

LLM 기반 AI agent 개발 과정

위블 CEO 노규남
bardroh@weable.ai

2026/01/26

- 클라우드 업체(가비아)와 회선 업체(케이아이엔엑스) CTO 역임
- 현재는 인공지능 스타트업인 위블(<https://www.weable.ai>) 의 CEO로 재직 중: 머신러닝과 클라우드, DevOps 등에 대한 **컨설팅** 및 **솔루션 개발**을 영역으로 비즈니스를 전개, **교육사업**도 진행
- AI를 중심으로 운영과 관리에 클라우드 기술을 접목하려고 시도하고 있음(MLOps, AIOps)
- AI: GBM 등 Classical Machine Learning부터 CNN, RNN, Transformer 기반의 NLP, DQN, CFR, LLM, 모방학습까지 머신러닝 전반에 걸친 솔루션 개발 및 컨설팅, 교육
- 클라우드: Public, Private, Hybrid IaaS, docker, kubernetes, MSA(MicroService Archiectrue), CI/CD 등 클라우드 네이티브한 영역에서의 개발 및 운영 역량 보유
- AI 중에서도 1) sLLM 2) 온디바이스 AI 3) 로봇 AI가 회사에서 가장 관심있게 연구하고 있는 영역임

검색 증강 챗봇

선택 업로드

검색에 사용할 파일을 선택해주세요.

(교육교재) 삼성전자 CRA 과...

파일 삭제

글포인트 쪽으로 간너씨비드린 투겁에사 인니. 이 공개 아진 고드시 노 난걸세자 푸시포구시
용이해집니다.

- Dependency Inversion Principle (의존성 역전 원칙): 고수준 모듈은 저수준 모듈에 의존해서는 안 된다. 즉, 추상화된 인터페이스에 의존하게 되어야 한다. 이렇게 설계되면 코드가 더 유연해져 확장성이 좋아집니다.

챗봇 리스코프 치환 원칙에 대해서 더 상세하게 설명해주세요.

챗봇 리스코프 치환 원칙(Liskov Substitution Principle, LSP)은 객체지향 프로그래밍에서 중요한 원칙 중 하나입니다. 이 원칙은 상위 타입의 인스턴스를 하위 타입의 인스턴스로 교체할 수 있는지 여부에 대한 규칙을 제공합니다. 즉, 프로그램의 다른 부분이 상위 타입으로 선언된 변수에 하위 타입의 객체를 할당할 수 있는지 확인하는 것입니다.

챗봇 리스코프 치환 원칙은 다음과 같은 조건을 만족해야 합니다:

- 상위 타입의 객체와 하위 타입의 객체가 서로 대체 가능해야 한다. 이렇게 하면, 프로그램의 다른 부분은 상위 타입으로 선언된 변수에 상위 타입의 객체를 할당하더라도, 실제로 하위 타입의 객체가 할당되어 있는 경우에도 정상적으로 작동해야 합니다.
- 하위 타입의 객체가 상위 타입의 인터페이스를 구현해야 한다. 이렇게 하면, 상위 타입의 변수에 하위 타입의 객체를 할당하더라도, 상위 타입의 메서드와 프로퍼티를 사용할 수 있습니다.
- 하위 타입의 객체가 상위 타입의 인터페이스를 구현할 때, 상위 타입과 동일한 방식으로 작동해야 합니다. 즉, 상위 타입과 하위 타입 간에는 일관성이 있어야 합니다.

챗봇 리스코프 치환 원칙은 코드의 유연성과 재사용성을 높여주며, 객체지향 설계를 더 안정적으로 만들어줍니다. 이 원칙을 따르면, 프로그램의 다른 부분에서 상위 타입의 변수에 하위 타입의 객체를 할당하더라도, 프로그램이 정상적으로 작동하게 됩니다. 이렇게 함으로써, 코드가 더 유연하고 재사용 가능해지며, 설계의 안정성이 향상됩니다.

대화를 입력해주세요.



X

⋮

NL2SQL

회사의 데이터에 대해서 자연어로 질의하신다면 해당하는 BigQuery 쿼리를 생성합니다.

상세하게 질의하실 수록 정확한 답을 얻을 수 있습니다.

데이터 카탈로그는 매일 업데이트 됩니다.



2023년 9월 1일부터 2023년 9월 30일까지 '닥터핀스' 브랜드의 재고 총합을 구해줘.



```
SELECT SUM(quantity) AS total_stock
FROM company_in.daily_stock_shop_stock
WHERE date_logged BETWEEN '2023-09-01' AND '2023-09-30'
AND brand = '닥터핀스';
```



