

【대구은행】 iM DiGital Banker Academy  
데이터 분석 전문가 양성과정

GPT-4 and ChatGPT

# 1. LLM 소개

---

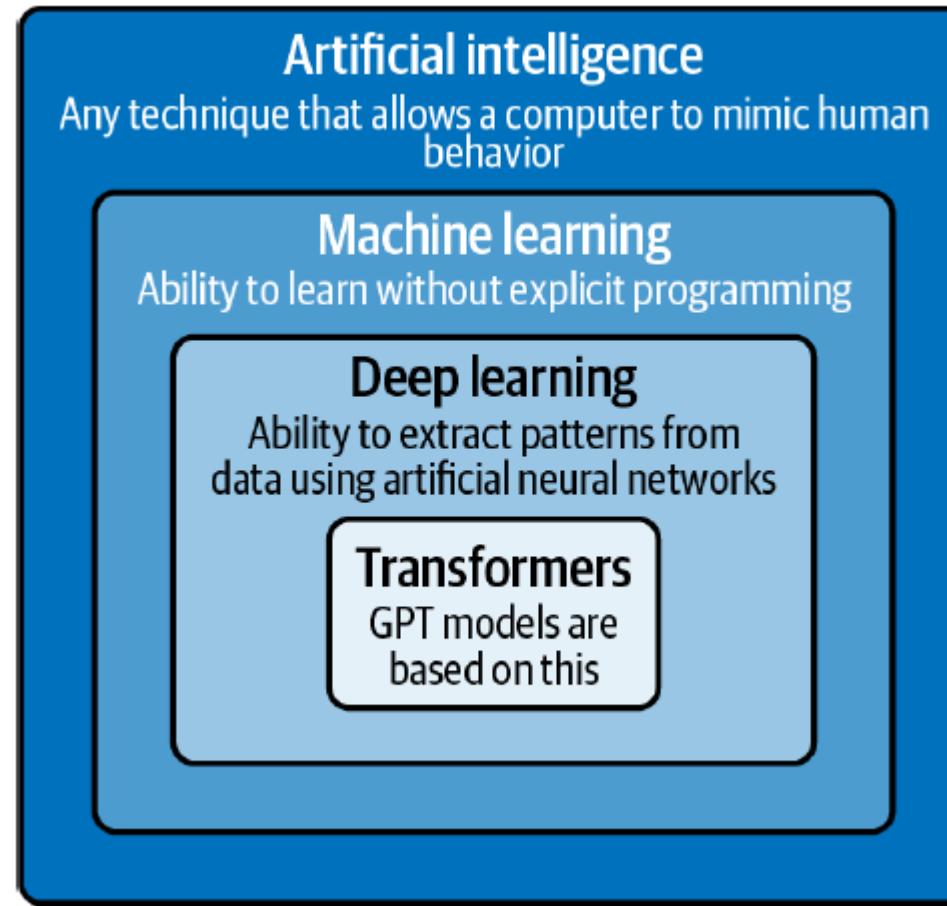
## 1. 언어 모델과 자연어 처리의 기초 탐구

### ▶ 주요 작업

- 텍스트 분류 : 감성 분석 및 토픽 모델링
- 기계 번역 : 한국에서 영어로, 파이썬에서 C++로 번역
- 질의 응답 : 제품 관련 문의 답변, 학생 질문 답변
- 텍스트 생성 : 프롬프트라고도 하는 주어진 입력 텍스트를 일관되고 관련성 있는 출력 텍스트 생성

## 2. 트랜스포머 아키텍처와 역할

### ▶ 트랜스포머 개념 이해도



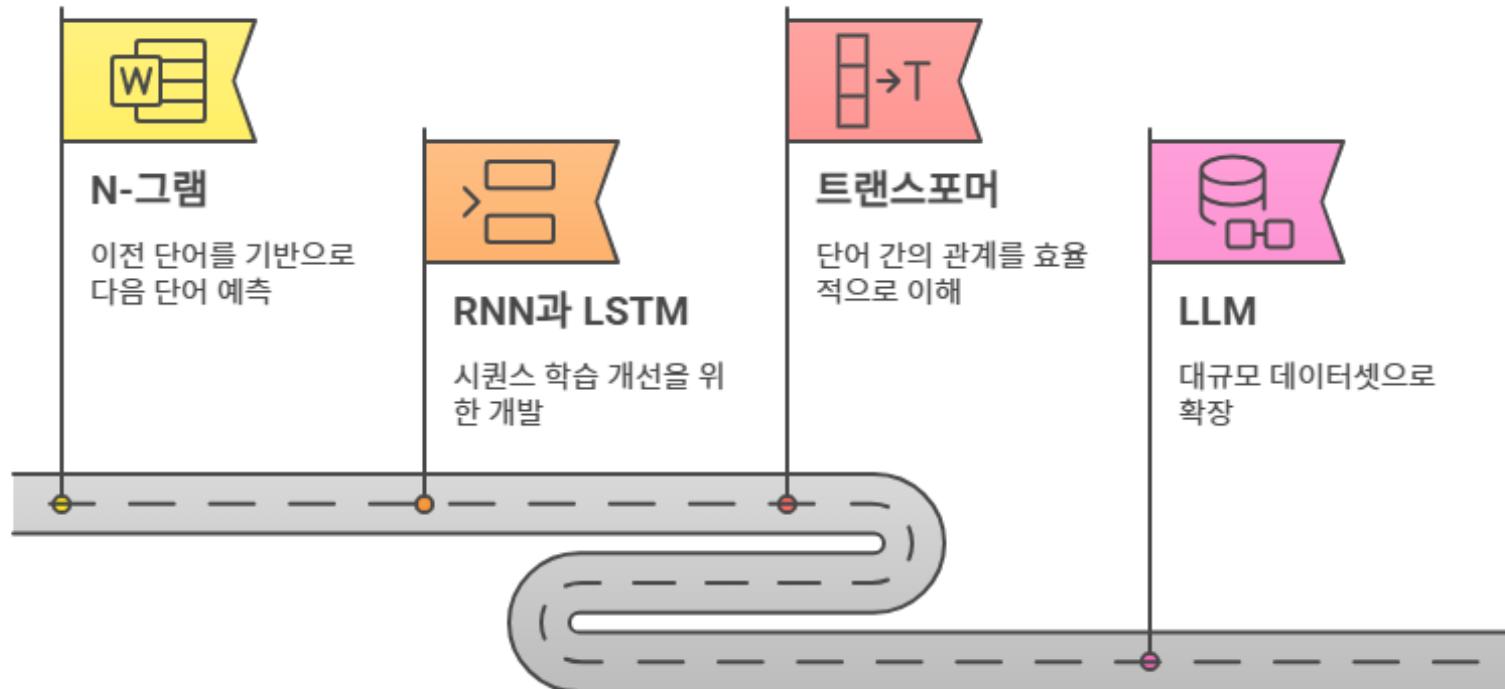
## 2. 트랜스포머 아키텍처와 역할

### ▶ 트랜스포머 이론 요약

- 트랜스포머는 Context를 효과적으로 처리하고 인코딩하는 기능 갖춤 (병렬처리)
- 어텐션 매커니즘
  - 시퀀스의 모든 단어를 똑같이 중요하게 다루지 않고, 작업의 각 단계에서 가장 관련성이 높은 부분에 집중
- 교차 어텐션
  - 입력 텍스트의 여러 부분의 관련성을 판단해 모델이 출력 텍스트의 다음 단어 정확히 예측
- 셀프 어텐션
  - 모델이 입력 텍스트의 여러 부분에 집중할 수 있는 기능
  - 문장 내의 각 단어의 중요도를 다른 단어와 함께 평가해 단어 간의 관계 더 잘 이해
  - 모델이 입력 텍스트의 여러 단어로부터 새로운 개념 구축

## 2. 트랜스포머 아키텍처와 역할

### ▶ NLP 기술 발전 요약



### 3. GPT

#### ▶ GPT 소개

- 트랜스포머 아키텍처 기반(인코더 + 디코더)으로, 그 중 디코더 부분 활용하는 모델
  - 인코더에서 생성된 임베딩을 통합하는 교차 어텐션 불필요
  - 디코더 내의 셀프 어텐션 메커니즘에만 의존해 Context 인식과 예측 생성
- 반면, 버트는 인코더 부분 활용하는 모델

### 3. GPT

#### ▶ GPT 모델의 토큰화 및 예측 단계

- 프롬프트가 LLM으로 전송되면 먼저 입력 내용을 토큰(Token)이라고 부르는 작은 조각으로 나눔
- 단일 단어, 단어의 일부, 공백, 구두점 등을 모두 토큰으로 나타냄
- GPT 모델별 토크나이저 [링크](#)

### 3. GPT

#### ▶ 입력 컨텍스트 윈도 (2024년 11월 기준)

- 128,000 토큰 지원 (종이책 300페이지 분량)
- 컨텍스트에 따라 각 잠재적 후속 토큰에 확률값 적용

### 3. GPT

#### ▶ 프롬프트 입력 처리 과정 도식화

##### 1. Receive prompt

- Example : "The weather is nice today, so I decided to"

##### 2. Break input into tokens

- Example : ["The", "wea", "ther", "is", "nice", "today", "", "so", "I", "de", "ci", "ded", "to"]

##### 3. Process tokens with Transformer architecture

- Understand relationships between tokens
- Identify the overall meaning of the prompt

##### 4. Predict next token based on context

- Assign probability scores to possible token
- Example : {"go":0.7, "stay":0.2, "wri":0.1}

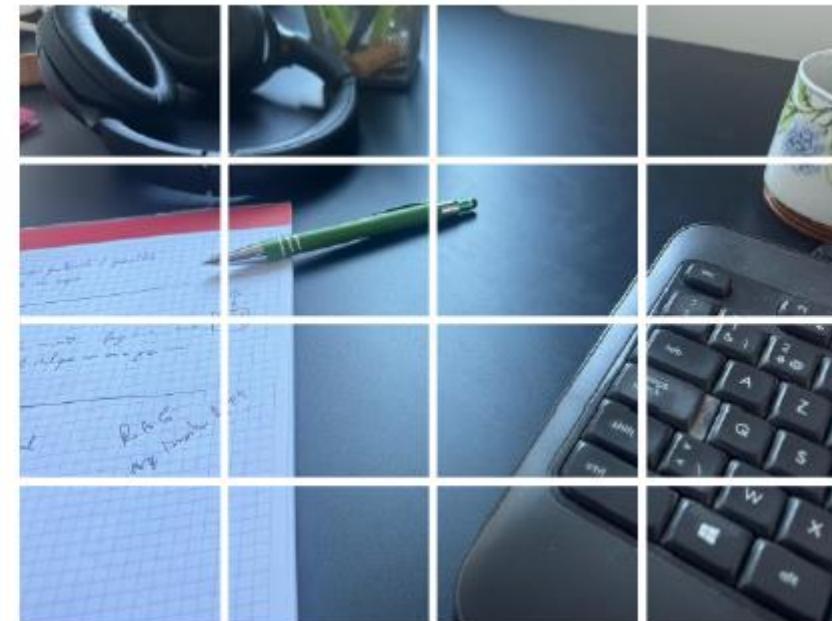
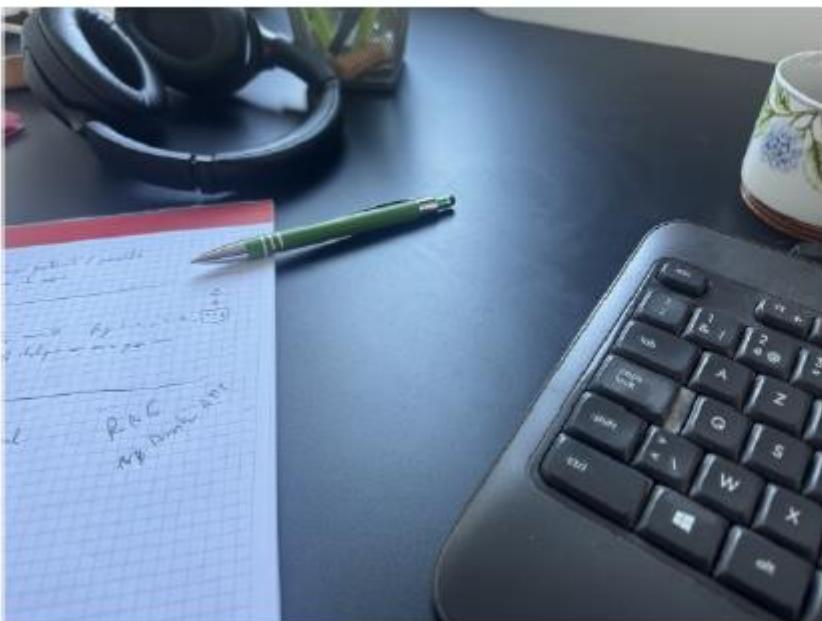
##### 5. Select a token based on this probability score

- Example : "go"

## 4. LLM과 Vision 인식의 통합

### ▶ GPT의 발전

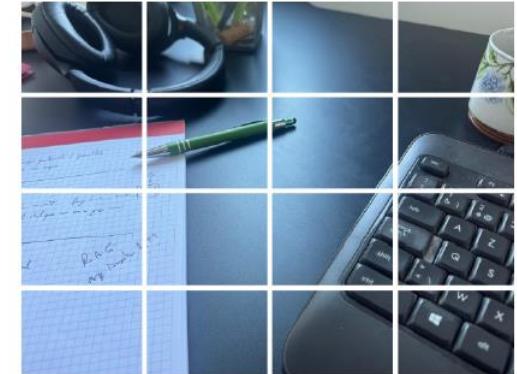
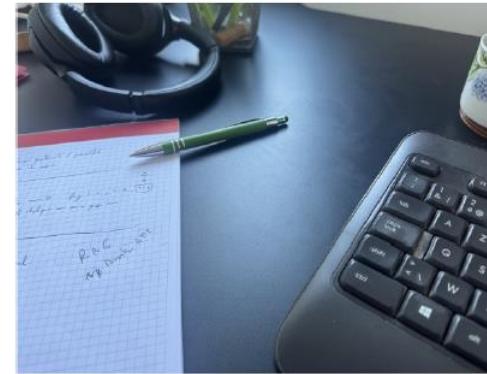
- 2023년 10월 GPT-4 Vision 모델 출시
- Vision Transformer (2021)의 등장



## 4. LLM과 Vision 인식의 통합

### ▶ GPT의 발전

- 2023년 10월 GPT-4 Vision 모델 출시
- Vision Transformer (2021)의 등장
- 한장의 이미지는 단어  $16 \times 16$  개와 같음
- 이미지 패치와 텍스트 토큰은 하나의 입력 시퀀스로 통합



## 5. From GPT-1 To GPT-4o

### ▶ GPT-1 (2018)

- GPT-1 이전 : 지도학습 (레이블 필요)
- GPT-1 : 비지도 사전 학습 단계 도입 (레이블 불필요)
- 11,000권의 미출간 도서 텍스트가 포함된 BookCorpus 데이터셋 사용

## 5. From GPT-1 To GPT-4o

### ▶ GPT-2 (2019)

- GPT-1의 10배 이상의 크기를 가진 GPT-2 모델 확장
  - GPT-2 파라미터 개수는 15억개
  - 40GB 데이터 학습

## 5. From GPT-1 To GPT-4o

### ▶ GPT-3 (2020)

- GPT-2보다 모델의 크기가 더 커짐
  - 1750억 개의 파라미터
  - 수십억 개의 웹 페이지나 위키백과 같은 공개자료 활용

## 5. From GPT-1 To GPT-4o

### ▶ Instruct Series (2021)

- 인간 피드백을 통한 강화학습(RLHF: Reinforcement Learning From Human Feedback)
- 사람의 피드백 재학습하는 과정 반복하며 성능 개선
- 관련 논문

Ouyang. L., etc. (2022). [Training language models to follow instructions with human feedback](#)

- 학습 과정은 지도형 파인 투닝(Supervised Fine-Tuning)과 인간 피드백을 활용한 강화 학습

## 5. From GPT-1 To GPT-4o

### ▶ 1단계 : 지도형 파인 투닝(Supervised Fine-Tuning)

- 목적 : 기존 GPT-3 모델을 보다 일관성 있는 응답 제공하도록 미세 조정
- 방법
  - 사용자 질문 (프롬프트) 수집
  - 사람(Labeler)이 프롬프트에 대한 이상적인 답변 예시 작성
  - 수천개의 프롬프트-답변 데이터셋 구축
  - 이 데이터 사용해 GPT-3 모델 미세 조정

## 5. From GPT-1 To GPT-4o

### ▶ 2단계 : 보상 모델(RM : Reward Model)

- 목적 : 답변의 품질을 자동으로 평가하는 모델 생성
- 방법
  - SFT 모델이 동일한 질문에 대해 여러 개의 답변 생성
  - 사람(Labeler)이 답변을 순위별로 평가 (적절성, 유해성 등 기준 적용)
  - 이 데이터를 사용해 보상 모델(RM) 학습

## 5. From GPT-1 To GPT-4o

### ▶ 3단계 : 강화 학습

- 목적 : 모델이 보상 모델(RM)의 평가를 바탕으로 지속적으로 개선되도록 학습
- 방법
  - 프롬프트 입력 → SFT 모델이 답변 생성 → RM이 평가(보상 점수 부여)
  - 보상 점수를 기반으로 모델을 반복적으로 업데이트
  - 이 과정을 자동으로 반복하여 성능 개선
  - 강화 학습은 계산된 보상을 기반으로 보상을 극대화하기 위해 PPO(Proximal Policy Optimization) 활용

## 5. From GPT-1 To GPT-4o

### ▶ GPT-3.5

- 2021년 6월까지의 데이터로 학습
- 대화형 도구 ChatGPT 공개 (GPT-3.5 turbo)

## 5. From GPT-1 To GPT-4o

### ▶ GPT-4

- 2023년 3월 공개
- 고급 추론 능력에서 기존 모델 능가 ([모델의 성능 평가 기술 보고서 확인](#))
- 텍스트 외에도 이미지도 입력할 수 있는 첫 번째 멀티모달 모델

## 6. 모델 비교

▶ 다양한 모델의 발현 및 생성 → 객관적 모델 평가의 비교성 대두

- 시험 결과
  - [Uniform Bar Exam](#)
  - [국제 생물학 올림피아드](#)
- 그 외 벤치마크 비교 : [Model Evaluations](#)

## 6. 모델 비교

- ▶ 사람들이 서로 다른 모델과의 상호작용 알지 못하게 하고, 이를 평가하는 것
- ▶ LMSYS 챗봇 아레나 리더보드
  - 사용자가 무작위로 선택된 두 개의 모델과 동시에 대화
  - 사용자는 어떤 모델과 이야기하고 있는지 모르는 상태에서 두 응답에서 더 나은 응답에 투표
  - 토너먼트형 대회와 유사
  - ELO 시스템 사용
    - 상대적 평가 : 자신보다 높은 점수의 상대를 이기면 더 많은 점수 획득
    - 승패 기록이 많아질수록 정확한 실력 측정 가능

## 2. Open AI API

---

## 1. 오픈 AI API 가용 모델

### ▶ 최신 모델

- [온라인 문서](#)에서 확인 가능

The screenshot shows the OpenAI Platform's 'Models' section. On the left, a sidebar menu includes 'GET STARTED' (Overview, Quickstart, Models), 'Models' (selected), Pricing, Changelog, Terms and policies, Cookbook, Forum, and Help. The main content area is titled 'Models' and features a heading 'Flagship models'. It lists three models: 'GPT-4o', 'GPT-4o mini', and 'o1 & o1-mini', each with a brief description, input/output capabilities, context length, and price information.

