



Spring Camp 2024

# LLM에도 봄이 찾아온다

황민호 | Kakao



**robin.hwang**

카카오 기술전략

**WIPRO**  
(2009)

**KISTI**  
(2010)

**DAUM**  
(2013)

**KAKAO**  
(현재)

SQL BNF 문법 기반의 자동  
SQL 생성기를 이용한 DBMS 테스트

문장 패턴분석을 통한  
온톨로지 인스턴스 구축

Daum 클라우드 서비스   
Daum 검색 

Kakao Moment 광고 플랫폼   
Kakao OLIVE 오픈소스 플랫폼   
Tech Talk TF  
컨퍼런스 TF  
Tech Portal TF  
AI 교육 자문 TF  
기술전략 | 기술기획

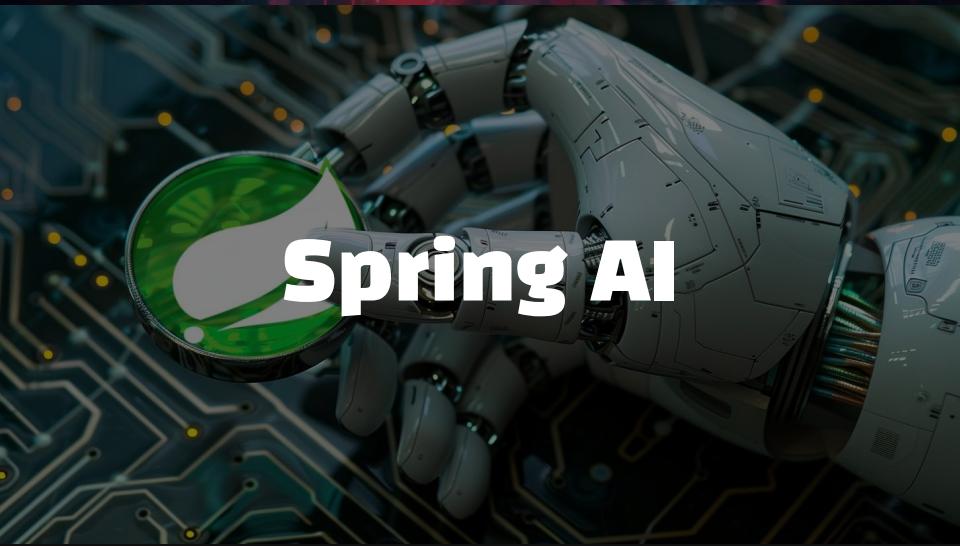




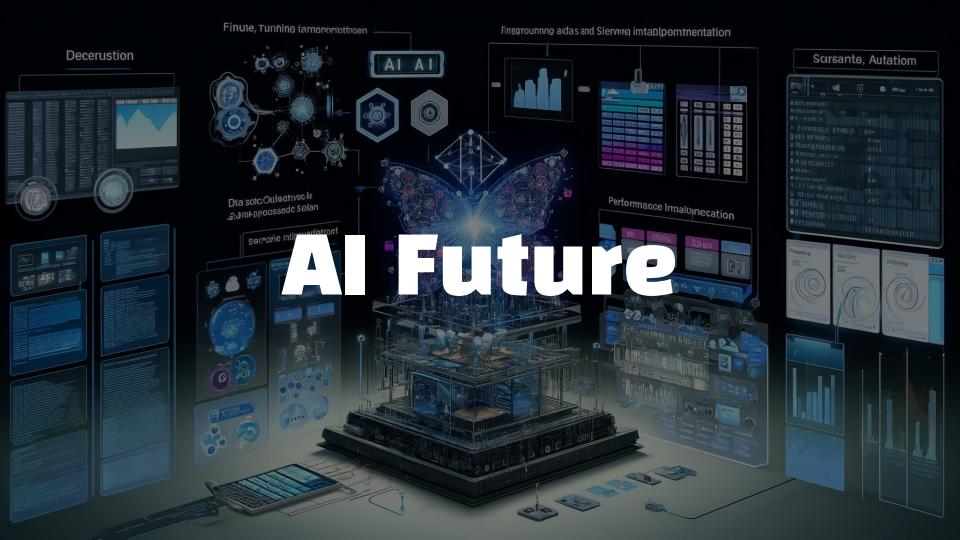
# Generative AI



# AI Development Concepts



# Spring AI



# AI Future



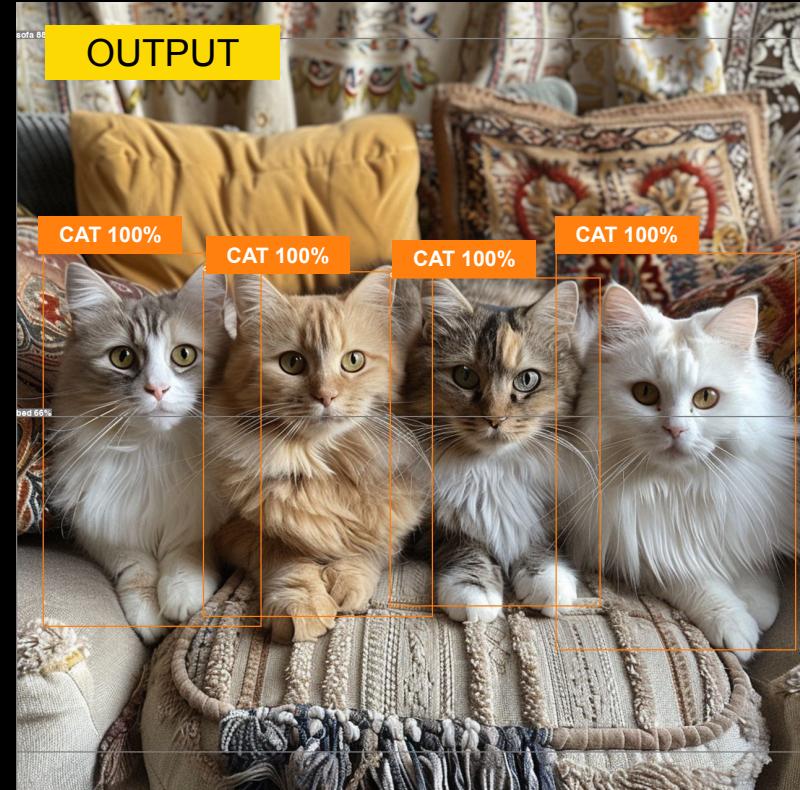
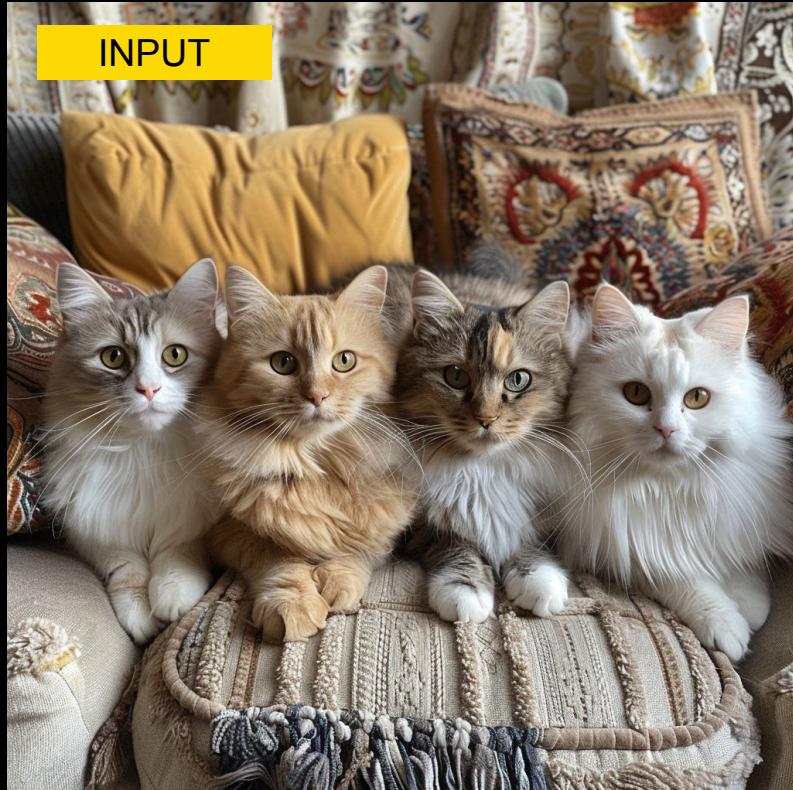
# Generative AI

# AI



# Generative AI

YOLO v3 객체 식별 - <https://v-iashin.github.io/detector>



# Generative AI

Midjourney v6

INPUT

TEXT

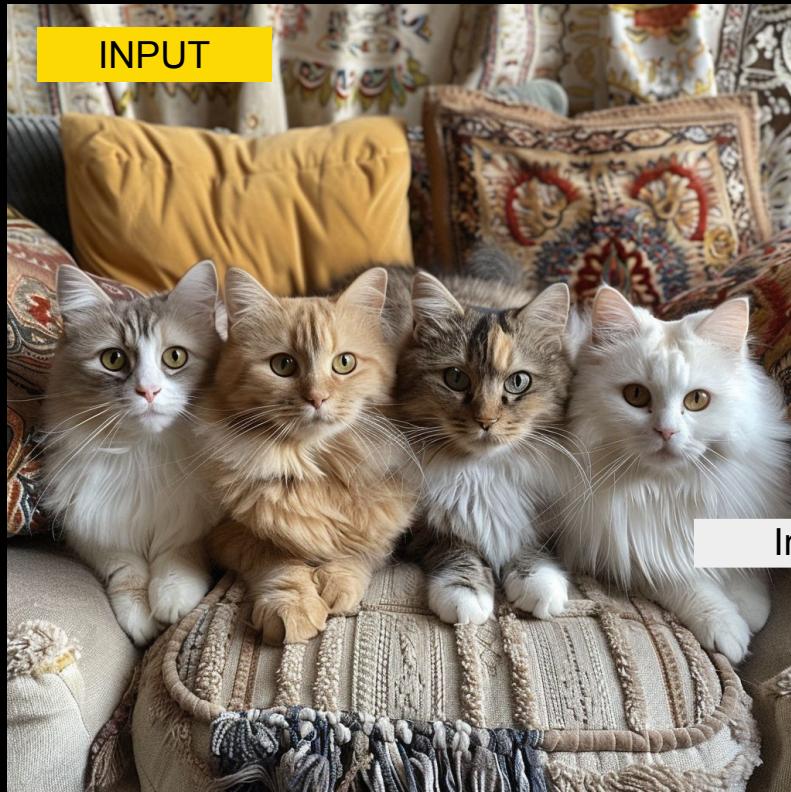
Cats, winter, anthropomorphic,  
fashionable --ar 16:9

