족
 - LLM 내부에 존재하는 정보를 기반으로 추론
 - 매일 변하는 외부 정보 (날씨, 버스정보, 현재 방안의 상태 등)를 이용한 추론 불가
 - 응답만 할 뿐이지, 실제 행동을 제공하지는 못함 (로봇제어, IoT제어 등)
- Collaboration : 다른 사람 혹은 LLM 간의 협력은 시작 단계

Agentic AI Beyond LLM

이 문제를 어떻게 하면 해결할 수 있을까요? 완화할 수 있을까요?

- Hallucination
→ 추론 알고리즘 강화 + 근거에 기반한 추론
 - Domain Knowledge
→ 타깃 도메인의 문서를 검색하여 그 안에서 추론
 - Real World 연계 부족
→ 필요한 외부 정보를 API로 찾아와서 추론에 사용
 - Collaboration
→ 여러 LLM 간의 정보교류, 순서/체계화된 정보흐름 제어
-
- The diagram illustrates the four solutions listed above, categorized by blue curly braces. The first three solutions (Hallucination, Domain Knowledge, and Real World) are grouped under the label 'Retrieval Augmented Generation'. The last solution (Collaboration) is grouped under the label 'AI Agent'.

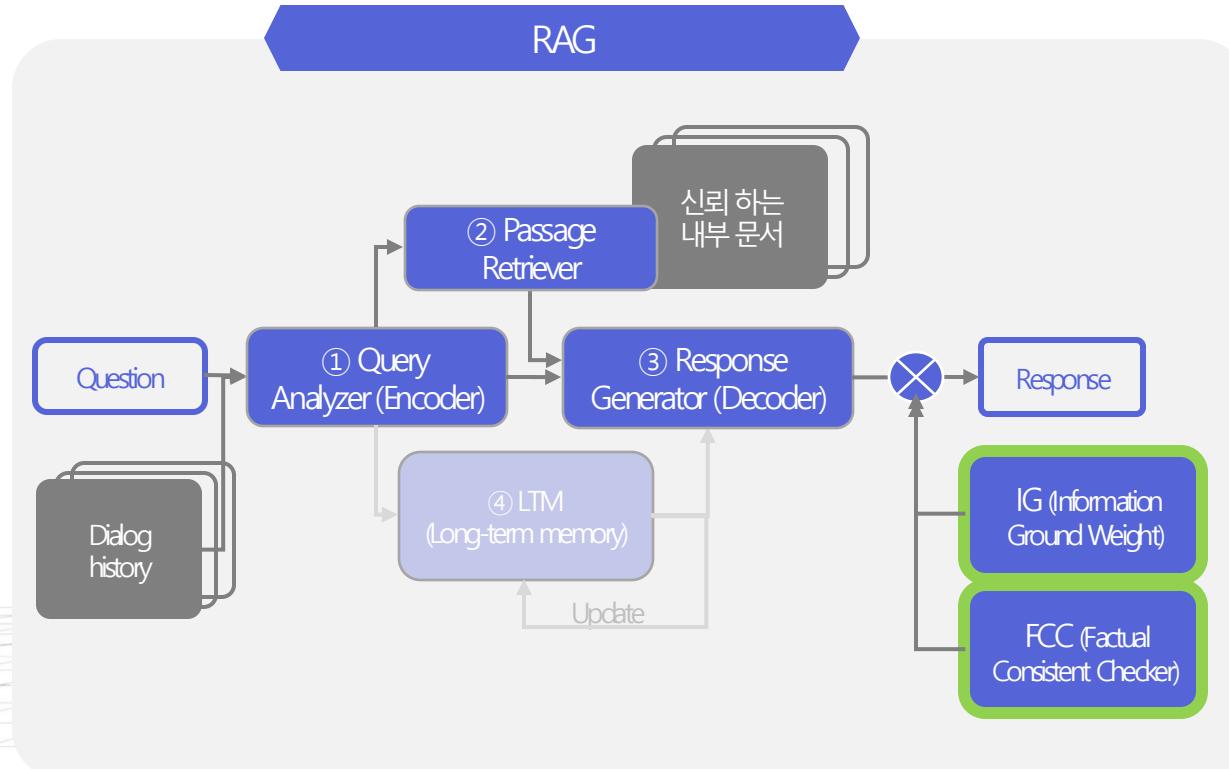
Agentic AI LLM / RAG / Agent

구분	정의	예시
LLM	LLM(Large Language Model)은 방대한 양의 텍스트 데이터를 학습하여 주어진 질문이나 문단에 이어질 내용을 생성하는 인공지능 모델.	<ul style="list-style-type: none">질문: "최근 출시된 전기차 모델들의 특징은?"답변: "전기차는 전통적인 내연기관 차량과 달리 전기를 동력으로 사용해 환경 친화적이며 운영 비용이 낮습니다."
RAG	RAG(Retrieval-Augmented Generation)는 LLM의 한계를 극복하기 위해 고안된 기술로, 검색을 통해 최신의 정보를 가져와 답변에 활용.	<ul style="list-style-type: none">질문: "최근 출시된 전기차 모델들의 특징은?"RAG: <24년도 전기차 모델 검색>답변: "2024년에 출시된 모델 A는 향상된 배터리 수명과 자율 주행 기능을 갖추고 있으며, 최근 출시된 모델 B는 비용 효율성이 높은 것으로 평가받습니다."
AI Agent	AI 에이전트는 단순히 질문에 답하는 것 이상의 복잡한 업무를 수행하도록 설계된 AI 시스템. LLM과 RAG를 통한 output text를 기반으로, 사용자가 원하는 다음 행동이나 명령 등을 파악해 이를 수행하는 역할을 함.	<ul style="list-style-type: none">질문: "최근 출시된 전기차 모델들의 특징은?"AI 에이전트: <구매 니즈 감지, 요구에 따라 행동을 취함>답변: "최신 전기차 분석을 완료했습니다. [분석 결과 제공] 귀하의 위치와 예산을 고려한 가까운 딜러를 찾았으며, 테스트 드라이브 예약을 도와드릴 수 있습니다."

1. Agentic AI?
2. RAG
 - 1) Embedding & Vector DB
 - 2) Indexing
 - 3) Retrieval
 - 4) Prompt & Generation
3. AI Agent
4. AI Agent platforms
5. Use Cases

RAG Retrieval Augmented Generation: Trustworthy Large AI

- 타깃 영역 내에서 검색된 문서를 입력으로 질문에 대한 응답을 생성도록 유도
- 타깃 영역에서 신뢰성 있는 응답만을 제공하는 Large AI 기술



① Query Analyzer

- 발화/문맥을 활용한 Query Generation
- 검색 키워드 추출, 주제/영역 반영

② Passage Retriever

- External Knowledge (신규문서, 도메인문서) 검색
- Neural Dense Retrieval

③ Response Generator

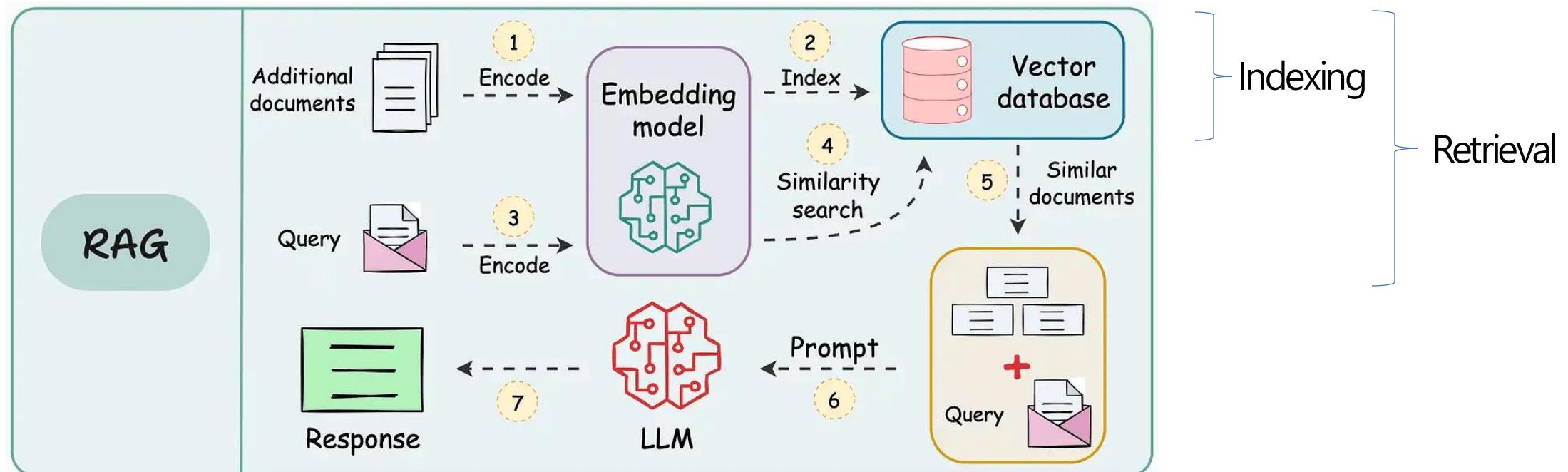
- 검색된 Passage와 대화문맥/장기기억을 반영한
- 응답/요약/변환 생성모델
- 추론에 참조한 정보원의 신뢰도 평가를 추론결과 선택에 반영
- 생성된 결과가 이전 발화 및 검색된 Passage의 내용과 일관성을 가지는지를 평가하여 필터링

④ Long-term memory

- (개인화 대화) 장기기억 저장을 위한 누적 Memory Encoder

RAG Retrieval Augmented Generation

- 타깃 영역 내에서 검색된 문서를 입력으로 질문에 대한 응답을 생성도록 유도
- 타깃 영역에서 신뢰성 있는 응답만을 제공하는 Large AI 기술



<https://blog.dailydoseofds.com/p/rag-vs-agentic-rag>

Mi:dm

36개월 아기의 IQ에 가장 영향을 많이 주는 요인은?

36개월 아기의 IQ에 가장 영향을 많이 주는 요인은 언어 자극이에요.

각 기업별 내부 문서가 접근되지 않으므로 부족한 응답 (혹은 잘못된 정보 노출)

Context (관련 내부문서 탐색결과) :

육아와 비슷한 개념인 양육에 관한 Benasich & Brooks-Gunn(1996) 연구에서 생후 12개월의 양육환경이 36개월의 IQ에 가장 영향을 많이 주는 강력한 예측인자로 나타났다.

Mi:dm

36개월 아기의 IQ에 가장 영향을 많이 주는 요인은?

양육환경이에요. [관련근거]

각 기업별 내부 문서에 기반하여 존재하는 지식을 응답 (근거 동시 제공)

부모·자녀 건강학회지 저?권 제1호

어머니의 자녀 들보는 활동에 관한 연구에 의하면 심장허(1997)는 산후 1~6개월 된 영아도 138명을 대상으로 한 연구에서 임신질경의 주체가 어내와 남편이었지만 후 임신 전까지의 기간이 12개월 이상일 때 임신의도가 강하고 육아태도가 긍정적인 것으로 보고하였다. 이러한 연구에서는 임신 및 출산질경이 육아에 포함됨을 의미한다. 또한 육아에 대한 인지 및 책임감과 같은 어머니의 자각과 태도에 영향을 주는 것은 임신 전에 가졌던 생활이나 임신의도와 관계 있으며 실제로 진행되는 육아과정 중에 모사 상호관계를 통해서도 강화될 수 있다.

조산모들은 산생아 들보기 활동 중 목욕시키기, 배변 소독하기, 온을 파악하기, 아기상태 파악하기 등을 어려워한 것으로 보고되었으며(이자형, 1996) 영아의 어머니는 아버지와 위생, 퀴리, 자녀의 성장을 말. 부모·자녀 상호작용에 대한 교육요구도가 높은 것으로 나타났다(한경자, 1997). Fenwick, Barclay & Schriedel(2001)는 산 후 7주 된 산모 1161명에게 필요한 육아와 관련된 정보는 아기의 청결, 성장발달, 아기 담래기 등에 관한 것이라고 보고하였다. 또한 어머니들이 육아와 관련된 사랑에 대해서 진화 상당한 내용을 본문에 보면 이상 칭호, 수유 및 염유, 육아농식, 성장발달, 배설, 칭보제공, 염려행동, 수면, 예방접종, 시고촌선, 약전 등에 대한 관심이 많은 것으로 나타났다(송지호 외, 2001).

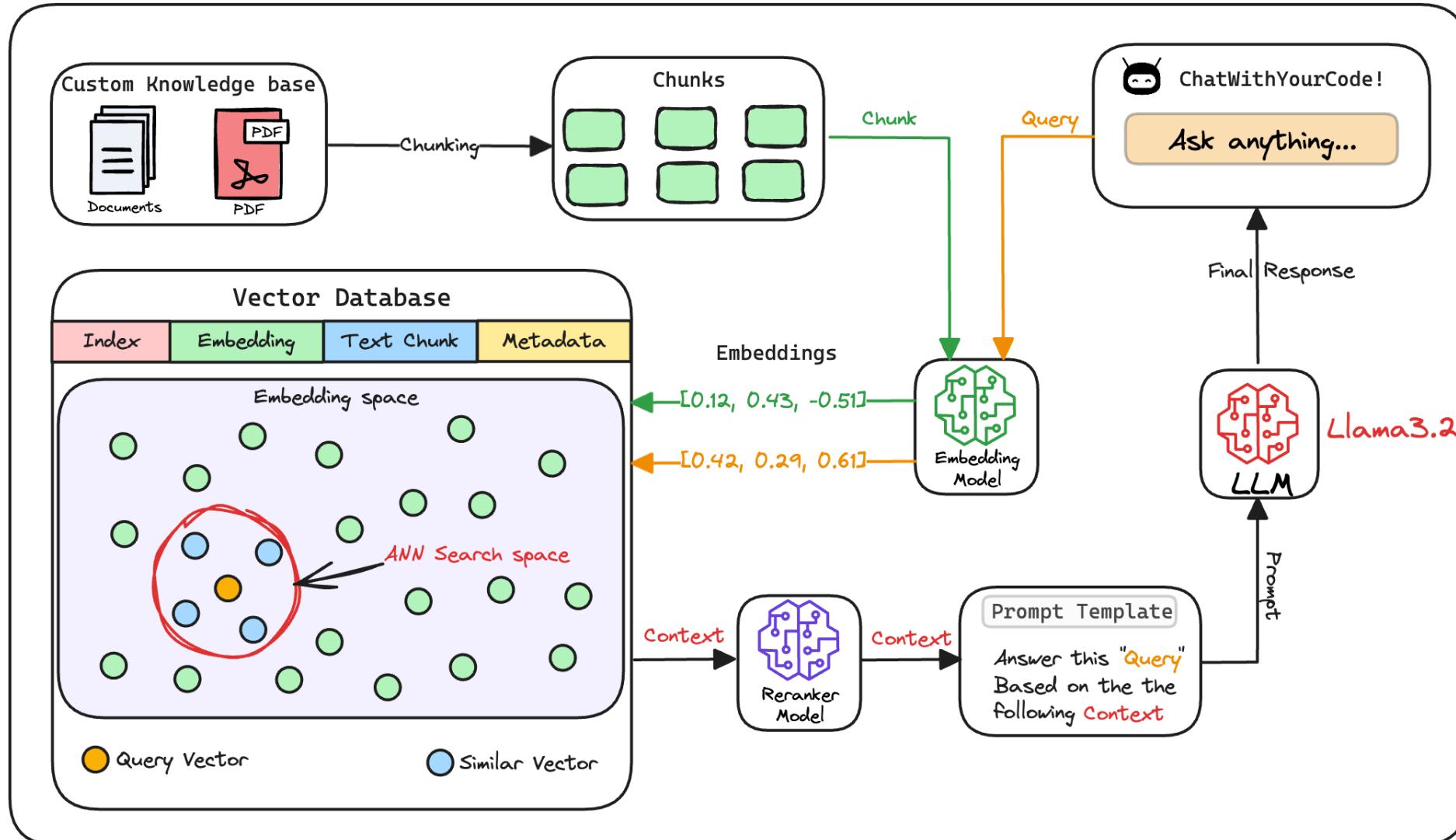
1) 영양과 글 : 솔직하고 균형 있는 영양을 공급하는 것을 의미한다. 문현고(김순이, 1991; 유안진, 1994; 이정은, 1992; 이자형, 1996; Bick MacArthur & Lancashire, 1998; Boback, et al., 1989; Chege, et al., 2002)에서 육아는 모유 먹이기, 수유 및 염유, 육아농식, 성장발달, 배설, 칭보제공, 염려행동, 수면, 예방접종, 시고촌선, 약전 등에 대한 관심이 많은 것으로 나타났다(송지호 외, 2001).