Model	Description	Input/Output	Context Length	Price
GPT-4o	Our versatile, high-intelligence flagship model	Text and image input, text output	128k context length	Smarter model, higher price per token
GPT-4o mini	Our fast, affordable small model for focused tasks	Text and image input, text output	128k context length	Faster model, lower price per token
o1 & o1-mini	Reasoning models that excel at complex, multi-step tasks	Text and image input, text output	128k context length	Uses additional tokens for reasoning

## 1. 오픈 AI API 가용 모델

### ▶ GPT 베이스 모델

- 인간 피드백을 통한 강화학습의 형태로 보완되지 못함
- 파인 투닝 등 관련 기능도 점점 중단함
- 베이스 모델의 종류
  - Babbage-002, davinci-002
- OpenAI는 GPT-4o 미니의 사용을 권장함

## 1. 오픈 AI API 가용 모델

### ▶ GPT-3.5

- gpt-3.5-turbo 모델 업데이트
  - gpt-3.5-turbo-1106 모델
  - gpt-3.5-turbo-0125 모델 (16,385토큰)
- 가격 정책
  - 1천 토큰당 \$0.0005, 출력 가격은 1천 토큰당 \$0.0015

## 1. 오픈 AI API 가용 모델

### ▶ GPT-4o

- gpt-4-0613 : 입력 컨텍스트 윈도우 8,192 토큰
- gpt-4-32k-0613 : 32,768 토큰의 입력 컨텍스트 윈도를 가진 같은 모델
  - 24,000 단어 또는 약 50p에 해당
- gpt-4o와 gpt-4o 미니
  - 128,000 토큰의 입력 컨텍스트와 비전 인식 지원
  - 텍스트와 이미지를 입력받아 API 텍스트 출력
  - gpt-4o-audio-preview 모델만 지원
  - 영어가 아닌 언어에서도 탁월한 성능 보임

## 2. 오픈 AI 파이썬 라이브러리

### ▶ API 키의 용도

- API 메서드 호출하는 권한 부여 & API 호출 사용료 청구 계정 추적

The screenshot shows the OpenAI API keys management interface. At the top left, there's a navigation bar with 'Personal' and 'Default project'. On the right, there are links for 'Playground', 'Dashboard', 'Docs', 'API reference', and a user profile icon. A sidebar on the left contains sections like 'SETTINGS' (Your profile), 'ORGANIZATION' (General), and 'PROJECT' (General, Cookbook, Forum, Help). The main content area is titled 'API keys' and includes a note about managing keys for the organization. It features a large button to 'Create an API key to access the OpenAI API' and a green button to '+ Create new secret key'.

## 2. 오픈 AI 파이썬 라이브러리

### ▶ Open AI API 사용

- 계정에 크레딧 추가
- 자동 충전, 사용 한도 설정, 사용에 따른 알림 등 상세한 설정
- 그 외 내용은 [관련 페이지 참조](#)
- 계정에 조직 연동 가능
  - [Organization settings](#)
  - [멤버 초대](#)

## 2. 오픈 AI 파이썬 라이브러리

### ▶ 키값을 환경 변수로 설정 (블로그 링크 필요)

- Linux & MacOS
- Windows

## 2. 오픈 AI 파이썬 라이브러리

▶ 키값을 환경 변수로 설정 (블로그 링크 필요)

- Google Colab & Jupyter Notebook

### 3. 채팅 완성 엔드포인트의 입력 옵션

▶ **create()** 메서드 필수 매개변수

필드 이름	유형	설명
Messages	배열	A list of messages comprising the conversation so far. Depending on the <a href="#">model</a> you use, different message types (modalities) are supported, like <a href="#">text</a> , <a href="#">images</a> , and <a href="#">audio</a> .
model	문자열	ID of the model to use. See the <a href="#">model endpoint compatibility</a> table for details on which models work with the Chat API.

- [그 외 부분 입력 매개변수](#)

### 3. 채팅 완성 엔드포인트의 입력 옵션

#### ▶ 대화 흐름

- 시스템 메시지
  - 어시스턴트의 응답 설정 시 도움
  - 대화 전반에 방향성 부여 시, 시스템 메시지 통해 설정
- 사용자 메시지
  - ChatGPT 인터페이스에서 질문이나 문장 입력
- 어시스턴트 메시지의 두 가지 역할
  - 하나는 대화를 이어가기 위해 이전 응답 저장
  - 원하는 행동의 예시 제시하는 지침으로 설정

### 3. 채팅 완성 엔드포인트의 입력 옵션

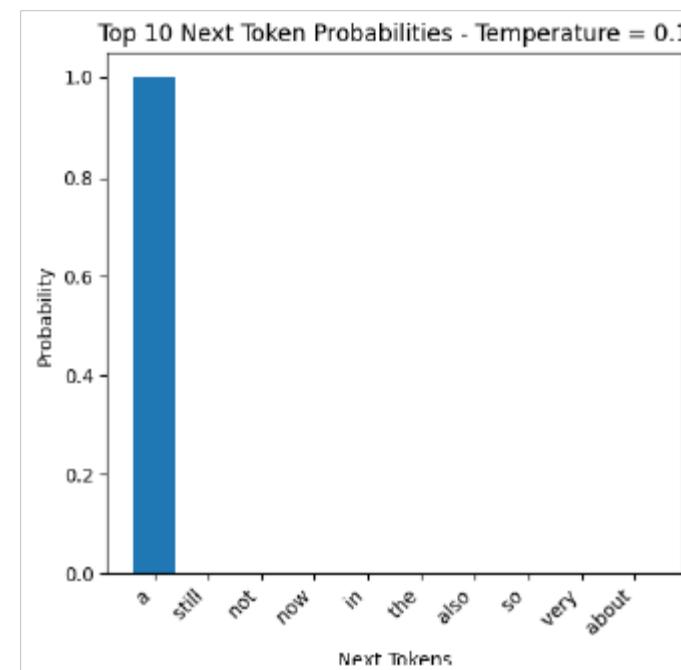
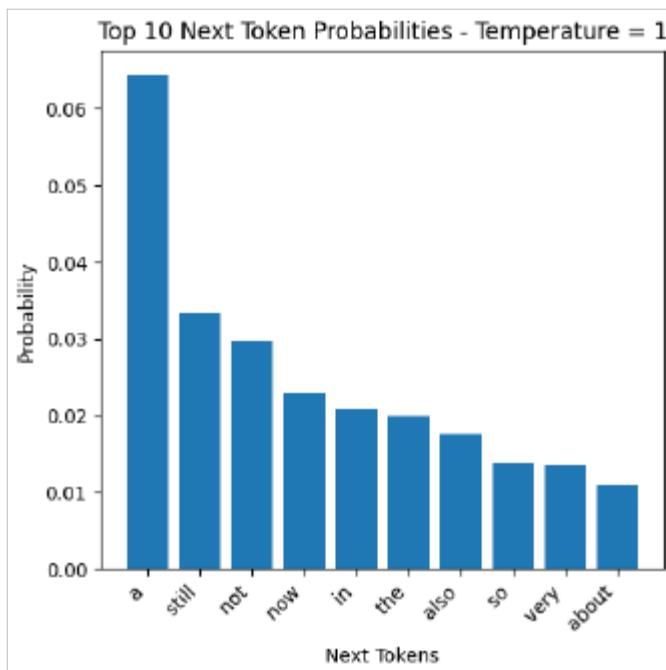
#### ▶ 대화 길이와 토큰 수

- 사용료 : 토큰당 사용료 책정
- 시간 : 토큰이 많을수록 응답 시간 길어짐, 응답에 분 단위로 소요
- 한도 : 토큰 총수는 모델의 최대한도보다 적어야 함
- 대화의 길이 관리 필수 → 토큰화 라이브러리 [tiktoken](#) 제공

### 3. 채팅 완성 엔드포인트의 입력 옵션

#### ▶ 기타 매개변수

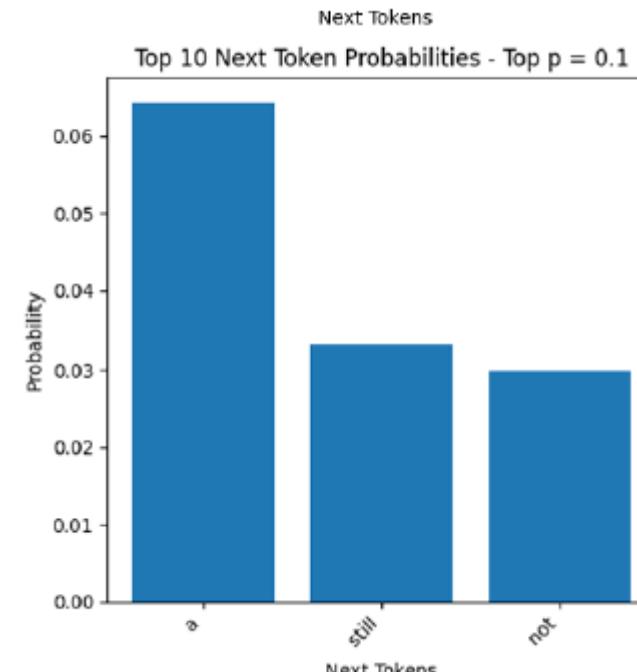
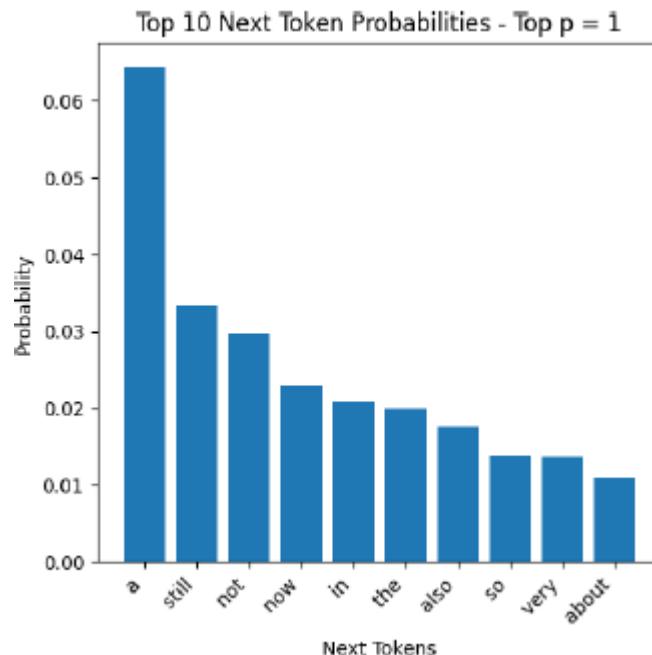
- temperature & top\_p : 생성된 텍스트의 일관성<sup>coherence</sup>과 변동성<sup>creativity</sup> 사이의 균형 조정
  - temperature : 확률 분포 자체 변화, 값이 클수록 낮은 확률의 토큰 선택될 가능성도 커짐



### 3. 채팅 완성 엔드포인트의 입력 옵션

#### ▶ 기타 매개변수

- temperature & top\_p : 생성된 텍스트의 일관성<sup>coherence</sup>과 변동성<sup>creativity</sup> 사이의 균형 조정
  - top-p : 뉴클리어스 샘플링<sup>nucleus sampling</sup> top\_p 값에 해당하는 부분 집합만을 고려
  - 그림 예시 : top\_p = 0.1, temperature=1 설정 → 확률 합이 0.1을 넘는 a, still, not의 토큰만 유지



### 3. 채팅 완성 엔드포인트의 입력 옵션

#### ▶ 기타 매개변수

- temperature & top\_p : 생성된 텍스트의 일관성<sup>coherence</sup>과 변동성<sup>creativity</sup> 사이의 균형 조정
  - 내용과 스타일이 일관적 : temperature와 top\_p 값을 낮게 선택
  - 객관적 사실이 중요한 출력 : temperature 높게, top\_p 값을 낮게 선택
  - 창의성이 필요한 출력 : temperature와 top\_p 값을 높게 선택

### 3. 채팅 완성 엔드포인트의 입력 옵션

#### ▶ ChatCompletion 모델 구조 (JSON 호환)

- 기본 정보
- 응답 내용
- 토큰 사용량
- 시스템 정보

## 4. 비전

### ▶ 멀티모달 LLM

- 텍스트 외의 다른 형식의 데이터 처리
- GPT-4o 계열의 모델을 통해 이미지 인식 기능 지원
- 파일 지원
  - PNG, JPEG, WEBP, GIF 형식 지원
  - 이미지당 최대 크기 20MB
- 모델에 이미지 제공하는 방법 두가지
  - 이미지를 가리키는 링크 전달,
  - base64로 인코딩된 이미지를 직접 요청에 전달

## 4. 비전

### ▶ 멀티모달 LLM

- 이미지 처리 사용료 : 토큰 기반
- 예시
  - 150px 정사각형 이미지 → 255개 토큰 사용
  - 2048px × 1024px → 1,105개 토큰 사용

## 5. JSON 출력

### ▶ JSON 객체 출력 방식

- `response_format` 매개변수 활용 `{'type' : 'json_object'}`
- 함수 활용 → Agent의 개념에 가까움
  - 현재는 사용자 정의 함수를 만들어서 전달하는 형태

## 6. 텍스트 완성 모델

### ▶ 텍스트 완성 엔드포인트 위한 입력 모델

- 주요 매개변수 : model, prompt, max\_tokens, suffix
  - 입출력 토큰 길이에 따라 비용 다름
  - 줄 바꿈, 문장 부호 등 접미사에 신경 써야 함
  - 출력값의 경우 max\_tokens 낮게 설정해 의도치 않게 과도한 비용이 청구되는 일 방지

## 7. 사용료

- ▶ 사용료는 계속 변동할 가능성 있음
  - 웹 페이지 : <https://openai.com/chatgpt/pricing/>

## 8. 기타 오픈AI API 및 기능

### ▶ 파운데이션 모델과 관련된 다른 기능도 제공

- 참고 : <https://platform.openai.com/docs/api-reference/introduction>
- 주요 기능
  - 임베딩
  - 모데레이션 모델
  - 텍스트 음성 변환
  - 음성인식
  - 이미지 모델 API

## 8. 기타 오픈AI API 및 기능

### ▶ 임베딩 Embedding

- 참고 : <https://platform.openai.com/docs/api-reference/embeddings>
- 수치가 아닌 값을 수치형인 벡터로 개념화 하는 것
- 임베딩된 데이터
  - 검색 : 쿼리 문자열과 관련성을 기준으로 결과 정렬
  - 추천 : 쿼리 문자열과 관련된 텍스트 문자열이 포함된 문서 추천
  - 군집화 : 유사도별로 문자열 그룹화
  - 이상 탐지 : 다른 문자열과 관련이 없는 텍스트 문자열
- 검색 증강 생성(RAG) 시스템

## 8. 기타 오픈AI API 및 기능

### ▶ 모더레이션 모델

- OpenAI 사용 정책 준수 : <https://openai.com/policies/usage-policies/>
- 다음 범주에서 영문으로 된 컨텐츠 검사
  - 차별, 혐오, 자해, 성적인 묘사, 미성년자와의 성적 묘사, 폭력적이거나 잔인한 컨텐츠