OUTPUT



# Generative AI

Midjourney v6

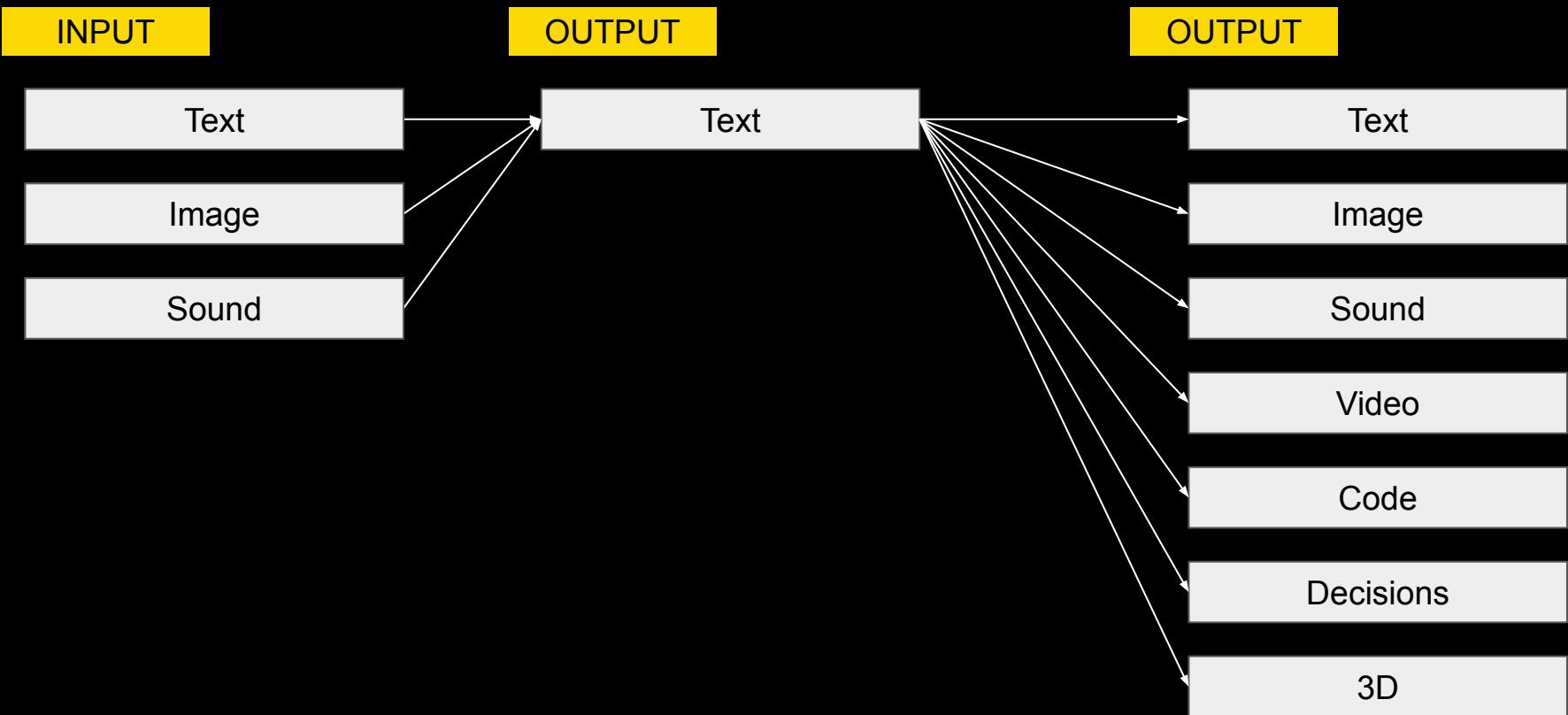


Inpainting



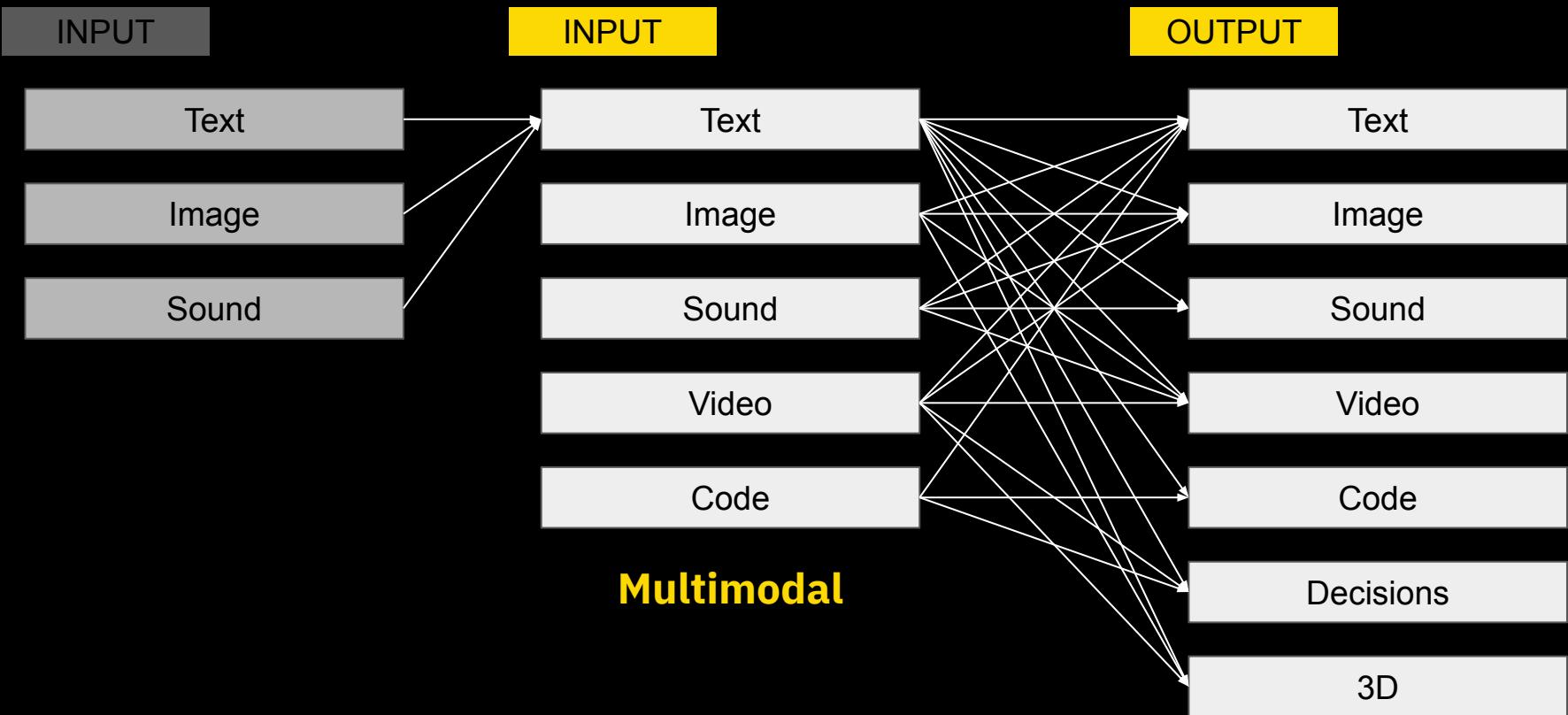
# Generative AI

Deep learning >>> Generative AI



# Generative AI

Generative AI with Multimodal



# 대표적인 LLM



## ChatGPT

- ChatGPT Plus, Team
- \$20/m, \$25/m
- GPT-3.5, GPT-4, GPT-4o
- 32,000 tokens
- GPTs, 이미지생성, Vision, 검색, 음성입출력, 고급데이터분석

### API 비용

- GPT-3.5 Turbo: \$0.5/1M, \$1.5/1M
- GPT-4 Turbo: \$10/1M, \$30/1M
- GPT-4o: \$5/1M, \$15/1M



## Gemini

- Gemini, Gemini Advanced
- ₩29,000/m
- Gemini Pro, Ultra
- 32,000 tokens
- Vision, 음성입력

### API 비용

- 1.0Pro(5/2): \$0.5/1M, \$1.5/1M
- 1.5Pro(Prev): \$7/1M, \$21/1M



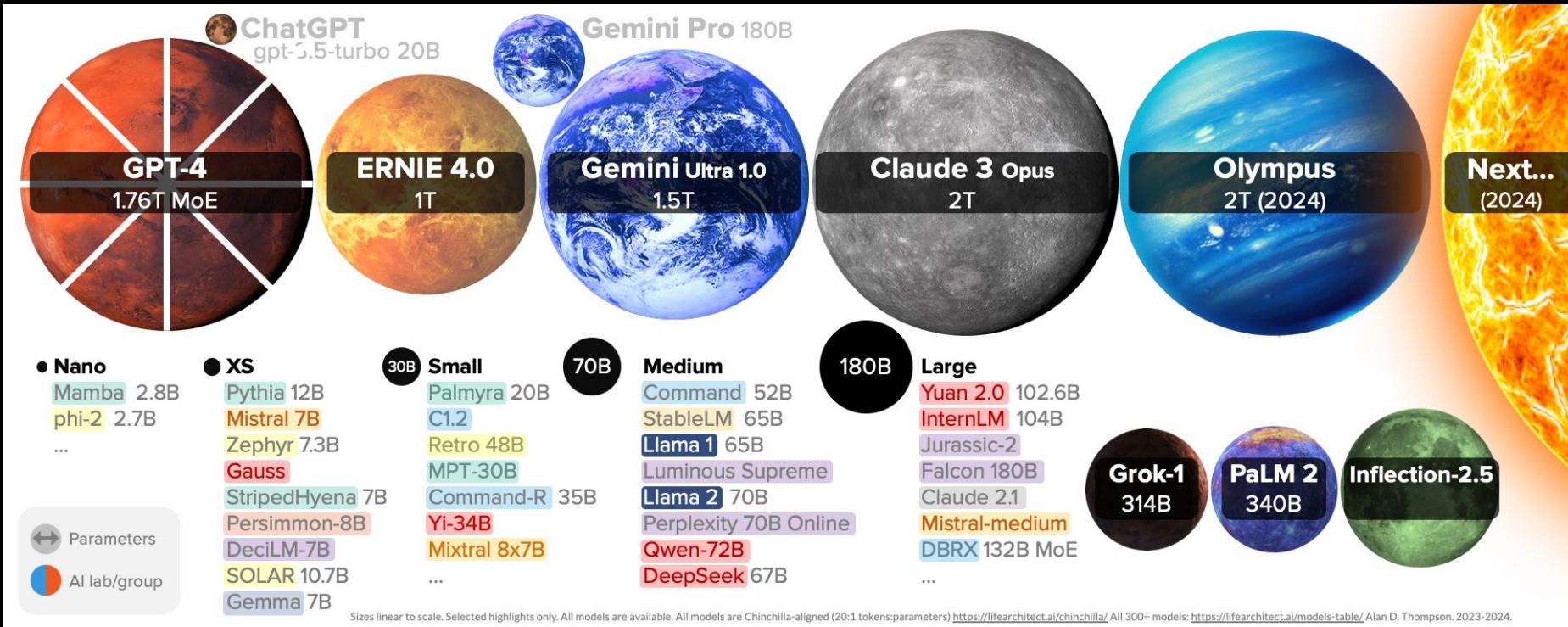
## Claude

- Claude, Claude Pro
- \$20/m
- Claude Haiku, Sonet, Opus
- 200,000 tokens
- Vision, 파일입력

### API 비용

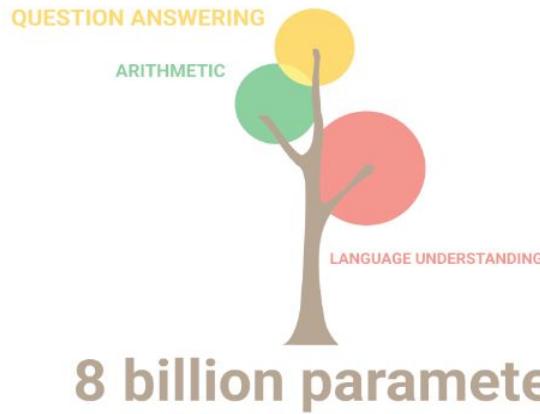
- Haiku: \$0.25/1M, \$1.25/1M
- Sonnet: \$3/1M, \$15/1M
- Opus: \$15/1M, \$75/1M

# 파라미터 크기별 LLM



<https://lifearchitect.ai/models/>

# 일반적으로 파라미터가 커질수록 성능이 높습니다



80억 : 간단한 질문 답변, 약간의 수리 능력

620억 : 문서요약, 번역, 상식적 추론

5,400억 : 코딩, 패턴파악, 대화, 농담 등등

GPT-3 : 1,750억개

GPT-4 : 2조개 (비공식)

# GPT-4 수준은 이해력과 추론 능력이 좋습니다

GPT-3.5 Turbo 20B

ChatGPT 3.5 ▾



You

오늘 사과가 5개 있습니다. 지난주에 사과를 3개 먹었습니다. 나는 지금 사과를 몇 개 가지고 있을까요?



ChatGPT

지금은 2개의 사과를 가지고 있습니다. 처음에 5개의 사과가 있었고, 그 중 3개를 먹었으므로  $5 - 3 = 2$ 개가 남게 됩니다.



GPT-4 1.76T

ChatGPT 4 ▾



You

오늘 사과가 5개 있습니다. 지난주에 사과를 3개 먹었습니다. 나는 지금 사과를 몇 개 가지고 있을까요?