2) 사랑하는 마음 : 자녀에 대한 부드러운 대도나 따뜻한, 아름다운 요구에 대한 지속적인 관심을 가지는 것을 의미한다. 문현고(김순이, 1992; 이자형, 1996; Boback, et al., 1989; Fenwick, et al., 2001; Newman & Newman, 1995; Siegel, 1999)에서 육아는 아동의 여러 가지 상태에 관심을 가지고, 부드럽고 따뜻한 태도로 대하며, 민감하게 반응하면서 헌신하는 것으로 언급하

RAG Indexing & Retrieval

<https://www.dailydoseofds.com/a-crash-course-on-building-rag-systems-part-1-with-implementations/>

- Workflow of Indexing & Retrieval in RAG



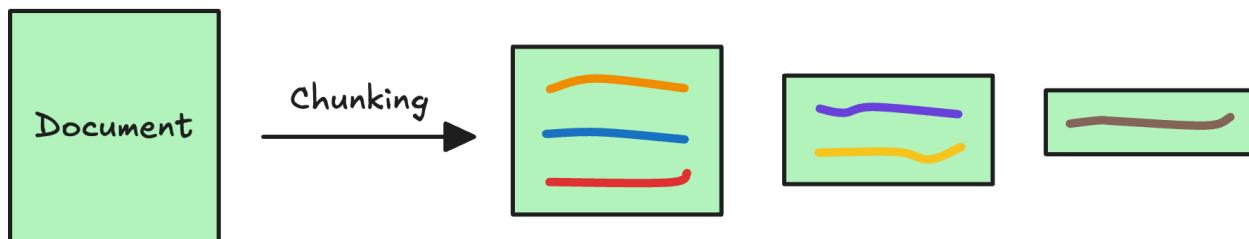
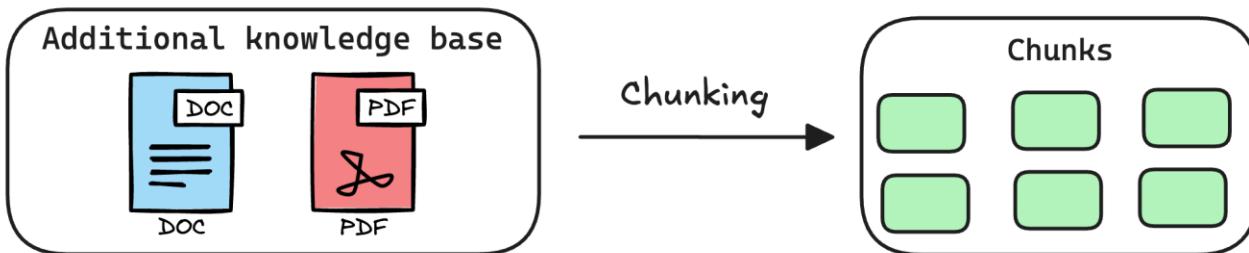
RAG Indexing & Retrieval

<https://www.dailydoseofds.com/a-crash-course-on-building-rag-systems-part-1-with-implementations/>

[0] Pre-processing

Document Loader / 문서변환

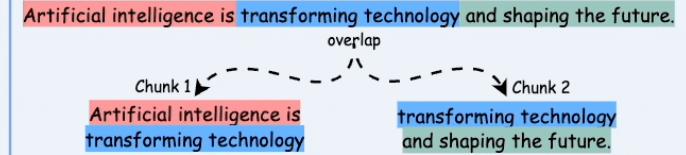
[1] Create chunks



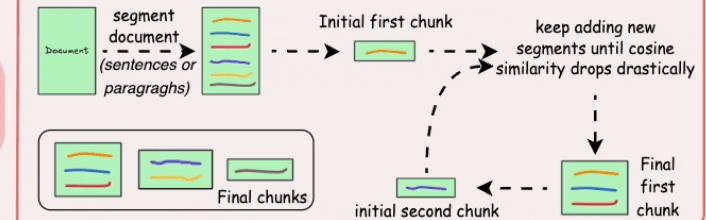
5 Chunking Strategies for RAG

blog.DailyDoseofDS.com

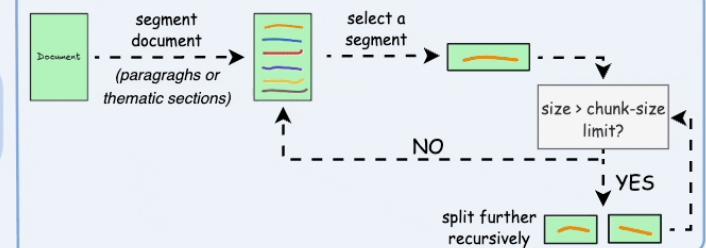
1) Fixed-size chunking



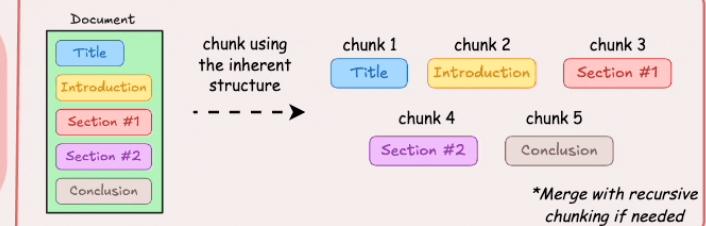
2) Semantic chunking



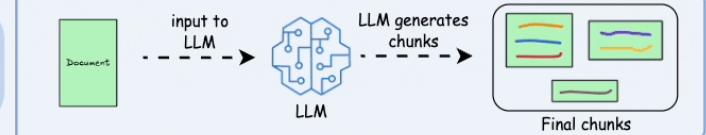
3) Recursive chunking



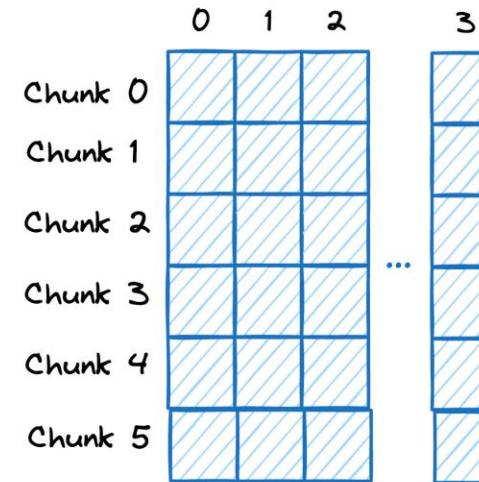
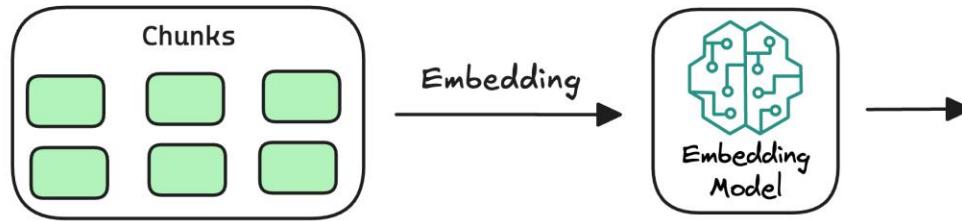
4) Document structure-based chunking



5) LLM-based chunking

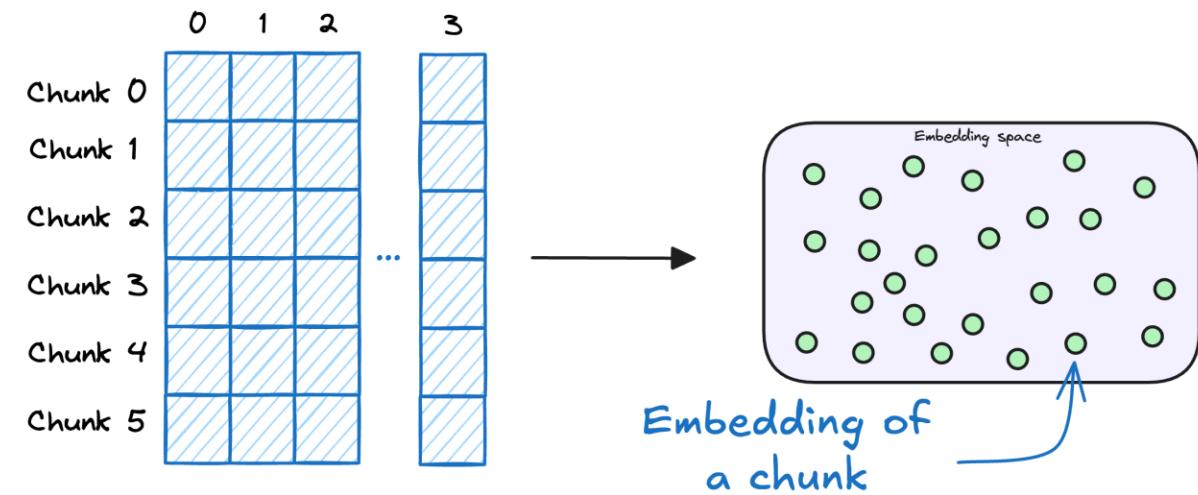


[2] Generate embeddings



[3] Store embeddings in a vector DB

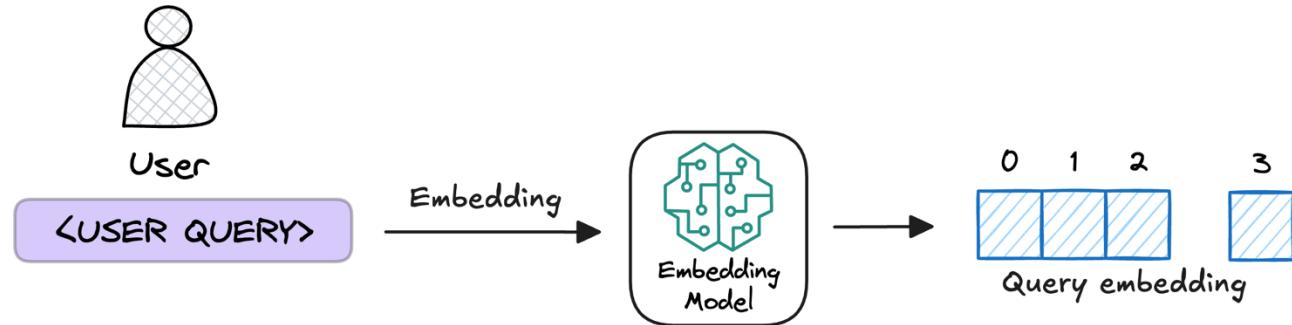
- A vector database also stores the **metadata** and **original content** along with the vector embeddings.
- Multimodal contents
 - figure, chart, video, audio, ...
- Metadata
 - Document information, Author, Date, Summarization, ...



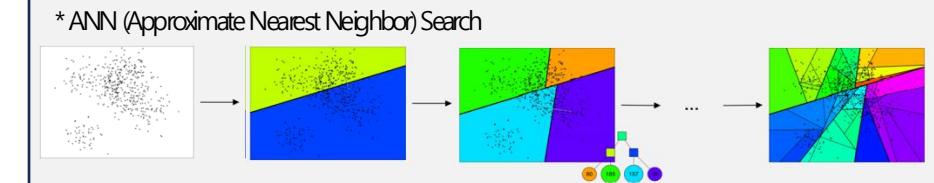
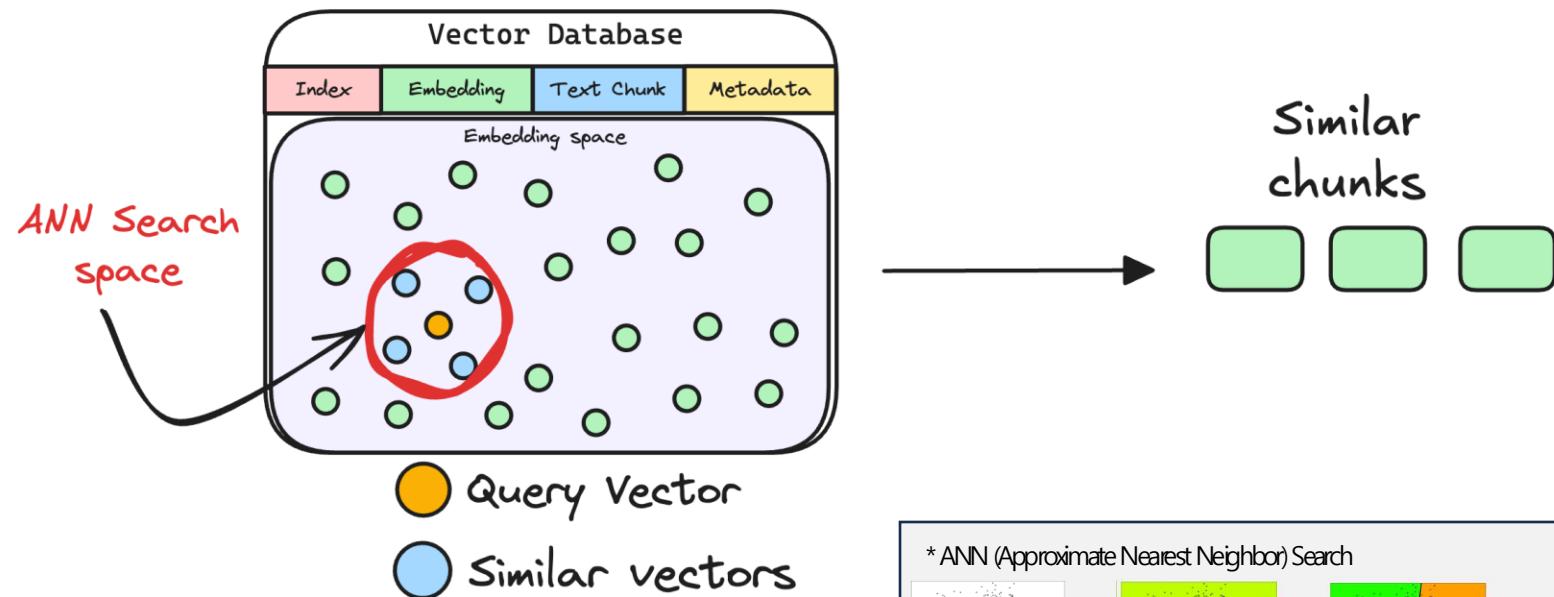
RAG Indexing & Retrieval

<https://www.dailydoseofds.com/a-crash-course-on-building-rag-systems-part-1-with-implementations/>

[4] Embed user query



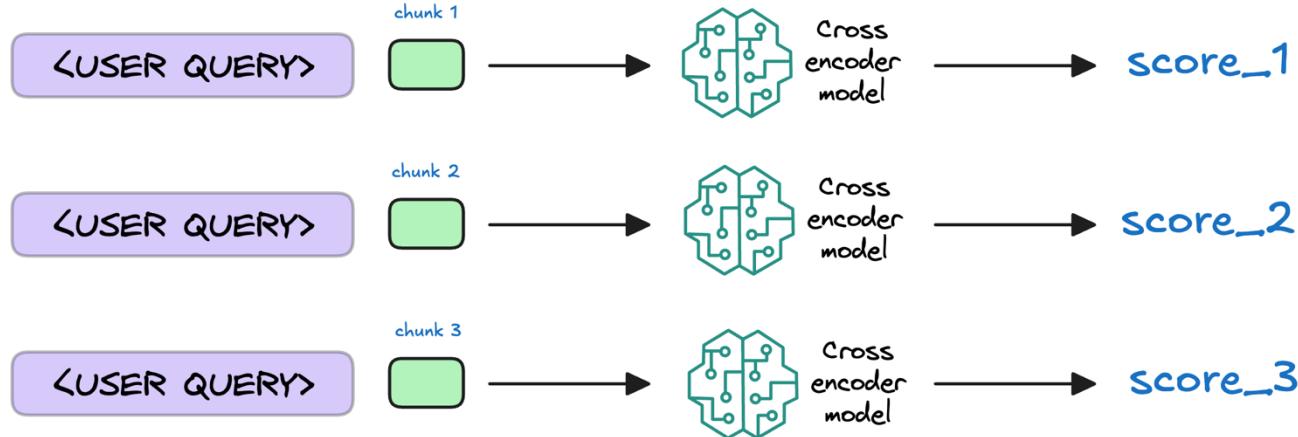
[5] Retrieve similar chunks



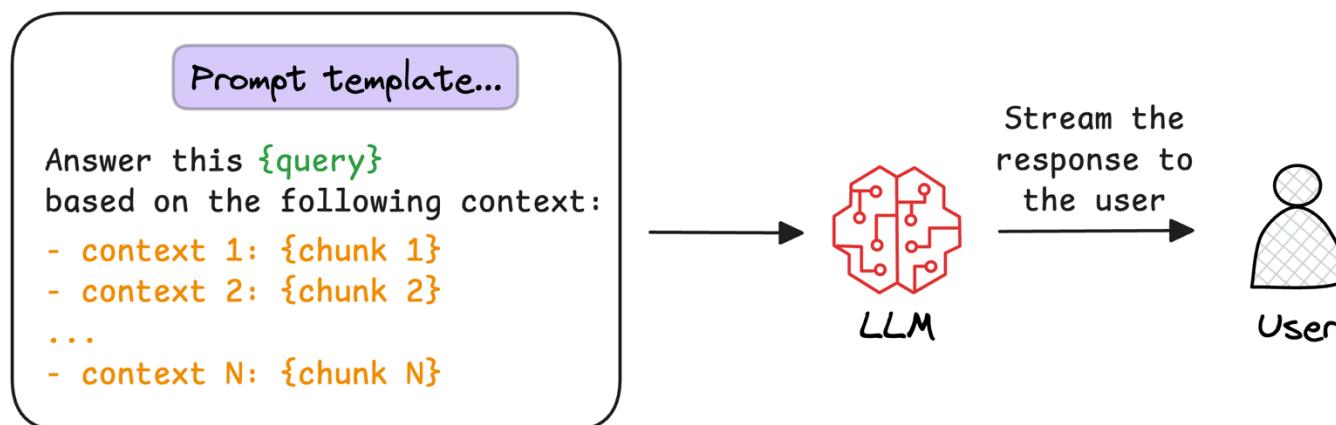
RAG Indexing & Retrieval

<https://www.dailydoseofds.com/a-crash-course-on-building-rag-systems-part-1-with-implementations/>