## 8. 기타 오픈AI API 및 기능

### ▶ 텍스트 음성 변환 : <https://platform.openai.com/docs/guides/text-to-speech>

- 애플리케이션이 음성인식 기능 지원
- OpenAI는 텍스트를 음성으로 변환하는 TTS(Text-To-Speech) 모델과 함께 오디오 API 제공
  - 가상 비서와 챗봇
  - 접근성 향상
- OpenAI의 TTS 웹 페이지에서 목소리 샘플 청취 가능
- 영어 이외에 다양한 언어 제공 (한국어 포함)
- mp3외에 opus, aac, flac 오디오 포맷 지원

## 8. 기타 오픈AI API 및 기능

▶ 음성인식 : <https://github.com/openai/whisper>

- 음성인식에 매우 유용한 모델

## 8. 기타 오픈AI API 및 기능

### ▶ 이미지 모델 API

- 텍스트 통해 이미지 처리하는 세 가지 방법
  - 생성 : 텍스트 입력으로 원하는 이미지 생성 (DALL-E 2 & DALL-E 3)
  - 편집 : 텍스트 입력으로 기존 이미지 편집 (DALL-E 2 지원)
  - 변형 : 기존 이미지 변형해 원본의 컨셉 유지하면서 창의적인 이미지 생성
- C2PA in DALL-E 3 <https://help.openai.com/en/articles/8912793-c2pa-in-dall-e-3>
  - 미디어 콘텐츠의 출처와 히스토리 검증하는 기술적 프로토콜 확립해 인터넷에서 허위 정보 확산 막기

### 3. LLM 기반 애플리케이션 개발

---

## 1. API 키 관리

### ▶ API 키 사용하는 애플리케이션 설계 방법 2가지

- 사용자가 직접 API 키 입력하도록 설계
- 애플리케이션이 고유한 API 키 사용하도록 설계

## 1. API 키 관리

### ▶ API 키 사용하는 애플리케이션 설계 방법 2가지

- 사용자가 직접 API 키 입력하도록 설계
  - (1) 사용자에게 필요할 때만 키 제공 요청, 원격 서버에 키 저장 및 사용하지 않도록 하기
  - (2) 데이터베이스에 키 저장해 관리
- (1)의 경우
  - 애플리케이션 시작 시, 사용자에게 키 요청 또는 처음 사용자가 입력한 키를 로컬 환경 변수로 저장
- (2)의 경우
  - 키가 외부로 전송되어 서버에 저장 → 보안 관점에서 관리 영역 ↑ 유출 위험 ↑
  - 웹 애플리케이션에서는 키를 브라우저 저장소가 아닌 메모리에 보관
  - 전송중 또는 미사용 중인 키는 항상 암호화

## 1. API 키 관리

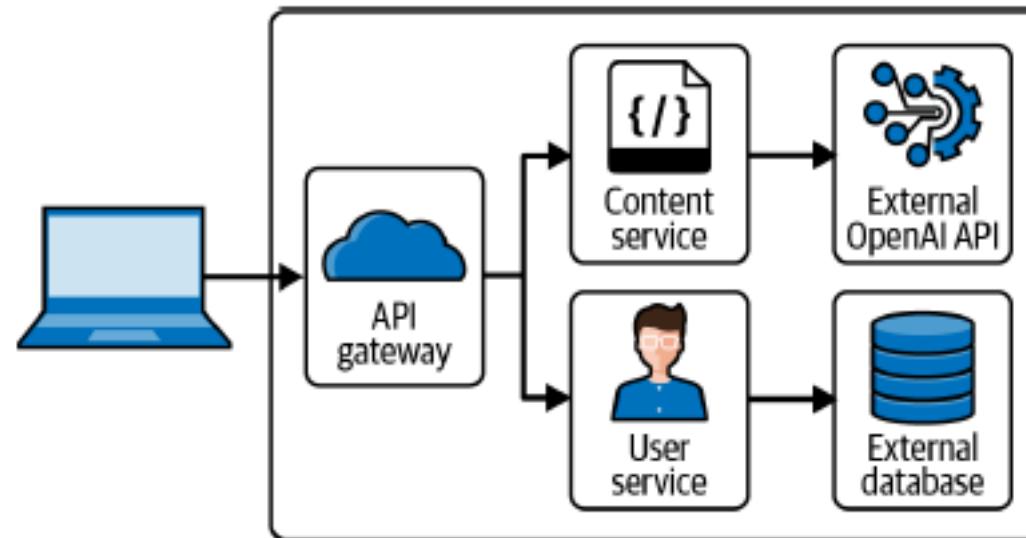
### ▶ API 키 사용하는 애플리케이션 설계 방법 2가지

- 애플리케이션이 고유한 API 키 사용하도록 설계
  - API 키를 코드에 직접 작성하지 않음
  - API 키를 애플리케이션의 소스 트리에 미 저장
  - 사용자의 브라우저나 개인 디바이스에서 API 키에 접근할 수 없도록 설정
  - 사용량 제한 설정하여 예산 관리
  - API 키를 정기적으로 갱신

## 2. 아키텍처 디자인 패턴

### ▶ OpenAI API에 의존적이지 않은 방식으로 구축하는 편이 좋음

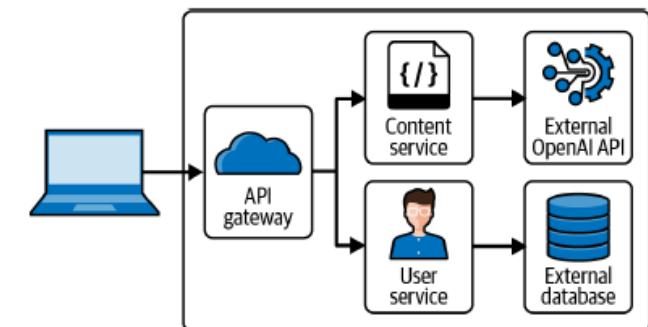
- API가 변경되어도 애플리케이션을 완전히 다시 작성해야 하는 일이 없도록 코드 작성
- 소프트웨어 아키텍처 디자인 패턴



## 2. 아키텍처 디자인 패턴

### ▶ OpenAI API에 의존적이지 않은 방식으로 구축하는 편이 좋음

- API가 변경되어도 애플리케이션을 완전히 다시 작성해야 하는 일이 없도록 코드 작성
- 소프트웨어 아키텍처 디자인 패턴
  - API 게이트웨이 : 사용자의 브라우저에서 전송한 요청 관리
  - 사용자 서비스 : 사용자 관리하는 서비스로 데이터베이스에 접근
  - 콘텐츠 서비스 : 오픈 AI, API를 호출해 콘텐츠 생성 및 처리와 관련된 작업 실행



### 3. LLM 기반 애플리케이션의 능력

#### ▶ 기존 애플리케이션에 GPT 모델같은 LLM 결합

- 대화능력
- 언어 처리 능력
- 인간-컴퓨터 상호작용 능력
- 능력 결합

### 3. LLM 기반 애플리케이션의 능력

#### ▶ 기존 애플리케이션에 GPT 모델같은 LLM 결합 프로젝트 예시

- 이메일
- 계약서 및 공식 문서
- 창의적인 글쓰기
- 단계별 수행 계획
- 브레인스토밍
- 광고
- 직무요건 설명

### 3. LLM 기반 애플리케이션의 능력

#### ▶ 기존 애플리케이션에 GPT 모델같은 LLM 결합

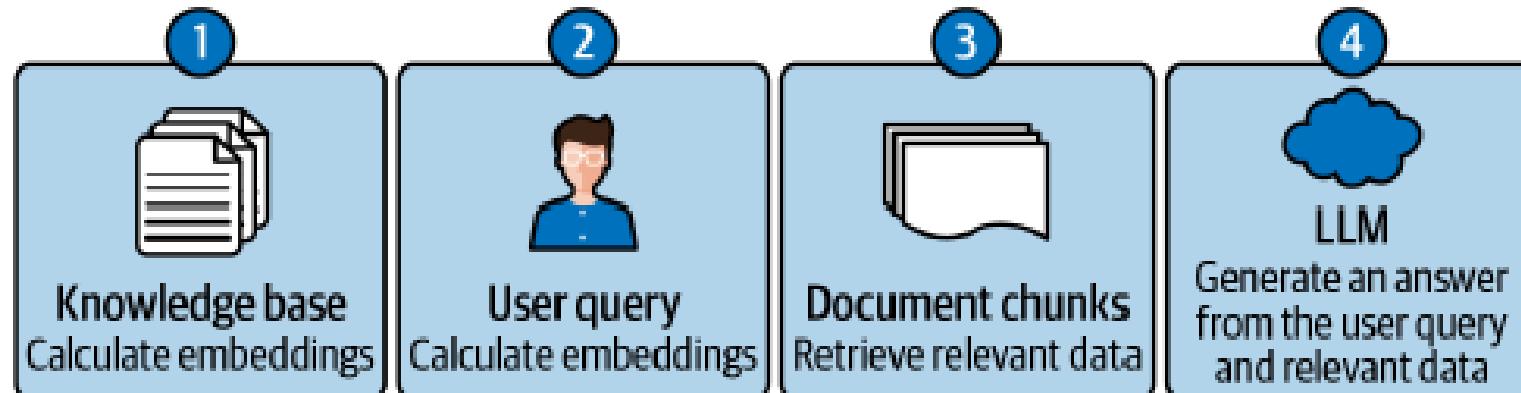
- 대화능력

- 프롬프트 엔지니어링 : 챗봇이 주목적임을 상기
- 가드레일 : 할루시네이션과 프롬프트 인젝션 제어
- 비용 : API 키의 사용 제어

### 3. LLM 기반 애플리케이션의 능력

#### ▶ 기존 애플리케이션에 GPT 모델같은 LLM 결합

- 언어 처리 능력
  - 사용자는 내부에서 작동하는 LLM의 존재 알지 못함
  - 검색 증강 생성(RAG) 활용



### 3. LLM 기반 애플리케이션의 능력

#### ▶ 기존 애플리케이션에 GPT 모델같은 LLM 결합

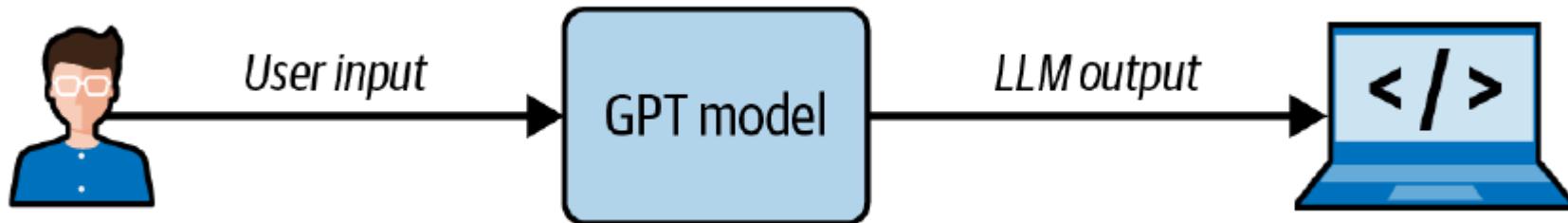
- 언어 처리 능력
  - 사용자는 내부에서 작동하는 LLM의 존재 알지 못함
  - 검색 증강 생성(RAG) 활용
    - 지식 베이스의 임베딩 생성
    - 질문이나 키워드의 임베딩 생성
    - 임베디드 지식 베이스에 쿼리 임베딩해 관련 데이터 조회
    - LLM 사용해 의미 검색에서 반환된 관련 데이터 분석해 정확한 답변 작성

### 3. LLM 기반 애플리케이션의 능력

#### ▶ 기존 애플리케이션에 GPT 모델같은 LLM 결합

- 인간-컴퓨터 상호작용 능력

- 첫번째 혁명 : 마우스와 그래픽 사용자 인터페이스 (1970년대)
- 두번째 혁명 : LLM, 자연어를 사용해 컴퓨터 시스템과 상호작용



### 3. LLM 기반 애플리케이션의 능력

#### ▶ 기존 애플리케이션에 GPT 모델같은 LLM 결합

- 능력 결합
  - 대화와 자연어 처리
  - 대화와 인간-컴퓨터 인터페이스

### 3. LLM 기반 애플리케이션의 능력

#### ▶ 기존 애플리케이션에 GPT 모델같은 LLM 결합 프로젝트 예시

- 뉴스 생성
- 유튜브 동영상 요약
- 챗봇
- 개인 어시스턴트
- 문서 정리
- 감정 분석

### 3. LLM 기반 애플리케이션의 능력

#### ▶ 기존 애플리케이션에 GPT 모델같은 LLM 결합 프로젝트 예시

- 뉴스 생성
  - 객관적 사실 입력 후 뉴스 기사 생성하는 도구 만들기
  - 뉴스 기사의 길이, 어조, 문체는 대상 매체와 청중에 따라 선택

### 3. LLM 기반 애플리케이션의 능력

#### ▶ 기존 애플리케이션에 GPT 모델같은 LLM 결합 프로젝트 예시

- 텍스트 요약

- 미디어 모니터링 : 방대한 정보에서 빠르게 주요 내용 확인
- 트렌드 탐색 : 기술 뉴스나 학술 논문 요약해 유용한 정보 얻기
- 고객 지원 : 고객이 이해하기 편하도록 상세 매뉴얼의 요약본 제공
- 이메일 축약 : 이메일에서 핵심을 빠르게 파악해 과도한 이메일 관련 업무 줄이

- 유튜브 동영상 요약

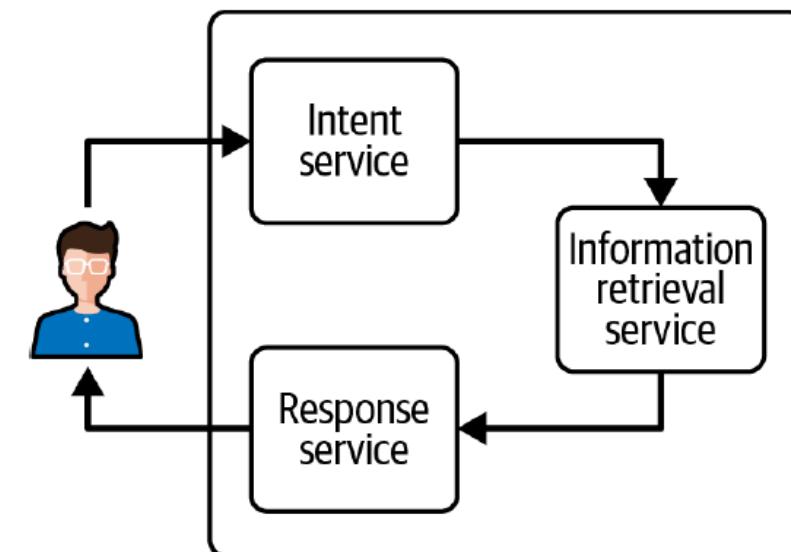
- 스크립트로 요약 : 동영상에서 스크립트 추출 후, GPT 모델 이용해 요약
- 비전 기능 활용한 요약 : GPT-4o의 비전 기능 사용해 동영상의 프레임 분석
- 위 두가지 방법 결합

### 3. LLM 기반 애플리케이션의 능력

#### ▶ 기존 애플리케이션에 GPT 모델같은 LLM 결합 프로젝트 예시

- 챗봇

- 비공개 데이터나 ChatGPT가 학습에 사용하지 않은 데이터에 관한 질문에 답변 유도
- 가이드 PDF 파일 활용
- PDF 파일이 매우 커서 RAG 사용 필요



### 3. LLM 기반 애플리케이션의 능력

#### ▶ 기존 애플리케이션에 GPT 모델같은 LLM 결합 프로젝트 예시

- 챗봇 - 3가지 구성 요소
  - 의도 분류 서비스
    - 사용자의 질문이 데이터 관련 or 잡담인지 식별
    - 적절한 데이터 소스 선택, OpenAI 정책 위반 또는 민감 정보 포함 여부 감지
  - 정보 검색 서비스
    - 의도 분석 결과 기반으로 적절한 정보 검색
    - 사용자 질문과 데이터의 임베딩 비교 후 벡터 스토어에서 검색
  - 답변생성 서비스
    - 검색된 정보 바탕으로 OpenAI 모델 활용해 응답 생성

### 3. LLM 기반 애플리케이션의 능력

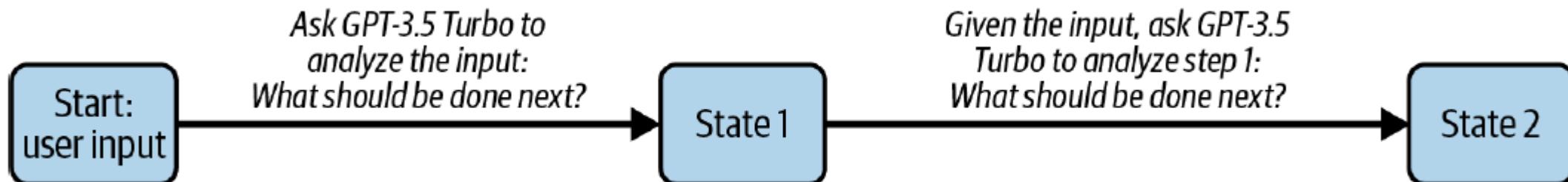
#### ▶ 기존 애플리케이션에 GPT 모델같은 LLM 결합 프로젝트 예시

- 챗봇 – 레디스 (메시지 브로커)
  - 레디스 : <https://redis.io/>
  - 인메모리 키-값 데이터베이스
  - 벡터 저장 기능과 벡터 유사도 검색 솔루션 두 가지 기본 제공 기능 사용
  - 관련 문서 : <https://rb.gy/5sr69a>