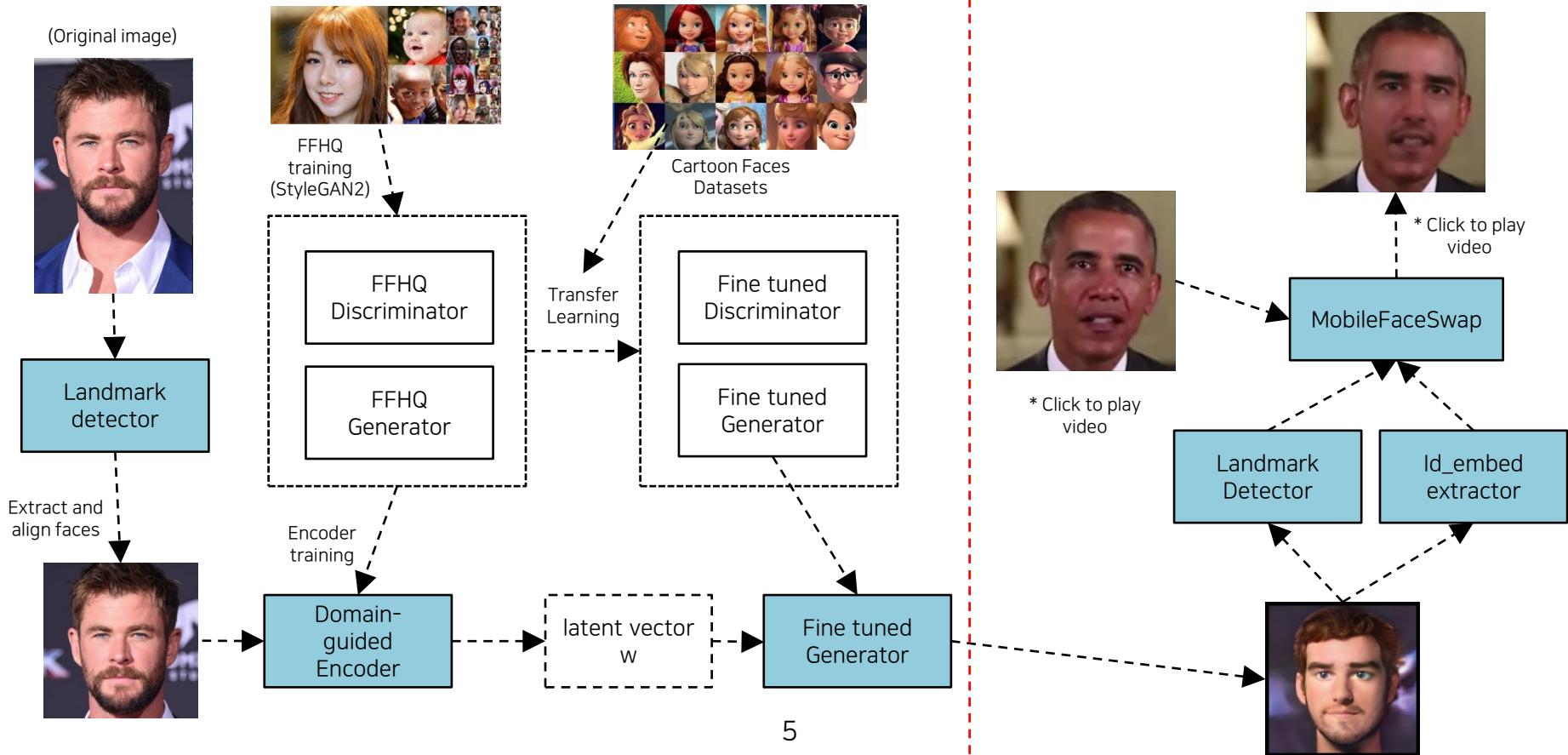
2023년 데이터로부터 전체 브랜드의 월별 재고 평균을 구해줘.

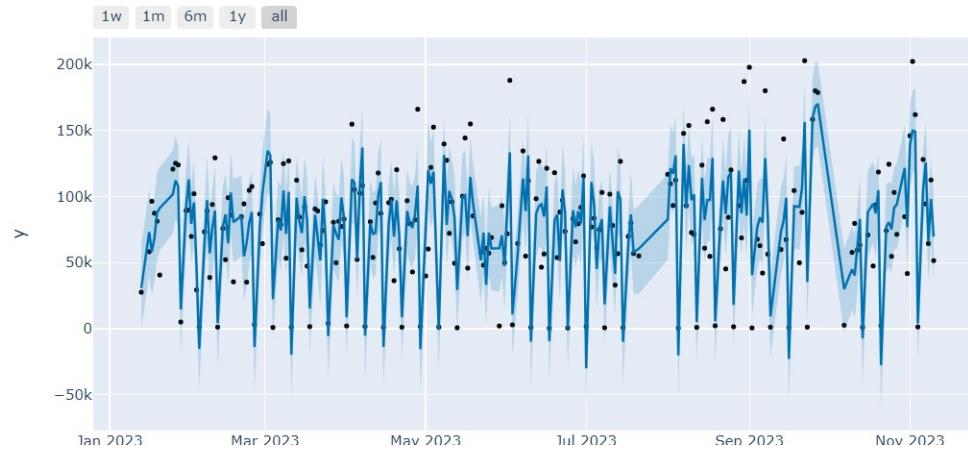
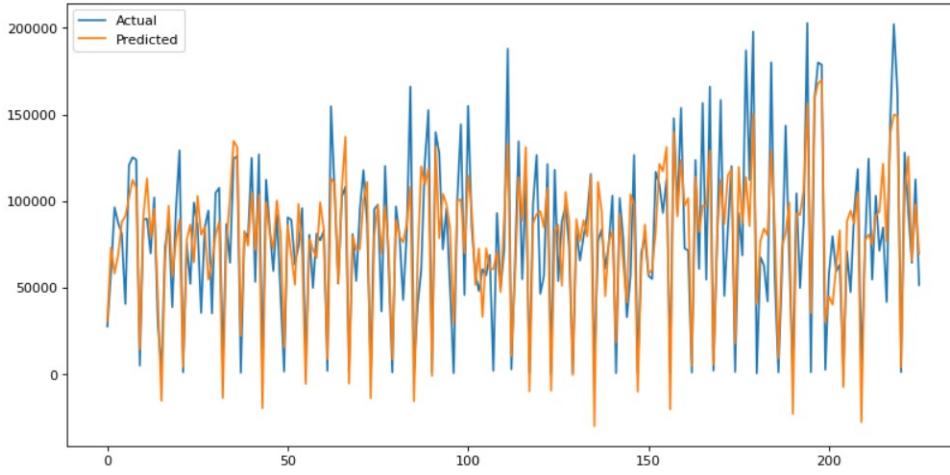


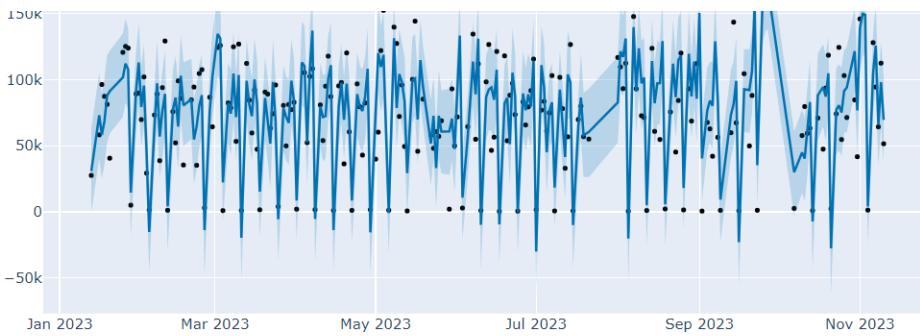
```
SELECT
    EXTRACT(YEAR FROM date_logged) AS year,
    EXTRACT(MONTH FROM date_logged) AS month,
    brand,
    AVG(quantity) AS average_stock
FROM
    company_in.daily_stock_shop_stock
WHERE
    EXTRACT(YEAR FROM date_logged) = 2023
GROUP BY
    year, month, brand
ORDER BY
    year, month;
```

| 대화를 입력해주세요.