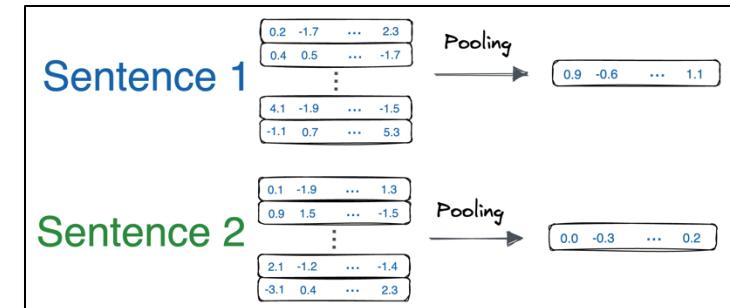
[6] Re-rank the chunks



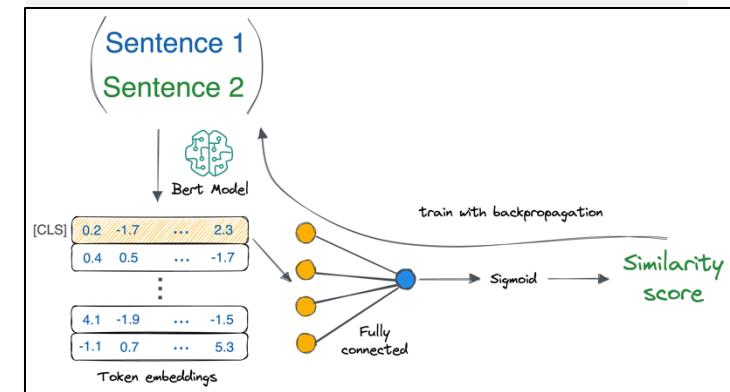
[7] Generate the final response



* Bi-encoder / Vector 비교만 수행 / 1차 Retrieval 단계에서 이용



* Cross-encoder / LLM 추론 필요 / 2차 re-ranker 단계에서 이용



RAG QA시스템 개발 Process

1. Domain 분석 & 평가셋 구축
 - 1) 해당영역에서 사용되는 질문의 종류와 유형/패턴 분석
 - 2) 지식원으로 사용 가능한 문서의 규모와 형식 파악
 - 3) 검증용 셋 구축 (검증용 질문 + 검색해야할 문서 + 도출해야할 정답)
2. 초기 RAG / QA Agent 구축 & 검증셋을 이용한 문제점 파악
 - 1) 검증용 셋을 이용한 성능 검증 ($Q \rightarrow D$ 성능, $Q \rightarrow A$ 성능의 2단계 검증 필요)
 - 2) 문제의 원인 분석
3. RAG 성능 최적화
 - 1) pdf 문서를 텍스트로 변환하는 과정에의 오류를 수정하여 성능의 향상
 - 2) 문서의 chunck 생성 방법을 변환함으로서 성능의 향상
 - 3) LLM을 호출할 때 사용되는 prompt를 바꿔봄으로서 성능의 향상
 - 4) 임베딩 모델을 바꿔봄으로서 성능의 향상
 - 5) re-ranker 알고리즘의 방법을 바꿔봄으로서 성능의 향상 비교
 - 6) 검색의 대상으로 메타정보를 같이 이용함으로서 성능의 향상
 - 7) 문서 유사도 측정 방법을 바꿔봄으로서 성능의 향상

RAG RAG / QA Agent 성능 최적화

4. QA Agent 성능 최적화

- 1) 입력의 유형에 따라 서로 다른 DB를 검색하게 하거나, 다른 문서를 검색하게 DB를 분리
- 2) 검색된 결과가 실제 문서에서 추출되었는지를 판단하는 Fact Checker를 도입하여, 가능한 여러 방법론에 따른 성능의 향상 여부를 검증
- 3) 여러 턴에 걸쳐서 질문이 들어오는데 이들을 해결하기 위해 질의응답 경과를 Memory에 저장하고 사용하는 여러 알고리즘을 적용하여, 성능의 향상 여부 검증
- 4) 한번의 검색결과에서 적절한 문서가 검색되지 않았을 때, 질문 자체를 확장하거나 변형하여 반복적으로 검색함으로서, 정답을 포함한 적합한 문서가 검색에 누락되지 않도록 검색, 이를 통한 성능의 향상 비교
- 5) Q+D-A set을 이용한 선호학습(SFT+RL)으로 사용자가 보다 선호하는 응답의 형태로 출력
- 6) CoT형태의 Reasoning 능력 부여로, 여러 단계의 심층추론을 사용한 응답 제공

RAG RAG 성능평가

- 검색(Q→D) 성능 평가지표

- Precision
- Recall
- F1-score

		Precision		Recall	
		Actual		Actual	
		Positive	Negative	Positive	Negative
Predicted	Positive	TP	FP	TP	FP
	Negative	FN	TN	FN	TN

$$F1\text{-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- QA(Q→A) 성능 평가지표

- 단답형질문
 - EM(Exact Match)
- 서술형질문
 - Human 수작업 평가
 - LLM-as-a-Judge (G-eval)

You will be given a user_question and system_answer couple. Your task is to provide a 'total rating' scoring how well the system_answer answers the user concerns expressed in the user_question.

Give your answer on a scale of 1 to 4, where 1 means that the system_answer is not helpful at all, and 4 means that the system_answer completely and helpfully addresses the user_question.

Here is the scale you should use to build your answer:

- 1: The system_answer is terrible: completely irrelevant to the question asked, or very partial
- 2: The system_answer is mostly not helpful: misses some key aspects of the question
- 3: The system_answer is mostly helpful: provides support, but still could be improved
- 4: The system_answer is excellent: relevant, direct, detailed, and addresses all the concerns raised in the question

Provide your feedback as follows:

Feedback:::

Evaluation: (your rationale for the rating, as a text)

Total rating: (your rating, as a number between 1 and 4)

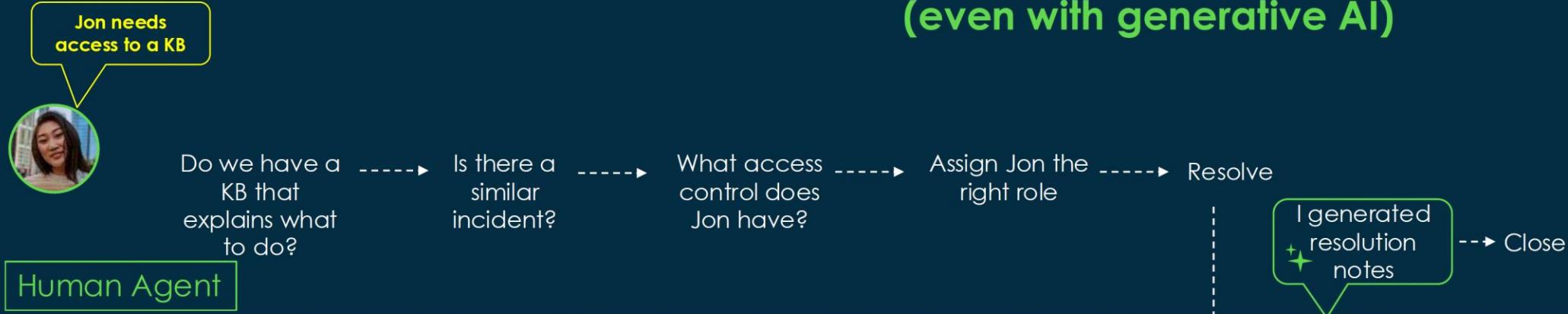
You MUST provide values for 'Evaluation:' and 'Total rating:' in your answer.

Now here are the question and answer. Question: {question} Answer: {answer}

Agentic AI

- 
1. Agentic AI?
 2. RAG
 3. AI Agent
 - 1) Component
 - 2) Tools / Environment
 - 3) Memories
 - 4) Multi-agent
 4. AI Agent platforms
 5. Use Cases

Today's Enterprise Workflows Remain Quite Manual (even with generative AI)

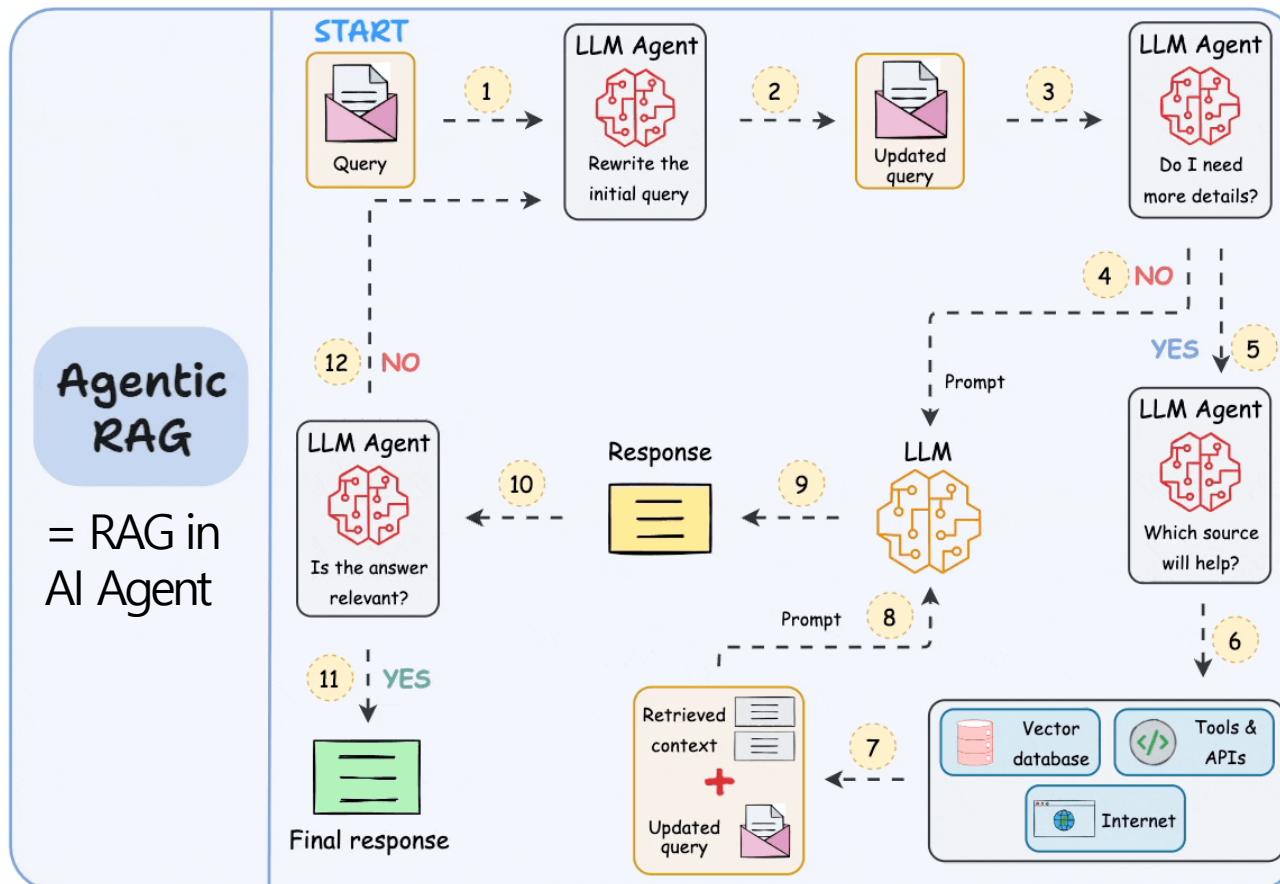


Resolution Generation skill

GenAI

AI Agent

자율적으로 도구를 사용하여 환경을 인식하고,
목표달성을 위해 의사결정 및 행동을 수행하는 지능형 SW



<https://blog.dailydoseofds.com/p/rag-vs-agentic-rag>

(1) Query Analyzer

- 질의 분석/분해

(3~5) Planning

- 응답을 위해 정보가 필요한가?
- 어디에서 찾을 것인가?
- 어떤 도구를 사용할 것인가?

(6) 정보 습득 (Context)

- Retrieval (Local/Web)
- Tools for Environment (APIs)

(7) LLM / Generation

(10) Evaluation / Validation

(11) Q/R+Context → Memory 저장

(1) Query Analyzer

- Query Reformulation / Rewriting

AI Agent

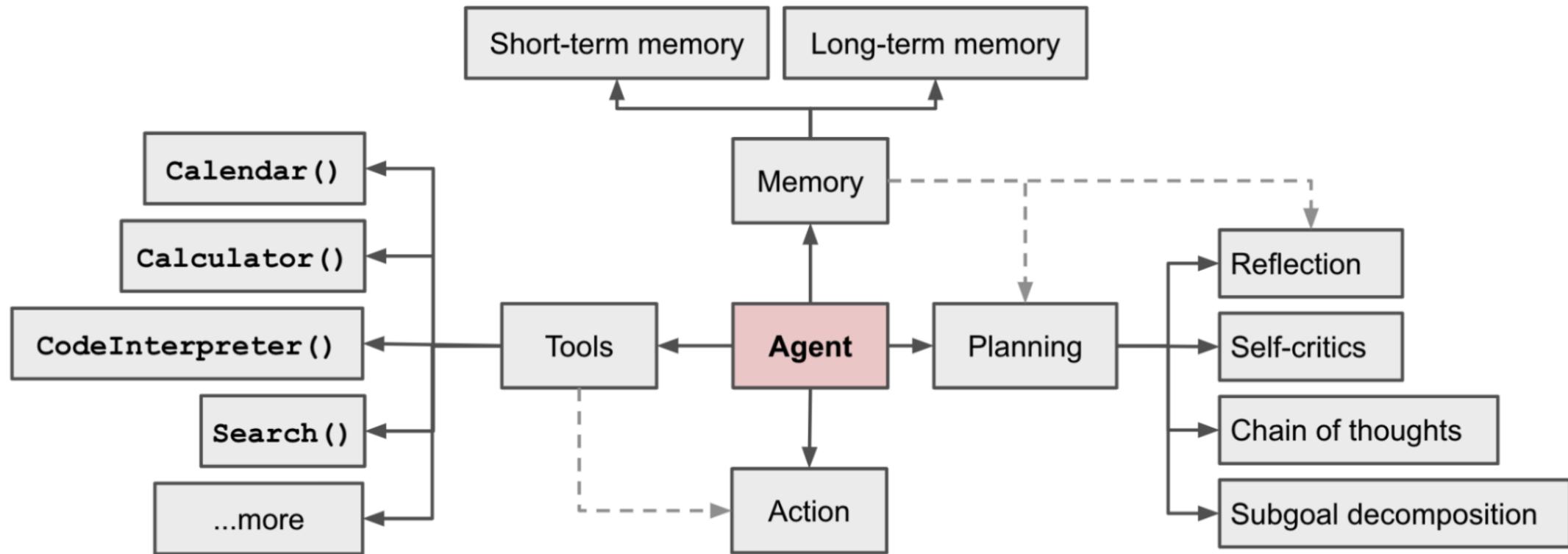
자율적으로 도구를 사용하여 환경을 인식하고,
목표달성을 위해 의사결정 및 행동을 수행하는 지능형 SW

LLM	에이전트
지식은 학습 데이터에 포함된 정보로만 제한	외부 시스템과 도구 연결을 통해 지식이 확장
- 사용자 질의에 기반한 단일 추론/예측 수행 - 대화 기록이나 지속적인 문맥을 관리하지 않음(예: 대화 기록 없음)	- 세션 기록(즉, 챗 기록)을 관리하여 다중 턴 추론/예측을 지원 - 사용자 질의와 오케스트레이션 계층에서 이루어지는 의사결정을 기반으로 동작
기본적으로 도구 사용 기능 없음	에이전트 아키텍처에 도구 사용 기능이 기본적으로 구현되어 있음
별도의 논리 레이어가 기본적으로 구현되어 있지 않음	CoT, ReAct 같은 추론 프레임워크나 LangChain 같은 사전 구축된 에이전트 프레임워크를 활용하는 인지 아키텍처가 구현되어 있음

AI Agent 구성요소

- 구성요소 & 동작방식

- Short-term memory : In-context prompt
- Long-term memory : external vector store and fast retrieval



AI Agent Planning

- Task Decomposition
 - Chain of thought (CoT; Wei et al. 2022) “Let’s think step by step”
 - Tree of thought (Yao et al., 2023)
 - “Step for XYZ. #n 1.”
 - “What are the subgoals for achieving XYZ?”
 - “Write a story outline first.”
 - LLM+P (Liu et al., 2023)
 - Translate the problem into Planning Domain Definition Language (PDDL)
- Self-Reflection
 - ReAct (Yao et al. 2023)
 - Reflexion (Shinn & Labash 2023)
 - Chain of Hindsight (CoH; Liu et al., 2023)
 - Self-RAG (Asai et al. 2024)

A Failure Example of GPT-4 in Planning

Problem (P1): You have 5 blocks. One cannot place more than one block on another block. b5 is on top of b3. b4 is on top of b2. b2 is on top of b1. b3 is on top of b4. b1 is on the table. b5 is clear. Your arm is empty.