### 3. LLM 기반 애플리케이션의 능력

#### ▶ 기존 애플리케이션에 GPT 모델같은 LLM 결합 프로젝트 예시

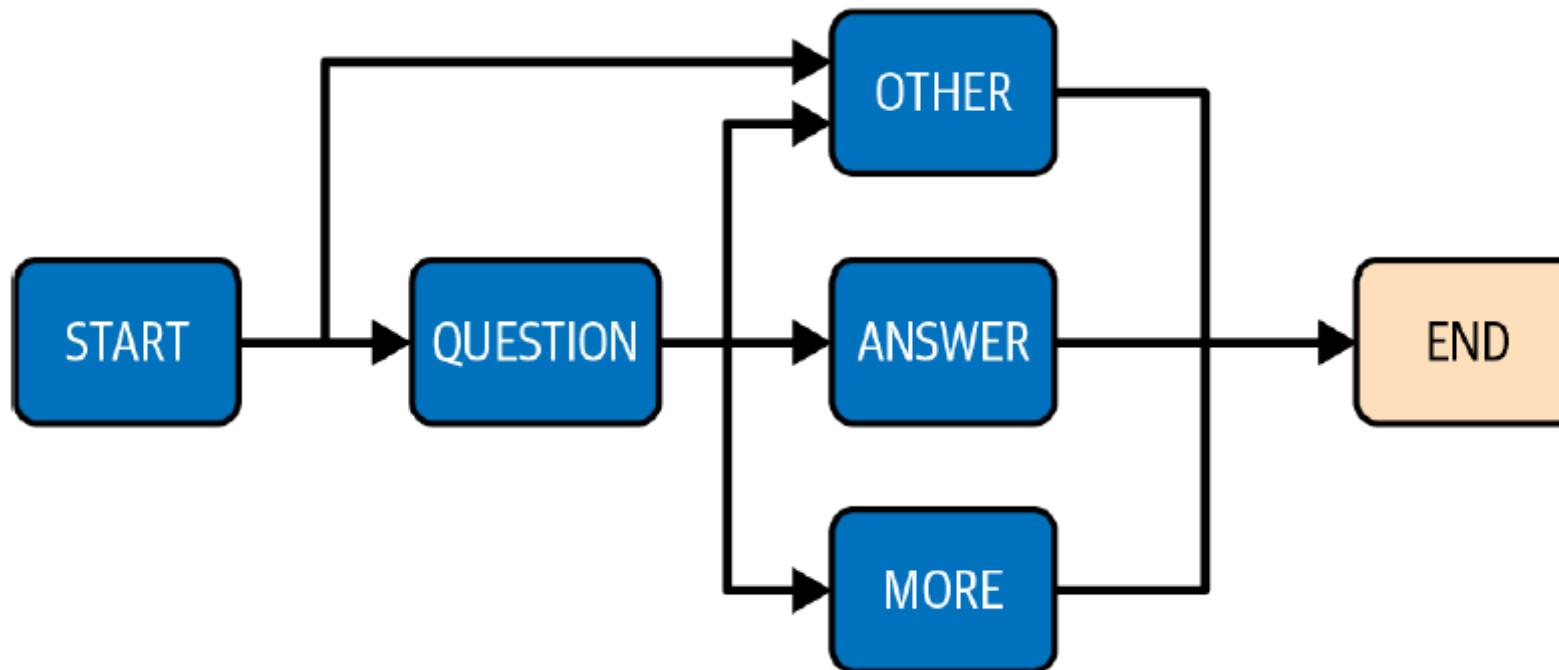
- 개인 어시스턴트
  - LLM의 기능 사용해 사용자가 무엇이든 요청할 수 있는 인터페이스 제공
- 단계별로 살펴보기
  - 시작 : 사용자의 입력에 대해 답변이 필요함 감지 → 다음 상태1를 Question으로 설정
  - 상태 1 : 사용자의 입력이 질문임 확인, GPT 모델에 답변 요구 → 다음 상태2를 Answer로 설정
  - 상태 2 : 사용자에게 답변



### 3. LLM 기반 애플리케이션의 능력

#### ▶ 기존 애플리케이션에 GPT 모델같은 LLM 결합 프로젝트 예시

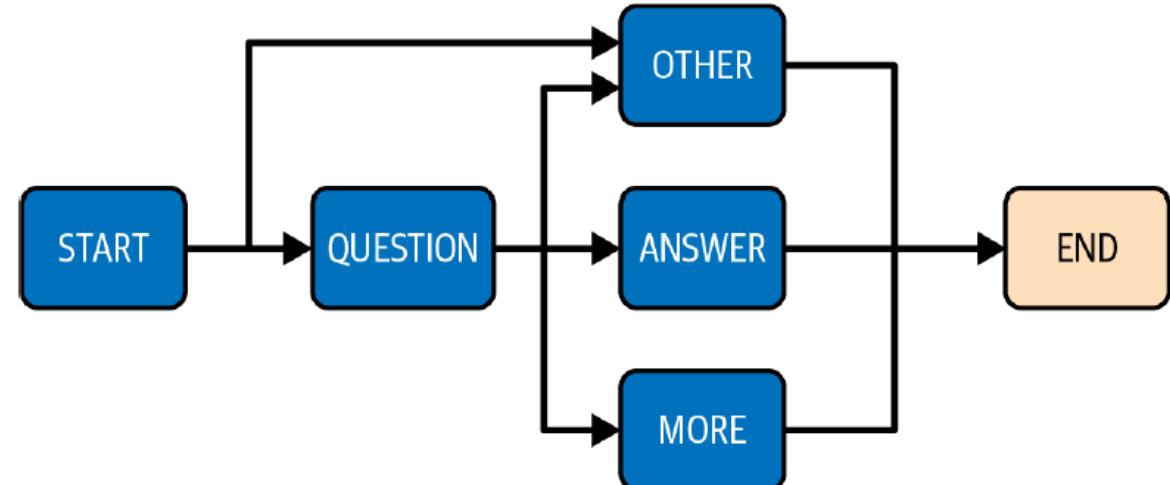
- 개인 어시스턴트 : 질문에 답변하도록 네 가지 상태 정의



### 3. LLM 기반 애플리케이션의 능력

#### ▶ 기존 애플리케이션에 GPT 모델같은 LLM 결합 프로젝트 예시

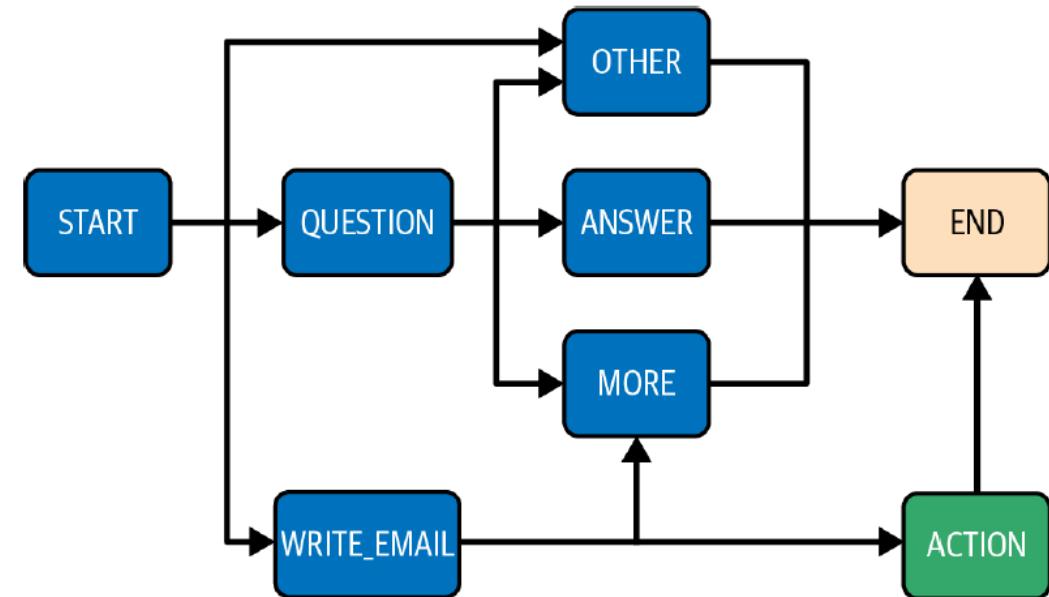
- 개인 어시스턴트 : 질문에 답변하도록 네 가지 상태 정의
  - QUESTION : 사용자의 질문 감지
  - ANSWER : 질문에 답
  - MORE : 자세한 정보 필요
  - OTHER : 질문에 답변할 수 없음



### 3. LLM 기반 애플리케이션의 능력

#### ▶ 기존 애플리케이션에 GPT 모델같은 LLM 결합 프로젝트 예시

- 개인 어시스턴트 : 질문에 답변하도록 네 가지 상태 정의
  - QUESTION : 사용자의 질문 감지
  - ANSWER : 질문에 답
  - MORE : 자세한 정보 필요
  - OTHER : 질문에 답변할 수 없음
  - WRITE\_EMAIL : 이메일 작성 감지



### 3. LLM 기반 애플리케이션의 능력

#### ▶ 기존 애플리케이션에 GPT 모델같은 LLM 결합 프로젝트 예시

- 위스퍼로 음성-텍스트 변환
  - 위스퍼 라이브러리 설치 : <https://github.com/openai/whisper>
  - ffmpeg 설치 필요 (사전에 윈도우는 Chocolatey 설치 필요 / 직접 파일 다운로드 받아서 환경변수 설정)

## 4. 문서 정리

- ▶ 다른 도구에서 쉽게 파싱할 수 있는 JSON 형식으로 결과 출력 권장
- ▶ 다른 유형의 파일에도 확장
  - 음성 : 음성-텍스트 API 사용해 음성 파일을 텍스트로 변환해 일반 텍스트 파일과 같은 방식 처리
  - 이미지 : GPT-4o의 다중 모달 기능 활용해 이미지 처리
  - 동영상 : 동영상 파일에서 프레임 추출해 이미지처럼 처리

## 5. 감정 분석

### ▶ 매개변수 logprobs 활용

- 토큰에 대한 모델의 신뢰도 파악 및 여러 용도로 사용
- 텍스트의 감정이 긍정적인지 부정적인지 분석, 감정별 확률 확인

## 6. 비용 관리

### ▶ OpenAI 도입 전

- 내가 만들고자 하는 기능이 프로젝트에 실질적으로 도움이 되는가?
- OpenAI API가 가장 적합한 해결책인가?
- 다른 도구나 설계로 비슷한 결과를 낼 수 있는가?
  - 정규표현식, 규칙 기반 파싱, 간단한 키워드 검색도 여전히 유효함 (무료)
  - XGBoost, LightGBM과 같은 알고리즘
  - 허깅페이스
  - GPT-4o 대신 GPT-4o mini로도 충분히 좋은 결과를 낼 수 있는가?
  - 컨텍스트 크기를 줄일 수 있는가?
  - RAG 디자인 사용 가능한가?

## 6. 비용 관리

### ▶ API 활용 시, 비용 관리 방법

- 메시지 길이 제한
- 대화 길이 제한
- 짧은 메시지 권장
- 짧은 대화
- 대화 요약
- 전통적인 소프트웨어 설계 기법
- 프롬프트 압축 알고리즘 (<https://github.com/microsoft/LLMLingua>)

## 7. 외부 API와 작업

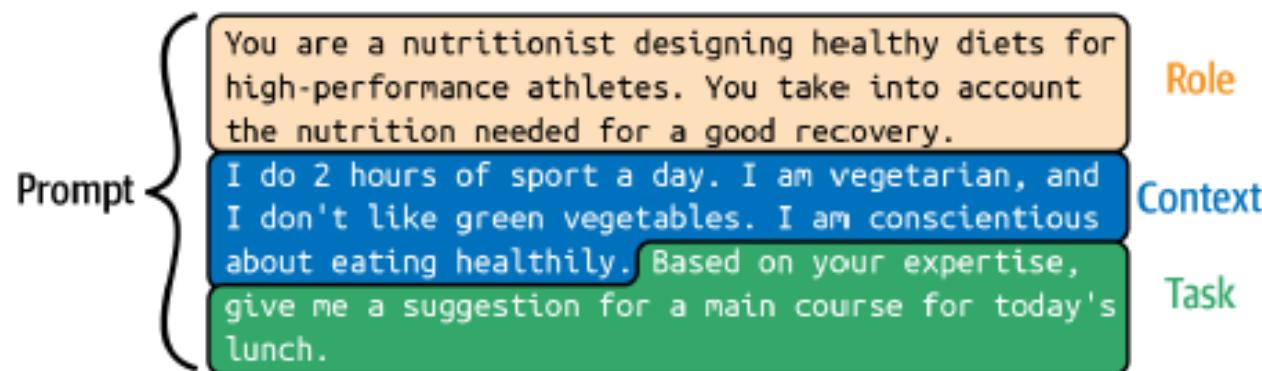
- ▶ 요청 제한 필수
- ▶ LLM 생성 오래 걸릴 시, 응답성 높이는 것이 프로젝트에 난관으로 작용
- ▶ 에러 처리
  - try-catch문 활용
  - 지수 백오프 전략 (재시도 처리 표준 방법) : backoff 라이브러리, tenacity 라이브러리 활용
  - 서킷 브레이커 패턴 (<https://martinfowler.com/bliki/CircuitBreaker.html>)
- ▶ 응답성과 사용자 경험 향상
  - 답변을 사용자에게 스트림으로 전달(stream 옵션 True)
  - 비동기 프로그래밍 (asyncio) / 교재 : <https://www.yes24.com/Product/Goods/99474986>

## 4. GPT-4o 및 ChatGPT 활용 고급 기법

---

## 1. 프롬프트 엔지니어링

- ▶ LLM이 최대한 적합한 출력 생성할 수 있도록, 최적의 입력 형태와 사례 개발
- ▶ 프롬프트 엔지니어링에 관한 가이드
  - 참고 : <https://platform.openai.com/docs/guides/prompt-engineering>
- ▶ 효과적인 프롬프트 설계 (예시 : <https://platform.openai.com/examples>)
  - 역할, 컨텍스트, 작업이라는 세 가지 요소 정의



## 1. 프롬프트 엔지니어링

### ▶ 다양한 테스크

- 문법 교정
- 어린이 대상 요약
- 코드 설명
- 코드 시간 복잡도 계산
- 파이썬 버그 수정
- SQL 작성
- 노트 요약
- 스프레드시트 생성

## 1. 프롬프트 엔지니어링

### ▶ 유의점

- GPT는 연산에는 적합하지 않음



## 1. 프롬프트 엔지니어링

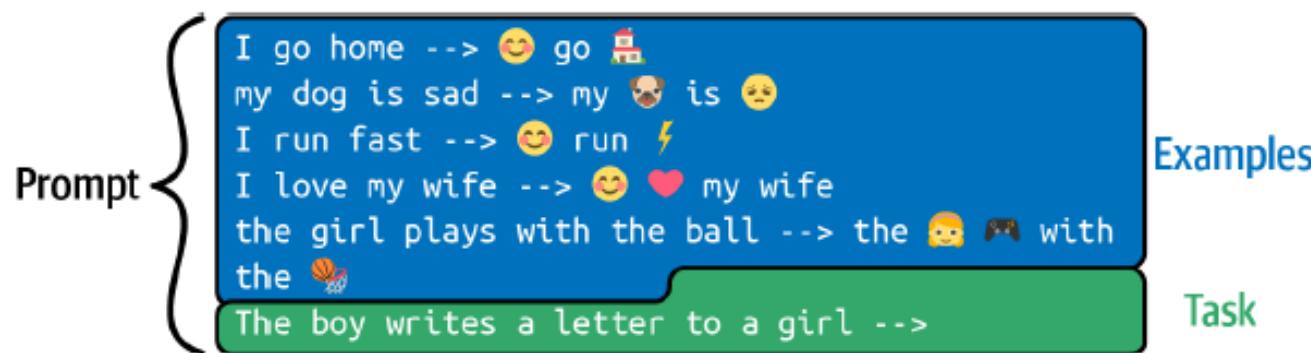
### ▶ 해결방안

- 프롬프트 끝에 ‘단계별로 생각하세요’ 지시 추가 시, 더 복잡한 추론 문제 해결
  - 제로샷-CoT 추론 (<https://arxiv.org/pdf/2205.11916>)

## 1. 프롬프트 엔지니어링

### ▶ 퓨샷 러닝 구현

- LLM의 퓨샷 러닝, 몇 개의 예시만으로 일반화해 가치 있는 결과 도출 Few-Shot 러닝 소개
  - 논문 : <https://arxiv.org/pdf/2005.14165>



## 1. 프롬프트 엔지니어링

### ▶ 퓨샷 러닝 구현

- LLM의 퓨샷 러닝, 몇 개의 예시만으로 일반화해 가치 있는 결과 도출 <sup>Few-Shot</sup> 러닝 소개
  - 논문 : <https://arxiv.org/pdf/2005.14165>
  - 주어진 예시의 패턴을 고려해 완성 연산 수행
- 컨텍스트가 명확하고 관련성이 있는지 확인

## 1. 프롬프트 엔지니어링

### ▶ 원샷 러닝 구현

- 모델이 작업을 실행하는 데 도움이 되는 예시 단 하나만 제공
- 간단한 작업 또는 LLM이 충분한 배경지식을 가진 작업을 할 때 효과적
- API 비용 저렴하다는 장점
- 복잡한 작업 또는 정확한 결과 보장 원할 시, 퓨샷 러닝이 더 적합한 편