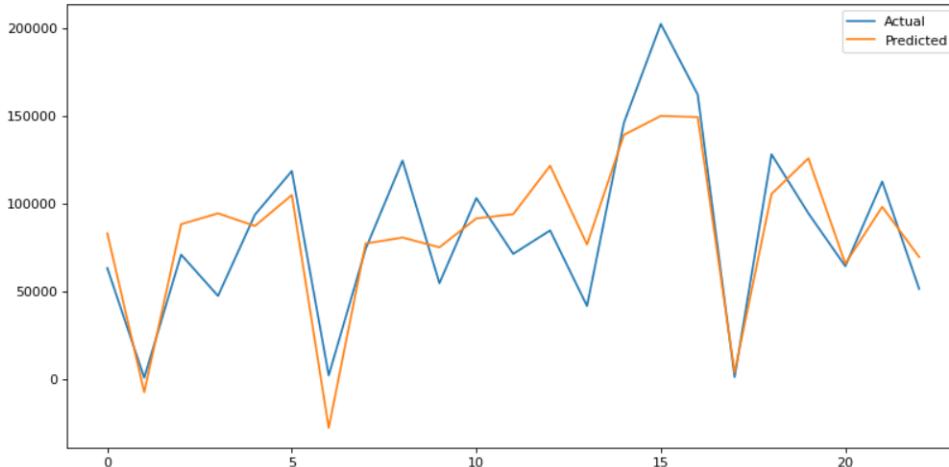


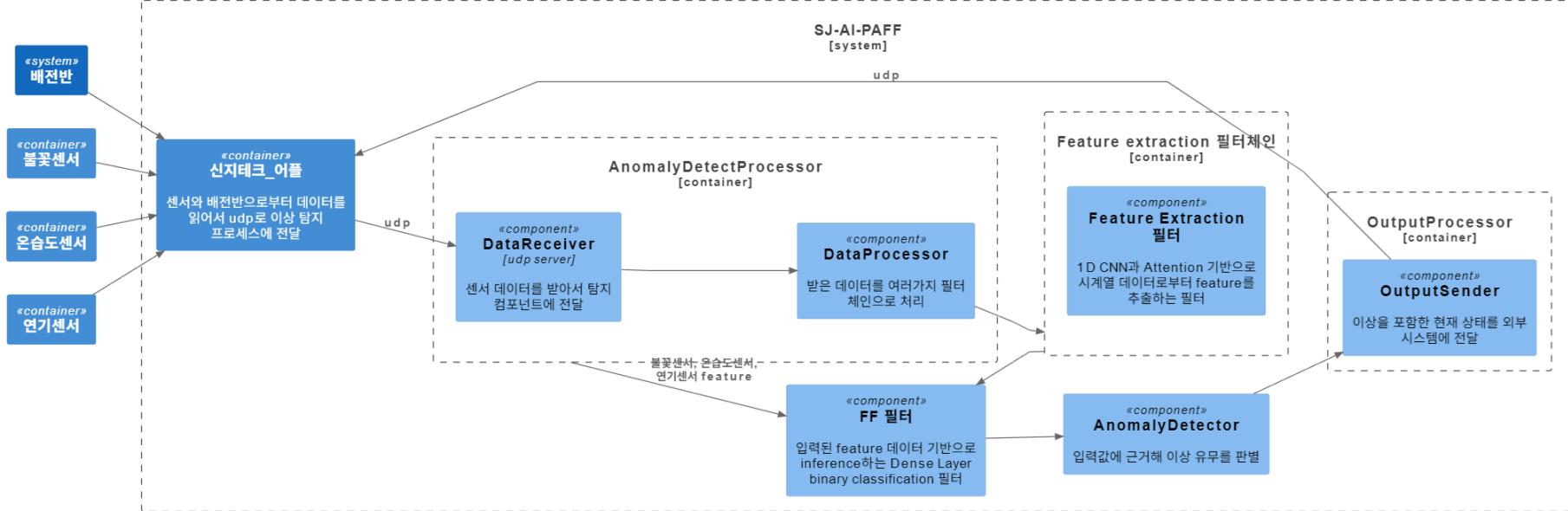






ds





하드웨어 박스 기반 로컬 LLM 솔루션

- 공유 메모리 128G 구성으로 최대 96GB VRAM 할당,
하드웨어 구성은 변경 가능
- 고객의 용도에 맞춰서 커스터마이즈 가능한 sLLM 구
동을 위한 하드웨어 박스: HW와 SW 모두 최적화
- 일반적인 챗봇, 다양한 종류의 RAG, 워크플로우 자동
화, 이미지 및 동영상 생성을 하나의 박스에서 모두 처
리할 수 있으므로 보안적인 측면에서의 문제가 원천적
으로 해소
- 추가적인 비용 없이 무제한으로 사용할 수 있으며, 사
용량 증가에 따라 유연하게 확장



Attention Mechanism Improve x +

localhost:8080/c/5b217ee4-1233-4cc9-860b-4b9b25f45ecd

Open WebUI w4 +

새 채팅

검색

노트

워크스페이스

채팅

온라인

Attention Mechanism Improve ...

Hello World

👋 Korean Greeting

새 채팅

👋 Greeting & How Are You?

어제

Hello Korean Conversation 😊

AI w4 Today at 오후 2:38

Attention 메커니즘은 입력 시퀀스의에서 각 위치의 중요도를 계산하여 가중치를 부여하는 방식으로 동작합니다. 가장 기본적인 형태의 Attention 계산 공식은 다음과 같이 설명할 수 있습니다.

Scaled Dot-Product Attention 공식

Transformer 모델에서 사용되는 **Scaled Dot-Product Attention**의 계산 과정은 다음과 같습니다:

입력

- **Query (Q): \mathbf{Q} : 입력 벡터에서 '질문'을 나타내는 벡터로, shape는 (seq_len, d_k) .
- **Key (K): \mathbf{K} : 입력 벡터에서 '키'를 나타내는 벡터로, shape는 (seq_len, d_k) .
- **Value (V): \mathbf{V} : 입력 벡터에서 '값'을 나타내는 벡터로, shape는 (seq_len, d_v) .