Your goal is to move the blocks.

b1 should be on top of b2.
b3 should be on top of b5.
b4 should be on top of b1.

GPT-4:

1. Move b5 from b3 to the table.
2. Move b3 from b4 to b5.
3. Move b4 from b2 to the table.
4. Move b2 from b1 to b5.
5. Move b1 from the table to b2.
6. Move b4 from the table to b1.

Initial State: b1 (table), b2 (b1), b3 (b4), b4 (b2), b5 (clear).
Goal State: b1 (b2), b2 (b5), b3 (clear), b4 (b1), b5 (clear).

An Example PDDL Problem File Written by GPT-4 with Context

Context (shaded):

An example planning problem is: You have 5 blocks. b2 is on top of b5. b5 is on top of b1. b1 is on top of b4. b3 is on top of b2. b4 is on the table. b3 is clear. Your arm is empty. Your goal is to move the blocks. b4 should be on top of b3. The problem PDDL is:

```
(:objects b1 b2 b3 b4 b5)
(:init (arm-empty) (on b1 b4) (on b2 b5) (on b3 b2)
      (on-table b4) (on b5 b1)
      (clear b3))
(:goal (and (on b4 b3))))
```

Prompt: Description of Problem (P1) + Context
+ Provide me with the problem PDDL file that describes the planning problem directly without further explanations.

GPT-4 (the generated problem PDDL):

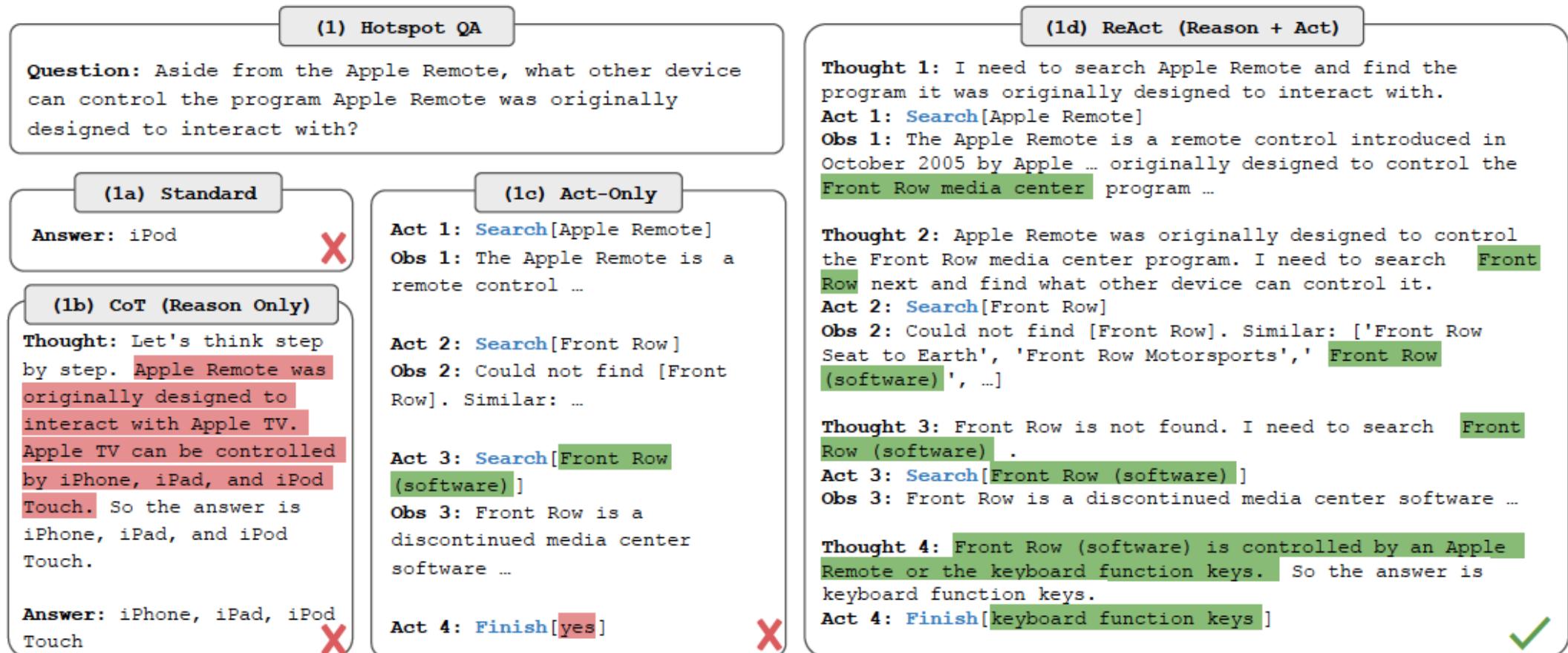
```
(:objects b1 b2 b3 b4 b5 )
(:init (arm-empty) (on-table b1)
(on b2 b1) (on b3 b4)
(on b4 b2) (on b5 b3) (clear b5)))
(:goal (and (on b1 b2) (on b3 b5)
(on b4 b1))))
```

Planner:

```
(unstack b5 b3) (putdown b5)
(unstack b3 b4) (stack b3 b5)
(unstack b4 b2) (putdown b4)
(unstack b2 b1) (putdown b2)
(pickup b1) (stack b1 b2) (pickup b4) (stack b4 b1)
```

AI Agent Planning

- Self-Reflection
 - ReAct (Yao et al. 2023)



AI Agent Memories

<https://lilianweng.github.io/posts/2023-06-23-agent/>

Embedding representation
for raw inputs

In-context learning
Prompt in context window

External vector store
and fast retrieval context

AI Agent memory

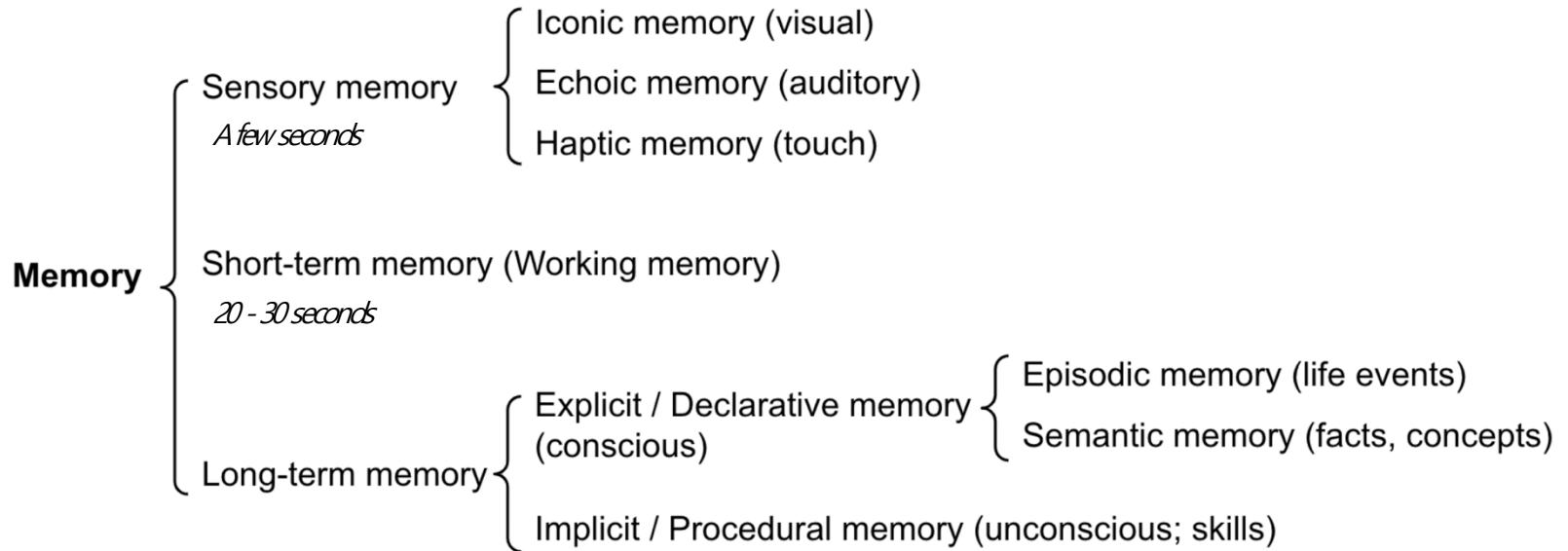


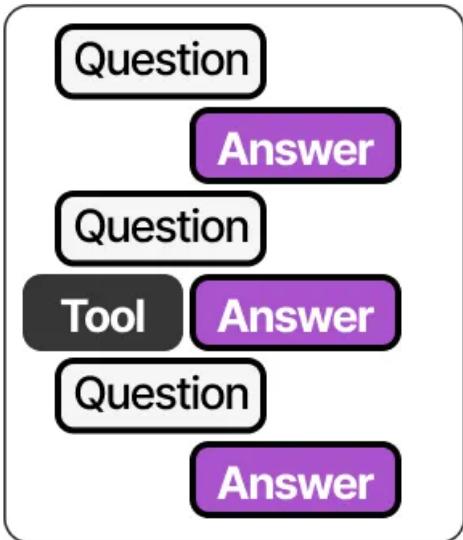
Fig. 8. Categorization of human memory.

Memory Type	Human example
Working	Agent's current and recent circumstances
Procedural	Instructions to determine the agent's behavior
Semantic	Facts about the world
Episodic	Sequences of the agent's past behaviors

AI Agent Memories

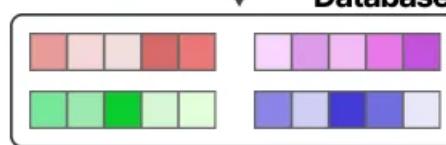
Short-term memory

(recent conversations and actions)



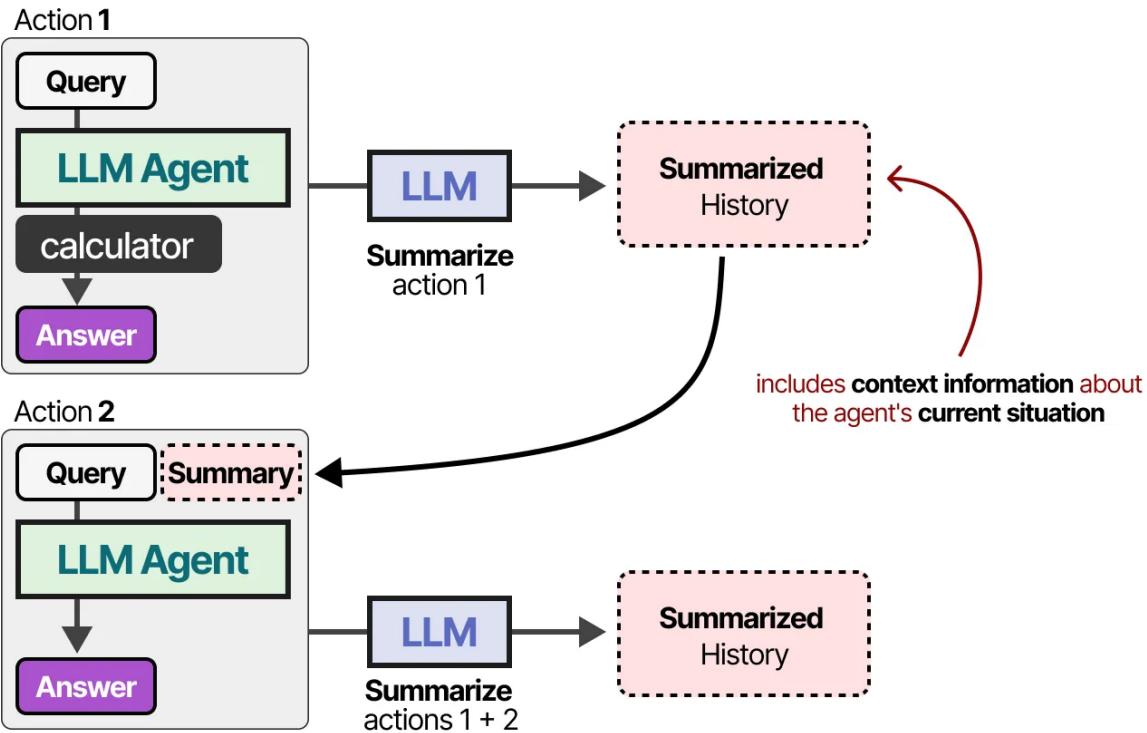
Long-term memory

(conversations and actions over an **extended period** or across **sessions**)