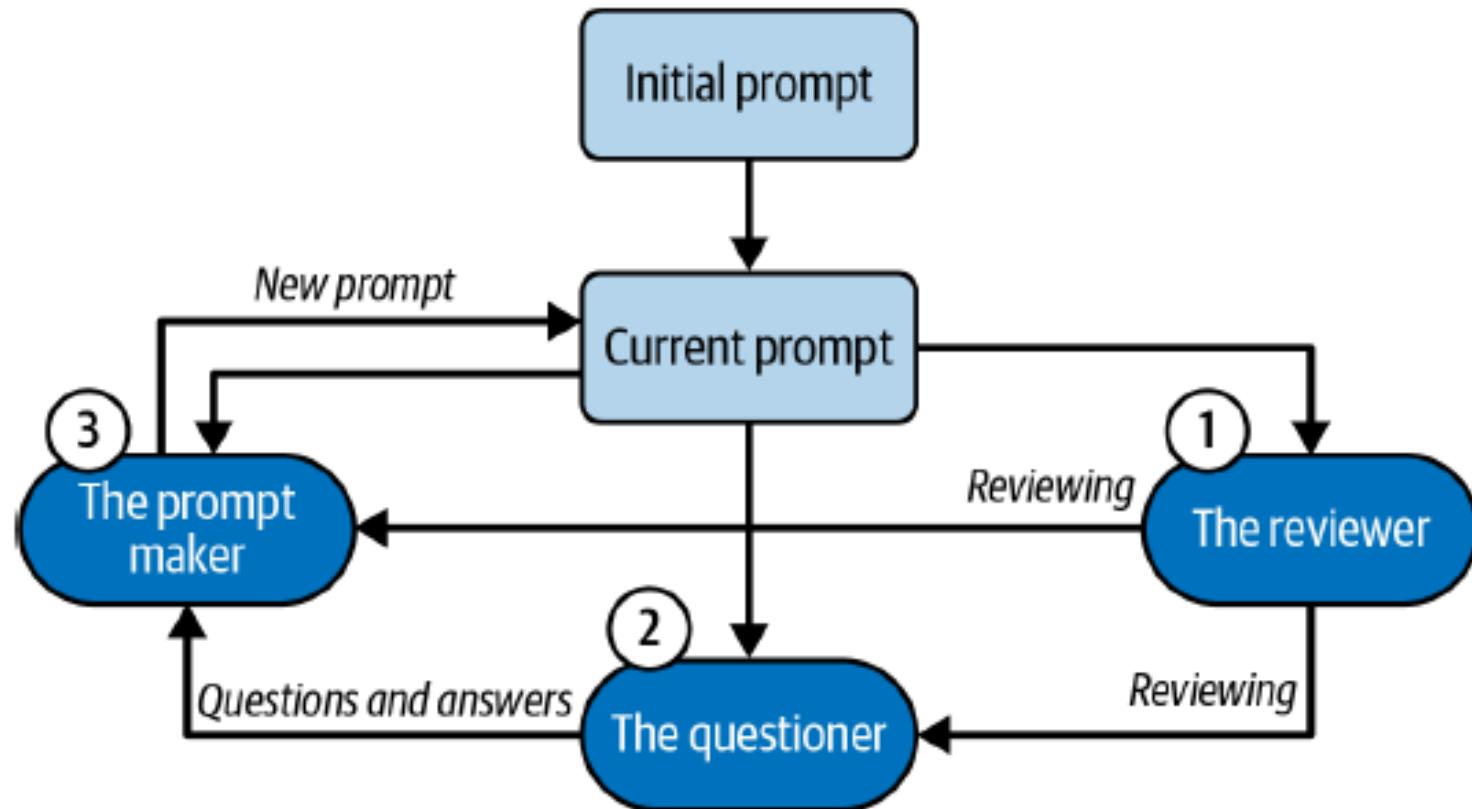
## 1. 프롬프트 엔지니어링

### ▶ 사용자 피드백을 통한 반복적 개선

- 초기 입력 프롬프트를 조금씩 개선해 나가는 방식
- LLM은 기존 프롬프트를 다시 작성해 더 나은 프롬프트 만듬
- LLM에 프롬프트 최적화를 적용해 모델의 성능 크게 향상
- 프롬프터의 구성
  - 검토자, 질문자, 제작자 등 세 가지 에이전트로 구성됨

## 1. 프롬프트 엔지니어링

### ▶ 사용자 피드백을 통한 반복적 개선



## 1. 프롬프트 엔지니어링

### ▶ 프롬프트 개선

- 프롬프트 종료 시, 모델이 질문을 이해했는지 물어보고, 이해하지 못했다면 모르는 부분 질문하도록 지시
- 출력 서식 지정 (JSON), 지침 반복하기
- 네거티브 프롬프트 사용
  - 출력에 표시하고 싶지 않은 내용 지정해 응답 조정하는 방법
  - 특정 유형의 응답을 걸러내는 제약 조건 또는 가이드라인 역할
- 길이 제한하기
- **프롬프트 체이닝**<sup>prompt chaining</sup> : 하나의 작업을 여러 개의 하위 작업으로 나누어 달라고 요청
- **섀도 프롬프트**
  - 작업을 명시적으로 밝히지 않는 대신 프롬프트에 힌트 포함해 모델이 원하는 결과 도출하도록 유도

## 1. 프롬프트 엔지니어링

### ▶ 프롬프트 관리 도구

- promptfoo : <https://www.promptfoo.dev/>
  - 테스트 주도 LLM 개발 위한 CLI
- DSPy : <https://github.com/stanfordnlp/dspy>
  - 개발자가 프롬프트 대신 코드로 LLM과 상호작용하도록 프롬프트 최적화하는 과정

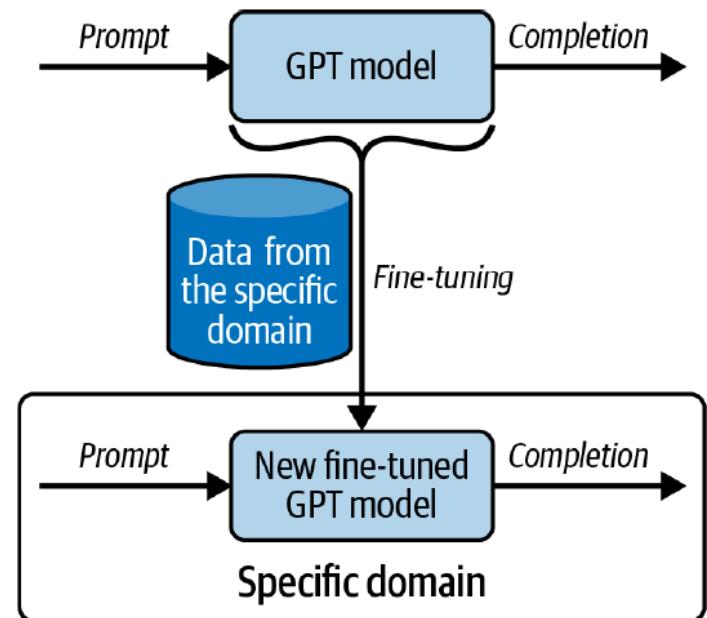
## 2. 파인 투닝

- ▶ GPT 모델은 폭넓은 작업에서 탁월한 성능 발휘
- ▶ 파인 투닝 사용 시기
  - LLM의 어조와 스타일 조정
  - LLM이 가진 도메인별 지식에 집중하도록 유도
  - 신뢰성 향상 및 할루시네이션 감소
  - 프롬프트에서 묘사하기 어려운 복잡한 작업 수행
  - 모델의 출력 형식을 변경(예: 자연어에서 JSON)
- ▶ 프롬프트 엔지니어링보다 더 복잡하고 비용이 많이 듬

## 2. 파인 투닝

### ▶ 시나리오 : 회사에서 사용할 이메일 응답 생성기

- 특정 어휘 사용하는 업계의 경우 이메일 응답에도 기존 문체를 유지 필요
- 생성 전략
  - 프롬프트 엔지니어링 기법 : 모델이 원하는 텍스트 출력하도록 강제
  - 기존 모델 파인 투닝



## 2. 파인 투닝

### ▶ 도메인별 요구 사항에 따라 GPT 기본 모델 조정

- 파인 투닝 리스트에서 모델을 정기적으로 추가하고 제거
- (2025) 파인 투닝에 사용할 수 있는 최신 모델
  - GPT-4o 제품군 모델

## 2. 파인 투닝

### ▶ 파인 투닝 vs 퓨샷 러닝 비교

비교 항목	파인 투닝(fine-tuning) 미세 조정	퓨샷 러닝(Few-Shot Learning)
학습 방식	모델을 추가로 학습하여 최적화	기존 모델을 그대로 사용, 추가학습 없음
데이터 필요량	수백~수백만 개의 예제(작업 난이도에 따라 증가)	5~10개 예제(간단한 작업)
유연성	특정 작업에 맞게 최적화됨	다양한 작업에 빠르게 적용 가능
비용	데이터 준비 및 학습 비용이 큼	비용이 낮고 효율적
적용 사례	특정 도메인에 맞춘 모델 개선	빠른 프로토타이핑, 실험
한계점	고품질 데이터가 필요하며 비용 부담이 큼	성능이 기존 모델에 의존

## 2. 파인 투닝

### ▶ 파인 투닝을 활용한 애플리케이션

- 법률 문서 분석
- 코드 리뷰 자동화
- 재무 문서 요약
- 기술 문서 번역
- 전문적인 뉴스 기사 작성

### 3. 검색 증강 생성(RAG)

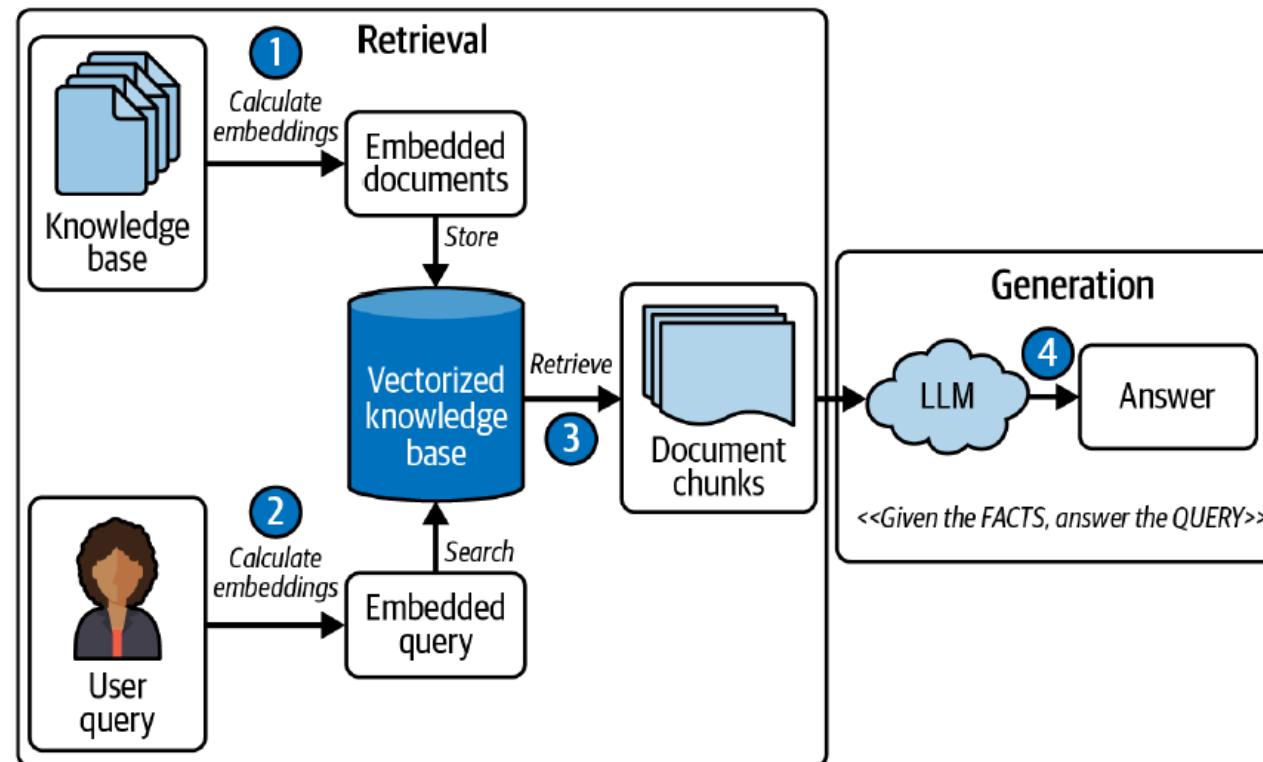
#### ▶ Retrieval-Augmented-Generation (RAG)

- GPT 모델의 한계 → 지식의 부재
  - 최신 데이터 : GPT 모델은 훈련 이후에 발생한 사건에 대한 정보를 가질 수 없음
  - 독점 데이터 : 접근하지 않은 개인 데이터에 관한 질문에 답변할 수 없음
  - 특정 도메인 또는 소수의 데이터 : 모델이 해당 데이터에 대해 배우기에 데이터가 충분하지 않음
- LLM 맞춤화 진행하는 파인 튜닝 보다 RAG 활용이 지식 주입 측면에서 더 뛰어함
  - 참고논문 : <https://arxiv.org/pdf/2312.05934>

### 3. 검색 증강 생성(RAG)

#### ▶ Retrieval-Augmented-Generation (RAG)

- RAG 과정의 세 가지 주요 단계



### 3. 검색 증강 생성(RAG)

#### ▶ Retrieval-Augmented-Generation (RAG)

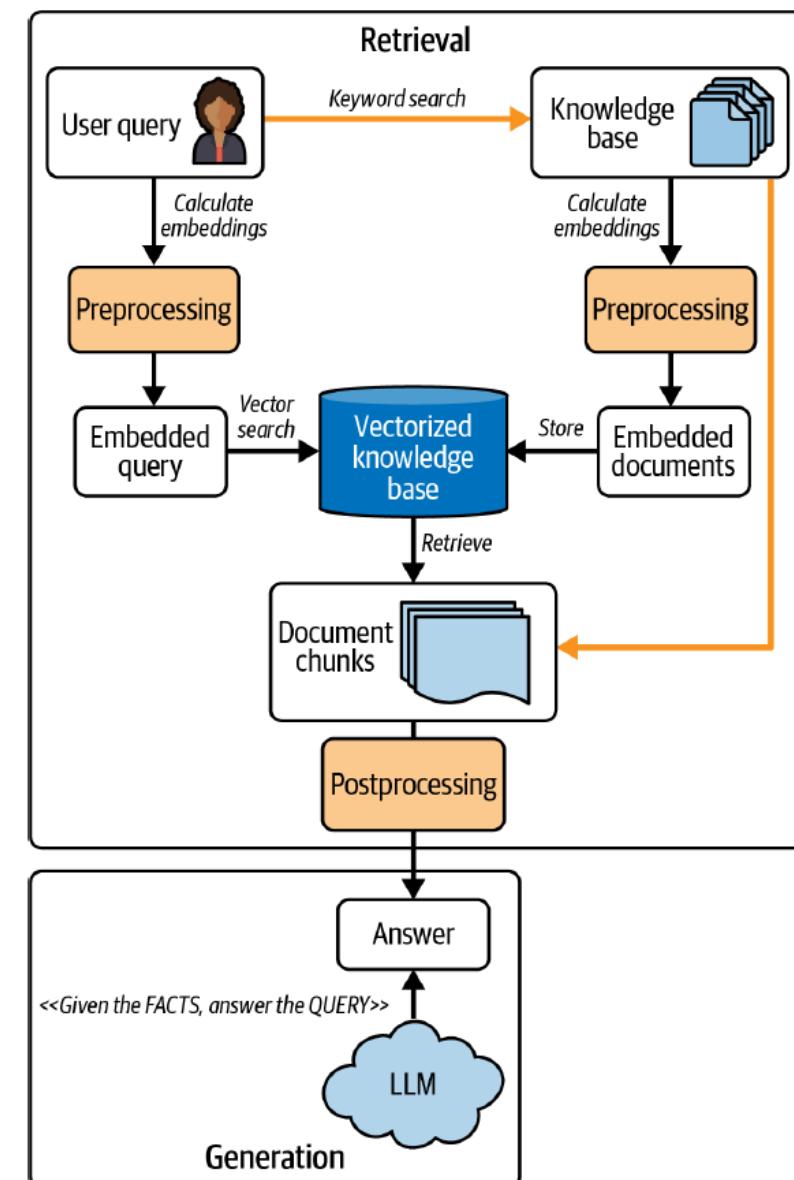
- 기본 RAG
  - 임베딩은 지식 베이스에서 계산되며, 임베딩은 추후 사용하기 위해 DB, 메모리, 디스크 등에 저장
  - 사용자의 쿼리마다 임베딩 계산
  - 2단계에서 계산한 임베딩은 이전에 계산된 임베딩의 벡터 검색 수행하는 데 사용 → 발췌문 생성
  - 프롬프트 사용해 GPT 모델 호출
- 기본 RAG가 좋은 성능을 내는 조건은 두 가지
  - 잘 구성된 질문과 잘 구조화한 양질의 데이터