단계별 계산

- Dot-Product**: Query와 Key 벡터의 내적을 계산합니다.
$$scores = \mathbf{Q} \cdot \mathbf{K}^T \quad (\text{shape: } (seq_len, seq_len))$$
- Scaling**: 내적 결과의 값을 $\sqrt{d_k}$ 로 나눕니다. 이는 다음과 같이 표현됩니다.
$$\text{scaled_scores} = \frac{\text{scores}}{\sqrt{d_k}}$$
- Softmax (정규화)**: Scaled scores에 Softmax 함수를 적용하여 각 위치의 가중치를 계산합니다.
$$\text{attention_weights} = \text{softmax}(\text{scaled_scores})$$

여기서 Softmax 함수는 각 열(column)마다 적용합니다.

- Attention 계산**: Obtained weights를 Value 벡터에 곱해 ↓ 최종 Attention 출력을 만듭니다.

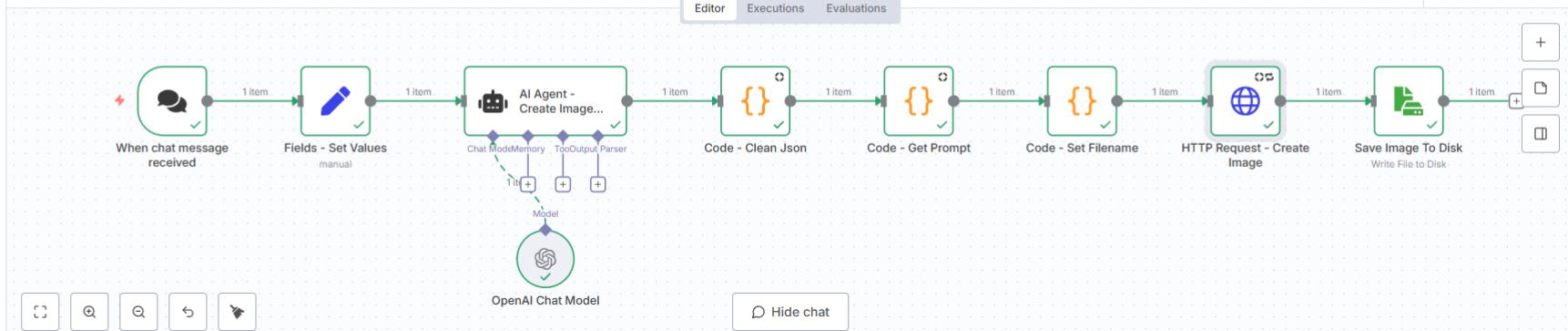
메시지 보내기

+ 코드 인터프리터

노규남

설계 Attention 계산하는 공식을 설명해줘

Overview



몸통 부위가 비어 있고 그 안에서 촛불이 타고 있는 여성을 그려줘. 촛불은 노란 불꽃을 내면서 타고 있어. 여성은 절로 금발에 머리가 길며 신비로운 분위기로 중세풍 드레스를 입고 웃고 있어. 배경은 낮의 마을이야. 최대한 사실적인 이미지로 그려주고 여성의 몸 전체가 나오도록 그려줘.

```
{
  "body": {
    "prompt": "Medieval village setting | Young woman with long blonde hair, wearing a mystical medieval dress, smiling, with a hollow torso containing a burning candle emitting a yellow flame | text elements: 'Luminary' - elegant calligraphy, integrated naturally into the dress's fabric ::7 | Cinematic dramatic lighting, natural daylight, soft shadows, depth of field blur, clean and slightly rustic environment ::7 | 8k resolution, hyperrealistic, photorealistic quality, octane render, cinematic composition -ar 16:9 --s 1000 --q 2 --chaos 20 --stylize 400",
    "image_size": {"width": "1080", "height": "1920"}, "num_inference_steps": 12, "guidance_scale": 3.5, "num_images": 1, "enable_safety_checker": true, "fileName": "c:\\work\\images\\${$('Code').item.json.fileName}"}}
}
```

Success in 55.963s | 1,304 Tokens

Fields - Set Values

AI Agent - Create Image...
OpenAI Chat Model

When chat message received

Code - Clean Json

Code - Get Prompt

Code - Set Filename

HTTP Request - Create Image

Save Image To Disk

HTTP Request - Create Image Success in 16.992s

PUT



Input Output ...

1 item

Templates

Variables

Insights

Help

What's New

Kyounam Roh

Type message, or press 'up' for prev one



SO ARM-101: 6자유도(6-DoF) 로봇팔

- Leader Arm과 Follower Arm간의 연동을 통한 Teleoperation
- 연결된 카메라와 모터의 센서를 통한 멀티 모달 데이터 수집 가능: 영상과 센서 데이터를 통합하여 학습
- Zero-shot Generalization: 사전 학습된 정책을 활용해 새로운 작업에 즉시 적용
- Behavioral Cloning 알고리즘을 적용하여 Transformer 기반 정책 네트워크(Action-Chunking-Transformer)와 Diffusion Policy를 통해 연속적인 동작을 구현



- 이 과정의 목적은 ChatGPT를 포함한 LLM 모델들을 사용하는 방법과, 이 LLM들을 활용하여 다양한 AI agent 및 AI 어플리케이션을 구현하는 방법에 대해서 숙지하는 것
- OpenAI API, 프롬프트 엔지니어링, gradio, streamlit, LangChain, LangGraph, AutoGen, docker, llama_index까지의 내용이 준비되어 있으나, 이해정도에 따라 진도는 유동적
- 교육시간은 월~목요일 오전 9시~오후 6시까지, 일일 8시간, 전체 32시간 분량
- 매 시간 정각에 시작하며, 50분 수업 후 10분 휴식을 반복
- 온라인으로 진행되는 수업 특성 상 커뮤니케이션이 적을 수 밖에 없음, 이를 보완하기 위해 수강생들 모두가 포함된 git repo(github)를 공유: 메일로 초청하여 멤버가 되며 이를 통해 필요한 파일들을 전달하고 실습, 과정 이해도를 간접적으로 평가
- 매일의 take-away가 있으며, 매일의 실습이 있음: 이전 과정의 내용을 기반으로 다음 과정을 진행하는 것이므로 실습을 꾸준하게 따라오는 것이 중요

[https://forms.gle/mY9JnHML
W8teK8p7A](https://forms.gle/mY9JnHMLW8teK8p7A)