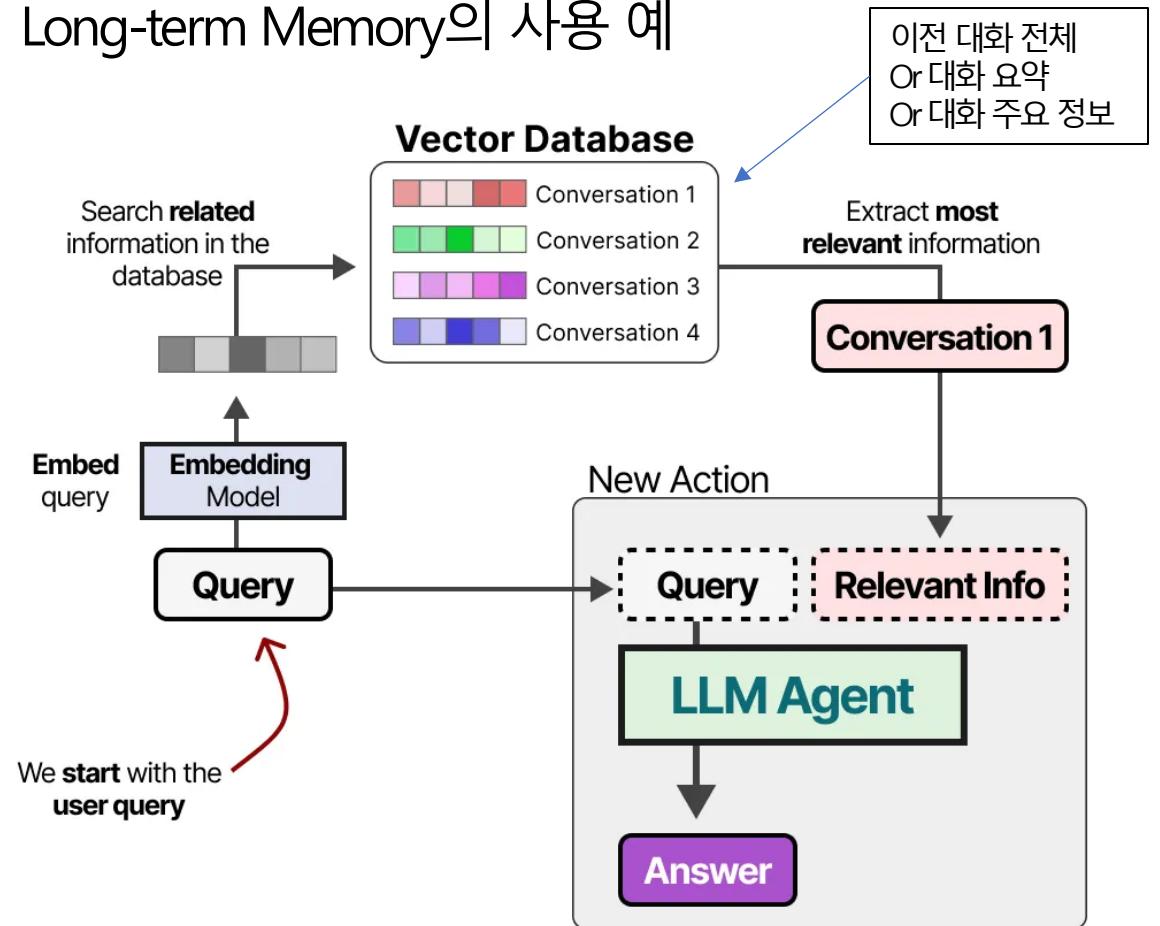


AI Agent Memories

Short-term Memory의 사용 예

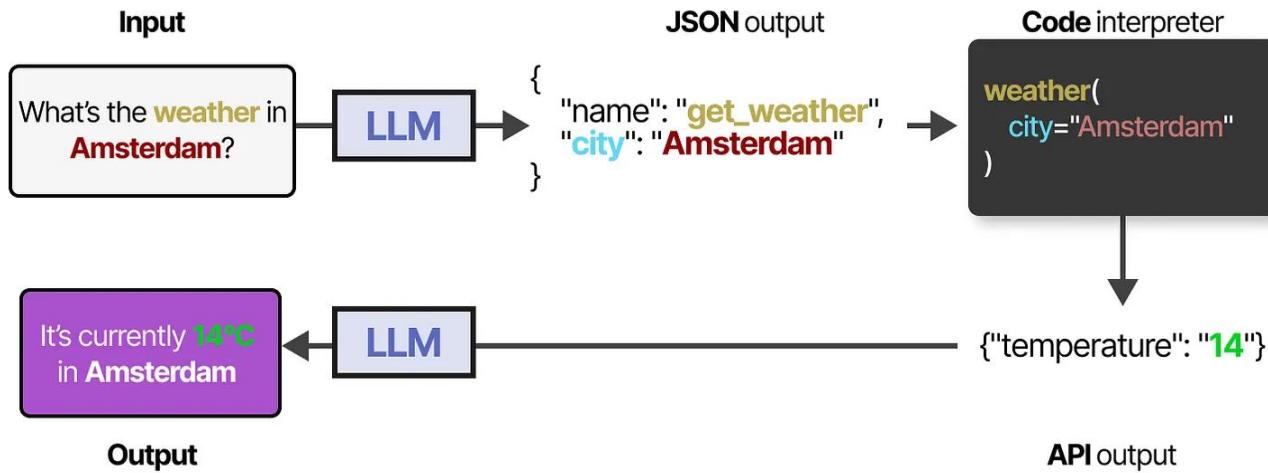


Long-term Memory의 사용 예

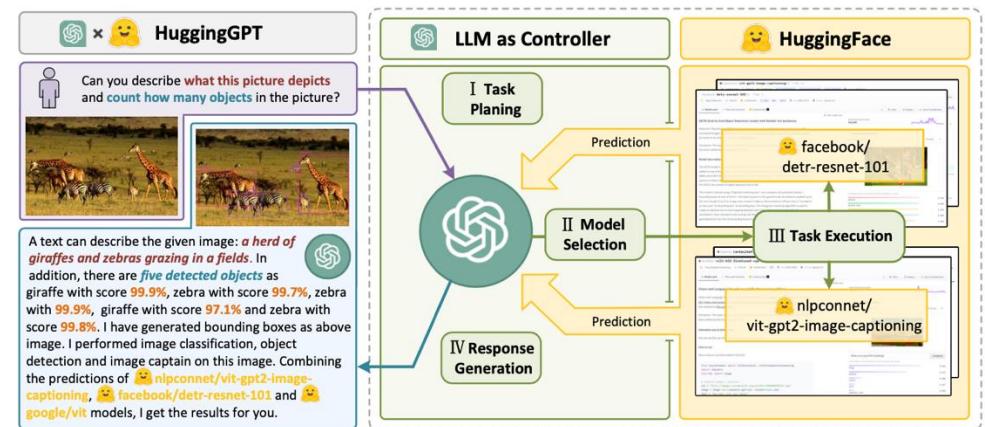


AI Agent Tool Use

- Human beings only create, modify and utilize external objects to do things.
- Tool Use for AI Agent



- MRKL (Karpas et al. 2022)
- Toolformer (Schick et al. 2023)
- HuggigGTP (Shen et al. 2023)



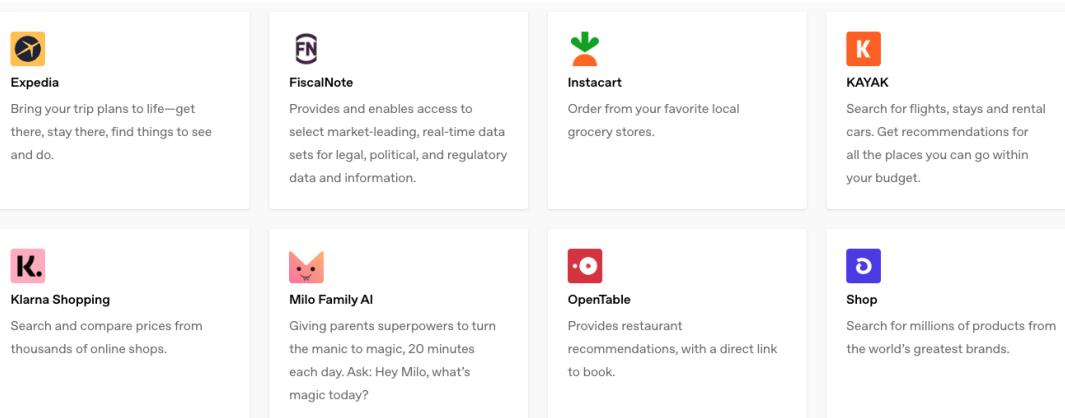
AI Agent Tool Use

- Tool Use for AI Agent
 - OpenAI function calling (2023)

What is the weather like in Paris today?

```
1 [{  
2     "id": "call_12345xyz",  
3     "type": "function",  
4     "function": {  
5         "name": "get_weather",  
6         "arguments": "{\"location\":\"Paris, France\"}"  
7     }  
8 }]
```

- ChatGPT Plugins/Actions (2023/2024)

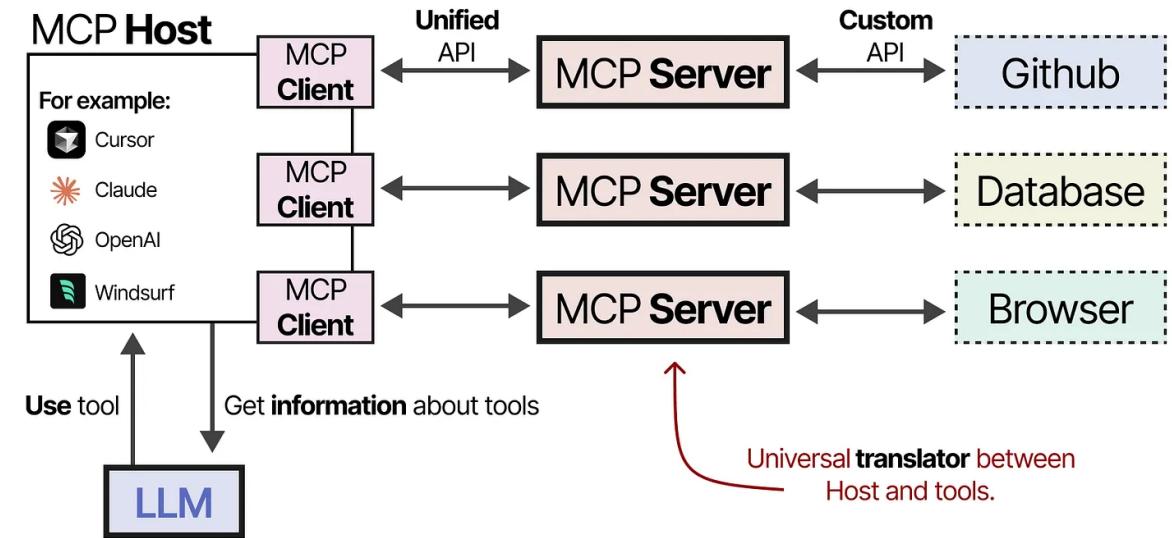
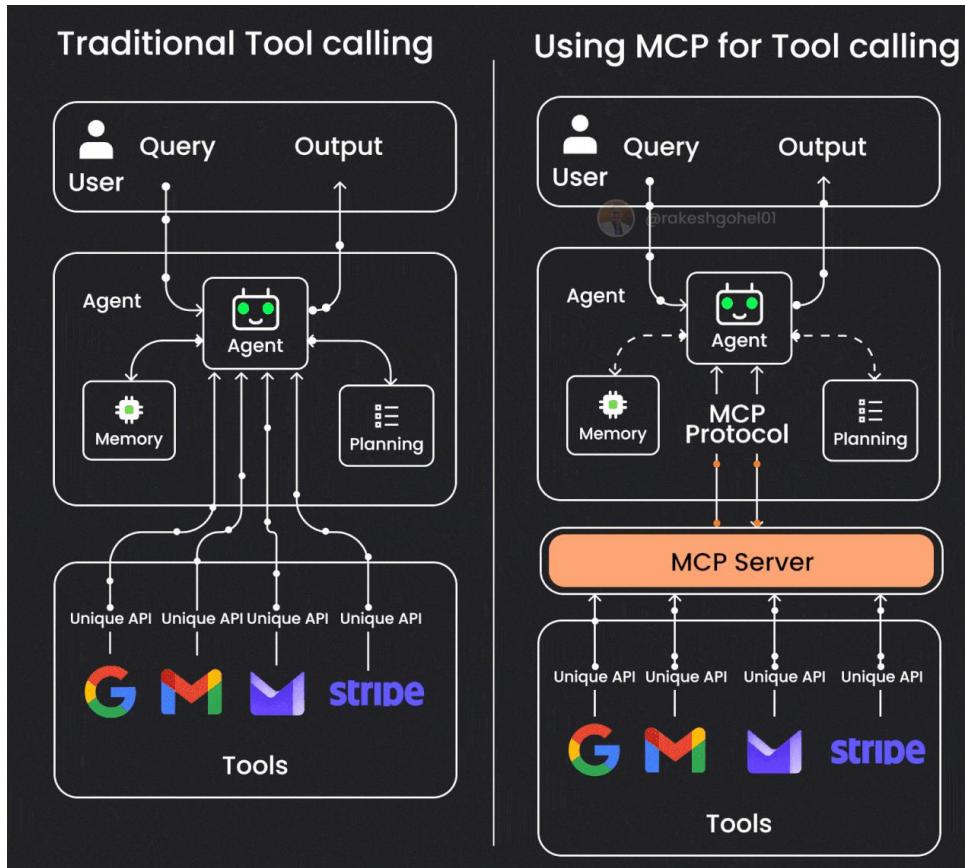


Function calling example with get_weather function

```
1  from openai import OpenAI  
2  
3  client = OpenAI()  
4  
5  tools = [  
6      {"type": "function",  
7      "function": {  
8          "name": "get_weather",  
9          "description": "Get current temperature for a given location.",  
10         "parameters": {  
11             "type": "object",  
12             "properties": {  
13                 "location": {  
14                     "type": "string",  
15                     "description": "City and country e.g. Bogotá, Colombia"  
16                 }  
17             },  
18             "required": [  
19                 "location"  
20             ],  
21             "additionalProperties": False  
22         },  
23         "strict": True  
24     }]  
25 }  
26  
27 completion = client.chat.completions.create(  
28     model="gpt-4o",  
29     messages=[{"role": "user", "content": "What is the weather like in Paris today?"}]  
30     tools=tools  
31 )  
32  
33 print(completion.choices[0].message.tool_calls)
```

AI Agent Tool Use

- Tool Use for AI Agent
 - Claude MCP (Model Context Protocol, Anthropic, 2024)
 - Cursor AI 채택, OpenAI 지원 발표 (2025), De facto standard로 발전 중
 - <https://docs.anthropic.com/ko/docs/agents-and-tools/mcp>



AI Agent Tool Use

- Tool Use for AI Agent
 - Claude MCP (Anthropic, 2024)

Prompt

Summarize the 5 latest commits

① send prompt to host

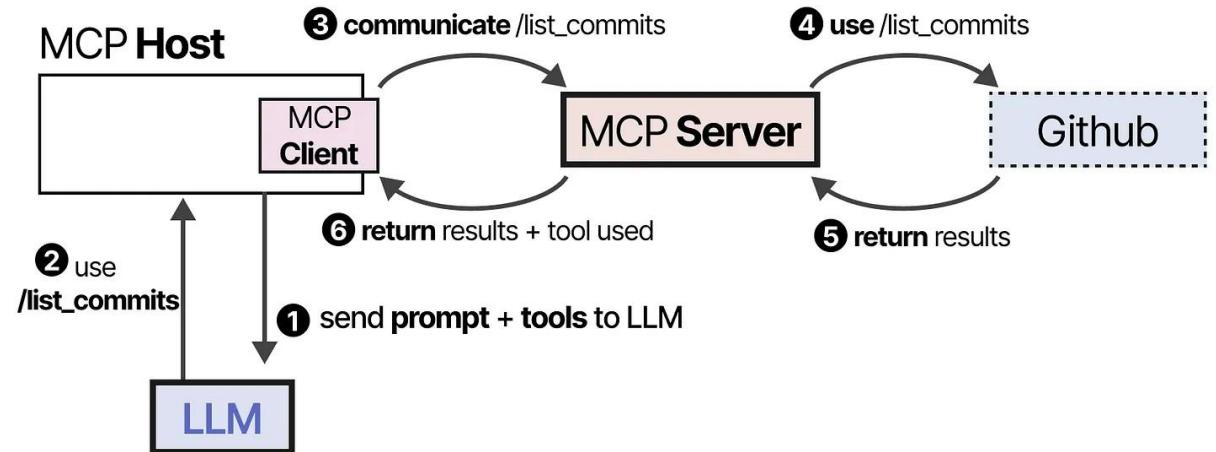
MCP Host

② list all available tools



③ return tools

- /list_commits
- /update_commits
- /pull_request
- ...

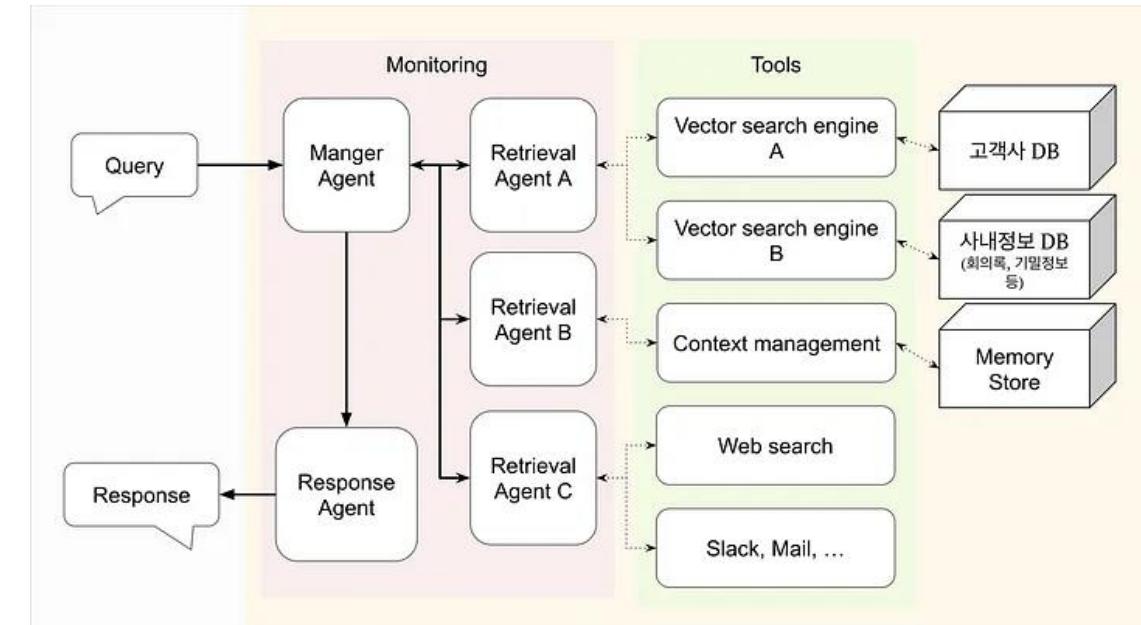
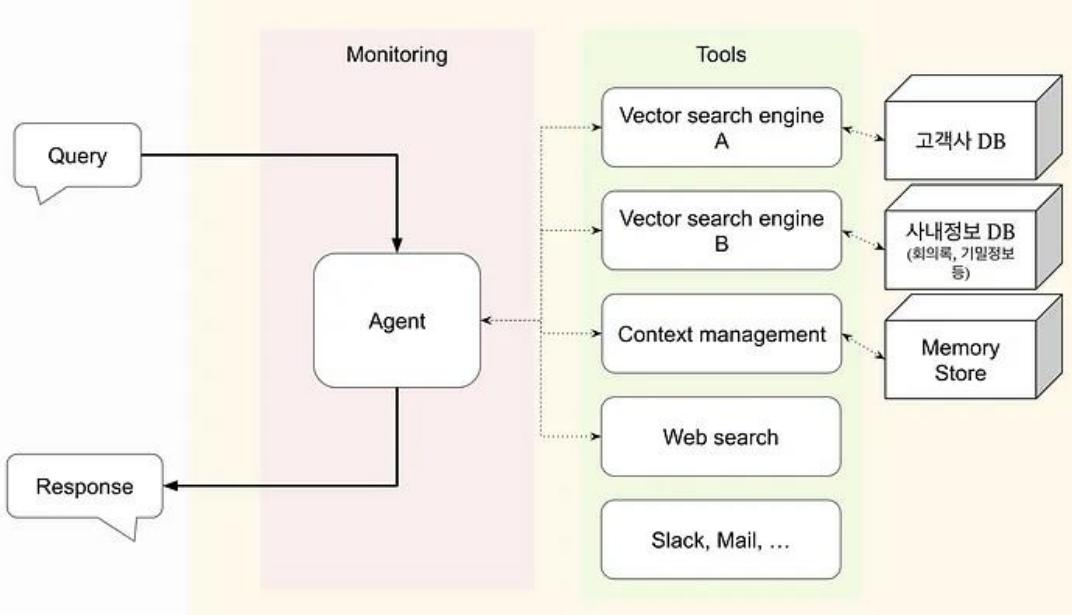
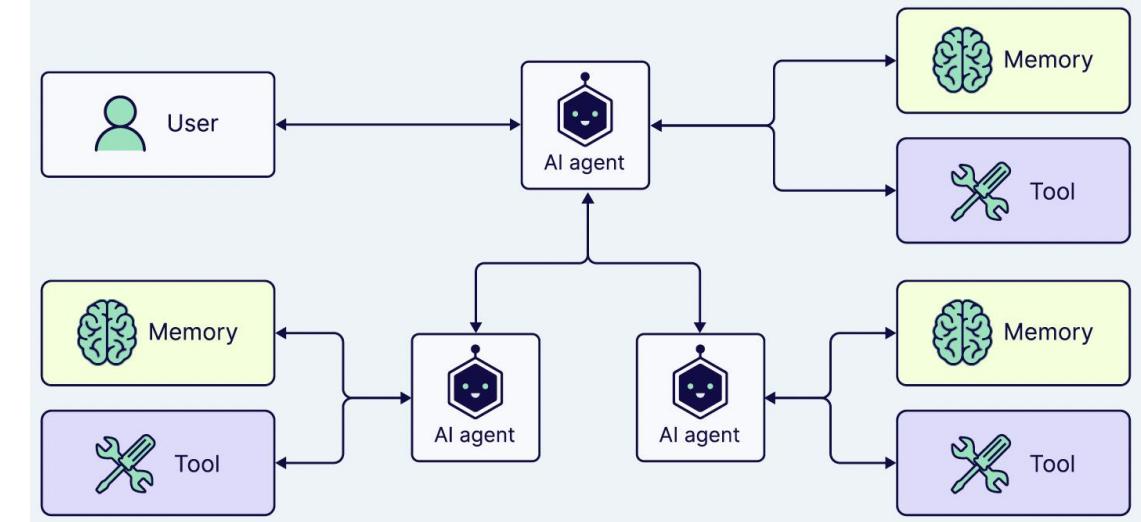
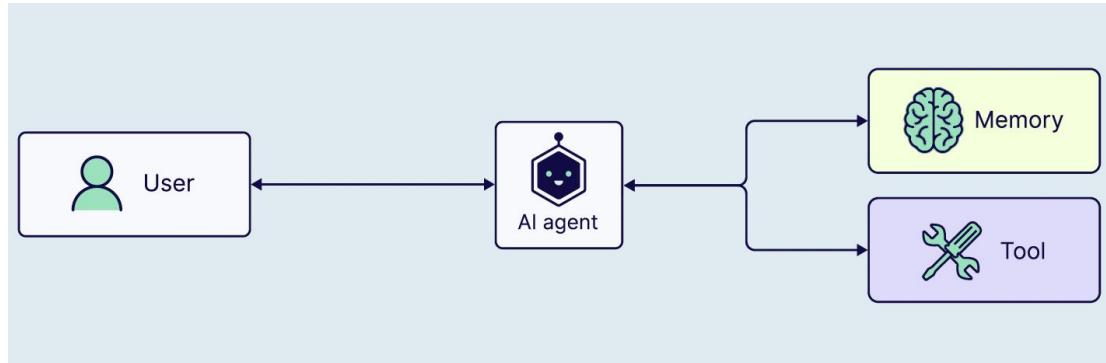