### 3. 검색 증강 생성(RAG)

#### ▶ Retrieval-Augmented-Generation (RAG)

- 고급 RAG

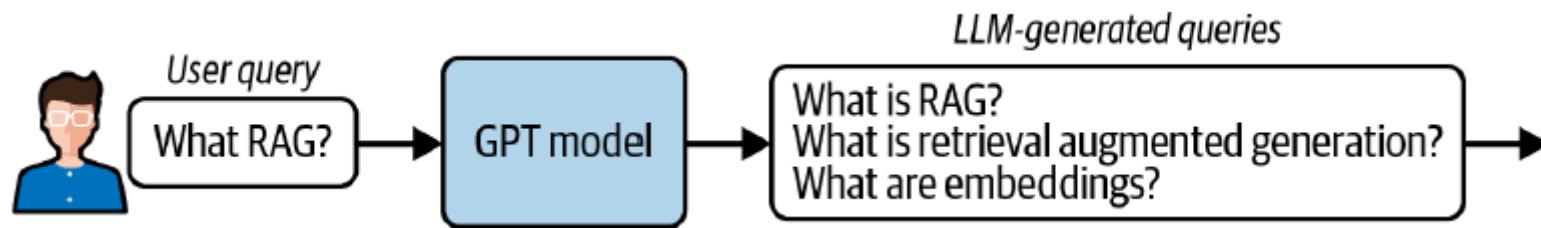
- 사용자 쿼리 전처리(User Query)
- 지식 베이스 전처리(Knowledge Base)
- 검색 개선(Improving search)
- 후처리(PostProcessing)



### 3. 검색 증강 생성(RAG)

#### ▶ Retrieval-Augmented-Generation (RAG)

- 고급 RAG – 사용자 쿼리 전처리
  - 오타 혹은 부적절한 표현이 결과에 영향을 주지 않도록 GPT 모델 재구성
  - 대화 내역 고려 예시

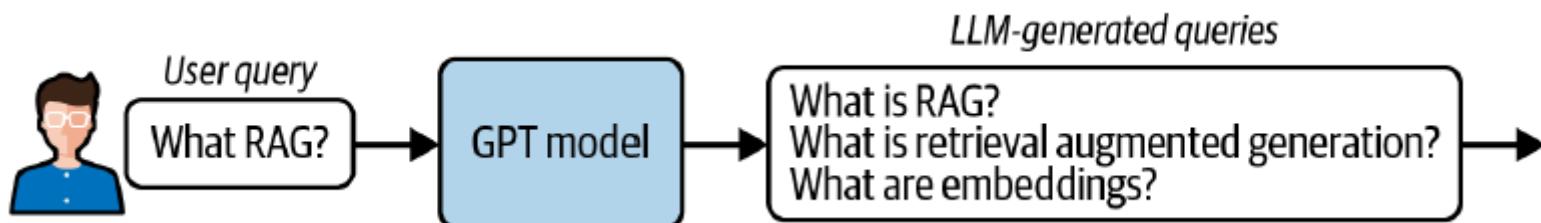


### 3. 검색 증강 생성(RAG)

#### ▶ Retrieval-Augmented-Generation (RAG)

- 고급 RAG – 사용자 쿼리 전처리

- 오타 혹은 부적절한 표현이 결과에 영향을 주지 않도록 GPT 모델 재구성
- 유사한 쿼리 생성
- 사용자의 입력을 여러 쿼리로 분해
- 검색 결과에 더 많은 컨텍스트가 포함되도록 더 넓은 범위의 쿼리 생성



### 3. 검색 증강 생성(RAG)

#### ▶ Retrieval-Augmented-Generation (RAG)

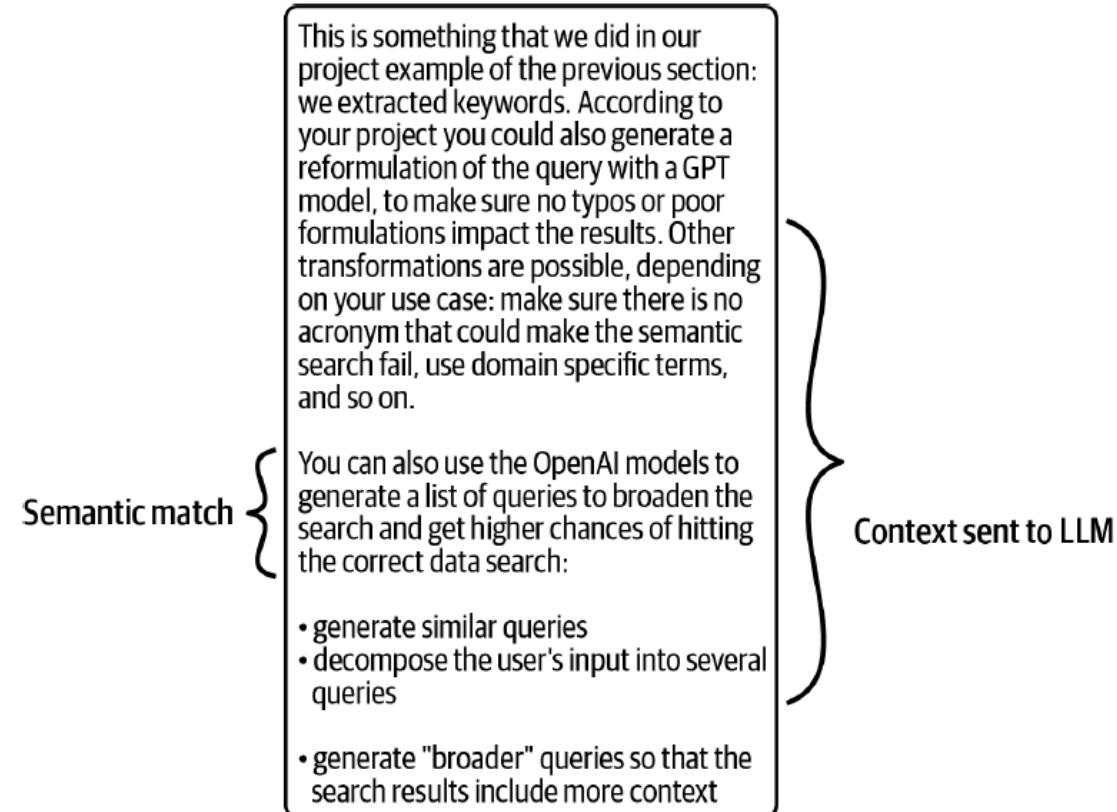
- 고급 RAG – 지식 베이스 전처리

- 청킹 : 문서를 문단이나 문장과 같이 의미 단위로 청크 분할
- 각 청크는 검색이 잘 수행될 수 있을 만큼 작아야 함
- 충분한 컨텍스트를 제공하려면 LLM이 처리할 수 있을 만큼 충분히 커야 함

### 3. 검색 증강 생성(RAG)

#### ▶ Retrieval-Augmented-Generation (RAG)

- 고급 RAG – 지식 베이스 전처리



### 3. 검색 증강 생성(RAG)

#### ▶ Retrieval-Augmented-Generation (RAG)

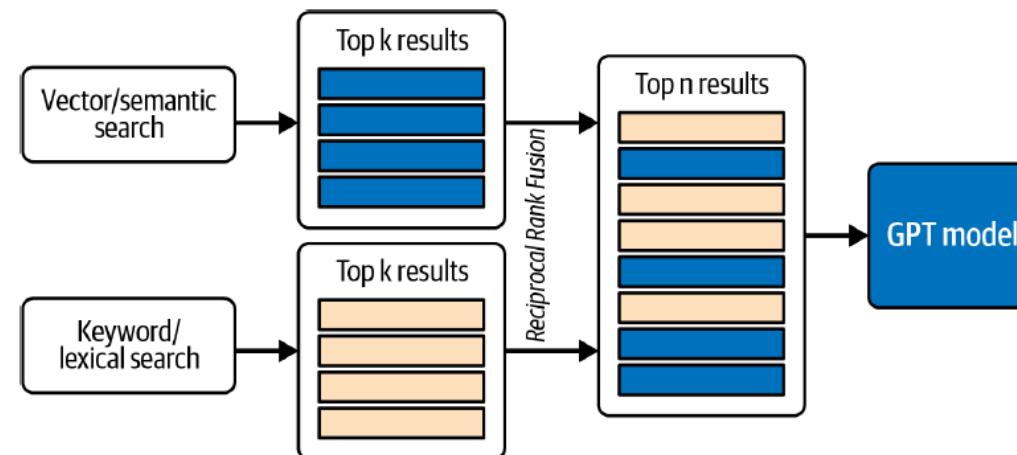
- 고급 RAG – 지식 베이스 전처리
  - 청크에 메타데이터 추가하여 해결
  - 메타데이터로 다시 검색하거나 메타데이터와 청크 자체를 벡터화하는 방법
  - 검색 결과 개선하기 위해 GPT-4o 호출
  - 가상 문서 임베딩(HyDE, Hypothetical Document Embedding)은 사용자의 쿼리 바탕으로 가상의 문서 생성

### 3. 검색 증강 생성(RAG)

#### ▶ Retrieval-Augmented-Generation (RAG)

- 고급 RAG – 검색 개선

- 벡터 검색 : 코사인 유사도 사용하는 KNN 검색 사용
- 근사 최근점 이웃(ANN : Approximate Nearest Neighbor) 알고리즘 적합
  - 정확한 거리 계산이 비싼 대규모 데이터셋에 적용하기 적합한 방법
- 하이브리드 검색



### 3. 검색 증강 생성(RAG)

#### ▶ Retrieval-Augmented-Generation (RAG)

- 고급 RAG – 후처리
  - 메타데이터 사용하여 필터링, 재정렬, 원래 질문에 답하는 데 도움이 되도록 변환
- 성공적인 고급 RAG 디자인
  - 여러 접근 방식 시도, 결과 분석, 재시도하는 방식
  - RAG 설계의 성능을 효과적으로 측정해 이 반복적인 접근 방식이 실제로 개선으로 이어지도록 하는

### 3. 검색 증강 생성(RAG)

#### ▶ Retrieval-Augmented-Generation (RAG)

- 고급 RAG
  - 여러 파이프라인
  - 병렬 처리
  - OpenAI API의 많은 호출
- 복잡성 증가 및 유지보수 어려움 증가, 비용 문제 발생

## 4. 전략 기법

### ▶ 다양한 기법의 비교

구분	제로샷 러닝	퓨샷 러닝	프롬프트 엔지니어링	파인 투닝	RAG
정의	사전 예제 없이 새로운 작업을 예측	입력과 원하는 출력을 포함하는 프롬프트 사용	컨텍스트, 역할, 작업 등을 포함하는 상세 프롬프트 또는 "단계별로 생각하기"와 같은 기법 활용	모델을 작은 특정 데이터셋으로 추가 학습하며, 프롬프트는 단순하게 사용됨	벡터 검색과 LLM(대형 언어 모델) 생성을 결합한 방식
사례	간단한 작업	명확하지만 복잡한 작업, 주로 특정 출력 형식이 요구되는 경우	창의적이고 복잡한 작업	매우 복잡한 작업 또는 특정 출력 형식, 톤, 스타일이 필요한 작업	독점적인 데이터나 LLM이 잘 모르는 데이터에 대한 쿼리
데이터	추가 예제 데이터가 필요 없음	몇 개의 예제만 필요	필요한 데이터 양은 프롬프트 엔지니어링 기법에 따라 다름	크고 고품질의 학습 데이터셋이 필요	잘 구성된 문서화된 데이터가 필요하며, 해당 데이터는 사용자의 질문에 대한 답을 포함해야 함

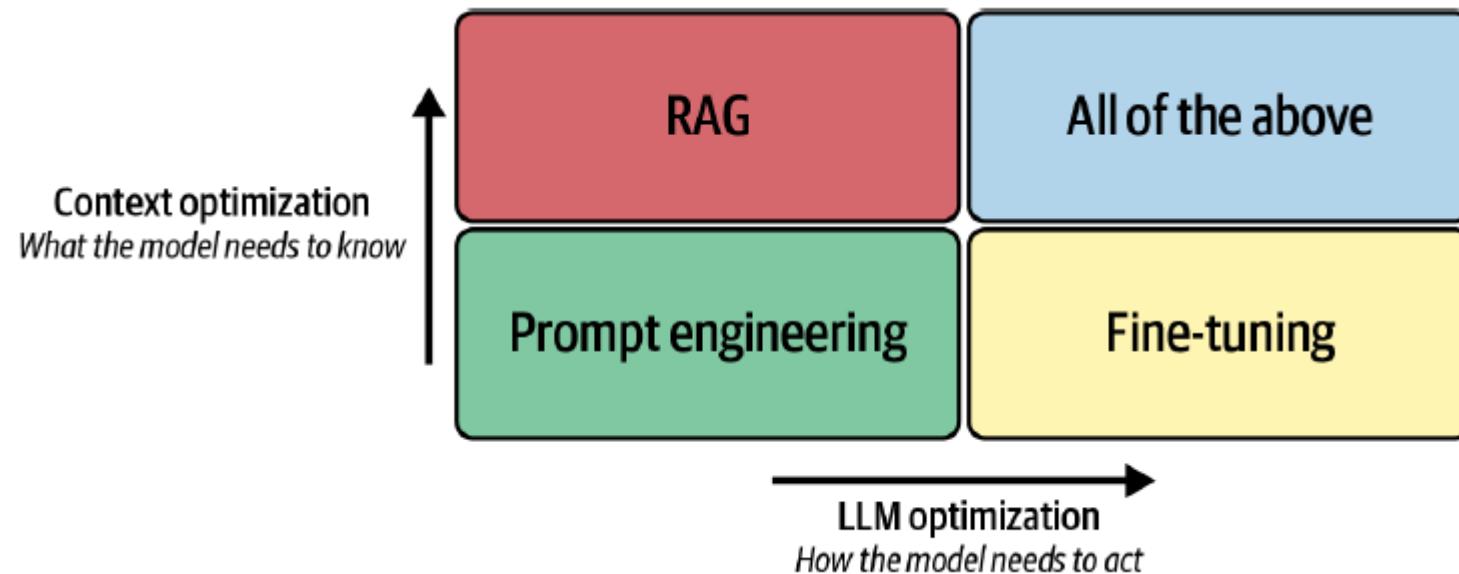
## 4. 전략 기법

### ▶ 다양한 기법의 비교

구분	제로샷 러닝	퓨샷 러닝	프롬프트 엔지니어링	파인 투닝	RAG
비용	사용: 입력 및 출력에 대해 토큰 단위로 과금됨	사용: 입력 및 출력에 대해 토큰 단위로 과금됨 (긴 프롬프트가 필요할 수도 있음)	사용: 입력 및 출력에 대해 토큰 단위로 과금됨 (긴 프롬프트가 필요할 수도 있음)	학습: 학습 데이터의 토큰당 비용 발생, 사용: 입력 및 출력의 토큰당 비용 발생 (미세 조정된 GPT-3.5 Turbo는 미세 조정되지 않은 모델보다 4~6배 더 비쌈 → 프롬프트가 기본 모델보다 6배 이상 길어질 경우 미세 조정이 경제적으로 유리할 수 있음)	설정: 지식 베이스 임베딩 생성 비용 + 사용: 임베딩 모델을 사용한 벡터화된 쿼리 생성 비용 + LLM이 벡터 검색 결과를 처리하는 비용
결론	기본적으로 사용	제로샷 학습이 작동하지 않고, 출력이 특정한 형식을 필요로 할 경우 사용	작업이 너무 복잡할 경우, 프롬프트 엔지니어링을 시도	매우 특정한 데이터셋이 있으며, 다른 방법으로는 충분한 결과를 얻지 못할 경우 사용 (프롬프트 엔지니어링을 먼저 시도한 후에 적용)	RAG는 설정이 쉬우며, 전체 문서를 LLM에 입력하는 것보다 더 나은 결과와 낮은 비용을 제공할 수 있음. 다만, 결과는 문서의 품질에 크게 의존하며, 고급 RAG 기법은 비용이 많이 들고 과도하게 복잡해질 수 있음

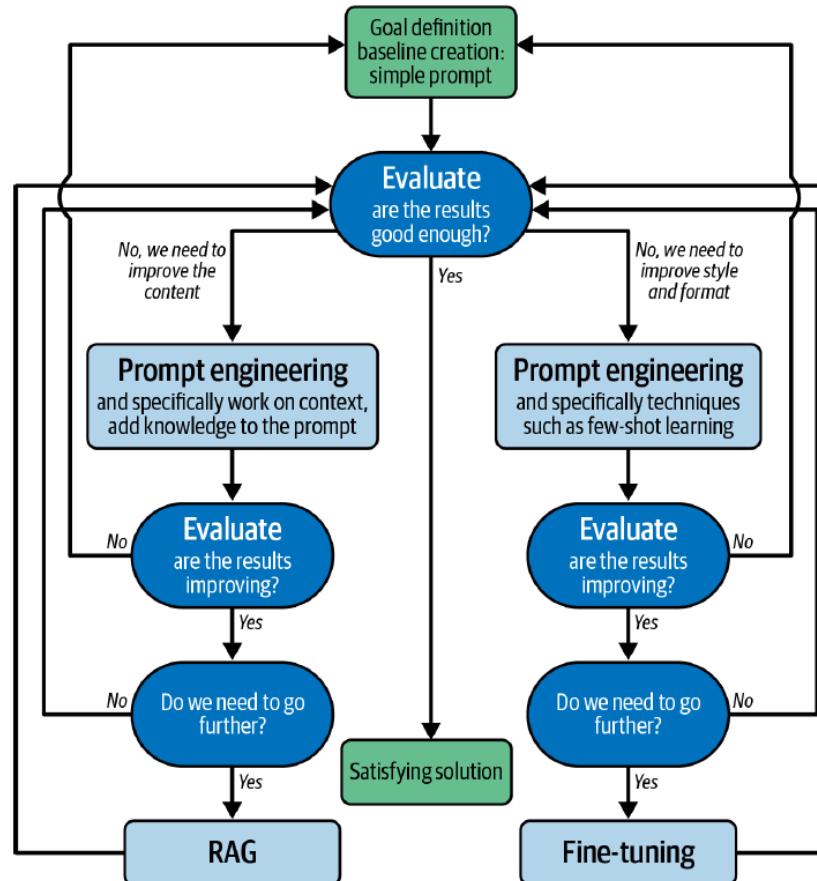
## 4. 전략 기법

### ▶ 다양한 기법의 비교



## 4. 전략 기법

### ▶ 최적화 작업의 흐름



## 5. 평가

### ▶ 솔루션과 구현을 다양한 기준에서 효과적으로 비교할 수 있도록 결과 평가

- 관련성
- 환각 현상
- 질문 답변의 정확도
- 유해성
- 검색 관련 지표

### ▶ 주요 평가 도구

- <https://mlflow.org/docs/latest/llms/llm-evaluate/index.html>
- <https://www.promptfoo.dev/>