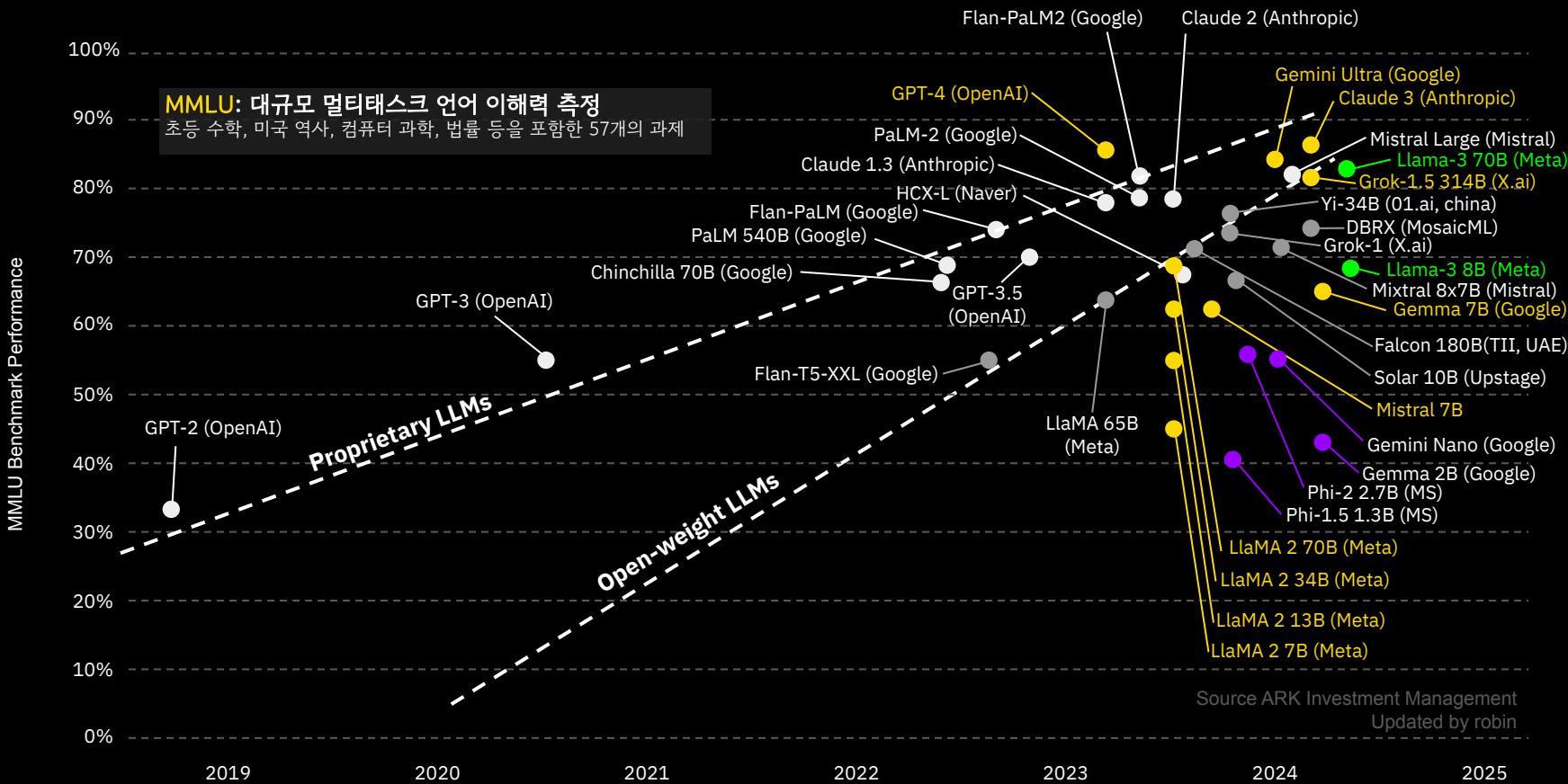
don't use advanced data analysis



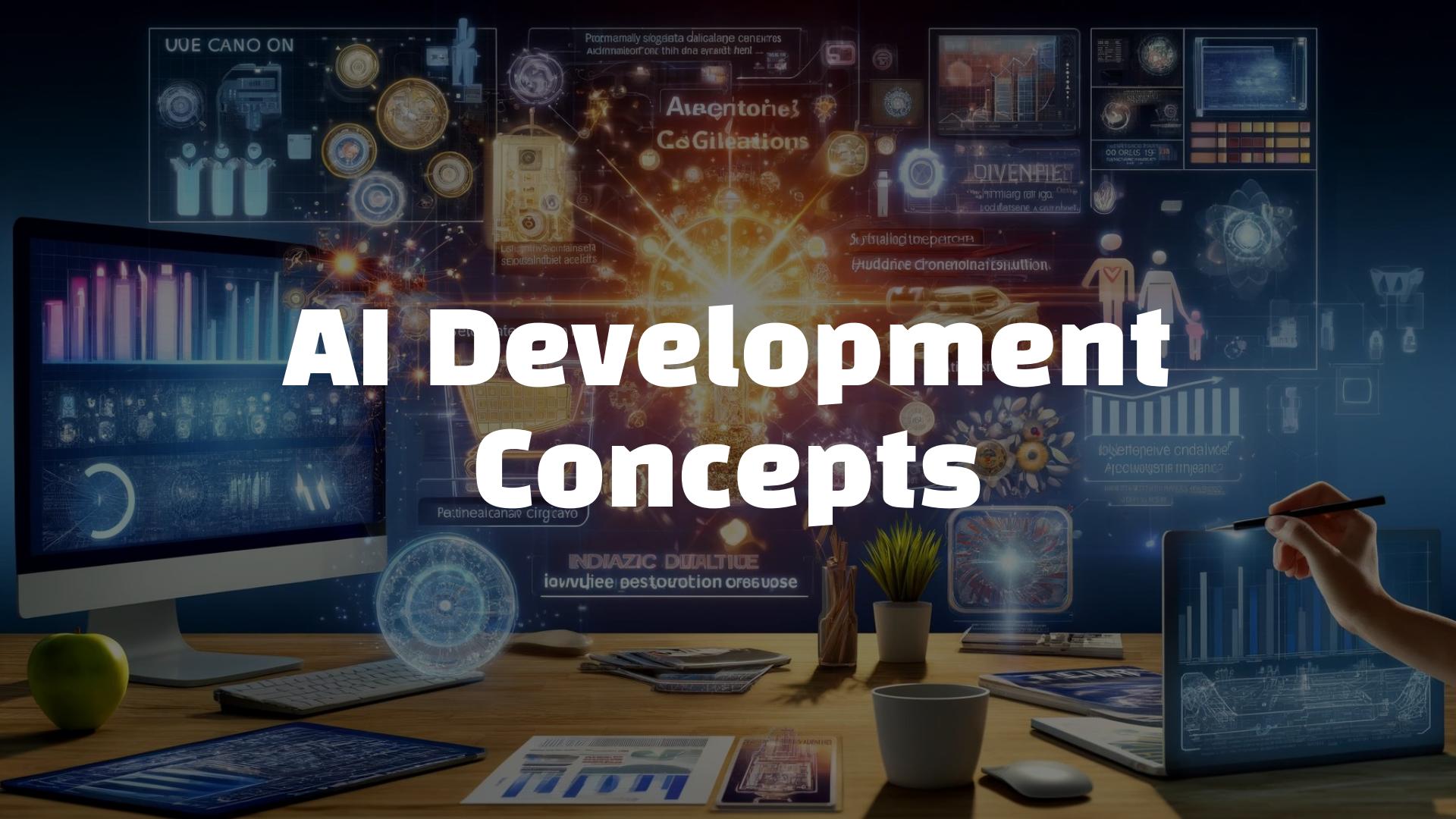
ChatGPT

오늘 사과가 5개 있고, 지난주에 3개를 먹었으므로, 현재 사과는 5개를 그대로 가지고 있습니다. 지난주에 먹은 사과는 이미 계산에 반영되어 있기 때문에, 현재 수량에 변동이 없습니다. 따라서 지금 5개의 사과를 가지고 있습니다.

# LLM 의 발전



# AI Development Concepts



# 생성형 AI 주요 개념

Models

Tokens

Function  
Calling

Prompt

Embeddings

Fine Tuning

Prompt  
Engineering

RAG

Evaluation

# AI Models

Models	Tokens	Function Calling
Prompt	Embeddings	Fine Tuning
Prompt Engineering	RAG	Evaluation

19

Type	Input	Output	Example
언어 모델	Text	Text	GPT-3.5, GPT-4, Gemini, Claude, Llama...
이미지 생성 모델	Text	Image	Dall-E, DeepAI..
음성 모델	Text or Sound	Sound or Text	TTS, Whisper..
임베딩 모델	Text	Vector	Embedding..
영상 생성 모델	Text	Video	Sora..
멀티모달	Multi-Modal (Text, Image, Video..)	Text ..	GPT-4V, GPT-4o, Gemini..

# AI Model Access

Models	Tokens	Function Calling
Prompt	Embeddings	Fine Tuning
Prompt Engineering	RAG	Evaluation

— API —

— Inference Engine —

— Local —



**ChatGPT**



**Gemini**



**Claude**

**TensorRT-LLM**

**LightLLM**

**DeepSpeed-MII**



**Ollama**



**LM Studio**



**LocalAI**

# AI 프롬프트

Models	Tokens	Function Calling
<b>Prompt</b>	Embeddings	Fine Tuning
Prompt Engineering	RAG	Evaluation

“고양이 이미지를 생성해줘”



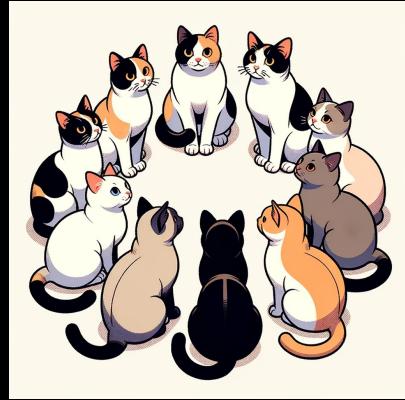
화분으로 둘러싸인 햇살 가득한 창턱에서 느긋하게 휴식을 취하고 있는 파란 눈이 인상적인 푸신한 흰색 고양이의 사진입니다.



꽃이 만발한 화려한 정원에서 펄럭이는 나비를 쫓는 장난기 가득한 얼룩 고양이의 일러스트입니다.



초승달 아래 돌담 위에 우아하게 앉아 있는 녹색 눈을 가진 매끈한 검은 고양이의 수채화입니다.



삼색, 삼, 치즈, 턱시도 고양이 등 다양한 고양이들이 원을 그리며 앉아 있는 벡터 이미지로, 모두 중앙을 주의 깊게 바라보고 있습니다.

# AI 프롬프트

Models	Tokens	Function Calling
<b>Prompt</b>	Embeddings	Fine Tuning
Prompt Engineering	RAG	Evaluation

“화분으로 둘러싸인 햇살 가득한 창턱에서 느긋하게 휴식을 취하고 있는 파란 눈이 인상적인 푸신한 흰색 고양이의 사진을 생성해줘”



화분으로  
둘러싸인

햇살 가득한  
창턱

느긋하게 휴식을  
취하고 있는

파란 눈이  
인상적인

푸신한

흰색 고양이

# AI 프롬프트

Models	Tokens	Function Calling
<b>Prompt</b>	Embeddings	Fine Tuning
Prompt Engineering	RAG	Evaluation

“나무로 둘러싸인 정원에서 느긋하게 휴식을 취하고 있는 파란 눈이 인상적인 푸신한 흰색 고양이의 사진을 생성해줘”



화분으로  
나무로  
둘러싸인

창턱  
정원

느긋하게 휴식을  
취하고 있는

파란 눈이  
인상적인

푸신한

흰색 고양이

# AI 프롬프트

Models	Tokens	Function Calling
<b>Prompt</b>	Embeddings	Fine Tuning
Prompt Engineering	RAG	Evaluation

“고양이 이미지를 생성해줘”



“나무로 둘러싸인 정원에서 느긋하게 휴식을 취하고 있는 파란 눈이 인상적인 푸신한 흰색 고양이의 사진을 생성해줘”



# AI 프롬프트

Models	Tokens	Function Calling
Prompt	Embeddings	Fine Tuning
Prompt Engineering	RAG	Evaluation

아이디어

“고양이 이미지를 생성해줘”

구체화

“화분으로 둘러싸인 햇살가득한 창턱에서 느긋하게 휴식을 취하고 있는 파란 눈이 인상적인 푸신한 흰색 고양이의 사진입니다.”

다양한 실험

“나무로 둘러싸인 정원에서 느긋하게 휴식을 취하고 있는 파란 눈이 인상적인 푸신한 흰색 고양이의 사진입니다.”

평가

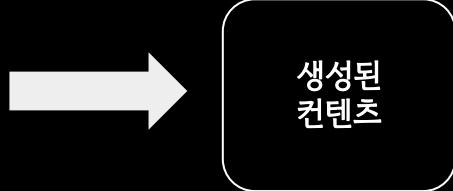
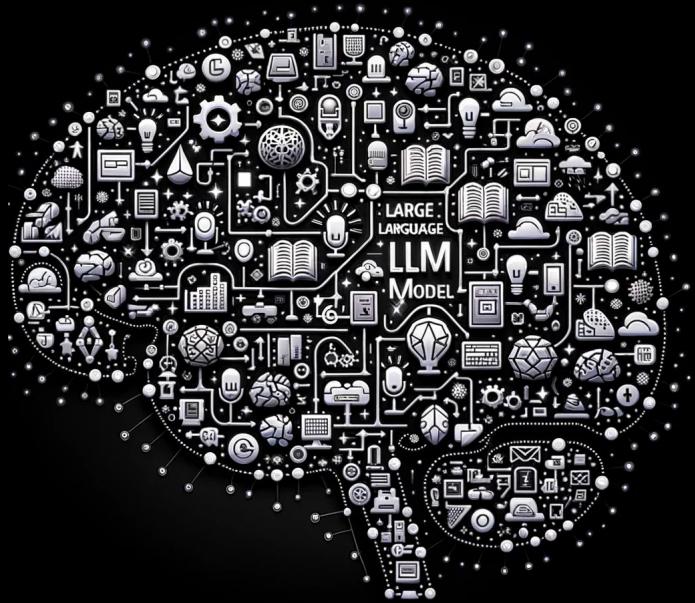
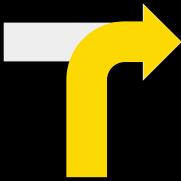
정확히 의도한대로 일관성 있게 재현이 되는가?

# 시스템 프롬프트

Models	Tokens	Function Calling
Prompt	Embeddings	Fine Tuning
<b>Prompt Engineering</b>	RAG	Evaluation

## Prompt Writing

“고양이 이미지를 생성해줘”



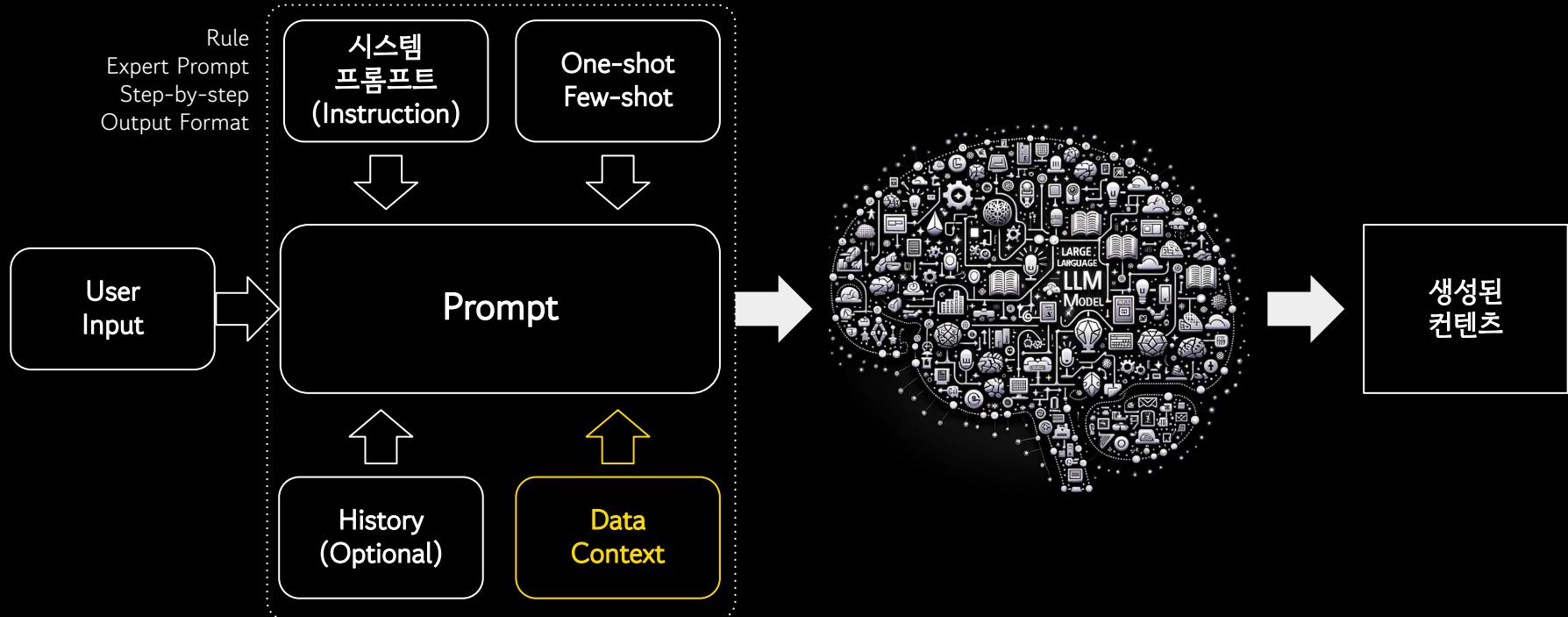
생성된  
컨텐츠

“만화 스타일로 생성해줘”

## Prompt Engineering

# AI 프롬프트 구성

Models	Tokens	Function Calling
Prompt	Embeddings	Fine Tuning
<b>Prompt Engineering</b>	RAG	Evaluation



응답 생성에 필요한 데이터를 함께 전달  
Hallucination 현상을 최소화

# 알려진 프롬프트 기법들

## 프롬프트 테크닉

- Zero-shot
- One-shot learning
- Few-shot learning
- Chain of Thought
- Zero-shot CoT
- Expert Prompting
- RAG (Retrieval Augmented Generation)
- Self-Consistency
- Tree-of-Thought
- Selection-Inference
- Least-to-Most
- ReAct Prompting
- Self Evaluation
- Plan-and-Solve Prompting
- Automatic Prompt Engineer
- Active-Prompt
- Generated Knowledge Prompting
- Faithful Reasoning
- Constitutional AI
- Graph Prompting
- Multimodal CoT
- Program-Aided Language Models
- ...