MCP Host



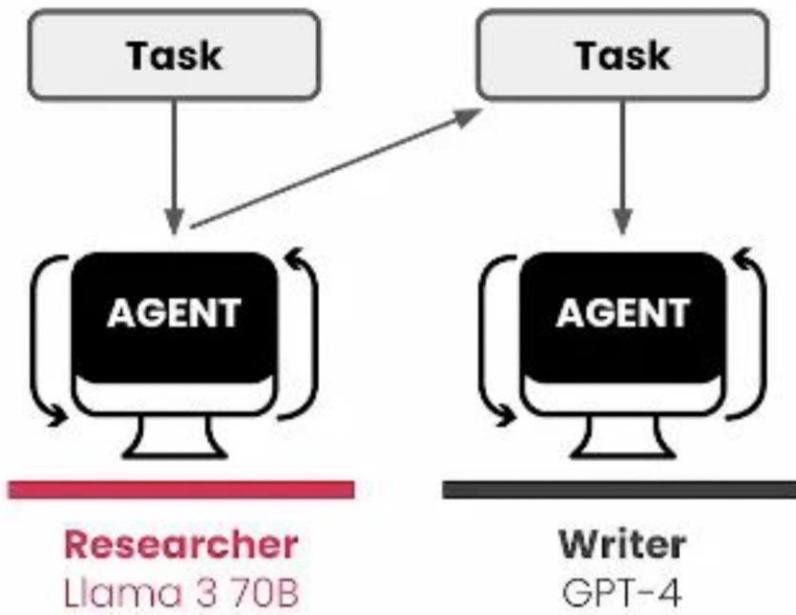
Multi Agent

Single Agent v.s. Multi Agent



Multi Agent

Single Agent v.s. Multi Agent



특징

1. Agent에 따라 다른 모델 할당
2. Agent에 따라 다른 도구 할당
3. Agent에 따라 다른 프롬프트 할당

기대효과

1. 비용 절감(저비용 모델 사용)
2. Agent별 도구 최적화(검색, 코딩, 파일 처리)
3. 추적 및 개선 효율성 증가(Writer 프롬프트 개선)

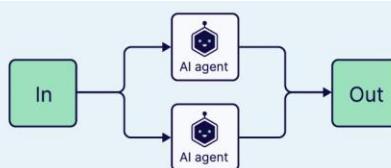
Multi Agent

Agentic workflow v.s. Agentic AI



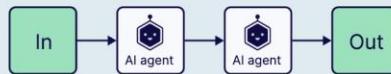
Multi Agent

Flow design patterns in multi agent system



Parallel

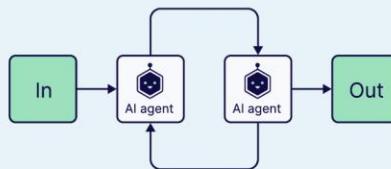
Multiple agents work simultaneously on different parts of a task.



Sequential

Tasks are processed sequentially, where one agent's output becomes the input for the next.

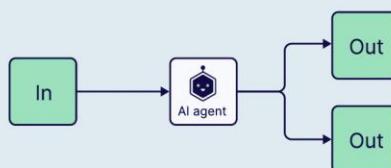
Example: Multi-step approvals.



Loop

Agents operate in iterative cycles, continuously improving their outputs based on feedback from other agent(s).

Example: Evaluation use cases, such as code writing and code testing.



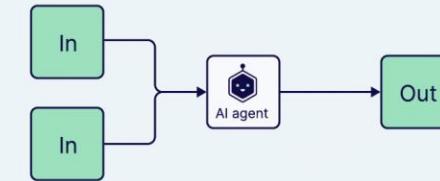
Router

A central router determines which agent(s) to invoke based on the task or input.

Aggregator

or synthesizer

Agents contribute outputs that are collected and synthesized by an aggregator agent into a final result.



Network

or horizontal

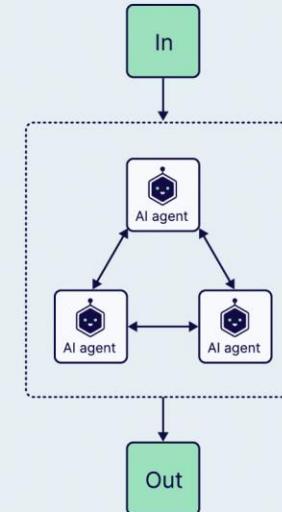
Agents communicate directly with one another in a many-to-many fashion, forming a decentralized network.

Pros:

- Distributed collaboration and group-driven decision-making.
- The system remains functional even if some agents fail.

Cons:

- Managing communication among agents can become challenging.
- More communication may cause inefficiencies and the possibility of agents duplicating efforts.



Hierarchical

or vertical

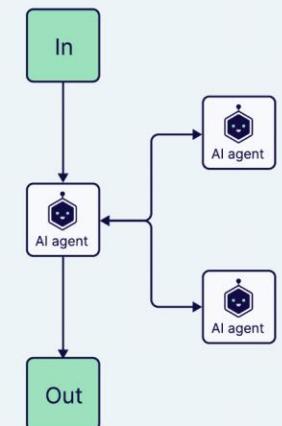
Agents are organized in a tree-like structure, with higher-level agents (supervisor agents) managing lower-level ones.

Pros:

- Clear division of roles and responsibilities among agents at different levels.
- Streamlined communication
- Suitable for large systems with a structured decision flow.

Cons:

- Failure at upper levels can disrupt the entire system.
- Lower-level agents have limited independence.

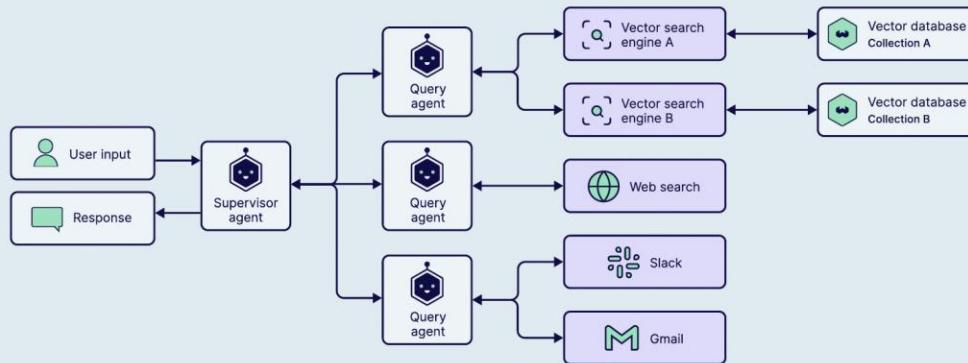


Multi Agent

Flow design patterns in multi agent system

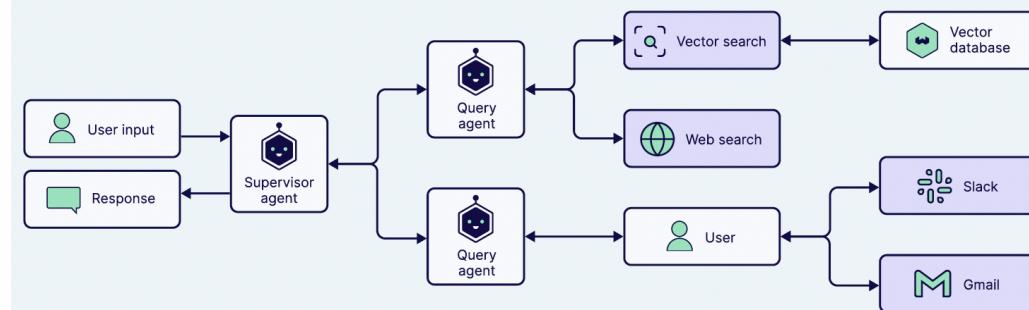
Hierarchical example

Let's begin with a simple hierarchical architecture, where you have one supervisor agent orchestrating multiple specialized agents.



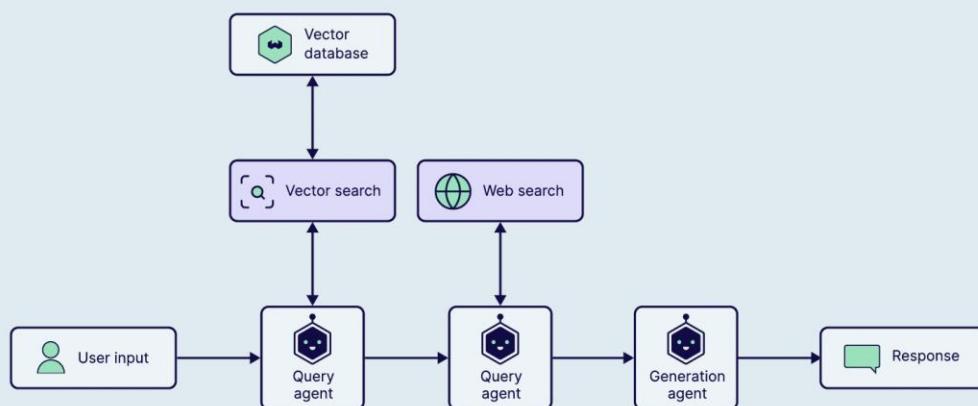
Human-in-the-loop example

If you don't want to give an agent access to sensitive information, such as an employee's personal emails, calendar, or chat messages, you can build a human-in-the-loop architecture. In this case, you can use a specialized agent that asks for human input before moving on to the next action.



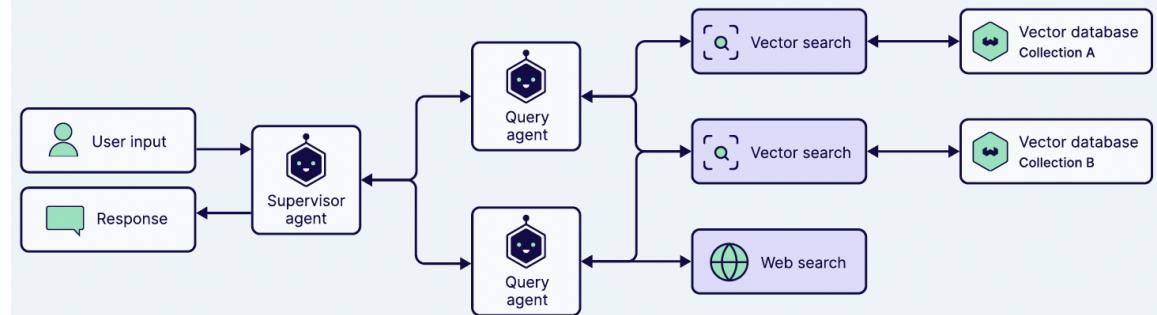
Sequential example

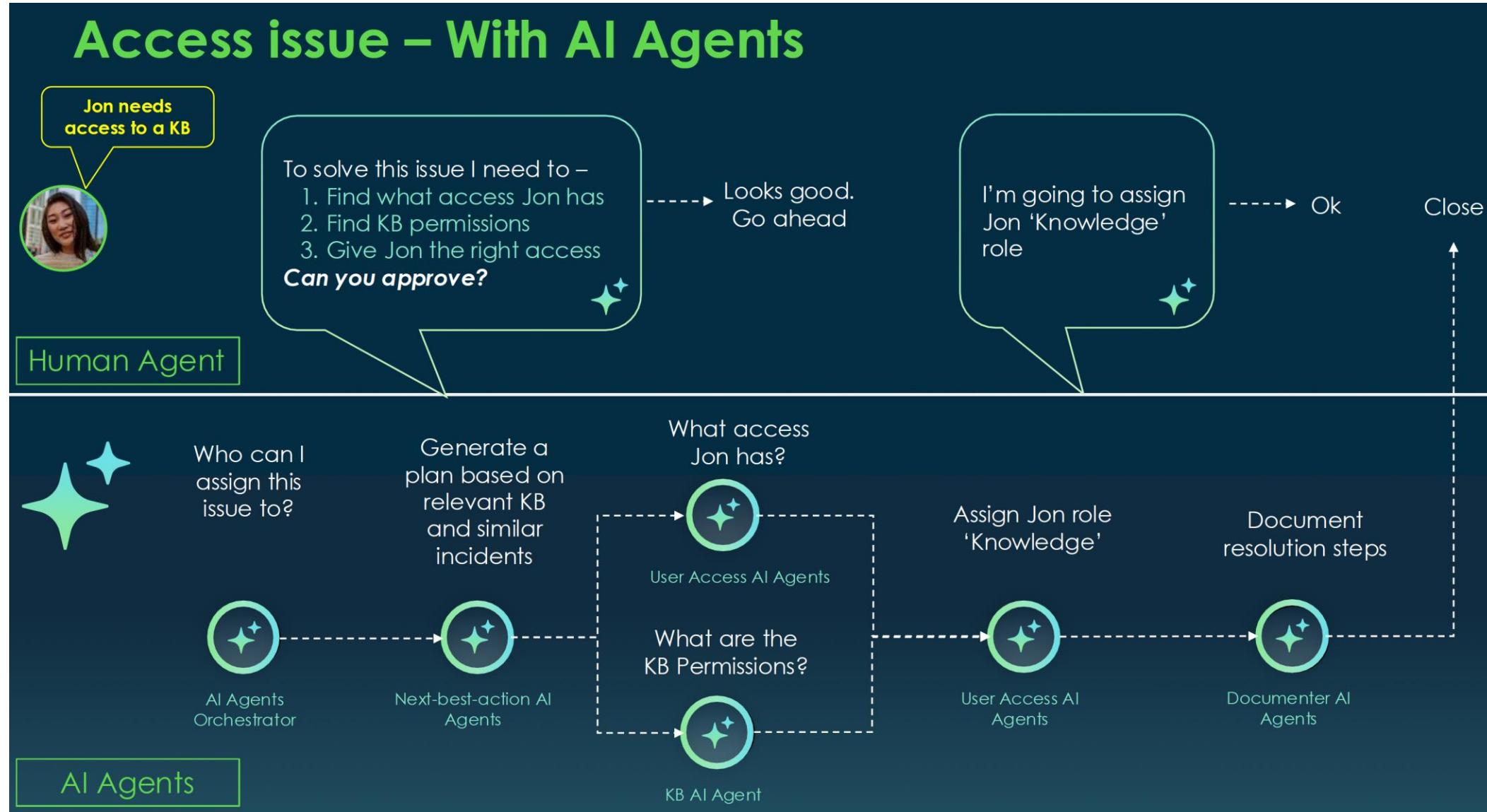
So far, we have looked at hierarchical multi-agent architectures only but if there is no need for a supervisor agent, you can also choose a network (or horizontal) architecture pattern.