- ChatGPT는 OpenAI에서 개발한 언어 모델(LLM)로, 자연어 이해 및 생성에 대한 혁신적인 능력을 제공
- 대화형 인터페이스를 통해 사용자와 상호작용하며, 텍스트 기반의 질문 응답, 대화 생성, 자동 요약 등 다양한 언어 작업 수행 가능
- 가입은 chatgpt.com, 무료계정은 ChatGPT로 모델이 고정되어 있으며, 한달에 \$20의 비용을 지불하면 ChatGPT plus로 모델을 업그레이드할 수 있음: 모델의 성능이 좀더 우수하며, 여러가지 플러그인을 활용하여 대화 외에 좀더 상세한 작업에 사용 가능
- 무료, 유료 모두 일정 수준의 request rate limit이 있으며, 그 이상의 요청에 대해서는 블록처리됨
- API로도 접근가능하나 API는 사용하는 token 양에 비례하는 요금을 과금
- 신용카드를 등록한 신규계정에 대해서는 3개월 기한의 \$18 -> \$5의 무료 크레딧이 제공되었으나 현재는 \$0로 시작: 일단 크레딧을 충전하고 난 후에 사용할 수 있음(최소 충전은 \$5)

- ChatGPT는 현재 나와 있는 언어 모델 중 가장 우수한 종류지만 ChatGPT만으로 AI 서비스를 만들 수는 없음: 실제 서비스 구축을 위해서는 LLM 외 다른 요소들이 상당히 많이 필요하다.
- ChatGPT를 실제 서비스에 적용하려면 부족한 점들이 상당히 있으며, 가격이 내려감과 동시에 점점 더 성능이 낮아지고 있다. API도 계속 변경된다. ChatGPT를 사용하려면 이런 변화에 지속적으로 대응해야 한다.
- 사용량이 많아지면 그에 비례하여 비용이 증가하며, 채팅을 통해 회사 내부의 정보가 외부로 흘러나갈 수 있다는 점을 고려해야 한다: 자체 인프라를 구축하거나 인터페이스에 필터를 적용할 수도 있다(가드레일).
- AI 서비스를 구축하는 방법에 대해서는 비교적 잘 알려져 있는 편이나, 서비스하기에 충분한 품질을 얻는 것은 생각보다 많이 어렵다. 대기업들(예. SK 텔레콤)도 비슷하게 혼매고 있다.
- AI 서비스를 구축할 때는 반드시 원하는 성능 지표를 설정하고, 측정을 위한 데이터셋을 준비해야 한다.
- ChatGPT의 비용은 성능을 고려할 때 매우 낮은 편이나, 만약 호출이 매우 많아지거나 context를 매번 길게 사용한다면 비용은 빠르게 증가한다. 항상 상위 모델을 사용할 필요는 없으며 적절한 모델을 선택하면 이런 비용은 최적화될 수 있다.

- ChatGPT는 API만 연결하면 되므로 특정 환경에 구애받지 않음
- 다만 결과를 확인하기 위해서는 좀더 interactive한 환경이 적합: jupyter notebook이나 Google colab
- Google colab 세션은 계속 유지되지 않으므로 사용하기에 적당하지 않으며, git repo 연동도 어려움
- 이 과정에서는 로컬 PC에 wsl을 설치하고, wsl에서 python환경을 구축한 후 jupyter notebook을 실행
- WSL2는 Windows Subsystem for Linux2로, 윈도우에서 리눅스 어플리케이션을 구동하게 해주는(정확하게 말하면 Linux system call을 처리해주는) 호환성 레이어로, 리눅스 커널의 기능을 제공
- WSL2를 사용하면 윈도우안에서 리눅스 배포본을 구동할 수 있고, 리눅스 어플리케이션도 모두 실행할 수 있음: 거의 대부분의 어플리케이션을 무리 없이 실행 가능
- 여러가지 오픈소스 프로젝트들이 리눅스를 기준하여 개발되고 있으므로, 윈도우에서의 설치로는 한계
- 가상화 기능이 있는 Windows 10 이상의 PC에서는 모두 사용 가능: Mac 사용자는 wsl을 사용할 필요가 없으며 자체적으로 python과 pip, jupyter, git을 설치해서 사용하면 됨

- 제어판 > 프로그램 > 프로그램 및 기능 > Windows 기능 켜기 또는 끄기
- Linux용 윈도우 하위시스템 켜기, 가상 머신 플랫폼 켜기
- 재부팅 후 Microsoft Store에서 Ubuntu를 검색하고 22.04를 선택하여 설치
- cmd.exe 실행
- wsl --install Ubuntu-22.04로 설치, ubuntu 계정 생성하고 암호 설정
- wsl -l -v로 현재 설치된 배포본 확인
- wsl -d Ubuntu-22.04로 wsl 콘솔 진입
- 종료는 wsl --terminate Ubuntu-22.04
- 배포본 삭제는 wsl --unregister Ubuntu-22.04

- 1) github 계정 생성
- 2) wsl 로그인, 작업 디렉토리 생성

```
mkdir ~/work  
cd ~/work
```

- 3) 계정 설정

```
git config --global user.name "Bard K. Roh"  
git config --global user.email "bardroh@weable.ai"  
git config --global --list
```

4-1) ssh키 설정 & git clone

ssh-keygen

cat ~/.ssh/id_rsa.pub

화면의 내용을 전부 복사해서 github의 Settings-SSH Keys에 붙임

git clone git@github.com:weable-kosa/kosa-ai-agent-2026-1st.git

4-2) github token 설정 & git clone

https clone시에는 basic token 설정, repo 관련 모든 권한 할당

git clone https://[사용자 id]:[access token]@github.com/weable-osa/kosa-ai-agent-2026-1st.git

5) 작업하고 git add; git commit; git push, 작업 전 항상 git pull (본인 폴더만 작업)

- <https://platform.openai.com/>에 접속하여 가입하고 카드 등록. Google, Apple, MS 계정으로 가입 가능
- token 사용량에 따라 다음 달에 과금하는 구조였으나, Credit Balance에서 충전하고 이를 소진하는 방식으로 변경되었음: Credit Balance가 \$0.00이 되면 rate limit이 걸렸을 때와 유사하게 429 에러가 발생, 최소 충전 금액은 \$5
- Auto recharge 옵션을 적용해 두면, Credit Balance가 일정 수준 이하로 떨어졌을 때 자동으로 일정액씩 충전하도록 해둘 수 있음
- API key를 생성하여 연동할 수 있으며 한번 생성된 key는 다시 보여지지 않음: 삭제하고 재생성
- 사용을 위해 python의 openai 패키지 설치가 필요(pip3 install openai)

- 1) sudo apt-get update
- 2) sudo apt-get install python3 python3-pip -y
- 3) VS Code 다운로드, 설치
- 4) VS Code Extension에서 WSL을 찾아서 설치 후 종료
- 5) 이후 WSL 실행

```
cd <git repo 경로>
```

```
code . # VS Code가 자동으로 실행됨
```

- 6) VS Code Extension에서 Python, Jupyter 등 권고하는 플러그인 설치: Install in WSL 을 확인할 것

Visual Studio Code - Code Edit

code.visualstudio.com

Visual Studio Code Docs Updates Blog API Extensions FAQ Learn Search Docs Download

Version 1.82 is now available! Read about the new features and fixes from August.