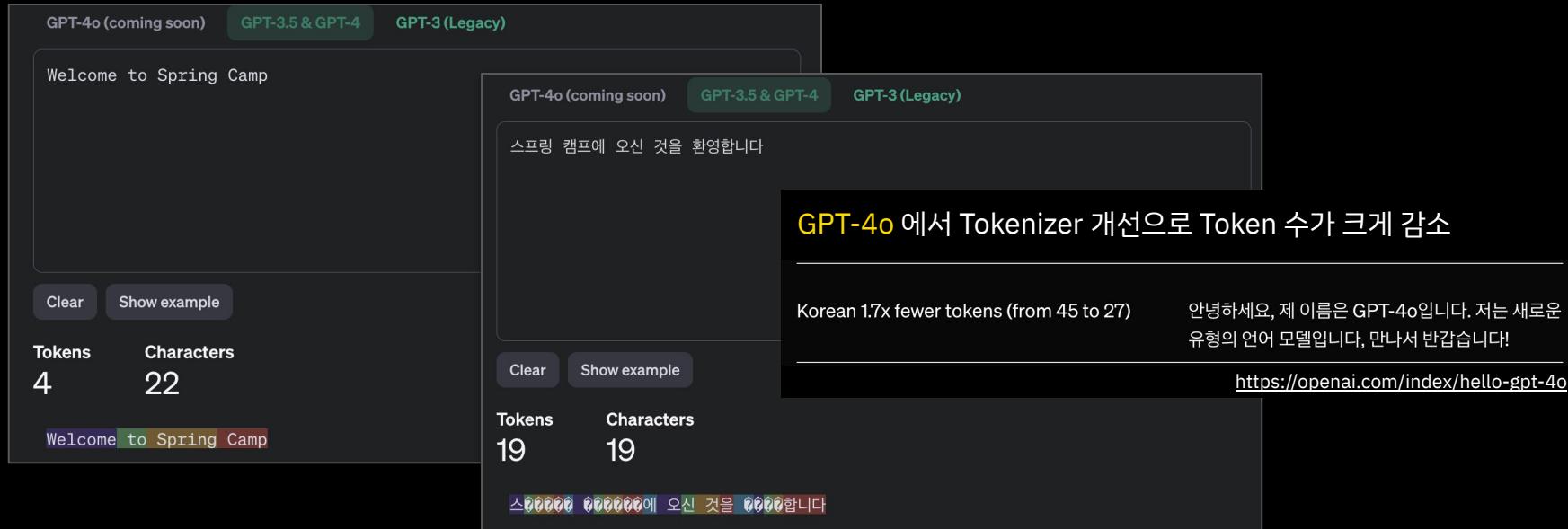
Models	Tokens	Function Calling
Prompt	Embeddings	Fine Tuning
<b>Prompt Engineering</b>	RAG	Evaluation



# Tokenizer

LLM 은 Token을 사용하여 텍스트를 처리  
 AI 모델이나 언어별에 따라 Token 수가 다름  
 LLM API 과금은 Token 수를 기준으로 산정 됨

Models	<b>Tokens</b>	Function Calling
Prompt	Embeddings	Fine Tuning
Prompt Templates	RAG	Evaluation



The screenshot shows two side-by-side tokenization results for the Korean sentence "스프링 캠프에 오신 것을 환영합니다".

**GPT-3.5 & GPT-4:** This section shows 45 tokens and 22 characters for the input text.

Tokens	Characters
4	22

**GPT-4o (coming soon):** This section shows 27 tokens and 19 characters for the same input text.

Tokens	Characters
19	19

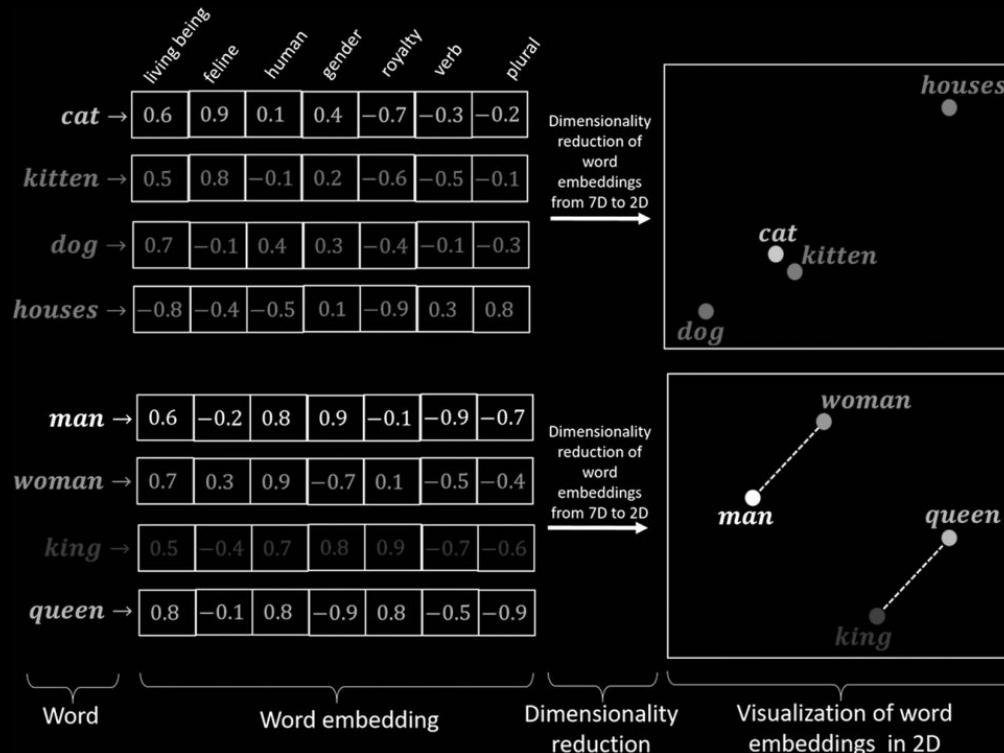
**Summary:** GPT-4o에서 Tokenizer 개선으로 Token 수가 크게 감소

Korean 1.7x fewer tokens (from 45 to 27)      안녕하세요, 제 이름은 GPT-4o입니다. 저는 새로운 유형의 언어 모델입니다, 만나서 반갑습니다!

<https://openai.com/index/hello-gpt-4o>

# Embedding

데이터의 패턴과 구조를 보존하면서 데이터를 벡터로 변환



Models	Tokens	Function Calling
Prompt	Embeddings	Fine Tuning
Prompt Templates	RAG	Evaluation

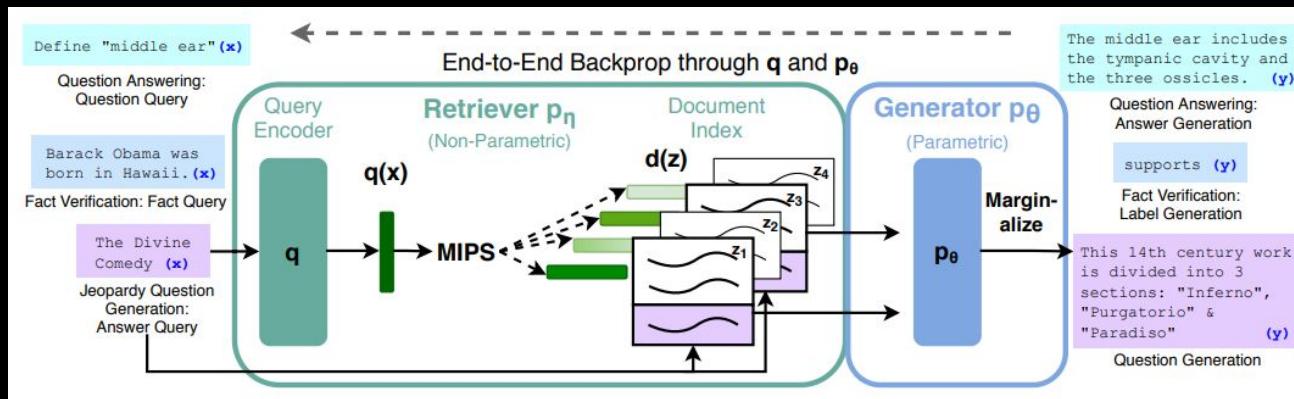
## Word Embedding 예시

- 각 단어를 7차원 벡터로 표현
- 7차원을 2차원으로 축소해서 시각화 해보면 'cat'과 'kitten'은 가깝고, 'dog'은 떨어져 있음
- man 과 woman, queen과 king 사이의 관계를 성별 차이를 반영하여 보여줌

# RAG (Retrieval Augmented Generation)

2020년 5월, Facebook AI 연구팀

Models	Tokens	Function Calling
Prompt	Embeddings	Fine Tuning
Prompt Templates	<b>RAG</b>	Evaluation



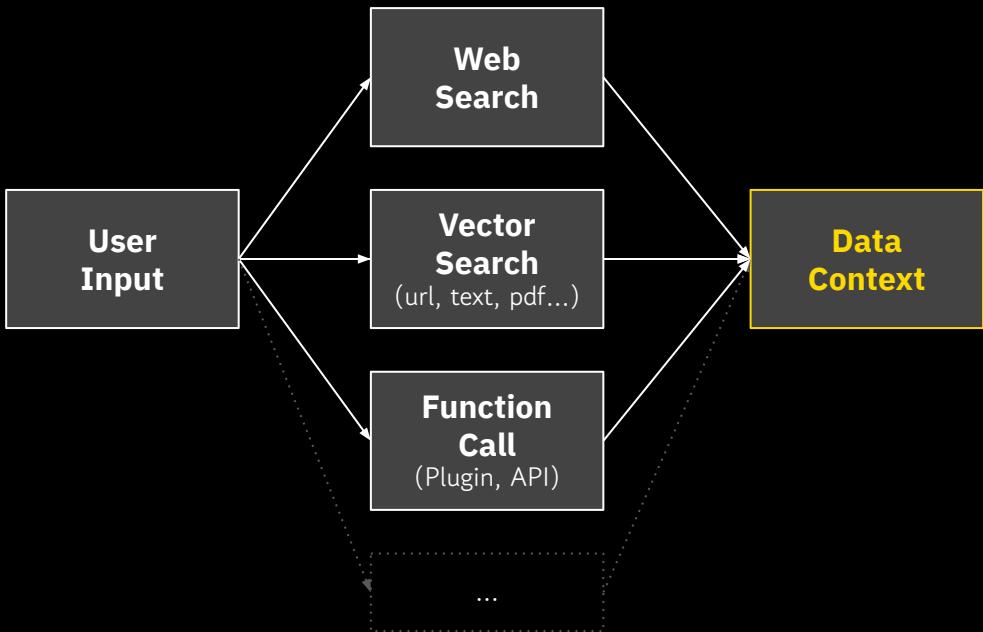
- LLM 과 함께 의미(Semantic) 기반 검색 방식인 Vector Search 도 크게 발전
- 주어진 질문에 관련된 지식을 검색해서 그 결과를 프롬프트에 컨텍스트로 제공하는 방식
- 현재 정확한 답변을 받기 위해 사용 가능한 확실한 방법
- Generative AI 서비스 개발 시 필수로 고려되는 구성

# RAG (Retrieval Augmented Generation)

Models	Tokens	Function Calling
Prompt	Embeddings	Fine Tuning
Prompt Templates	<b>RAG</b>	Evaluation



Open Book Test



# RAG (Retrieval Augmented Generation)

Models	Tokens	Function Calling
Prompt	Embeddings	Fine Tuning
Prompt Templates	<b>RAG</b>	Evaluation

ChatGPT 3.5

You  
강감찬 장군이 갤럭시 폰으로 주로 어떤 업무를 했나요?

ChatGPT  
강감찬 장군은 갤럭시 폰을 통해 다양한 업무를 수행했습니다. 예를 들어,

1. 전략적 의사 소통: 갤럭시 폰을 사용하여 부하 장군들이나 정부 당국과 전략적인 의사소통을 주고받았습니다. 이를 통해 전쟁이나 전략적인 결정에 필요한 정보를 신속하게 전달하고 응답할 수 있었습니다.
2. 전술적 정보 수집: 갤럭시 폰을 사용하여 정찰 정보나 적군의 움직임 등 전술적 정보를 수집하고 분석했습니다. 이를 통해 전투 계획을 세우고 적에 대비 할 수 있었습니다.
3. 부하 장군 지원: 갤럭시 폰을 통해 부하 장군들에게 지시를 내리고 지원을 제공했습니다. 이를 통해 전투 상황을 실시간으로 파악하고 효율적으로 지휘할 수 있었습니다.

ChatGPT 4

You  
강감찬 장군이 갤럭시 폰으로 주로 어떤 업무를 했나요?

ChatGPT  
강감찬 장군은 고려 시대의 인물로, 현대의 갤럭시 폰과 같은 기술을 사용할 수 없었습니다. 강감찬 장군은 10세기 말에서 11세기 초에 걸쳐 활동한 군사 지도자로, 주로 전략적인 군사 작전과 지휘를 통해 나라를 방어하는 업무를 수행했습니다. 그의 가장 유명한 전투는 1019년 거란의 침략을 성공적으로 막아낸 귀주대첩입니다.

현대 기술을 사용했다는 설정은 흥미로운 상상이지만, 실제 역사적 상황과는 거리가 있습니다. 현대 기술을 활용한 역사적 인물의 가상 시나리오를 탐구하는 것은 창의적인 작업이나 소설, 게임, 영화 등에서 흔히 볼 수 있는 주제입니다.

# 추가 정보를 주면 (RAG) 정확도가 올라갑니다

Models	Tokens	Function Calling
Prompt	Embeddings	Fine Tuning
Prompt Templates	RAG	Evaluation

You +강감찬 장군 + 갤럭시 폰 정보

강감찬 장군이 갤럭시 폰으로 주로 어떤 업무를 했나요?

ChatGPT 강감찬 장군은 갤럭시 폰을 통해 다양한 업무를 수행했습니다. 예를 들어,

1. 전략적 의사 소통: 갤럭시 폰을 사용하여 부하 장군들이나 정부 당국과 전략적인 의사소통을 주고받았습니다. 이를 통해 전쟁이나 전략적인 결정에 필요한 정보를 신속하게 전달하고 응답할 수 있었습니다.
2. 전술적 정보 수집: 갤럭시 폰을 사용하여 정찰 정보나 적군의 움직임 등 전술적 정보를 수집하고 분석했습니다. 이를 통해 전투 계획을 세우고 적에 대비 할 수 있었습니다.
3. 부하 장군 지원: 갤럭시 폰을 통해 부하 장군들에게 지시를 내리고 지원을 제공했습니다. 이를 통해 전투 상황을 실시간으로 파악하고 효율적으로 지휘할 수 있었습니다.