Shared tools example

On the other hand, depending on your use case, it can also be helpful to have different agents with access to the same tool. For example, if your agents need to have access to central user information, then it might be necessary to provide them with a search tool over a central database.



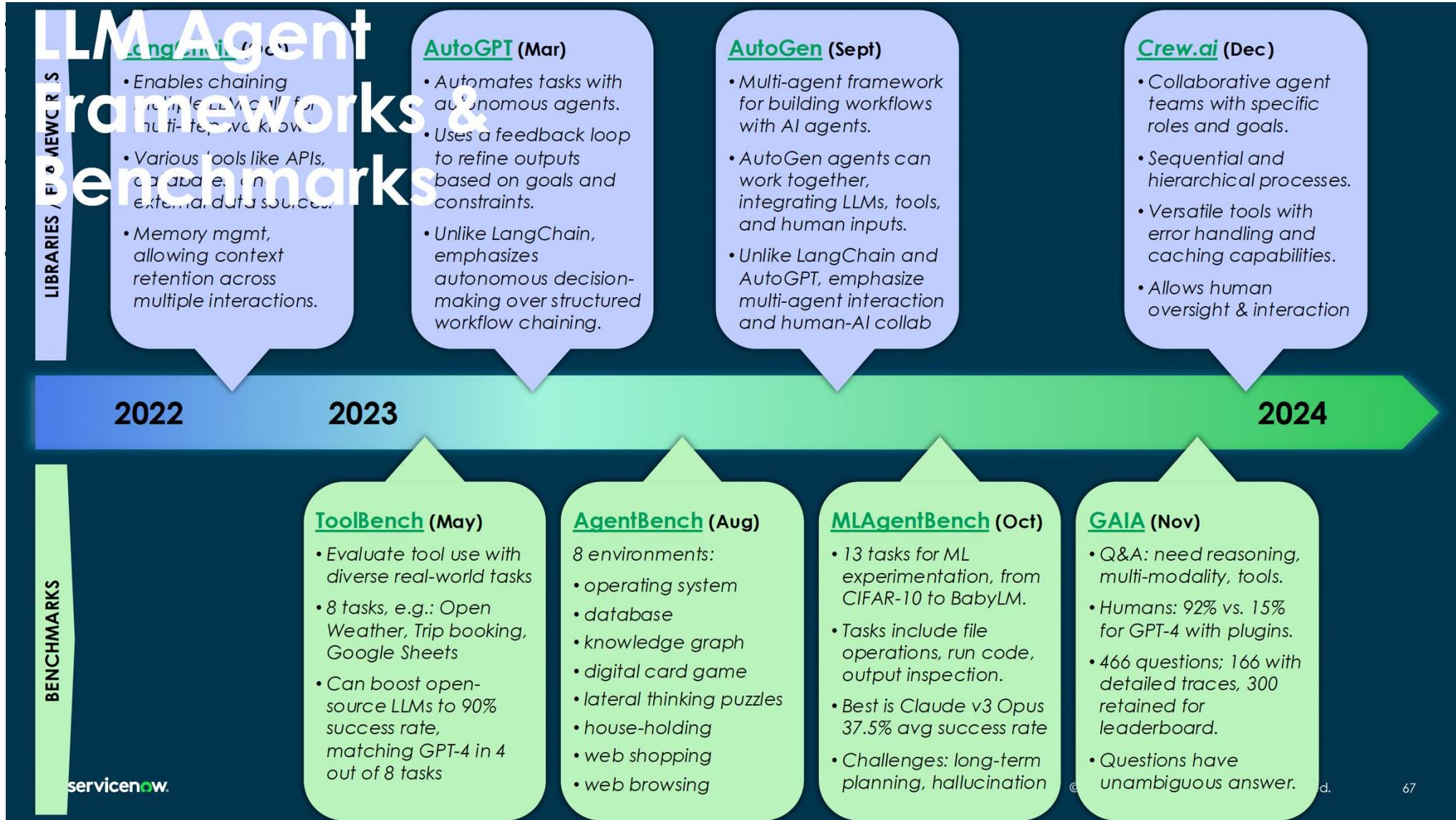


Agentic AI

1. Agentic AI?
2. RAG
3. AI Agent
4. AI Agent platforms
 - 1) LangChain
 - 2) Streamlit
 - 3) And more
5. Use Cases

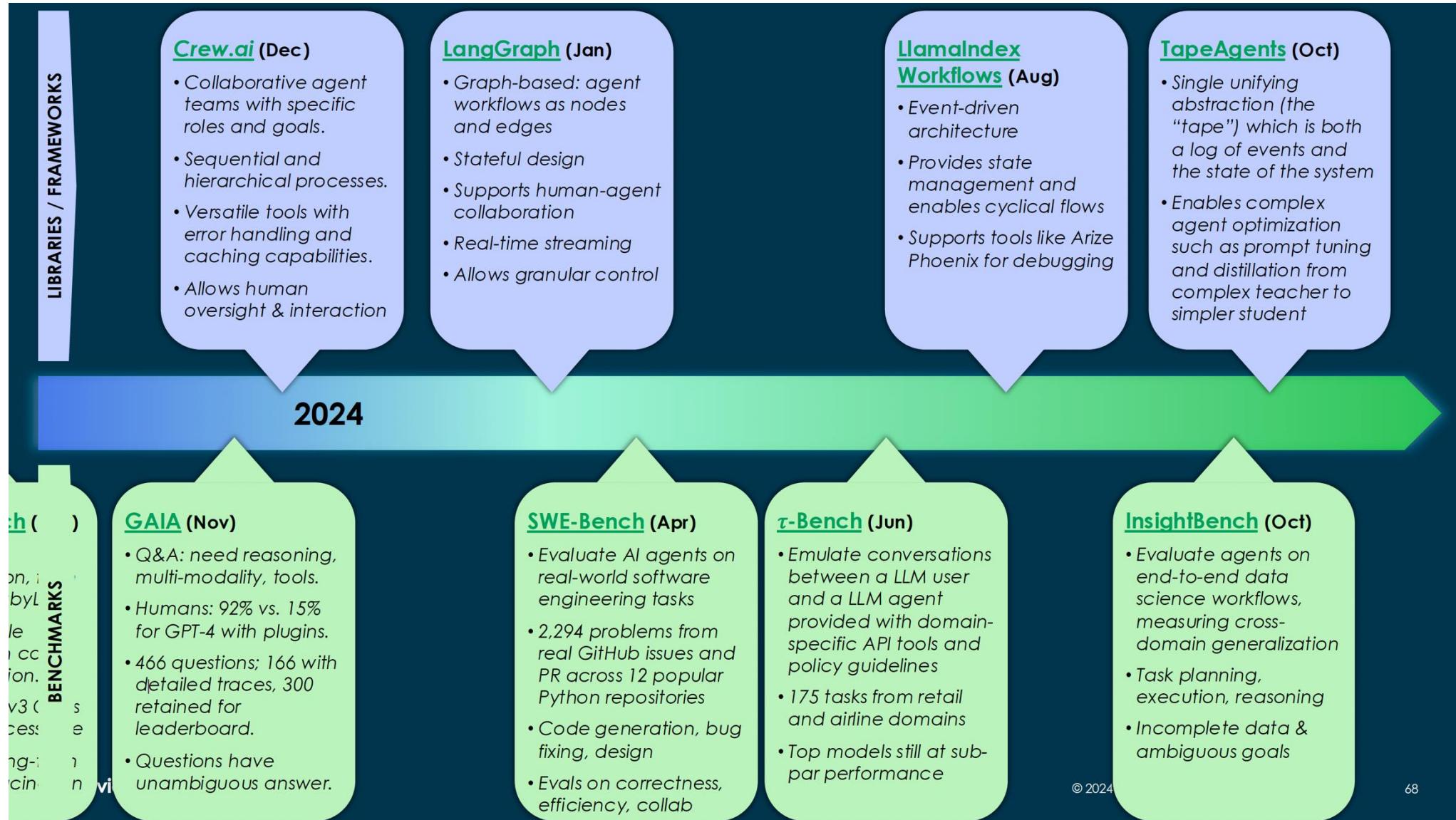
AI Agent platforms

Nicolas Chapados, "AI Agents for Enterprise Workflows", <https://llmagents-learning.org/f24>



AI Agent platforms

Nicolas Chapados, "AI Agents for Enterprise Workflows", <https://llmagents-learning.org/f24>



LangChain

Retrieval Augmented Generation Agents
Storage & Indexing Extraction

Build your app with LangChain

Build context-aware, reasoning applications with LangChain's flexible framework that leverages your company's data and APIs. Future-proof your application by making vendor optionality part of your LLM infrastructure design.

[Learn more about LangChain ↗](#)

Run at scale with LangGraph Platform

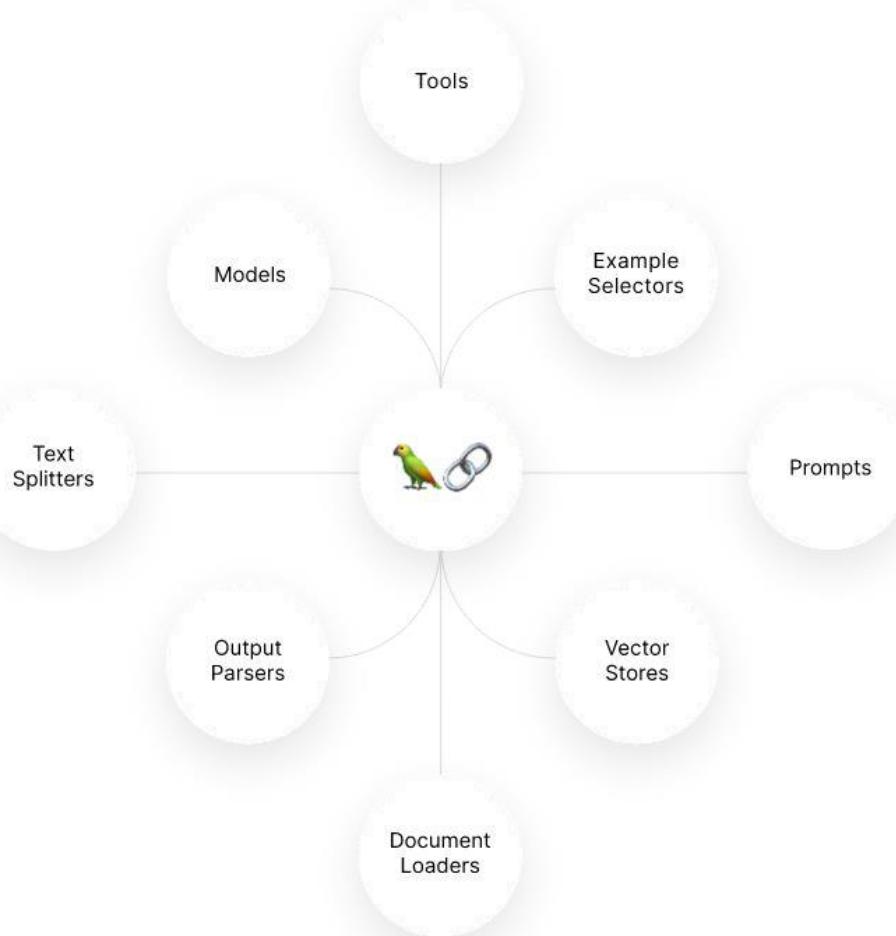
Use LangGraph Platform's APIs to design agent-driven user experiences featuring human-in-the-loop, multi-agent collaboration, conversation history, long-term memory, and time-travel. Deploy with fault-tolerant scalability.

[↗ Learn more about LangGraph Platform](#)

Manage LLM performance with LangSmith

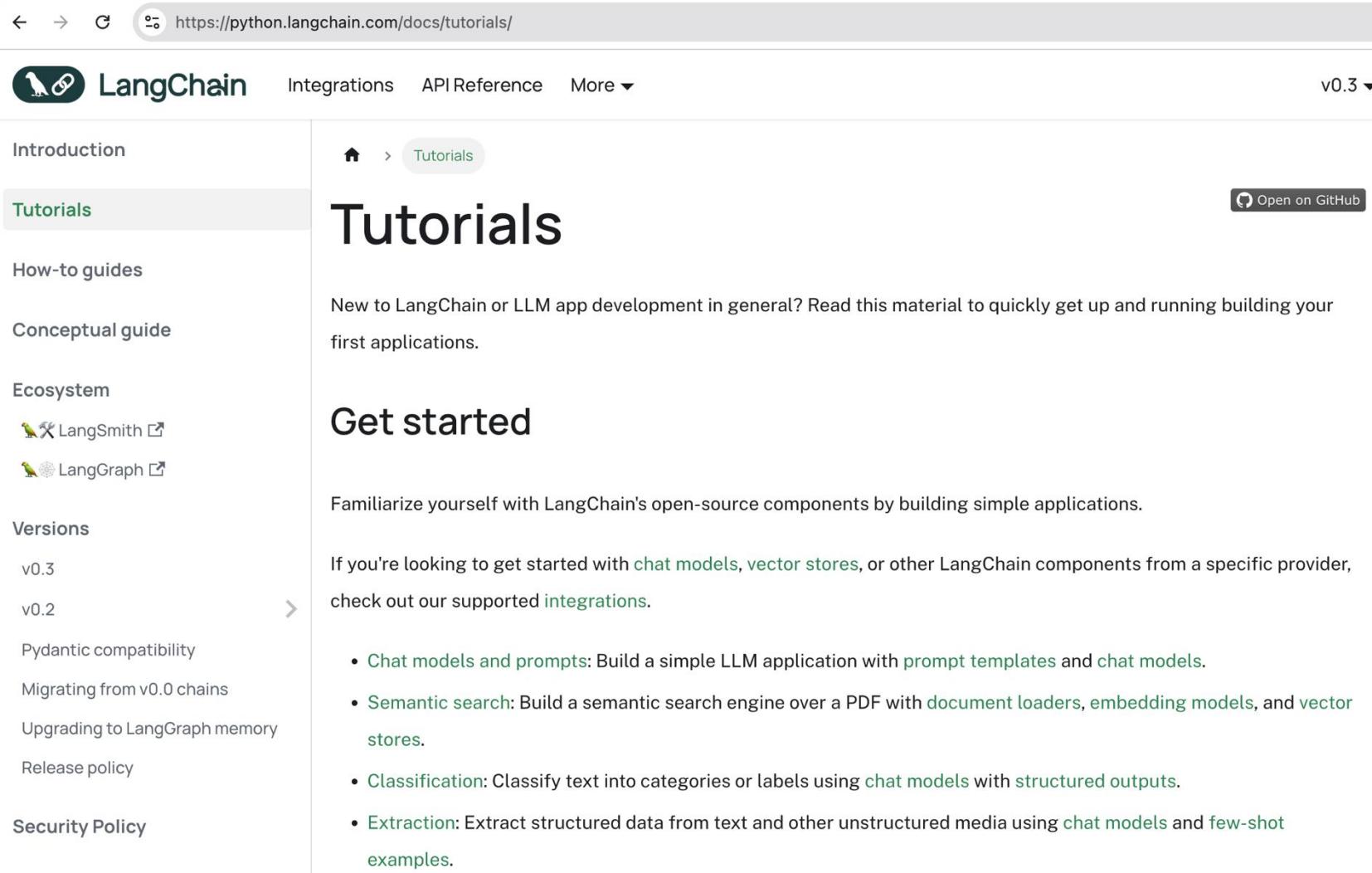
Ship faster with LangSmith's debug, test, deploy, and monitoring workflows. Don't rely on "vibes" – add engineering rigor to your LLM-development workflow, whether you're building with LangChain or not.

[Learn more about LangSmith ↗](#)



LangChain Tutorials

- <https://python.langchain.com/docs/tutorials/>



The screenshot shows the 'Tutorials' section of the LangChain documentation. The left sidebar has a 'Tutorials' tab selected. The main content area features a large title 'Tutorials' and a 'Get started' section with a list of five bullet points. A 'GitHub' button is visible in the top right.

https://python.langchain.com/docs/tutorials/

LangChain Integrations API Reference More ▾ v0.3 ▾

Introduction Tutorials How-to guides Conceptual guide Ecosystem Versions Pydantic compatibility Migrating from v0.0 chains Upgrading to LangGraph memory Release policy Security Policy

Open on GitHub

Tutorials

New to LangChain or LLM app development in general? Read this material to quickly get up and running building your first applications.

Get started

Familiarize yourself with LangChain's open-source components by building simple applications.

If you're looking to get started with [chat models](#), [vector stores](#), or other LangChain components from a specific provider, check out our supported [integrations](#).