Code editing. Redefined.

Free. Built on open source. Runs everywhere.

Download for Windows
Stable Build

Web, Insiders edition, or other platforms

By using VS Code, you agree to its license and privacy statement.

EXTENSIONS: MARKETPLACE

- Python 2019.6.24221 54.9M 4.5 Microsoft [Install](#)
- GitLens — Git sup... 9.8.5 23.1M 4.5 Eric Amadio [Install](#)
- C/C++ 0.24.0 23M 3.5 C/C++ IntelliSense, debugging, and Microsoft [Install](#)
- ESLint 1.9.0 21.9M 4.5 Integrates ESLint JavaScript into V... Dirk Baumer [Install](#)
- Debugger for Ch... 4.11.6 20.6M 4 Microsoft [Install](#)
- Language Supp... 0.47.0 18.7M 4.5 Java Linting, Intellisense, formatting, ... Red Hat [Install](#)
- vscode-icons 8.8.0 17.2M 5 Icons for Visual Studio Code VSCode Icons Team [Install](#)
- Vetur 0.21.1 17M 4.5 Vue tooling for VS Code Pine Wu [Install](#)
- C# 1.21.0 15.6M 4 C# for Visual Studio Code (powered ... Microsoft [Install](#)

File Edit Selection View Go Debug Terminal Help serviceWorker.js - create-react-app - Visual Studio Code - In...

src > JS serviceWorker.js > register > window.addEventListener('load') callback

```
checkValidServiceWorker(swUrl, config);  
  
// Add some additional logging to localhost, p...  
// service worker/PWA documentation.  
  
navigator.serviceWorker.ready.then(() => {  
  product  
  productSub  
  removeSiteSpecificTrackingException  
  removeWebWideTrackingException  
  requestMediaKeySystemAccess  
  sendBeacon  
  serviceWorker (property) Navigator.serviceWorke...  
  storage  
  storeSiteSpecificTrackingException  
  storeWebwideTrackingException  
} userAgent  
}  
  
function registerValidSW(swUrl, config) {  
  navigator.serviceWorker  
    .register(swUrl)  
    .then(registration => {  
      Local: http://localhost:3000/  
      On Your Network: http://10.211.55.3:3000/  
  
Note that the development build is not optimized.
```

TERMINAL 1: node

You can now view create-react-app in the browser.

Local: http://localhost:3000/
On Your Network: http://10.211.55.3:3000/
Note that the development build is not optimized.

Ln43, Col19 Spaces:2 UTF-8 LF JavaScript

File Edit Selection View Go Run Terminal Help ⏪ ⏩ Search

EXPLORER NO FOLDER OPENED You have not yet opened a folder. Open Folder Opening a folder will close all currently open editors. To keep them open, add a folder instead. Clone Repository You can clone a repository locally. Clone Repository To learn more about how to use git and source control in VS Code read our docs.

Start

- New File...
- Open File...
- Open Folder...
- Clone Git Repository...
- Connect to...

Recent

You have no recent folders, [open a folder](#) to start.

Welcome

Visual Studio Code

Editing evolved

Recommended

 GitHub Copilot Supercharge your coding experience for as little as \$10/month with cutting edge AI code generation.

Walkthroughs

 Get Started with VS Code Discover the best customizations to make VS Code yours.

 Learn the Fundamentals

 Boost your Productivity

Show welcome page on startup

File Edit Selection View Go Run Terminal Help

Extension: Jupyter

Jupyter v2024.2.0

Microsoft [microsoft.com](#) | ⚡ 75,659,524 | ★★★★☆ (304)

Jupyter notebook support, interactive programming and computing that supports Intellisense, debugging and more.

Disable Uninstall Switch to Pre-Release Version

Extension is enabled on 'WSL: Ubuntu-22.04'

DETAILS FEATURES CHangelog

Extension Pack (4)

- Jupyter Keymap
- Jupyter Notebook Renderers
- Jupyter Slide Show
- Jupyter Cell Tags

CATEGORIES

- Extension Packs
- Data Science
- Machine Learning
- Notebooks
- Visualization

RESOURCES

- Marketplace
- Issues
- Repository

PROBLEMS OUTPUT

Activation time: 8ms

Extension is enabled on 'WSL: Ubuntu-22.04'

WSL: Ubuntu-22.04 main 2 △ 0 WSL: Ubuntu-22.04

The screenshot shows the Microsoft Visual Studio Code (VS Code) interface with the Extensions Marketplace open. The search bar at the top contains the text "Extension: Jupyter". The results list several Jupyter-related extensions, with the top result, "Jupyter", circled in red. The "Jupyter" extension page is displayed on the right, showing its details: version v2024.2.0, developer Microsoft, popularity (75,659,524), and a rating of 3 stars from 304 reviews. It also mentions that the extension supports Intellisense, debugging, and more. Below the main extension page, a "Extension Pack (4)" section lists four related extensions: "Jupyter Keymap", "Jupyter Notebook Renderers", "Jupyter Slide Show", and "Jupyter Cell Tags", all developed by Microsoft. The "Jupyter Slide Show" extension is highlighted with a green border. The bottom of the page shows activation statistics: "Activation time: 8ms" and "Extension is enabled on 'WSL: Ubuntu-22.04'". On the left sidebar, there are icons for various extensions categories like Data Science, Machine Learning, and Notebooks. The bottom status bar shows the current workspace as "WSL: Ubuntu-22.04" and the file name "main".