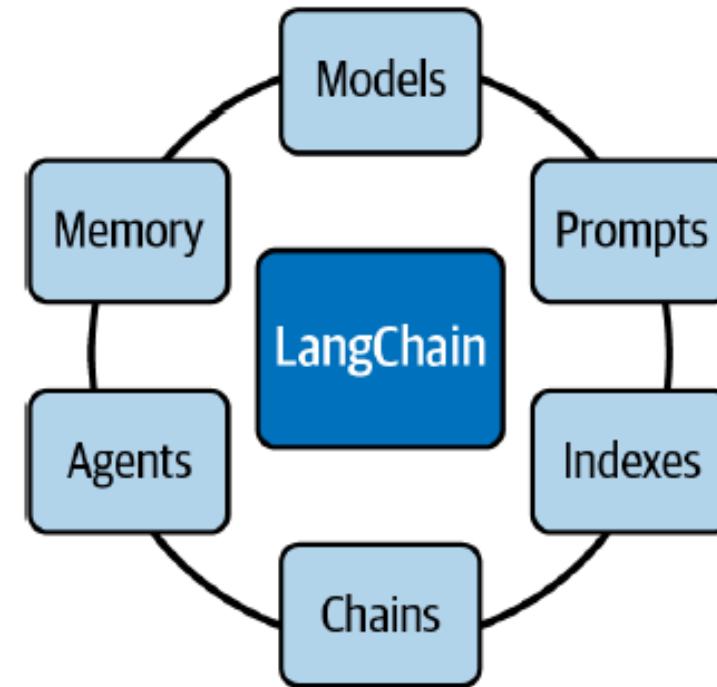
## 5. 랭체인 프레임워크로 LLM 기능 높이기

---

## 1. 랭체인

### ▶ LLM 기반 애플리케이션 개발 및 배포 위해 설계된 프레임워크

- 사이트 : <https://www.langchain.com/>
- OpenAI뿐만 아니라 다른 솔루션과도 호환
- 버전이 매우 자주 바뀌고 있음



## 1. 랭체인

### ▶ LLM 기반 애플리케이션 개발 및 배포 위해 설계된 프레임워크

- 각 모듈에 대한 간략한 설명

구분	설명
모델	다양한 LLM(대형 언어 모델)과 상호작용할 수 있는 표준 인터페이스 제공. OpenAI, Hugging Face, Cohere, GPT4All 등 다양한 모델 통합 지원.
프롬프트	LLM을 프로그래밍하는 표준 방식으로, 프롬프트 관리를 위한 다양한 도구 포함.
인덱스	LLM과 사용자의 데이터를 결합할 수 있도록 지원하는 모듈.
체인	여러 개의 모델 또는 프롬프트를 조합하여 연속적인 호출을 만들 수 있도록 하는 Chain 인터페이스 제공.
에이전트	사용자 입력을 처리하고, 도구를 선택하여 작업을 수행하는 에이전트 인터페이스 제공. 반복적인 행동을 통해 최종 해결책을 도출.
메모리	체인 또는 에이전트 호출 간 상태를 유지할 수 있도록 지원하는 모듈. 기본적으로 체인과 에이전트는 무상태(stateless) 방식으로 동작.

## 1. 랭체인

### ▶ LLM 기반 애플리케이션 개발 및 배포 위해 설계된 프레임워크

- 여러 패키지로 나눠짐

구분	설명
langchain-core	다른 패키지에 최소한으로 의존하면서 핵심 기능 제공
langchain-community	외부 서비스 및 플랫폼과의 통합 담당
langchain	더 높은 수준의 추상화와 체인에 초점을 맞춤, 체인, 에이전트, 고급 쿼리 기법, 일반화에 사용하는 오케스트레이션 조각 등 블록 빌딩에 필수적인 구성 요소들 포함
langchain-experimental	아직 안정화되지 않은 새로운 기능 포함, 애플리케이션 개발에는 사용 권장하지 않음

## 1. 랭체인

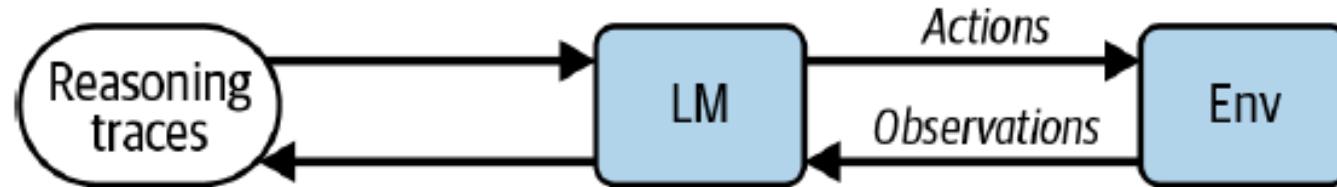
### ▶ 에이전트와 도구

- 랭체인 프레임워크의 핵심 기능 – 에이전트와 도구
  - 에이전트
    - 개발환경과 상호작용할 수 있는 소프트웨어
    - LLM 컨텍스트에서 에이전트는 LLM의 특정 프롬프트 통해 생성
  - 도구
    - 언어 모델이 함수와 더 쉽게 상호작용하도록 함
    - 사전 정의된 도구 : 구글 검색, 위키백과 검색, 파이썬 REPL, 계산기, 세계 날씨 API
      - 참조 : <https://python.langchain.com/v0.1/docs/integrations/tools/>
    - 사용자 지정 도구 구축하여 사용 중인 에이전트에 불러올 수 있음

## 1. 랭체인

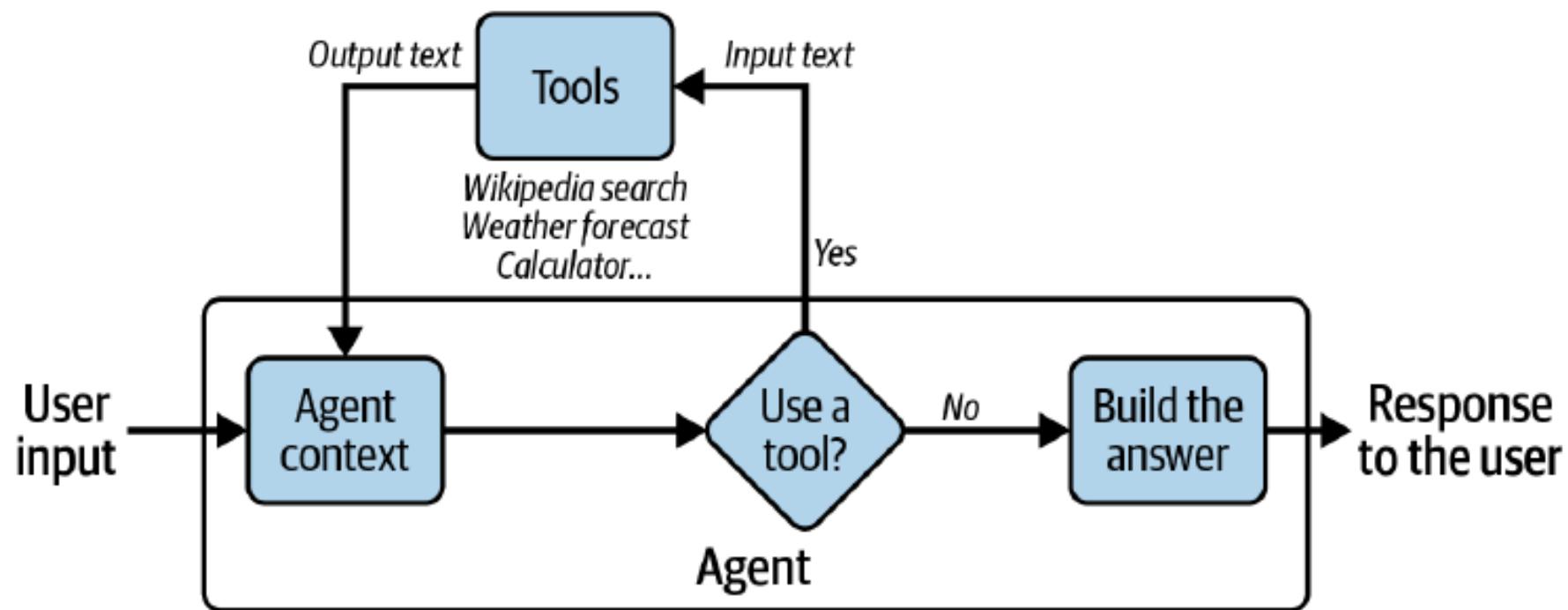
### ▶ ReAct 애이전트

- 관찰과 행동을 추론 과정에 결합하는 것



## 1. 랭체인

### ▶ 랭체인에서 에이전트와 도구 간의 상호작용



## 1. 랭체인

### ▶ 메모리 기능 지원

- 랭스미스 : <https://www.langchain.com/langsmith>
- 랭그래프에서 메모리 사용하기 위해선 MemorySaver 사용

## 1. 랭체인

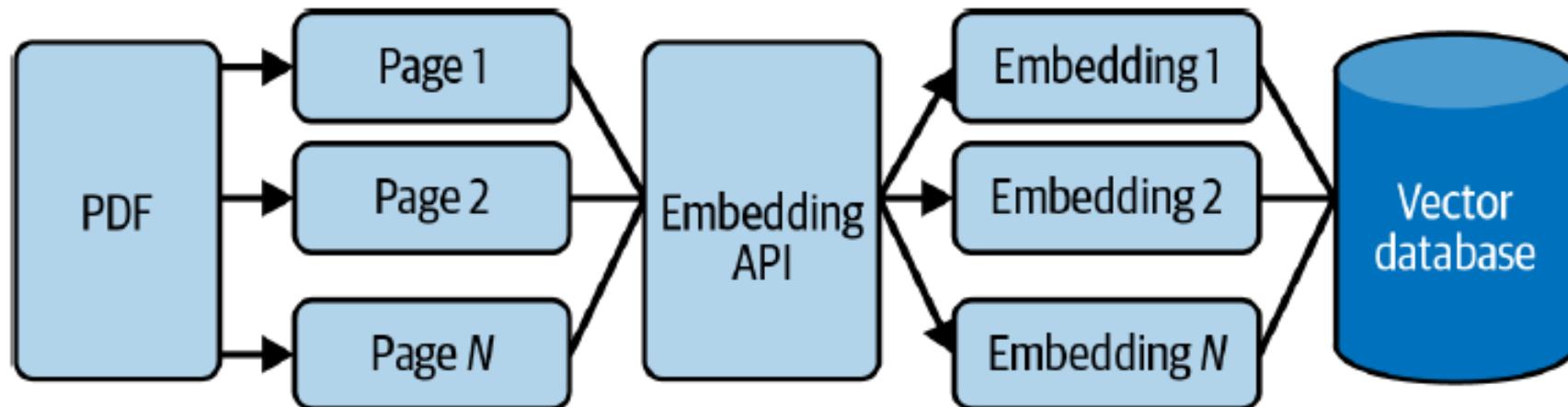
### ▶ 임베딩

- 검색 증강 생성(RAG)은 언어 모델과 사용자 데이터 결합해 앱에서 사용하는 모델의 지식 개인화
- 단계별 작업
  - 1단계 : 정보 검색으로, 사용자의 쿼리를 받아 가장 관련성이 큰 문서에서 내용 추출
  - 2단계 : 추출된 내용을 바탕으로 언어 모델이 출력 생성
- 랭체인 모듈 : `document_loaders`
  - 다양한 소스의 데이터를 애플리케이션으로 빠르게 불러오기 가능
  - 참조 : [https://python.langchain.com/docs/how\\_to/#document-loaders](https://python.langchain.com/docs/how_to/#document-loaders)

## 1. 랭체인

### ▶ 임베딩

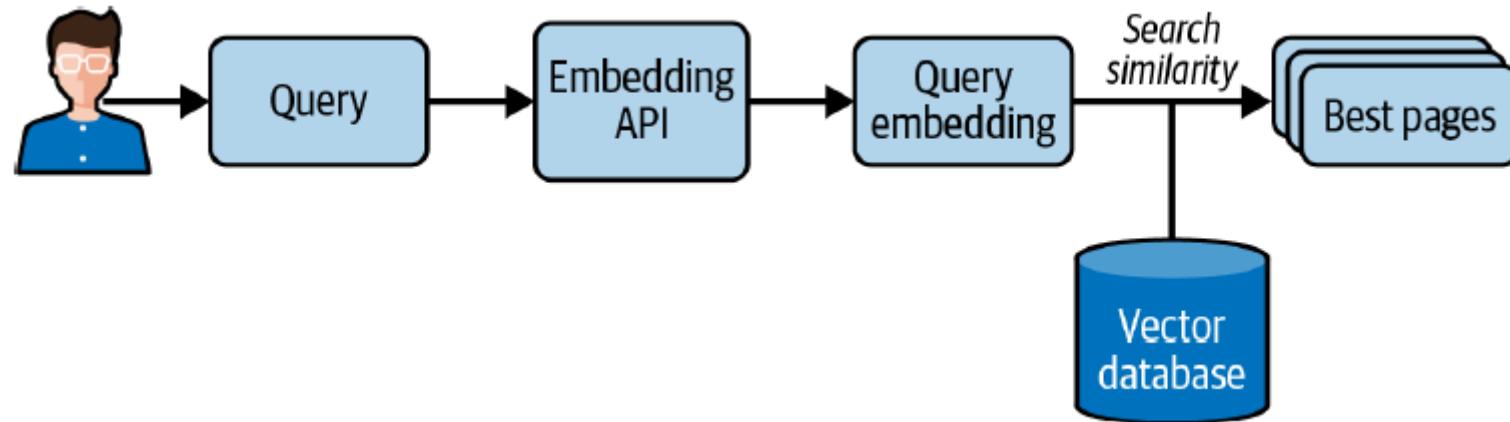
- PDF 문서에서 임베딩을 만들고 저장하기



## 1. 랭체인

### ▶ 임베딩

- 사용자의 질문에 답하기 위해 검색된 정보가 LLM의 컨텍스트에 추가됨
- 임베딩과 벡터 데이터베이스를 사용해 입력된 질문과 가장 유사한 페이지 식별하는 과정



## 2. 라마인덱스

### ▶ 랭체인과 유사한 기능 제공하는 프레임워크

- 랭체인 : 체인 개념을 중심으로 만듬
- 라마인덱스 : 컨텍스트가 추가된 거대 언어 모델 애플리케이션에 집중
  - 파싱, 데이터 수집, 검색 서비스가 포함된 기업 솔루션인 라마클라우드도 제공