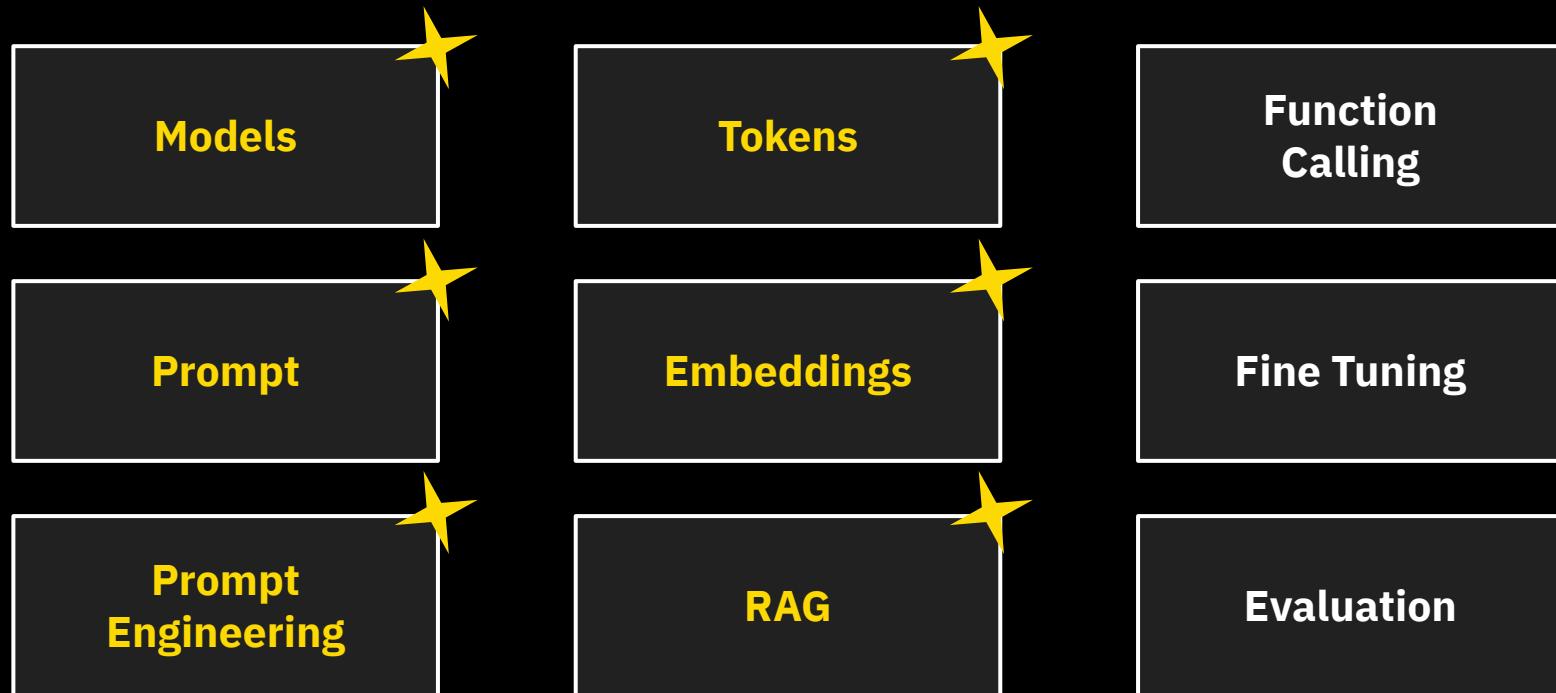
You +강감찬 장군 + 갤럭시 폰 정보 (참고)

ICL

강감찬 장군이 갤럭시 폰으로 주로 어떤 업무를 했나요?  
 - 강감찬 장군은 고려 시대 인물로 10세기 말에서 11세기 초에 활동했습니다.  
 - 갤럭시 폰은 삼성에서 2009년부터 지금까지 출시된 모바일 모델입니다.

ChatGPT 강감찬 장군이 실제로 갤럭시 폰을 사용했다는 기록은 없습니다. 강감찬 장군은 10세기 말에서 11세기 초에 활동한 고려 시대의 뛰어난 무장 지휘관으로, 그 당시에는 현대적인 휴대전화가 없었습니다. 그러나 만약 그가 현대의 기술을 이용할 수 있었다면, 갤럭시 폰을 사용하여 전략적인 의사 소통, 정보 수집 및 분석, 그리고 군사 작전을 조정하는 데 사용할 수 있을 것입니다. 이를 통해 실시간으로 전술을 조정하고 효과적인 지휘를 수행할 수 있었을 겁니다.

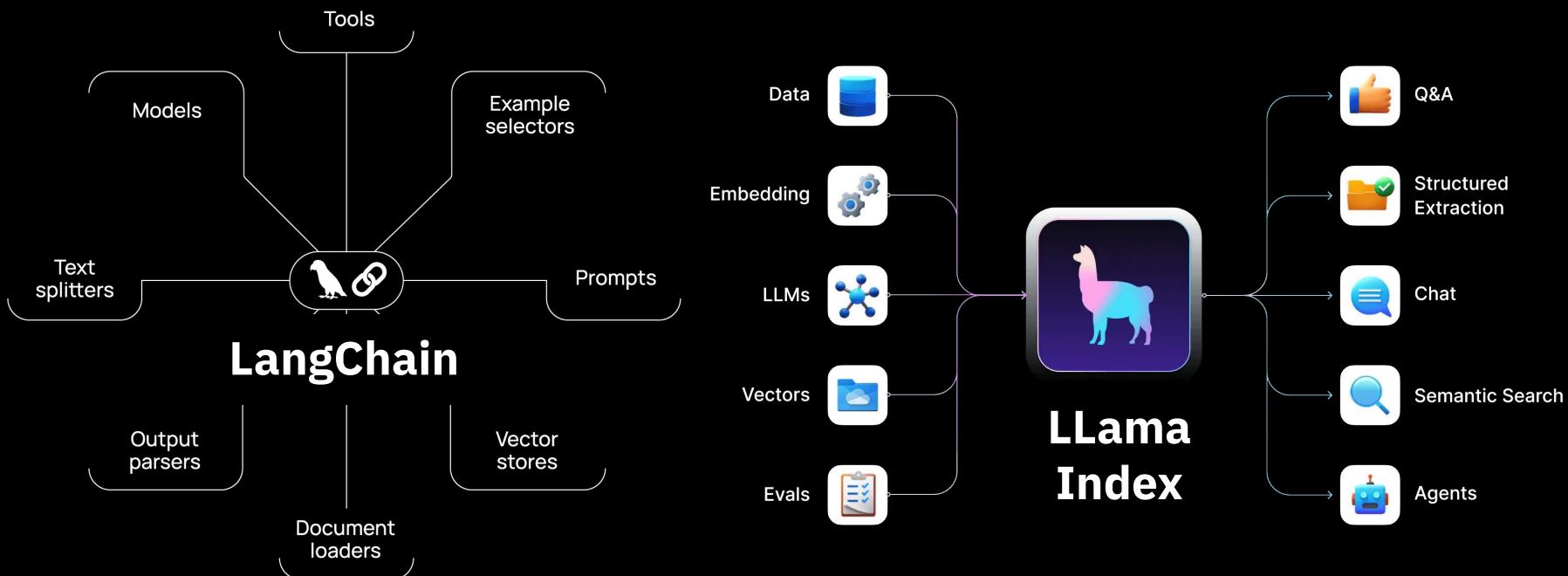
# 생성형 AI 주요 개념



<https://docs.spring.io/spring-ai/reference/concepts.html>

# AI Framework

Python, Javascript ...



# Spring AI

생성 AI 애플리케이션이 많은 프로그래밍 언어에서  
보편화 될 것이라는 믿음으로 시작

# Spring AI Features

## 주요 AI 모델 Provider 통합

- OpenAI, 마이크로소프트, 아마존, 구글, 허깅페이스 등
- 채팅, 텍스트-이미지 변환 모델 유형 지원 (추가 예정)
- 동기식 및 스트림 모두 지원

## 주요 벡터 DB 지원

- Azure Vector Search, Chroma, Milvus, Neo4j, PostgreSQL/PGVector, PineCone, Qdrant, Redis, Weaviate 등
- 벡터 스토어 제공업체 간 이식 가능한 새로운 SQL 유사 메타데이터 필터 API

## AI 개발 편의

- AI 모델 출력을 POJO 매팅
- Function Calling 지원
- AI 모델 및 벡터 스토어용 Spring Boot Starter 제공

# AI Model Support

## Chat Models



PaLM2, Gemini



## Image Models



Cohere, Llama  
Titan, Anthropic



## Embeddings Models

OpenAI

VertexAI

Azure OpenAI

MistralAI

Ollama

PostgresML

Bedrock Cohere

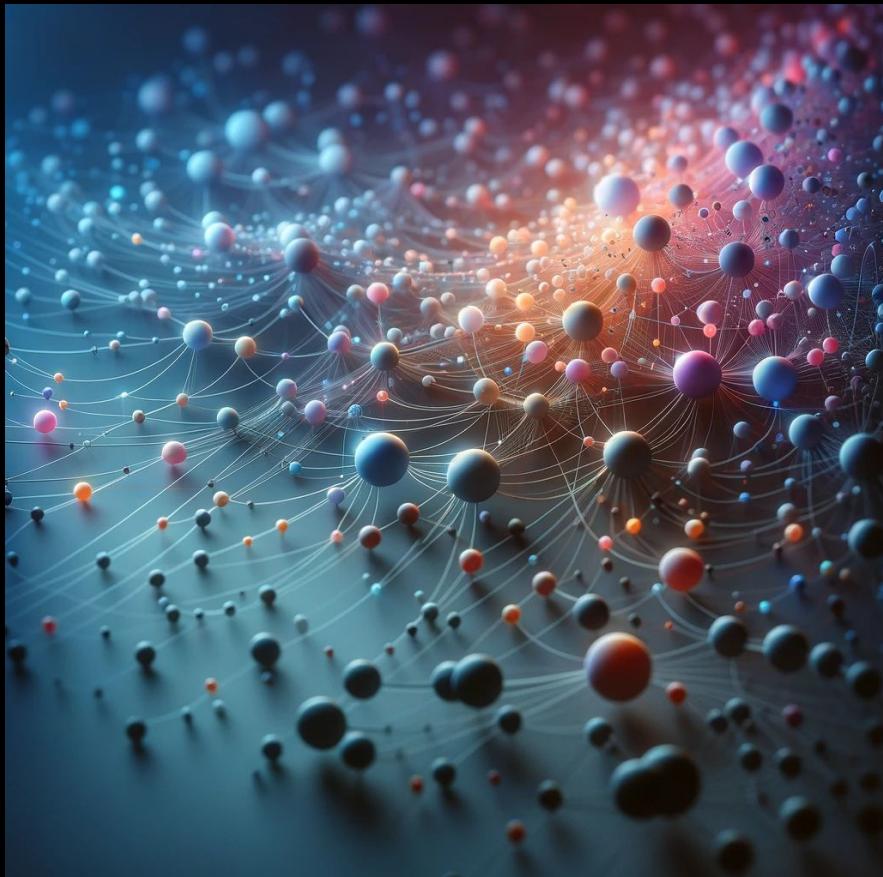
Transformers (ONNX)

Bedrock Titan

# Vector DB Support

## Vector DB

- Azure AI Vector Database
- Chroma Vector Database
- Milvus Vector Database
- Neo4J Vector Database
- PGvector Vector Database
- Pinecone
- Qdrant
- Weaviate



# Spring AI

## 마일스톤



# Spring AI Client

Spring 생태계의 다른 클라이언트 클래스와 유사한 스타일로 프롬프트를 만들고 실행하기 위한 클래스

RestClient

WebClient

JdbcClient

ChatClient

StreamingChatClient

EmbeddingClient

ImageClient

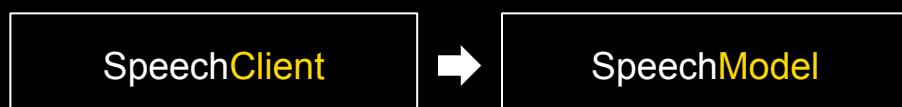
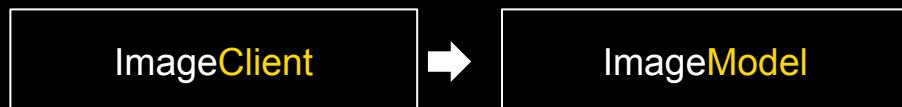
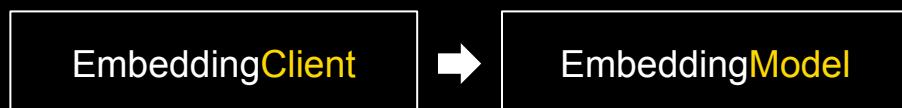
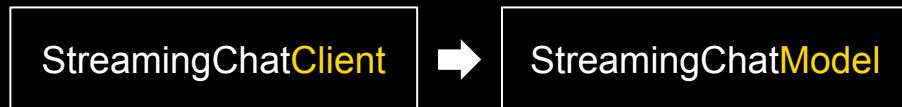
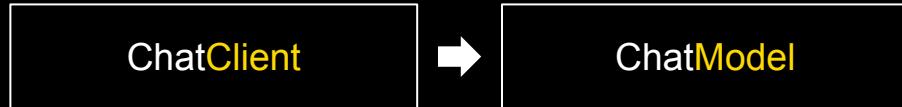
SpeechClient

...

# Spring AI

최근 변경사항 - 1.0.0-M1 / 5월 22일

이제 ChatClient는 ChatModel의 인스턴스를 사용



Deprecate 되고, 패키지 위치도 변경...

```

| Deprecated
@Deprecated(
    since = "1.0.0 M1",
    forRemoval = true
)
default String call(String message) {
    Prompt prompt = new Prompt(new UserMessage(message));
    Generation generation = this.call(prompt).getResult();
    return generation != null ? generation.getOutput().getContent() : "";
}

| Deprecated
@Deprecated(
    since = "1.0.0 M1",
    forRemoval = true
)
default String call(Message... messages) {
    Prompt prompt = new Prompt(Arrays.asList(messages));
    Generation generation = this.call(prompt).getResult();
    return generation != null ? generation.getOutput().getContent() : "";
}
  
```

# Spring AI

## 방법 1. xxxClient 대신 xxxModel 사용

```
● ● ●
@RestController
public class OldSimpleAiController {

    private final ChatClient chatClient;

    @Autowired
    public OldSimpleAiController(ChatClient chatClient) {
        this.chatClient = chatClient;
    }

    @GetMapping("/ai/simple")
    public Map<String, String> completion(@RequestParam(value =
    "message", defaultValue = "Tell me a joke") String message) {
        return Map.of("generation", chatClient.call(message));
    }
}
```



```
● ● ●
@RestController
public class SimpleAiController {

    private final ChatModel chatModel;

    @Autowired
    public SimpleAiController(ChatModel chatModel) {
        this.chatModel = chatModel;
    }

    @GetMapping("/ai/simple")
    public Map<String, String> completion(@RequestParam(value =
    "message", defaultValue = "Tell me a joke") String message) {
        return Map.of("generation", chatModel.call(message));
    }
}
```