File Edit Selection View Go Run Terminal Help

EXTENSIONS: MARKETPLACE

WSL

Open any folder in the Windows Subsystem for Linux (WSL) and take advanta... Microsoft

WSL workspaceFolder

Returns `workspaceFolder` as a WSL path. Useful where WSL format workspace... Ifurzewadock

Linux/Unix/WSL paths

Provides commands to have unix-like paths (with slashes instead of backslash... Krzysztof Grzywocz

Remote Development

An extension pack that lets you open any folder in a container, on a remote m... Microsoft

Open (WSL support)

Opens the current file with the default application for the OS (in Windows for... NoThInG

wsl-split

Iamjava

WSL Path Tools

Tools to convert paths between Windows and WSL format Rob Clouth

OCaml and Reason IDE - for WSL

OCaml and Reason language support for WSL raiscui

Save and Run with wsl

Run commands when a file is saved in Terminal (add wsl path) raiscui

Json Editor for VS code - WSL

Generate a tree view editor of the active JSON document pick68

Laravel Artisan

Run Laravel Artisan commands within Visual Studio Code Ryan Naddy

WSLSchoolAux

Bruno School

WSL: Recommender

Recommendations using the Windows Subsystem for Linux (WSL) and the WSL ext...

7. LCELipynb 8. Assistant-math.ipynb Extension: WSL

WSL v0.86.0 Microsoft microsoft.com 25,383,614 ★★★★★ (77)

Open any folder in the Windows Subsystem for Linux (WSL) and take advantage of Visual Studio Code's full feature set.

Disable Uninstall

This extension is enabled globally.

DETAILS FEATURES CHangelog

Visual Studio Code WSL

The **WSL extension** lets you use VS Code on Windows to build Linux applications that run on the [Windows Subsystem for Linux \(WSL\)](#). You get all the productivity of Windows while developing with Linux-based tools, runtimes, and utilities.

The **WSL extension** lets you use VS Code in WSL just as you would from Windows.

Why do I need the WSL extension?

Why WSL?

WSL lets you run a Linux environment -- including command-line tools and applications -- directly on Windows, without the overhead of a traditional virtual machine or dualboot setup. WSL especially helps web developers and those working with Bash and Linux-first tools (i.e. Ruby, Python) to use their toolchain on Windows and ensure consistency between development and production environments.

When you install a version of Linux on Windows, you're getting a full Linux environment. It's isolated from Windows- the UI is the

PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER

```
ubuntu@DESKTOP-JJ3PK15:~/work/kosa-chatgpt-2024$ git pull
Already up to date.
ubuntu@DESKTOP-JJ3PK15:~/work/kosa-chatgpt-2024$
```

Categories

Other

Resources

Marketplace Issues Repository License Microsoft

More Info

Published 2019-05-03, 03:40:03 Last released 2024-01-27, 02:31:24 Last updated 2024-03-01, 10:43:51 Identifier ms-vscode-wsl bash - kosa-chatgpt-2024

Billing overview - OpenAI API

platform.openai.com/account/billing/overview

Billable API requests

Playground Assistants Fine-tuning API keys Storage Usage Settings Organization Team Limits Billing Profile Documentation Help All products Weable, Inc

Billing settings

Overview Payment methods Billing history Preferences

Pay as you go

Credit balance ①

\$0.00

Auto recharge is off

When your credit balance reaches \$0, your API requests will stop working. Enable automatic recharge to automatically keep your credit balance topped up.

Enable auto recharge

Add to credit balance Cancel plan

Payment methods

Add or change payment method

Billing history

View past and current invoices

Preferences

Manage billing information

Usage limits

Set monthly spend limits

Pricing

View pricing and FAQs

Model	Input	Cached input	Output
gpt-5	\$1.25	\$0.13	\$10.00
gpt-5-mini	\$0.25	\$0.03	\$2.00
gpt-5-nano	\$0.05	\$0.01	\$0.40
gpt-5-chat-latest	\$1.25	\$0.13	\$10.00
gpt-4.1	\$2.00	\$0.50	\$8.00
gpt-4.1-mini	\$0.40	\$0.10	\$1.60
gpt-4.1-nano	\$0.10	\$0.03	\$0.40
gpt-4o	\$2.50	\$1.25	\$10.00
gpt-4o-mini	\$0.15	\$0.08	\$0.60
gpt-realtime	\$4.00	\$0.40	\$16.00
gpt-4o-realtime-preview	\$5.00	\$2.50	\$20.00
gpt-4o-mini-realtime-preview	\$0.60	\$0.30	\$2.40
gpt-audio	\$2.50	-	\$10.00
gpt-4o-audio-preview	\$2.50	-	\$10.00
gpt-4o-mini-audio-preview	\$0.15	-	\$0.60
o1	\$15.00	\$7.50	\$60.00
o1-pro	\$150.00	-	\$600.00
o3-pro	\$20.00	-	\$80.00
o3	\$2.00	\$0.50	\$8.00
o3-deep-research	\$10.00	\$2.50	\$40.00
o4-mini	\$1.10	\$0.28	\$4.40
o4-mini-deep-research	\$2.00	\$0.50	\$8.00
o3-mini	\$1.10	\$0.55	\$4.40
o1-mini	\$1.10	\$0.55	\$4.40
codex-mini-latest	\$1.50	\$0.38	\$6.00
gpt-4o-mini-search-preview	\$0.15	-	\$0.60
gpt-4o-search-preview	\$2.50	-	\$10.00
computer-use-preview	\$3.00	-	\$12.00
gpt-image-1	\$5.00	\$1.25	-

| 참고

모델	Input /1M	Output /1M	Context window	Max output tokens	비고
o3-pro	20	40	200K	100K	
o1-pro	150	600	200K	100K	
o4-mini	1.1	4.40	200K	100K	
o3-mini	1.1	4.40	200K	100K	
o1-mini	1.1	4.40	32K	64K	
o1	15.0	60.0	32K	32K	
gpt-5-nano	0.05	0.4	400K	128K	분류, 요약 특화
gpt-5-mini	0.25	2.0	400K	128K	
gpt-5	1.25	10.0	400K	128K	
gpt-4o	2.5	10.0	128K	16K	
gpt-4o-mini	0.15	0.6	128K	16K	
gpt-4-turbo	10	30	128K	4K	프롬프트 캐싱 없음
gpt-4	30	60	8K	8K	프롬프트 캐싱 없음
gpt-3.5-turbo	0.5	1.5	16K	4K	프롬프트 캐싱 없음
Code Interpreter			N/A		0.03/session
File Search			N/A		0.10/GB per day (1GB free)