## 2. 라마인덱스

### ▶ 라마인덱스 원칙

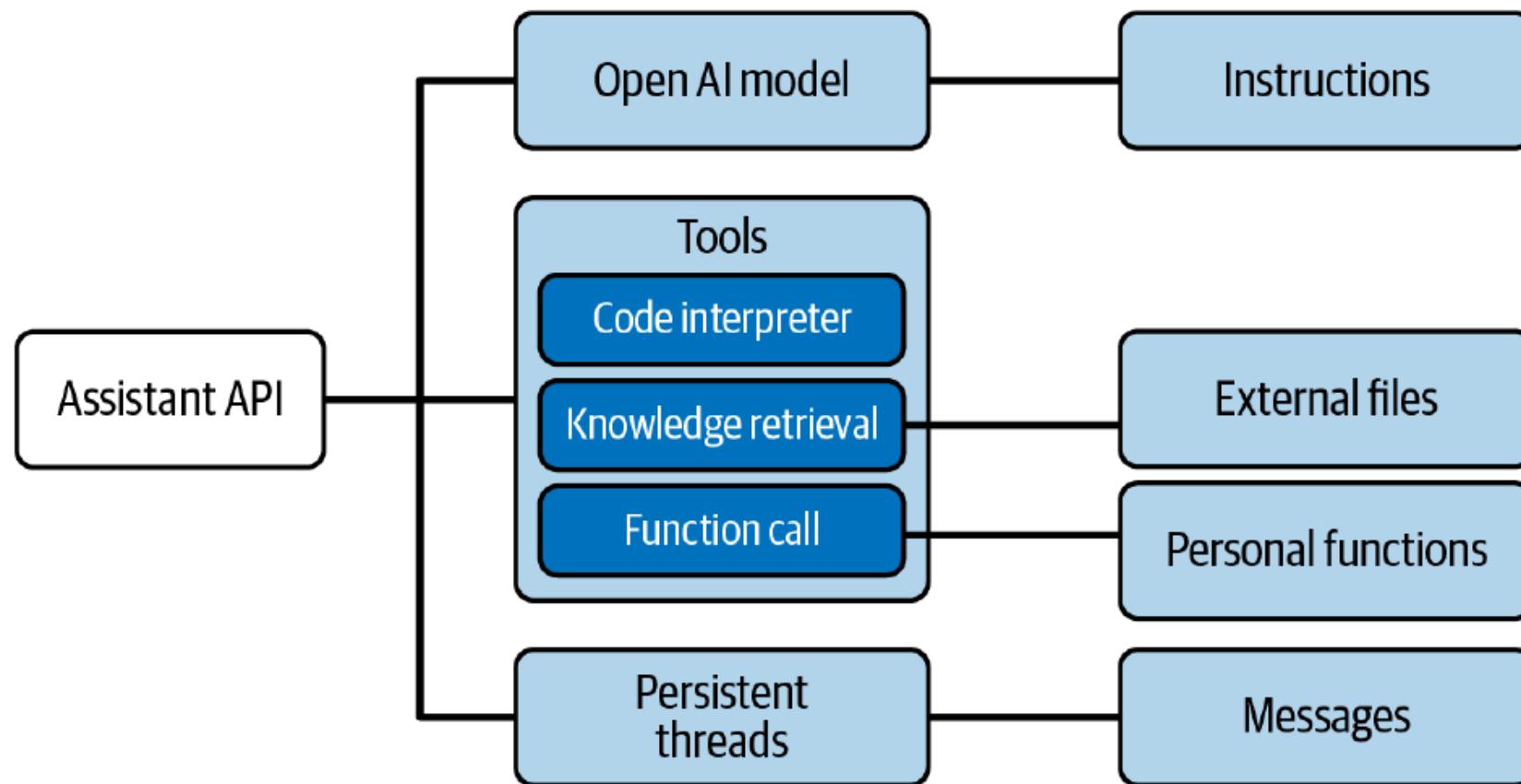
- RAG 파이프라인



- 로드 : 데이터 커넥터를 통해 기존 데이터를 쉽게 가져오고 여러 데이터 소스 및 데이터 형식 지원
- 색인 : 벡터 임베딩, 메타데이터 추가도 쉽게 할 수 있음
- 저장 : 임베딩을 만들어 쿼리할 수 있는 형태로 저장, 여러 스토리지 솔루션 제공
- 쿼리 : 라마인덱스를 통해 질문을 입력해 검색된 컨텍스트와 LLM의 답변 함께 얻을 수 있음
- 평가 : RAG 솔루션을 만들려면 디자인을 반복적으로 평가하는 과정 필요, 응답의 정확성, 신뢰성 도구 제공

### 3. 어시스턴트 API

#### ▶ 어시스턴트 API 주요 기능



### 3. 어시스턴트 API

#### ▶ 어시스턴트 API 주요 기능

- 개발자들이 OpenAI ChatGPT 인터페이스에서 GPTs처럼 다양한 특정 작업 수행할 수 있는 강력한 AI 어시스턴트를 만들 수 있게 함
  - 코드 인터프리터, 검색 모듈, 함수 호출 매커니즘 통해 다양한 도구에 접근 가능
- 무상태성을 가져 대화 미 저장
  - 어시스턴트 API 사용하여 진행 중인 대회에 지속적인 스레드 개념 도입

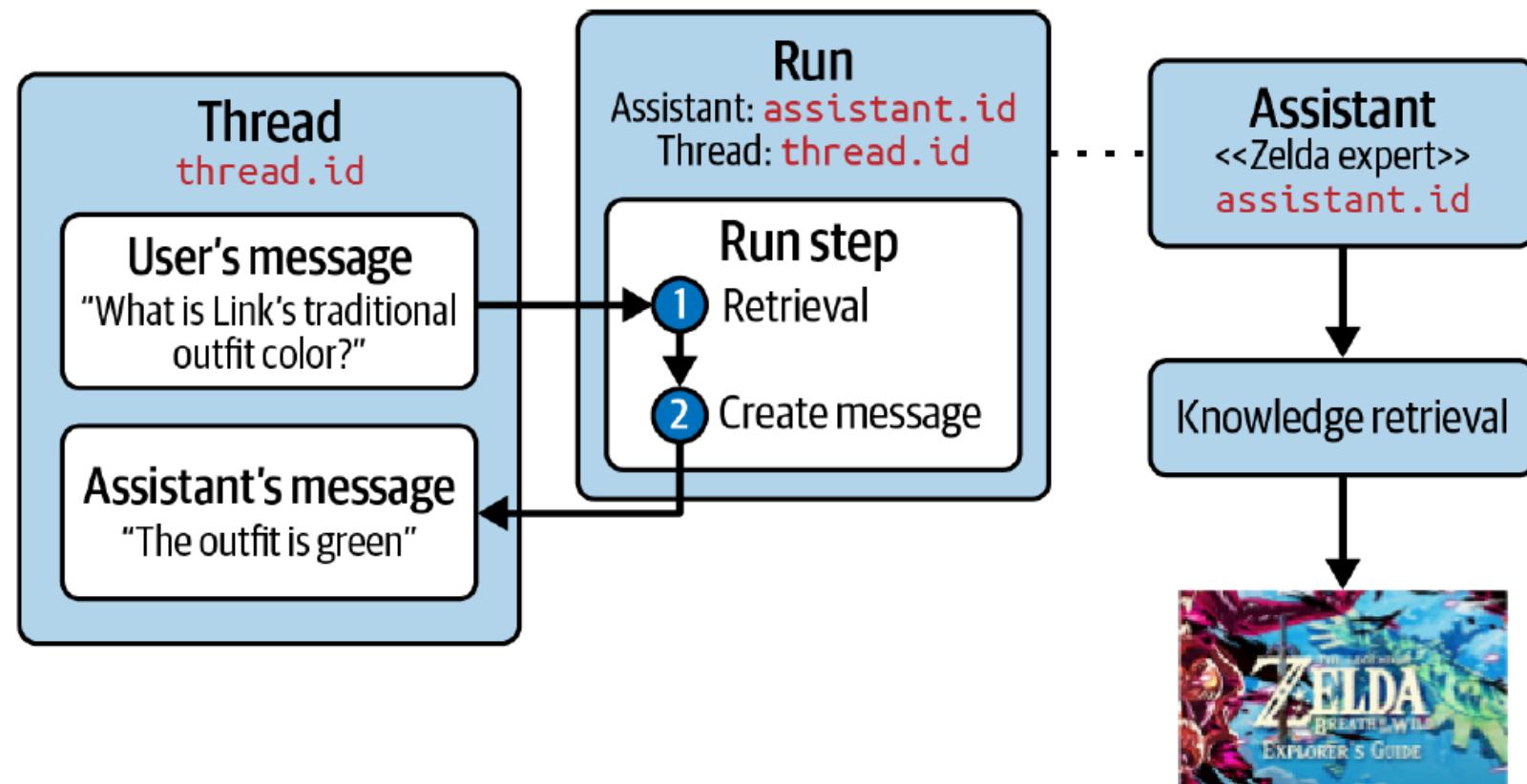
### 3. 어시스턴트 API

#### ▶ 어시스턴트 API를 통한 대화 관리

- 스레드 : 어시스턴트와 사용자가 대화하는 세션 나타냄, 메시지 저장이 주요 목적
- 메시지 : 메시지는 어시스턴트나 사용자가 작성한 텍스트 포함
- 실행 : 어시스턴트가 사용자의 메시지에 응답하려면 실행 생성, 실행 지속 시 스레드에 메시지 추가
- 단계 실행 : 어시스턴트는 실행 중에 도구 호출하거나 메시지 생성

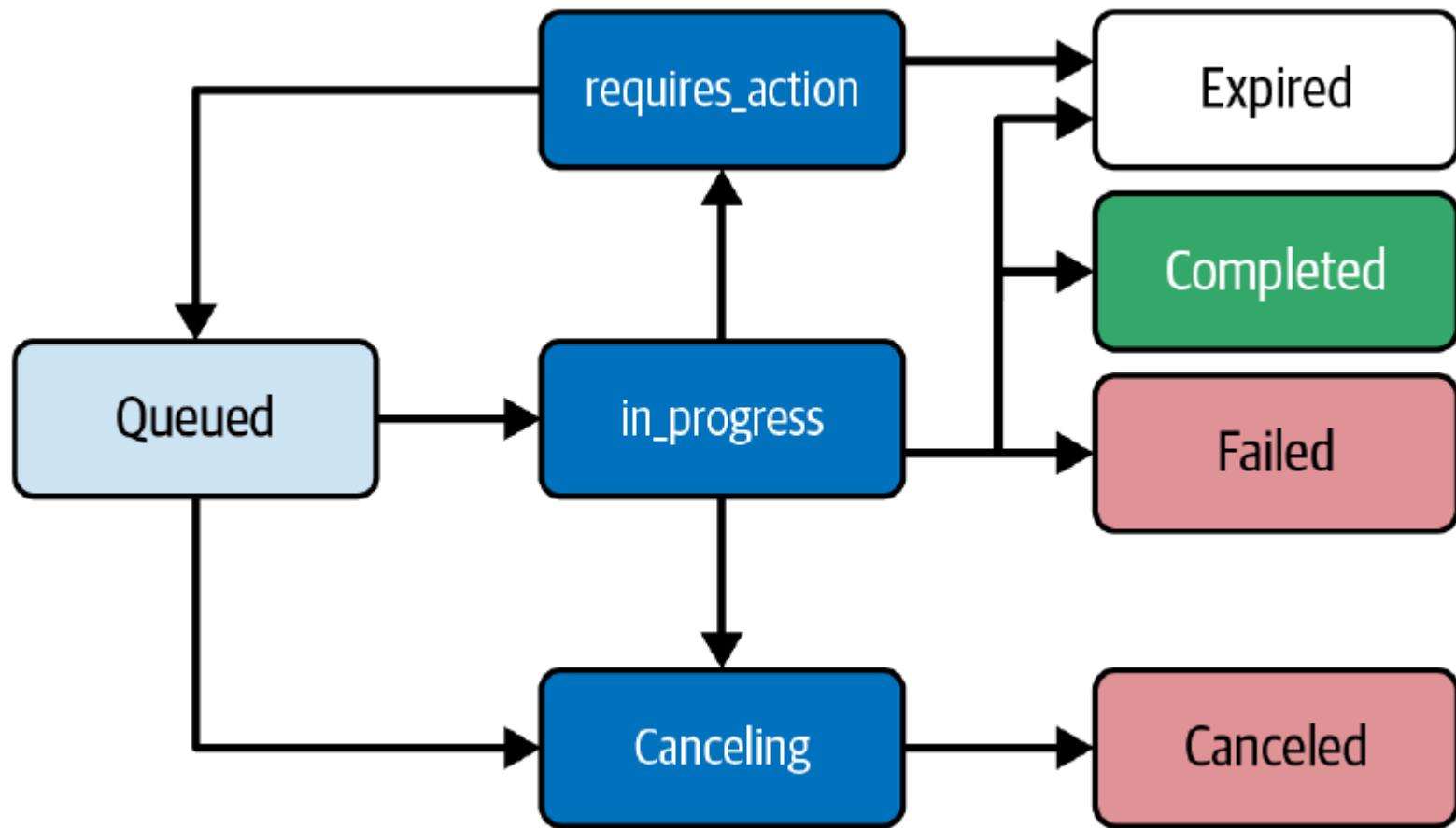
### 3. 어시스턴트 API

- ▶ 어시스턴트 API가 응답을 생성하도록 상호작용



### 3. 어시스턴트 API

- ▶ run 객체의 여러 가지 상태



## 6. 실전예제

---

## 1. 들어가기에 앞서

### ▶ OpenAI API 서비스 유료

- ChatGPT Plus와는 완전히 별개의 서비스
- [Open API 요금 정책](#)

## 1. 들어가기에 앞서

### ▶ OpenAI 토큰 수 계산 페이지

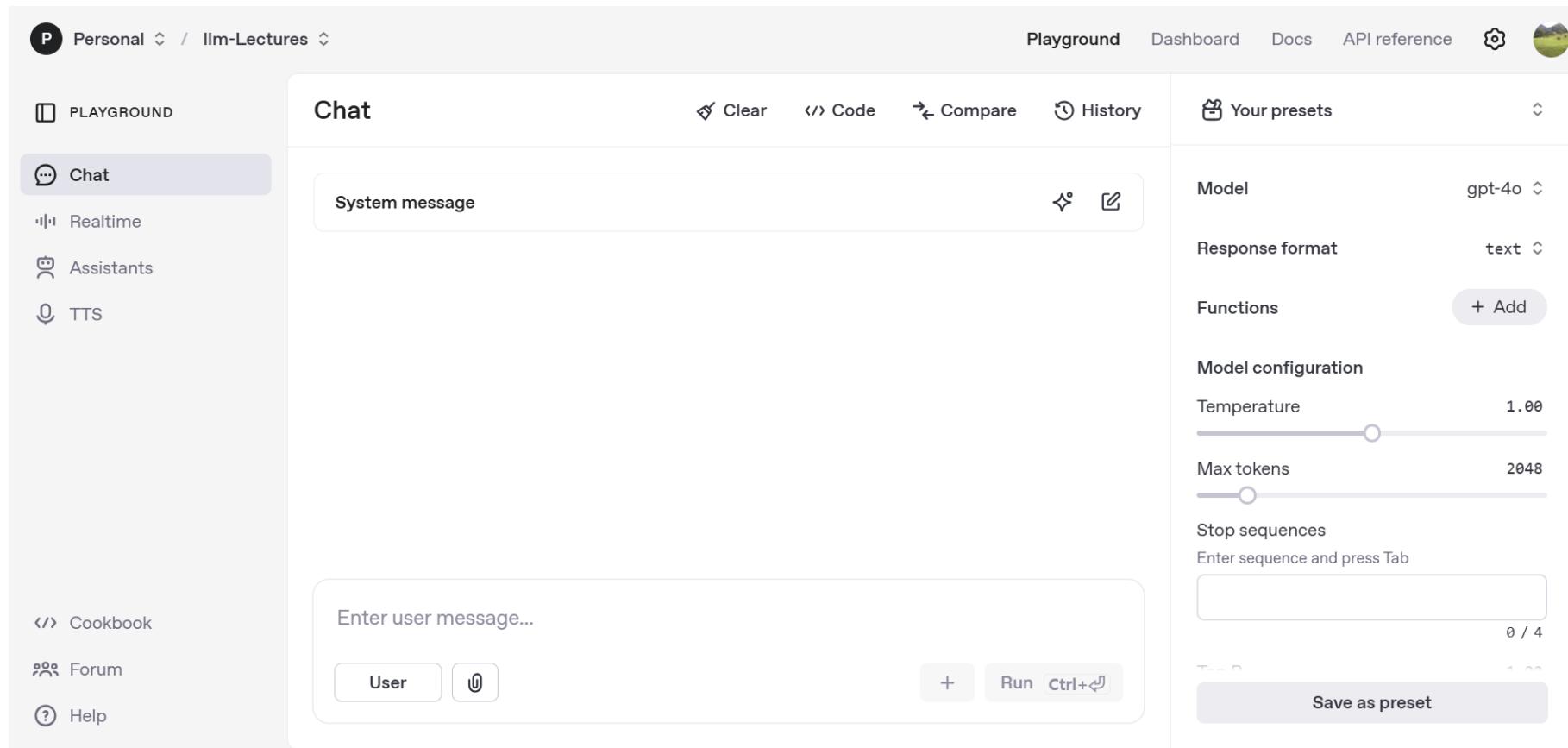
- 토큰은 인공지능 모델이 사용하는 단위 조각의 단위
- OpenAI 토큰 수 계산 페이지 : <https://platform.openai.com/tokenizer>

The screenshot shows the 'Tokenizer' page of the OpenAI Platform. At the top, there's a navigation bar with 'OpenAI Platform', 'Docs', 'API reference', 'Log in', and 'Sign up'. Below the navigation, the title 'Tokenizer' is centered above a section titled 'Learn about language model tokenization'. This section contains a paragraph explaining that OpenAI's large language models process text using tokens, which are common sequences of characters found in a set of text. It mentions that models learn to understand the statistical relationships between these tokens and excel at producing the next token in a sequence. A 'Learn more.' link is provided. Below this, another paragraph states that users can use the tool to understand how a piece of text might be tokenized by a language model and the total count of tokens in that piece of text. At the bottom, there are three tabs: 'GPT-4o & GPT-4o mini' (which is selected), 'GPT-3.5 & GPT-4', and 'GPT-3 (Legacy)'. Below the tabs is a text input field with placeholder text 'Enter some text'. At the very bottom, there are 'Clear' and 'Show example' buttons.

## 1. 들어가기에 앞서

### ▶ API의 다양한 모델 테스트

- [Playground](#) 서비스 지원 (주의 : 무료 아님)



## 2. 텍스트 생성 모델

▶ 사용자 요청에 따라 다음과 같은 다양한 애플리케이션 구축

- 문서 초안 작성
- 컴퓨터 코드 작성
- 다양한 분야의 지식에 대한 질문에 답변하기
- 글 요약
- 소프트웨어에 자연어 인터페이스 제공
- 언어 번역