## 방법 2. ChatClient Builder 사용

```
● ● ●
@RestController
public class OldSimpleAiController {

    private final ChatClient chatClient;

    @Autowired
    public OldSimpleAiController(ChatClient chatClient) {
        this.chatClient = chatClient;
    }

    @GetMapping("/ai/simple")
    public Map<String, String> completion(@RequestParam(value =
    "message", defaultValue = "Tell me a joke") String message) {
        return Map.of(
            "generation",
            chatClient.call(message)
        );
    }
}
```



```
● ● ●
@RestController
public class SimpleAiController {

    private final ChatClient chatClient;

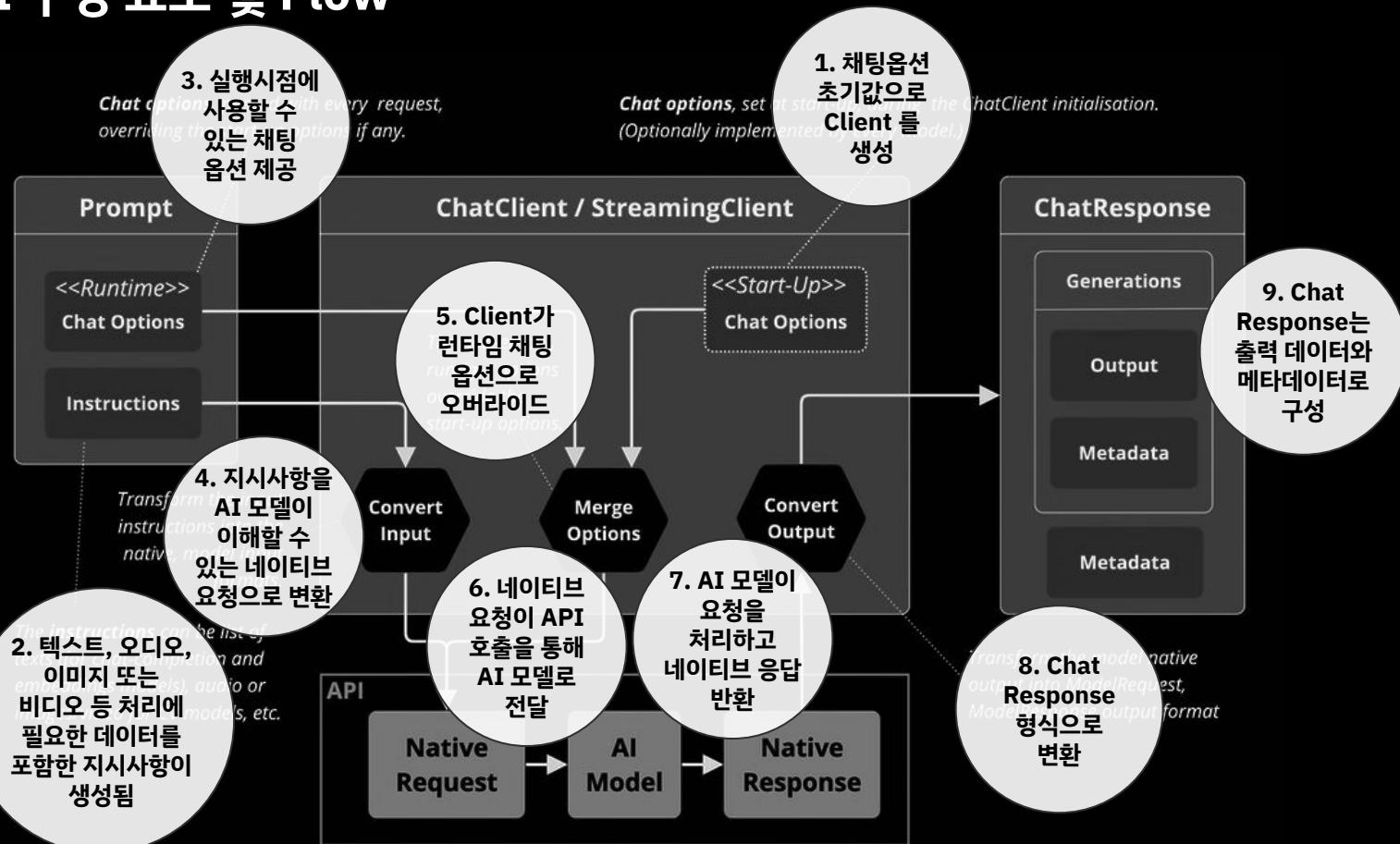
    @Autowired
    public SimpleAiController(ChatClient chatClient) {
        this.chatClient = chatClient;
    }

    @GetMapping("/ai/simple")
    public Map<String, String> completion(@RequestParam(value =
    "message", defaultValue = "Tell me a joke") String message) {
        return Map.of(
            "generation",
            chatClient.prompt().user(message).call().content()
        );
    }
}
```

```
● ● ●
@Configuration
public class ApplicationConfiguration {

    @Bean
    ChatClient chatClient(ChatModel chatModel) {
        return ChatClient.builder(chatModel).build();
    }
}
```

# Spring AI 구성 요소 및 Flow



# OpenAI API 연결

Curl, Python, Spring AI 비교

Curl

```
curl https://api.openai.com/v1/chat/completions \
-H "Content-Type: application/json" \
-H "Authorization: Bearer $OPENAI_API_KEY" \
-d '{
  "model": "gpt-3.5-turbo",
  "messages": [
    {
      "role": "system",
      "content": "You are a helpful assistant."
    },
    {
      "role": "user",
      "content": "Who won the world series in 2020?"
    },
    {
      "role": "assistant",
      "content": "The Los Angeles Dodgers won the World Series in 2020."
    },
    {
      "role": "user",
      "content": "Where was it played?"
    }
  ]
}'
```

Python (OpenAI SDK)

```
response = client.chat.completions.create(
    model="gpt-3.5-turbo",
    messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "Who won the world series in 2020?"},
        {"role": "assistant", "content": "The Los Angeles Dodgers won the World Series in 2020."}
    ]
)

return response.choices[0].message.content
```

Spring AI

```
ChatResponse response = chatClient.prompt()
    .system("Hello! How can I help you today?")
    .user("Who won the world series in 2020?")
    .messages(new AssistantMessage("The Los Angeles Dodgers won the World Series in 2020."))
    .user("Where was it played?")
    .options(OpenAiChatOptions.builder().withModel("gpt-3.5-turbo").build())
    .call().chatResponse();

return response.getResult().getOutput().getContent();
```

The 2020 World Series was played at Globe Life Field in Arlington, Texas.

# Spring AI 사용을 위한 기본 설정

아래 환경 설정은 **1.0.0-M1** 기준

## build.gradle

```

repositories {
    mavenCentral()
    maven {
        url 'https://repo.spring.io/milestone'
        content {
            includeGroup 'org.springframework.ai'
        }
    }
    maven {
        url 'https://repo.spring.io/snapshot'
        content {
            includeGroup 'org.springframework.ai'
        }
    }
}

ext {
    set('springAiVersion', '1.0.0-SNAPSHOT')
}

dependencyManagement {
    imports {
        mavenBom "org.springframework.ai:spring-ai-bom:${springAiVersion}"
    }
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.ai:spring-ai-openai-spring-boot-starter'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'
}

```

## application.yaml

```

spring.application.name: spring-ai-basic
spring:
    ai:
        openai:
            api-key: ${OPENAI_API_KEY}
        chat:
            options:
                model: 'gpt-4o'

```

## ChatClient

```

@SpringBootApplication
public class SpringAiBasicApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringAiBasicApplication.class, args);
    }

    @Bean
    ChatClient chatClient(ChatModel chatModel) {
        return ChatClient.builder(chatModel).build();
    }
}

```

# 예제1) Hello 민호

ChatClient, PromptTemplate 클래스

```
● ● ●  
@RestController  
@RequestMapping("/chat")  
public class ChatController {  
  
    private final ChatClient chatClient;  
  
    public ChatController(ChatClient chatClient) {  
        this.chatClient = chatClient;  
    }  
  
    @GetMapping("/hello")  
    public String hello(@RequestParam(value = "name", defaultValue = "robin") String name) {  
        PromptTemplate promptTemplate = new PromptTemplate("Hello! My name is {name}");  
        Prompt prompt = promptTemplate.create(Map.of("name", name));  
        return chatClient.prompt(prompt).call().chatResponse().getResult().getOutput().getContent();  
    }  
}
```

<http://localhost:8080/chat/hello?name=민호>

Hello, 민호! How can I assist you today?

## 예제2) 고양이 이미지 생성

ImageModel 클래스 : Dall-E 3 모델로 이미지 생성

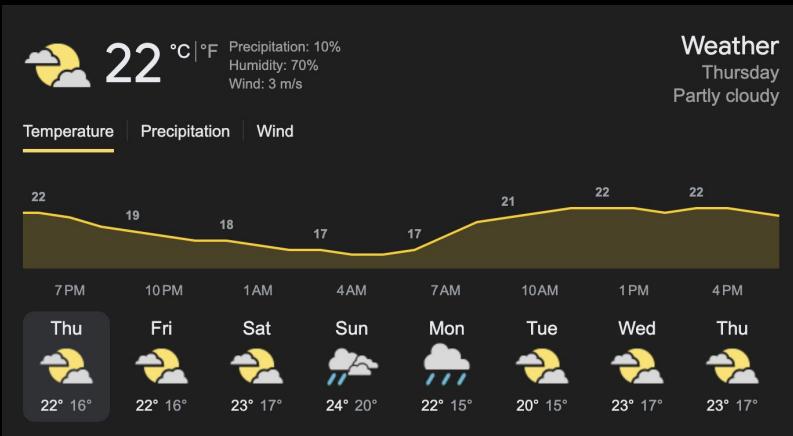
<http://localhost:8080/image?request=%EC%9E%90%ED%8A%8C%EB%A1%9C%EC%9D%BC>

```
● ● ●  
@Controller  
public class ImageController {  
  
    private final ImageModel imageModel;  
  
    public ImageController(ImageModel imageModel) {  
        this.imageModel = imageModel;  
    }  
  
    @GetMapping("/image")  
    public String imageGen(@RequestParam String request) {  
        ImageOptions options = ImageOptionsBuilder.builder()  
            .withModel("dall-e-3")  
            .build();  
  
        ImagePrompt imagePrompt = new ImagePrompt(request, options);  
        ImageResponse response = imageModel.call(imagePrompt);  
        String imageUrl = response.getResult().getOutput().getUrl();  
  
        return "redirect:" + imageUrl;  
    }  
}
```



# 예제3) 이미지 인식을 통한 일기예보 전달

일기 예보 이미지를 함께 전달하여 GPT-4 Vision 기능을 활용



```

@RestController
public class ForecastController {
    private final ChatClient chatClient;
    @Value("classpath:/forecast.jpg")
    Resource forecastImageResource;

    public ForecastController(ChatClient chatClient) {
        this.chatClient = chatClient;
    }

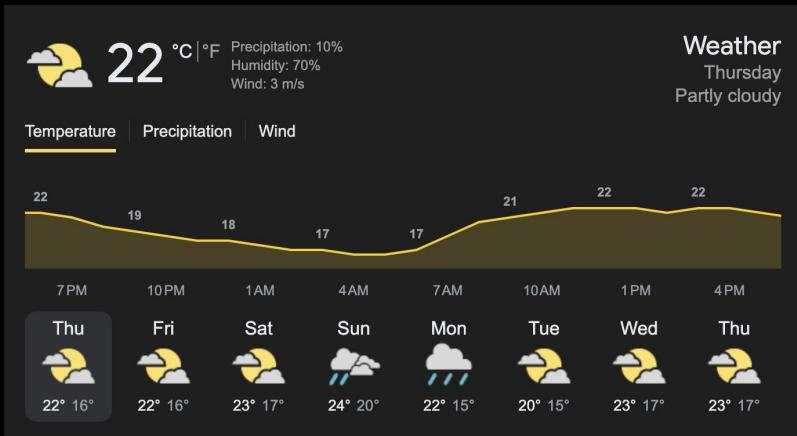
    @GetMapping("/forecast")
    public String forecast(@RequestParam String ask) throws Exception {
        return chatClient.prompt()
            .user(userSpec -> userSpec
                .text(ask)
                .media(MediaTypeUtils.IMAGE_JPEG, forecastImageResource))
            .call()
            .content();
    }
}

```

# 예제3) 이미지 인식을 통한 일기예보 전달

일기 예보 이미지를 함께 전달하여 GPT-4 Vision 기능을 활용

<http://localhost:8080/forecast?ask=날씨 정보 알려줘>



현재 날씨는 부분적으로 흐리고, 기온은 22°C입니다. 강수 확률은 10%, 습도는 70%, 바람은 시속 3m/s로 불고 있습니다.

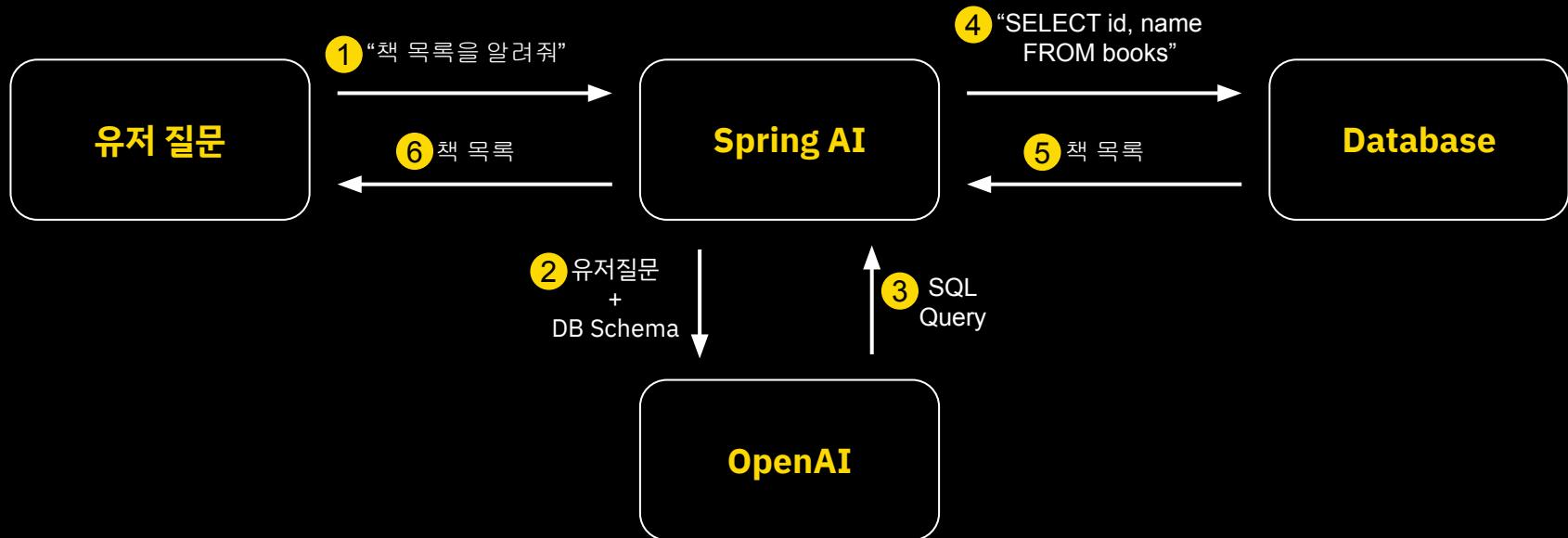
향후 일주일간의 일기 예보는 다음과 같습니다:

- 목요일 (오늘): 22°C / 16°C, 부분적으로 흐림
- 금요일: 22°C / 16°C, 부분적으로 흐림
- 토요일: 23°C / 17°C, 부분적으로 흐림
- 일요일: 24°C / 20°C, 비
- 월요일: 22°C / 15°C, 비
- 화요일: 20°C / 15°C, 비
- 수요일: 23°C / 17°C, 부분적으로 흐림
- 목요일: 23°C / 17°C, 부분적으로 흐림

날씨에 맞게 계획을 세우시길 바랍니다!