- [Chat models and prompts](#): Build a simple LLM application with [prompt templates](#) and [chat models](#).
- [Semantic search](#): Build a semantic search engine over a PDF with [document loaders](#), [embedding models](#), and [vector stores](#).
- [Classification](#): Classify text into categories or labels using [chat models](#) with [structured outputs](#).
- [Extraction](#): Extract structured data from text and other unstructured media using [chat models](#) and [few-shot examples](#).

LangChain Tutorials

- <https://wikidocs.net/book/14314>
- <https://github.com/teddylee777/langchain-kr>

The screenshot shows a web browser window with the URL <https://wikidocs.net/book/14314> in the address bar. The page content is a tutorial titled '**<랭체인LangChain 노트> - LangChain 한국어 튜토리얼**'. The page includes a sidebar with navigation links for chapters CH01 and CH02, and a main content area with sections on setting up LangChain, using OpenAI API, and more. A footer at the bottom lists various resources and contact information.

검색어를 입력하세요. [Search icon] [More icon]

■ <랭체인LangChain 노트> - LangChain 한국어 튜토리얼

<랭체인LangChain 노트> - LangChain 한국어 튜토리얼

CH01 LangChain 시작하기

- 01. 설치 영상보고 따라하기
- 02. OpenAI API 키 발급 및 테스트
- 03. LangSmith 추적 설정
- 04. OpenAI API 사용(GPT-4o 멀티모달)
- 05. LangChain Expression Language(LCEL)
- 06. LCEL 인터페이스
- 07. Runnable

CH02 프롬프트(Prompt)

- 01. 프롬프트(Prompt)
- 02. 퓨샷 프롬프트(FewShotPromptTemplate)
- 03. LangChain Hub
- 04. 개인화된 프롬프트(Hub에 업로드)

■ <랭체인LangChain 노트> - LangChain 한국어 튜토리얼

지은이 : 테디노트
최종 편집일시 : 2025년 3월 18일 4:09 오후
저작권 : (CC) BY-NC-ND

2,771 명이 추천

추천은 공유할 수 있는 무료 전자책을 집필하는데 정말 큰 힘이 됩니다. "추천" 한 번씩만 부탁 드리겠습니다🙏🙏

랭체인 한국어 튜토리얼 강의
패스트캠퍼스 - RAG 비법노트

랭체인 한국어 튜토리얼 코드저장소(GitHub)
<https://github.com/teddylee777/langchain-kr>

유튜브 "테디노트"
<https://www.youtube.com/c/@teddynote>

데이터 분석 블로그 <https://teddylee777.github.io>

문의 teddylee777@gmail.com

LangChain Tutorials

- <https://wikidocs.net/book/14314>
- <https://github.com/pinecone-io/examples>

https://www.pinecone.io/learn/series/langchain/langchain-intro/



The screenshot shows the Pinecone website's navigation bar with links for Product, Docs, Customers, Resources, Pricing, Contact, Log in, and Sign up. Below the navigation, there is a section titled "LangChain AI Handbook" featuring a blue background with a chain graphic and three colored shapes (blue square, yellow diamond, white circle). The main content area discusses the history and growth of Large Language Models (LLMs) since GPT-3's release in 2020, mentioning significant milestones like LaMDA, BLOOM, and ChatGPT. It also highlights the emergence of LangChain in late 2022.

Pinecone

Product Docs Customers Resources Pricing Contact Log in Sign up

LangChain AI Handbook

Large Language Models (LLMs) entered the world stage with the release of OpenAI's GPT-3 in 2020 [1]. Since then, they've enjoyed a steady growth in popularity.

That is until late 2022. Interest in LLMs and the broader discipline of generative AI has skyrocketed. The reasons for this are likely the continuous upward momentum of significant advances in LLMs.

We saw the dramatic news about Google's "sentient" LaMDA chatbot. The first high-performance and open-source LLM called BLOOM was released. OpenAI released their next-generation text embedding model and the next generation of "GPT-3.5" models.

After all these giant leaps forward in the LLM space, OpenAI released ChatGPT — thrusting LLMs into the spotlight.

LangChain appeared around the same time. Its creator, Harrison Chase, made the first commit in late October 2022. Leaving a short couple of months of development before getting caught in the LLM wave.

Despite being early days for the library, it is already packed full of incredible features for building amazing tools around the core of LLMs. In this article, we'll introduce the library and start with the most straightforward component offered by LangChain — LLMs.

LangChain AI Handbook

Chapters

1. An Introduction to LangChain

Streamlit

- <https://docs.streamlit.io/develop/tutorials>
- <https://github.com/teddylee777/streamlit-tutorial>

Streamlit 튜토리얼 😊

Streamlit 을 활용하여 파이썬 코드로 빠르게 웹앱(WebApp)을 생성할 수 있습니다. 사용법도 굉장히 쉽고, 무료 호스팅 기능도 지원해주기 때문에, 데모 앱을 만들어 볼 때 굉장히 유용합니다.

YouTube 튜토리얼 영상

- [EP01. Streamlit으로 빠르게 웹앱 생성하기 - 설치, 환경설정, 실습파일](#)
- [EP02. 텍스트 컴포넌트](#)
- [EP03. 데이터프레임\(DataFrame\), 테이블 출력](#)
- [EP04. 자주 사용하는 위젯\(Widgets\)들](#)
- [EP05. 로또 생성기 웹사이트 만들기](#)
- [EP06. 그래프, 차트 그리기](#)
- [EP07. 파일 업로드 기능 & 업로드된 파일 처리](#)
- [EP08. 주식 데이터 조회, 주가 차트 그리기](#)
- [EP09. 사이드바, 탭 추가하기](#)
- [EP10. MBTI 대백과사전 웹앱 만들기 \(NAVER Clova API 활용\)](#)
- [EP11. Naver Clova Studio API를 활용한 챗봇 서비스 만들기](#)
- [EP12. 기업의 평균 월급여, 연봉 추정하기 \(국민연금 데이터 활용\)](#)
- [EP13. 기업의 월급여, 연봉 추정 서비스 제작 & 웹으로 배포 \(무료 호스팅\)](#)

And more Data

- AIHub

<https://www.aihub.or.kr/>

The screenshot shows the homepage of AIHub. At the top, there are several circular icons representing different data domains: 한국어 (Korean), 영상이미지 (Video Image), 헬스케어 (Healthcare), 교통물류 (Transportation Logistics), 재난안전환경 (Disaster Safety Environment), and 농축수산 (Agriculture, Animal Husbandry, and Fisheries). Below these are sections for '추천 검색어' (Recommended Search Terms) and a search bar. On the left, there's a '로그인' (Login) section with fields for '아이디' (ID) and '비밀번호' (Password), and buttons for '로그인' (Login), '네이버 로그인' (Naver Login), and '카카오 로그인' (Kakao Login). On the right, there's a '추천 데이터' (Recommended Data) section featuring items like '[재난안전환경] 기반암 시추 시료를 이용한 암반 등급 분류 데이터' (Data classification of rock grade using samples from disaster prevention site investigation) and '[영상이미지] 물체 조작 손 동작 3D 데이터' (3D data of object manipulation hand movement).

This screenshot shows the 'AI 데이터 찾기' (AI Data Search) page. At the top, there are three filter sections: '분야별 보기' (Category Filter) with '한국어' checked; '데이터 유형 보기' (Data Type Filter) with '텍스트' checked; and '구축년도별 보기' (Year Filter) with '2022년', '2023년', and '2024년' checked. Below these filters is a search bar with placeholder text '검색어를 입력해주세요' (Please enter a keyword). The main area displays a grid of data sets, each with a thumbnail, title, download button, and file size. The first few items include '[한국어] 국회 회의록 기반 지식 검색 데이터' (Knowledge search data based on National Assembly proceedings), '[한국어] 법률안 검토 보고서 요약 데이터' (Summary data of legislative bill review reports), '[한국어] K-Culture 관광 콘텐츠 특화 일본어 말뭉치 데이터' (Japanese corpus for specialized K-Culture tourism content), and '[한국어] 국가기록물 대상 초거대 AI 학습을 위한 말뭉치 ...' (Corpus for large-scale AI learning for national records).

And more Data

- 국립국어원 <https://www.korean.go.kr/>

<https://kli.korean.go.kr/corpus/main/requestMain.do>

이 누리집은 대한민국 공식 전자정부 누리집입니다.

언어정보나눔터 모두의 말뭉치 인공지능(AI)말평 은용어

말뭉치 소개 말뭉치 신청 말뭉치 활용 알립니다 오픈 에이피아이 들어가기 회원 가입 KR

모두의 말뭉치 미래를 준비하는 소중한 우리말 자원

말뭉치 분류 배포 연도 검색어 입력하세요.

제목을 입력하세요.

선택 말뭉치 담기

신규

지역어 말뭉치 2021 일상 대화 음성 말뭉치 2023 일상 대화 말뭉치 2023 글쓰기 원시 자료 말뭉치 2023 글쓰기 채점 자료 말뭉치 2023 한국어-인도네시아어 병렬 말뭉치 2023

한국어-힌디어 병렬 말뭉치 2023 한국어-캄보디아 크메르어 병렬 말뭉치 2023 한국어-러시아어 병렬 말뭉치 2023 한국어-태국어 병렬 말뭉치 2023 한국어-필리핀 타갈로그어 병렬 말뭉치 2023 한국어-우즈베크어 병렬 말뭉치 2023

한국어-베트남어 병렬 말뭉치 2023 한국수어 원시 말뭉치 2024 한국수어-한국어 병렬 말뭉치 2024 한국수어 주석 말뭉치 2024 국회 회의록 요약 말뭉치 2023 목자-점자 병렬 말뭉치 2023

신청하기 신청하기

<https://kli.korean.go.kr/benchmark/home.do>

이 누리집은 대한민국 공식 전자정부 누리집입니다.

언어정보나눔터 모두의 말뭉치 인공지능(AI)말평 은용어

과제 참여마당 알립니다

제목을 입력하세요.

AI 말평 인공지능의 한국어 능력 종합 평가

6팀 참가 중
감정 분석

부적절 발언 탐지

4팀 참가 중
연결 문장 추론

1팀 참가 중
이야기 완성

상시 평가 국립국어원 주최

상시 평가 국립국어원 주최

상시 평가 국립국어원 주최

상시 평가 국립국어원 주최

And more Data



Hugging Face

- Hugging Face Datasets <https://huggingface.co/docs/datasets/index>

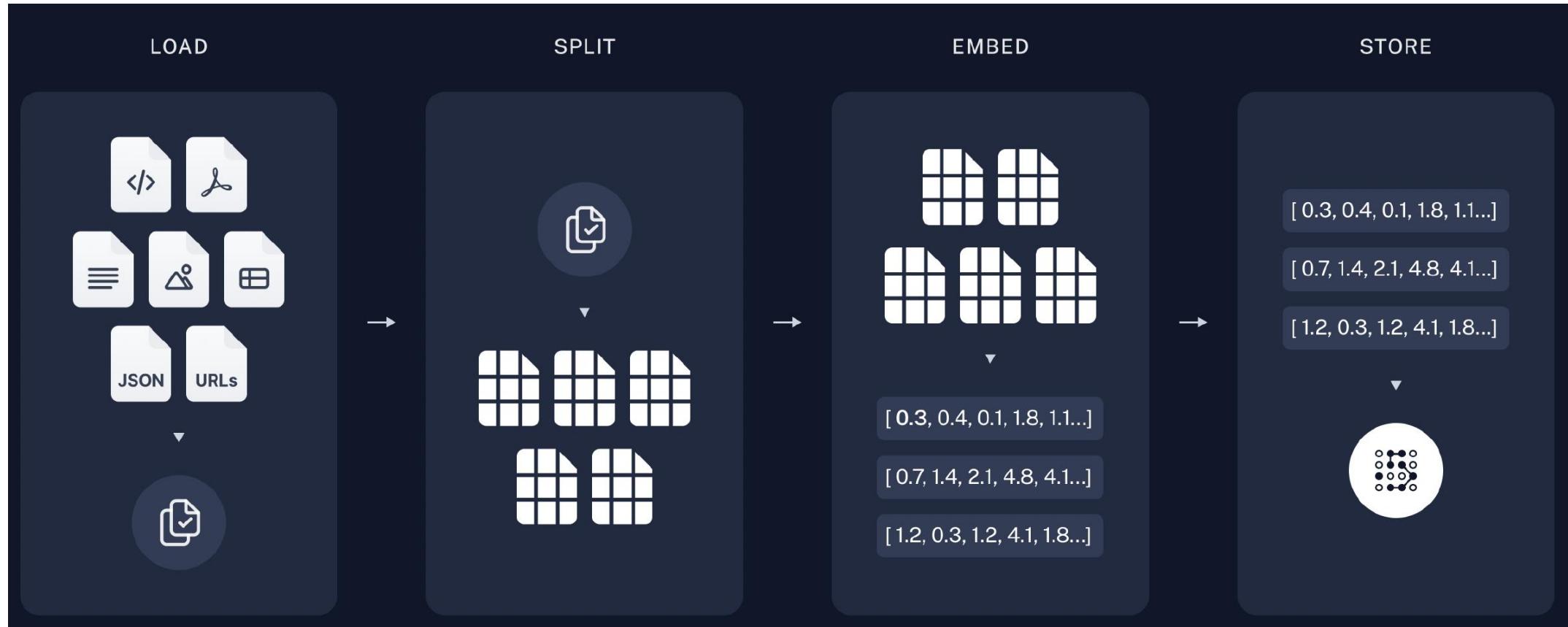
The screenshot shows the Hugging Face Datasets homepage. On the left, there are filters for 'Main' tasks like Tasks, Libraries, Languages, Licenses, Other, and modalities like 3D, Audio, Geospatial, Image, Tabular, Text, Time-series, and Video. There's also a slider for size (rows) from <1K to >1T and a section for file formats: json, csv, parquet, imagefolder, soundfolder, webdataset, text, and arrow. The main area displays a grid of dataset cards. Each card includes the dataset name, a small icon, a 'Viewer' link, the last update time, file size, number of views, and likes. The cards shown are:

- nvidia/Llama-Nemotron-Post-Training-Dataset-v1
- FreedomIntelligence/medical-o1-reasoning-SFT
- open-r1/codeforces-cots
- Congliu/Chinese-DeepSeek-R1-Distill-data-110k
- facebook/natural_reasoning
- nvidia/PhysicalAI-Robotics-GR00T-X-Embodiment-Sim
- glaiveai/reasoning-v1-20m
- open-r1/OpenR1-Math-220k
- Conard/fortune-telling
- BytedTsinghua-SIA/DAPO-Math-17k
- DropletX/DropletVideo-10M
- SmallDoge/SmallThoughts

And more Data & Pre-processing

- Pre-processing : 데이터 전처리

[Load] 문서를 검색에 적절한 형식으로 변환한 후, [Split] 검색의 Chunk 단위로 분할하여
[Embed] Passage Vector 로 변환한 후, [Store] 찾기 쉬운 구조로 Vector Store에 저장



And more Pre-processing > Document Loader

- 문서를 LLM이 잘 읽을 수 있는 파일 형식으로 변환하되 정보는 그대로 유지 필요
 - 실제 많이 사용되는 문서 형식 : Pdf, Excel, Word, pptx, hwpx
 - LLM이 편하게 읽는 파일 형식 : Text, Markdown, JSON
- 고려할 문제
 - 변환의 정확도 & 속도
 - 한글 인코딩, 특수 문자
 - 도표, 차트, 이미지의 내용 전달, 문서에서의 위치 정보
 - 글자 외 정보(메타정보) : 문서제목, 작성자, 단락구분, 페이지 번호, 각주/미주
- Utilities
 - PDFPlumber, UnstructuredPDFLoader, PyPDFLoader, fitz

And more Pre-processing > Text splitter

- 검색과 LLM의 입력으로 적절한 크기의 Chunk로 문서를 분할해야 합니다
 - 하나의 단위로 취급될 내용이 여러 Chunk로 분리되어서는 안됩니다.
- CharacterTextSplitter
- RecursiveCharacterTextSplitter
- TokenTextSplitter
- HuggingFace
- SemanticChunker(experimental)

And more Pre-processing > Embedding model

- Embedding 모델과 LLM 은 서로 목적이 다른 모델이고, 서로 다르게 학습되어 있습니다.
- Open source 모델
 - MTEB (Massive Text Embedding Benchmark) 리더보드 상위모델 선별하여 사용
 - 영문벤치마크 성능과 한국어에서의 성능은 별개
 - 로컬 구동 환경 구비 필요
- OpenAIEmbedding (+ CacheBackedEmbeddings)
 - OpenAI 유료과금, 비교적 낮은 비용이나 자주 사용되는 Embedding 특성 고려 필요
 - 한국어에서 검증된 높은 안정된 성능

And more Pre-processing > VectorStore

- VectorStore 는 각 회사별 고유의 indexing 알고리즘을 사용하므로, 검색 결과가 서로 다를 수 있음
- Pinecone
- Weaviate
- ElasticSearch
- Chroma
- FAISS

Agentic AI

1. Agentic AI?
2. RAG
3. AI Agent
4. AI Agent platforms
5. Use Cases