# 예제3) 자동으로 SQL 쿼리 작성해서 조회하기

전체 Flow



## 예제3) 자동으로 SQL 쿼리 작성해서 조회하기

프롬프트 템플릿을 통해 DB 정보와 사용자 질문을 함께 전달하도록 구현



```
@RestController
public class SqlController {
    @Value("classpath:/schema.sql")
    private Resource ddlResource;

    @Value("classpath:/sql-prompt-template.st")
    private Resource sqlPromptTemplateResource;

    private final ChatClient aiClient;
    private final JdbcTemplate jdbcTemplate;

    public SqlController(
        ChatClient.Builder aiClientBuilder,
        JdbcTemplate jdbcTemplate) {
        this.aiClient = aiClientBuilder.build();
        this.jdbcTemplate = jdbcTemplate;
    }
}
```

```
@GetMapping(path="/sql")
public Answer sql(@RequestParam String question) throws IOException {
    String schema = ddlResource.getContentAsString(Charset.defaultCharset());

    String query = aiClient.prompt()
        .user(userSpec -> userSpec
            .text(sqlPromptTemplateResource)
            .param("question", question)
            .param("ddl", schema)
        )
        .call()
        .content();
    if (query.toLowerCase().startsWith("select")) {
        return new Answer(query, jdbcTemplate.queryForList(query));
    }

    throw new SqlGenerationException(query);
}
```

## 예제3) 자동으로 SQL 쿼리 작성해서 조회하기

### 프롬프트 템플릿 예시

sql-prompt-template.st

DDL 섹션의 DDL이 주어지면 질문 섹션에서 질문에 답하기 위한 SQL 쿼리를 작성합니다.  
SQL 쿼리 이외에는 아무것도 작성하지 마십시오.

쿼리는 마크다운 포맷이 아닌 일반 텍스트로 작성합니다.

SELECT 쿼리만 생성합니다. 질문으로 인해 삽입, UPDATE 또는 DELETE가 발생하거나  
쿼리가 어떤 식으로든 DDL을 변경하는 경우 작업 작업이 지원되지 않는다고 말합니다.

질문에 대한 답을 얻을 수 없는 경우에는 DDL이 해당 질문에 대한 답을 지원하지 않는다고  
말합니다.

QUESTION  
{question}

DDL  
{ddl}

# 예제3) 자동으로 SQL 쿼리 작성해서 조회하기

GPT 가 작성한 쿼리로 DB에 수행하여 받아온 결과

<http://localhost:8080/sql?question=작가목록을알려줘>

<http://localhost:8080/sql?question=김유정작가의작품을알려줘>

```
{  
    "sqlQuery": "select id, name from authors;",  
    "results": [  
        {  
            "ID": 1,  
            "NAME": "박경리"  
        },  
        {  
            "ID": 2,  
            "NAME": "김유정"  
        },  
        {  
            "ID": 3,  
            "NAME": "이청준"  
        },  
        {  
            "ID": 4,  
            "NAME": "한강"  
        }  
    ]  
}
```

```
{  
    "sqlQuery": "select Books.title \nfrom Books \njoin  
Authors on Books.author_ref = Authors.id \nwhere  
Authors.firstName = '김유정';",  
    "results": [  
        {  
            "TITLE": "동백꽃"  
        },  
        {  
            "TITLE": "봄봄"  
        },  
        {  
            "TITLE": "만무방"  
        }  
    ]  
}
```

# OpenAI API 비용

```
spring.application.name: spring-ai-basic
spring:
  ai:
    openai:
      api-key: ${OPENAI_API_KEY}
      chat:
        options:
          model: 'gpt-4o'
```

## GPT-3.5 Turbo

GPT-3.5 Turbo is our fast and inexpensive model for simpler tasks.

`gpt-3.5-turbo-0125` is the flagship model of this family, supports a 16K context window and is optimized for dialog.

`gpt-3.5-turbo-instruct` is an Instruct model and only supports a 4K context window.

[Learn about GPT-3.5 Turbo ↗](#)

Model	Input	Output
gpt-3.5-turbo-0125	US\$0.50 / 1M tokens	US\$1.50 / 1M tokens
gpt-3.5-turbo-instruct	US\$1.50 / 1M tokens	US\$2.00 / 1M tokens

## GPT-4o

GPT-4o is our most advanced multimodal model that's faster and cheaper than GPT-4 Turbo with stronger vision capabilities. The model has 128K context and an October 2023 knowledge cutoff.

[Learn about GPT-4o ↗](#)

Model	Input	Output
gpt-4o	US\$5.00 / 1M tokens	US\$15.00 / 1M tokens
gpt-4o-2024-05-13	US\$5.00 / 1M tokens	US\$15.00 / 1M tokens

# 로컬 LLM으로 개발하기

Ollama 를 이용



Get up and running with large language models.

Run Llama 3, Phi 3, Mistral, Gemma, and other models. Customize and create your own.

Download ↓

Available for macOS, Linux, and Windows (preview)

<https://ollama.com>



Models

Filter by name...

llama3

Meta Llama 3: The most capable openly available LLM to date

8B 70B

↓ 1.5M Pulls ⚡ 68 Tags ⏲ Updated 3 days ago

phi3

Phi-3 is a family of lightweight 3B (Mini) and 14B (Medium) state-of-the-art open models by Microsoft.

3B 14B

↓ 246.4K Pulls ⚡ 70 Tags ⏲ Updated 3 hours ago

mistral

The 7B model released by Mistral AI, updated to version 0.3.

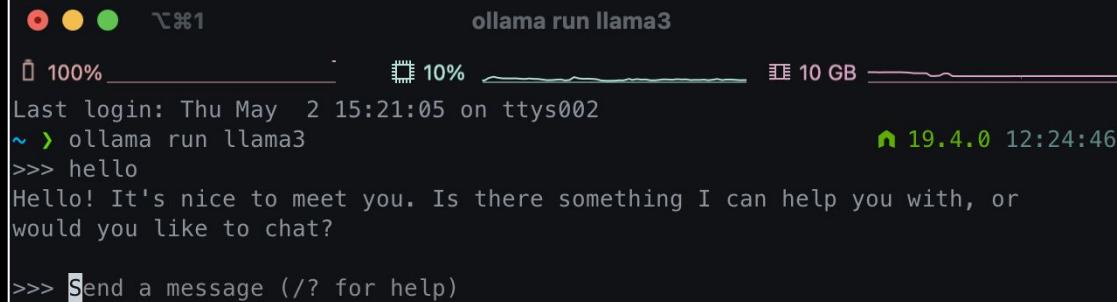
7B

↓ 833.5K Pulls ⚡ 84 Tags ⏲ Updated 2 days ago

# 로컬 LLM으로 개발하기

Ollama 설치 후 실행하기

```
> ollama run llama3
```



```
● ● ● ℗ 100% 10% 10 GB
ollen run llama3
Last login: Thu May  2 15:21:05 on ttys002
~ > ollama run llama3
~> hello
Hello! It's nice to meet you. Is there something I can help you with, or
would you like to chat?

>>> Send a message (/? for help)
```

## API

Example using curl:

```
curl -X POST http://localhost:11434/api/generate -d '{
  "model": "llama3",
  "prompt": "Why is the sky blue?"
}'
```

# 로컬 LLM으로 개발하기

Spring AI Ollama

 **spring initializr**

---

**Project**

Gradle - Groovy       Maven       Gradle - Kotlin       Groovy

**Language**

Java       Kotlin

---

**Spring Boot**

3.3.1 (SNAPSHOT)       3.3.0       3.2.7 (SNAPSHOT)

3.2.6

**Project Metadata**

Group: com.kakao.tech

Artifact: spring-ai-llama

Name: spring-ai-llama

Description: Demo project for Spring Boot

Package name: com.kakao.tech.spring-ai-llama

Packaging:  Jar       War

Java:  22       21       17

---

**Dependencies** ADD DEPENDENCIES... % + B

**Spring Web** WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

**Ollama** AI

Spring AI support for Ollama. It allows you to run various Large Language Models (LLMs) locally and generate text from them.

# Spring AI 텍스트 예제

Ollama를 이용하여 Llama3 8B 모델의 API에 연결

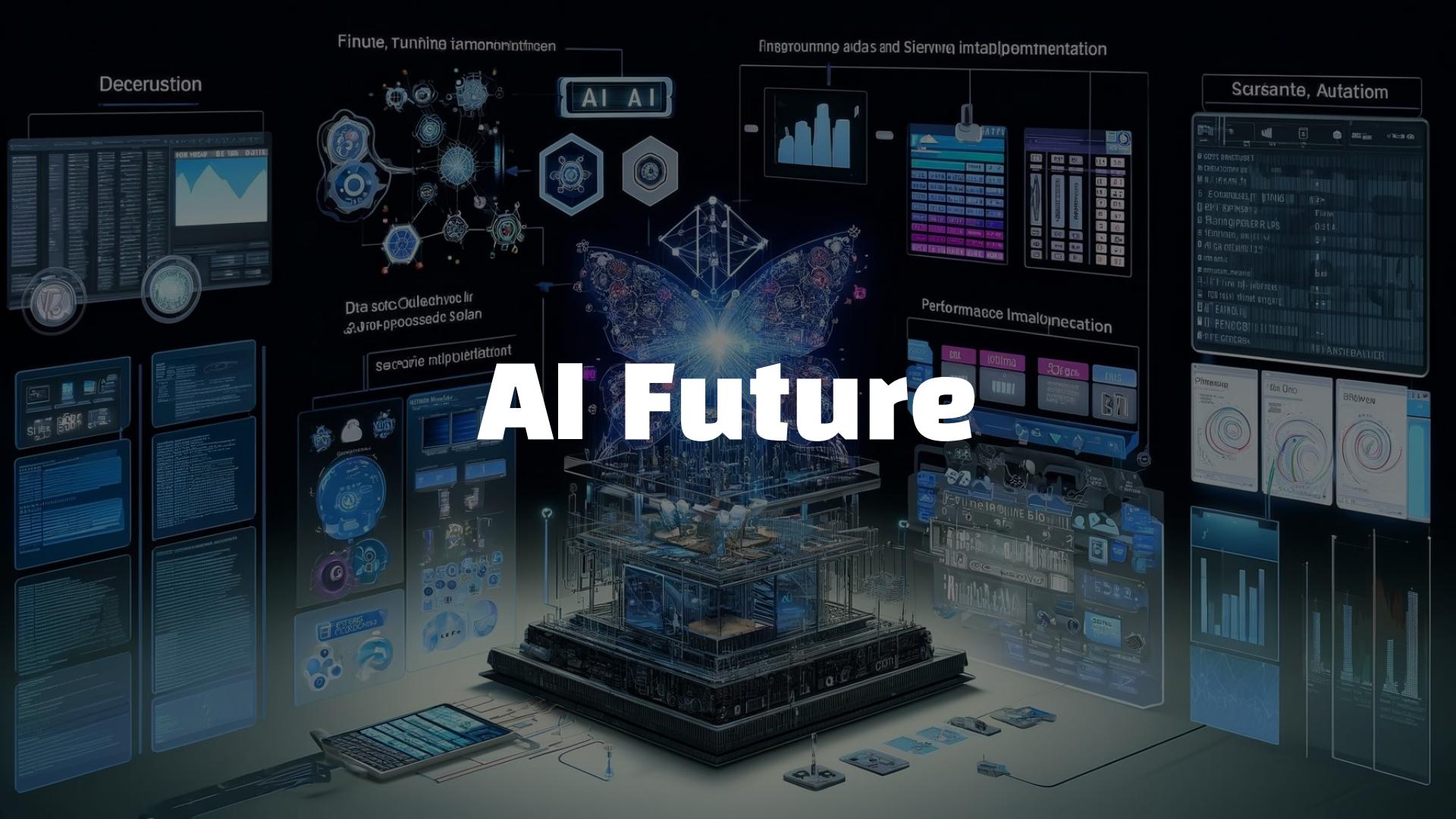
```
private final OllamaChatClient chatClient;

public LlamaChatController(OllamaChatClient ollamaChatClient) {
    this.chatClient = ollamaChatClient;
}

@GetMapping("/hello")
public String hello(@RequestParam(value = "name", defaultValue = "robin") String name) {
    PromptTemplate promptTemplate = new PromptTemplate("Hello! My name is {name}");
    Prompt prompt = promptTemplate.create(Map.of("name", name));
    return chatClient.call(prompt).getResult().getOutput().getContent();
}
```

<http://localhost:8080/chat/hello?name=민호>

Hi, Minho! It's great to meet you. What brings you here today?



Decerustion

Finute, Tunning tamoronttaen

Angrounng aidas and Sienva imaplmentation

Sarsante, Autation

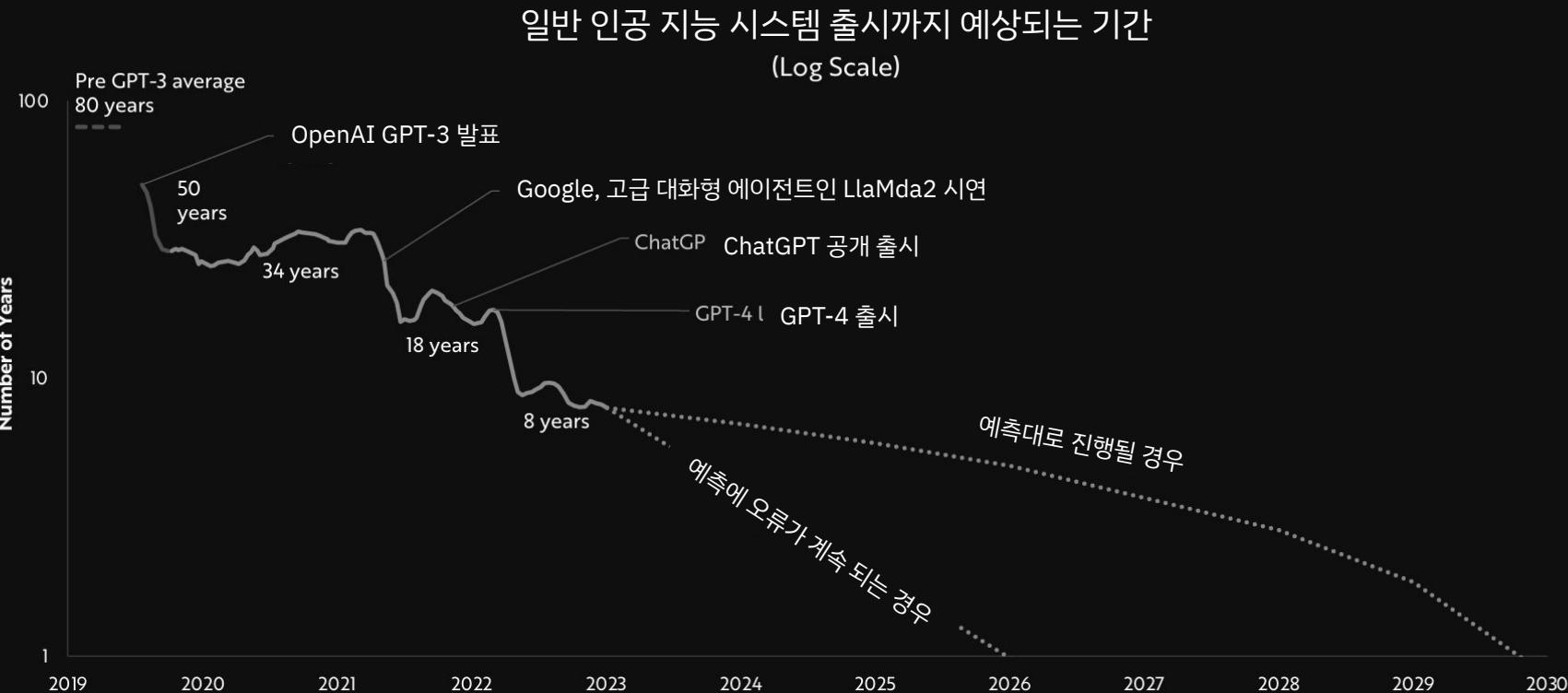
# AI Future

Data sotc Odlehyoc ir  
Qrot pprocessed selan

Secearie mlpoperatront

Performance Imalbination

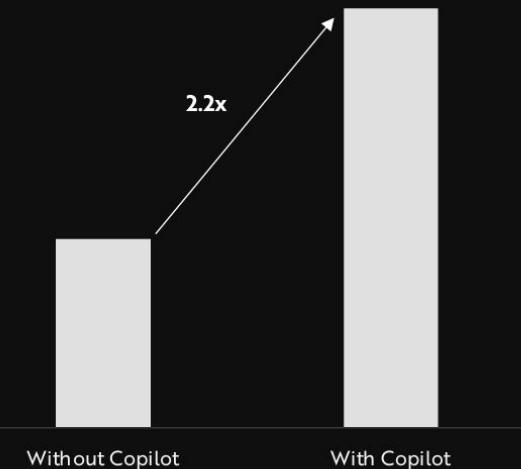
# 예측가들의 예상보다 빠른 속도로 가속화되는 AI



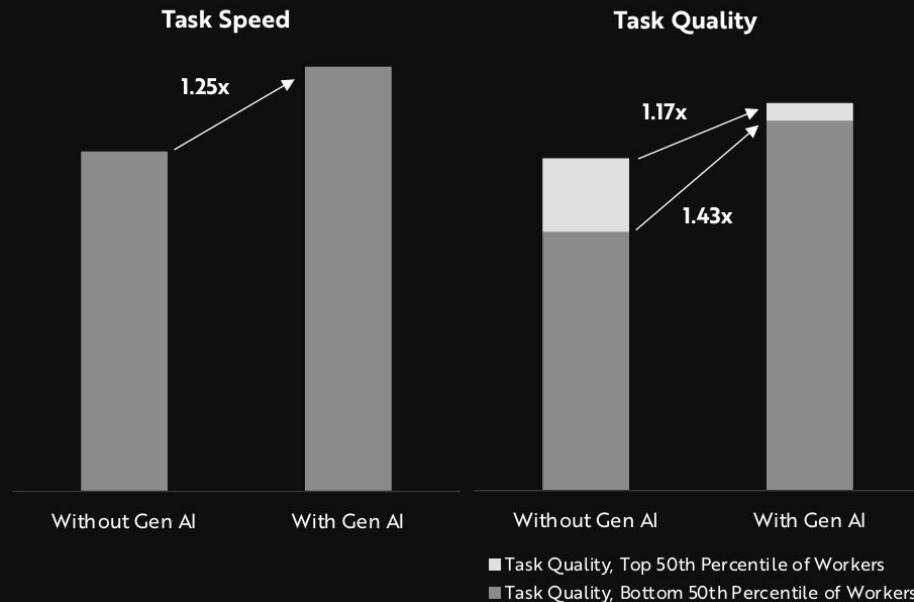
# AI는 이미 생산성을 크게 향상

AI 기반 Assistant는 생산성을 높이고 있으며, 고성과자보다 저성과자에게 상대적으로 더 많은 혜택을 주고 있음

2023년 GitHub Copilot 을 이용한  
코딩의 개발자 생산성



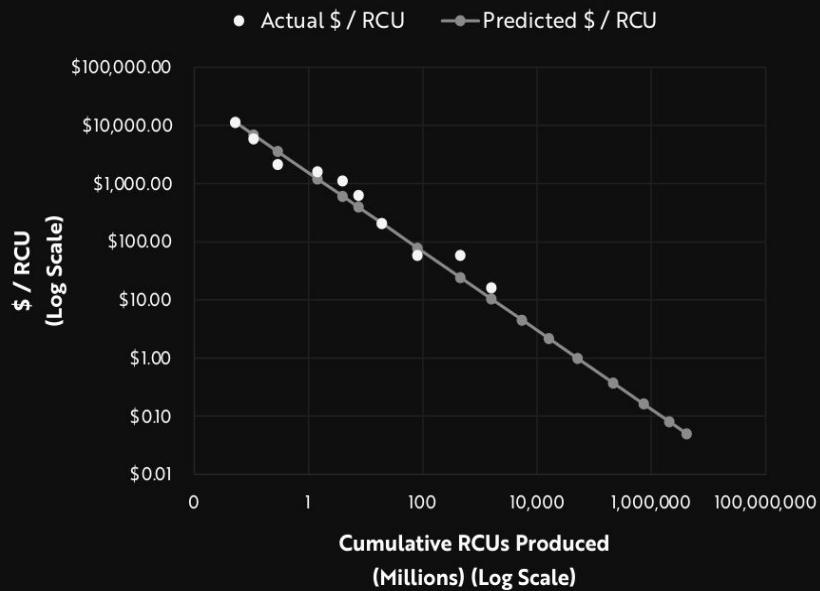
2023년 Generative AI를 사용하는 컨설턴트의 생산성



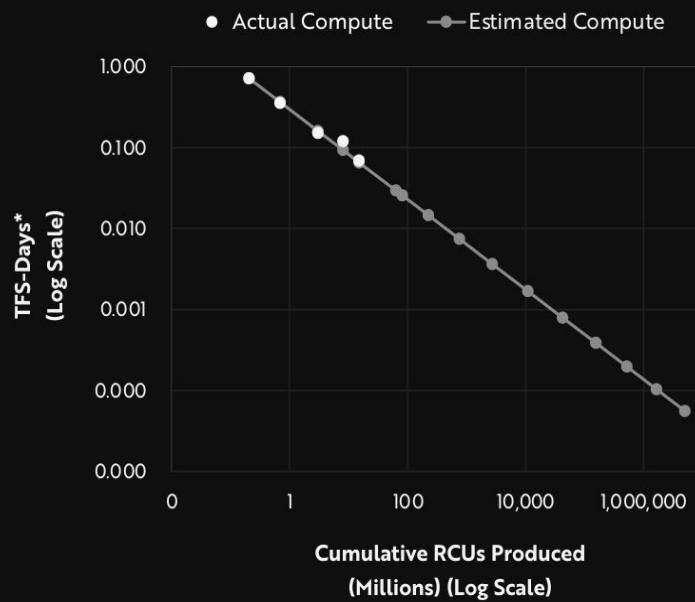
# 훈련 비용은 매년 75%씩 계속 감소

AI 상대 컴퓨팅 유닛(RCU) 비용이 연간 53% 감소  
 알고리즘 개선으로 학습 비용이 연간 47% 더 낮아짐  
 2030년까지 AI 학습 비용을 연간 75%까지 감소 예상

AI 트레이닝 하드웨어 비용



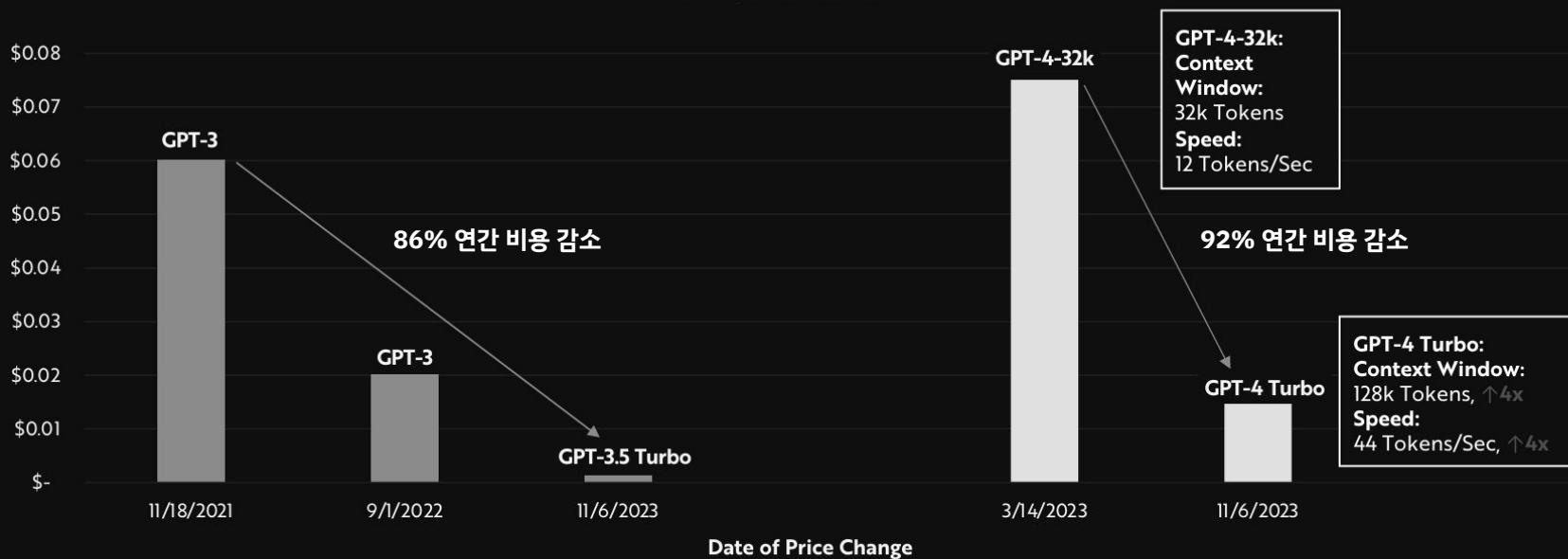
신경망을 사용한 AI 소프트웨어 학습 비용



# AI 추론 비용의 감소

추론 비용은 연간 약 86%의 속도로 훈련비용 보다 훨씬 빠른 속도로 하락  
현재 GPT-4 Turbo와 관련된 추론 비용은 1년 전 GPT-3의 추론 비용보다 낮음

토큰 1,000개당 GPT-3 및 GPT-4 API 추론비용

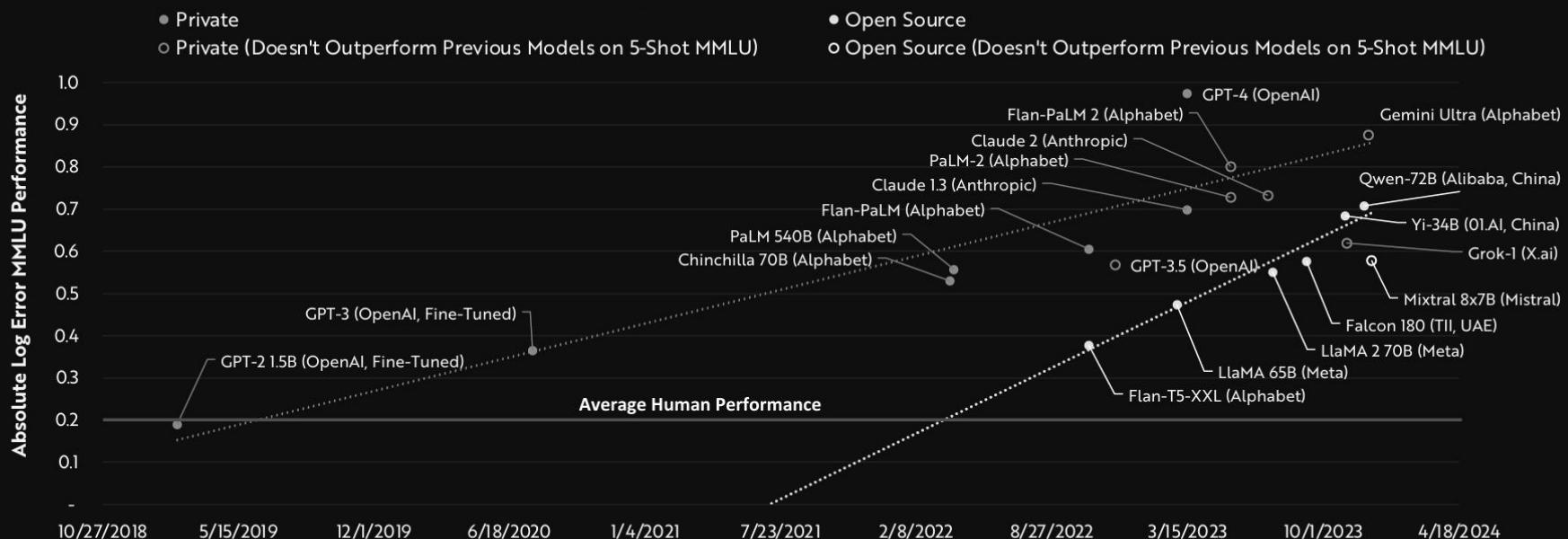


# 오픈소스 모델 성장 가속

Meta의 도전적인 폐쇄형 소스 모델은 제너레이티브 AI에 대한 액세스를 대중화

최근 중국의 모델에 힘입어 오픈 소스 모델의 성능이 프라이빗 모델보다 더 빠르게 향상

## 오픈소스 모델과 프라이빗 모델 5-shot MMLU 성능 비교





AI와 함께  
실험하고 성장하는  
AI 개발은  
**Spring AI로!**

[robin.hwang@kakao.com](mailto:robin.hwang@kakao.com)