- OpenAI에서 직접 계정을 만들고 API를 연동할 수 있는데, MS Azure 클라우드에서 거의 동일한 모델들을 거의 동일한 가격으로 판매하고 있음
- OpenAI 대비 MS Azure OpenAI Service가 갖는 기본적인 장점이라면, US West 단일리전이 아닌 다른 리전 선택이 가능하며(도쿄 리전이 가장 가까움) 여러가지 클라우드의 보안조치가 되어 있다는 점
- Azure OpenAI의 모델들도 지역 할당량이 있는데, gpt-3.5-turbo 모델 기준하여 분당 120K~300K까지 다양함. gpt-4인 경우는 분당 20~40K
- 할당량이 부족해지게 되면, Azure 포털에서 할당량 증가를 요청할 수 있으며, 수락되면 더 많은 token을 사용할 수 있게 됨: OpenAI를 대규모 서비스에 적용할 경우 가장 문제가 되는 부분이 이 할당량 quota
- MS Azure의 OpenAI 모델은 별도의 품으로 신청해야 하며, Azure의 체험 계정으로는 사용이 어려움

| OpenAI vs Azure OpenAI

Feature	OpenAI	Azure OpenAI
Security & Data Privacy	Basic Security	Enterprise Security, RBAC, Customer-Managed Keys, Private Link
Compliance	None	SOC2, ISO, HIPAA, CSA STAR
Reliability	No SLA (yet)	Azure SLA, Dedicated Capacity Option (soon)
Responsible AI	Separate Safety Classifier (adds latency)	Built-in, enterprise-grade, low latency moderation and harm prevention
Holistic Solution	Advanced LLM & Image Generation, Basic Speech	OpenAI Models, Complete AI Solution, and a Complete PaaS
Service Region	US West (샌프란시스코) Only	East US Region, South Central US Region, West Europe Region 기타

OpenAI playground

The screenshot shows the OpenAI Playground interface. On the left, there's a sidebar titled "Get started" with instructions and a "KEEP IN MIND" section containing several tips. The main area is titled "Playground" and contains a "SYSTEM" panel (empty) and a "USER" panel where users can enter messages. To the right of the panels are various configuration options: Mode (set to Chat), Model (set to gpt-3.5-turbo), Temperature (set to 1), Maximum length (set to 256), Stop sequences (empty input field), Top P (set to 1), Frequency penalty (set to 0), and Presence penalty (set to 0). At the bottom right, a note states: "API and Playground requests will not be used to train our models. Learn more". A "View code" button is circled in red at the top right of the configuration area.

| Completion mode (deprecated)

```
# Completion mode, 지시와 명령을 하나의 prompt로 전달
import os
import openai

openai.api_key = os.getenv("OPENAI_API_KEY")
# 텍스트 생성 요청을 보냅니다.
response = openai.completions.create(
    model="gpt-3.5-turbo-instruct",
    prompt="다음을 일본어로 번역하세요: 안녕하세요? 오늘 날씨가 참 좋군요!",
    max_tokens=256, # 생성된 텍스트의 최대 길이를 설정합니다.
)
# 생성된 텍스트를 출력합니다.
print(response.choices[0].text)
```

표준적인 Chat completion mode

```
import openai
```

```
res = openai.chat.completions.create(  
    model="gpt-4o-mini",  
    messages=[{"role": "user", "content": "다음을 일본어로 번역하세요: 안녕하세요? 오늘 날씨가 참 좋군  
요!"}],  
    temperature=0.5,  
    max_tokens=256,  
    top_p=1,  
    frequency_penalty=0,  
    presence_penalty=0  
)  
print(res.choices[0].message.content)
```

- system: 주로 채팅 대화의 시작 부분에 사용되며, 시스템 역할로 제공된 메시지는 Assistant에게 초기 컨텍스트를 제공하고 작동 방식을 지정. 예) 당신은 과학에 대한 전문가입니다.
- user: 사용자 역할은 API를 호출하는 클라이언트 측 역할. 이 역할은 Assistant에게 질문을 하거나 지시를 내릴 때 사용. 예) 과학 분야에서 어떤 발전이 있었나요?
- assistant: Assistant는 사용자 역할로부터 받은 메시지에 대답하고 정보를 제공하는 역할. 예) 새로운 에너지 저장 기술에 대해 이야기해드릴게요.
- function: Function 역할은 Assistant가 호출로 실행할 작업을 정의할 때 사용. 주로 외부의 API와 LLM을 연동하기 위한 목적으로 사용됨

- temperature: 생성 시 다양성을 조절하는 매개변수 (0~1)
- max_tokens: 생성된 텍스트의 최대 토큰 수
- stop: 종료 토큰(End-of-Sequence)을 나타내는 토큰의 리스트
- top_p: 다음 단어 선택 시 고려되는 확률 분포의 크기를 지정하는 매개변수 (0.1~1) 예) 0.8 -> 상위 80%의 확률을 가진 token 중 다음 단어를 선택
- frequency_penalty: 단어 반복에 대해서 패널티를 주어서 반복하지 않게 하는 매개변수
- presence_penalty: 새 토큰에 대한 출현빈도를 낮추기 위한 매개변수

```
gen = openai.chat.completions.create(  
    model="gpt-4o-mini",  
    messages=  
    ...  
    stream=True  
)
```

```
while True:  
    response = next(gen)  
    delta = response.choices[0].delta  
    if delta.content is not None:  
        print(delta.content, end="")  
    else:  
        break
```

- 프롬프트 엔지니어링은 다양한 애플리케이션과 연구 주제에 언어모델(LM)을 효율적으로 사용할 수 있도록 프롬프트를 개발하고 최적화하는 비교적 새로운 분야
- 연구자는 프롬프트 엔지니어링을 사용하여 질의응답 및 산술 추론과 같은 일반적 작업부터 복잡한 작업까지 다양한 범위에서 LLM의 역량을 향상
- 프롬프트 엔지니어링은 LLM과 상호 작용하고 개발하는 데 유용한 다양한 기술과 기법을 포함
- LLM과 인터페이스를 형성하고, 빌드하고, LLM의 기능을 이해하는 데 중요한 기술
- LLM에 따라 프롬프트의 유효성은 차이가 있음: 대부분의 프롬프트 엔지니어링은 ChatGPT를 기준으로 설명 하지만 llama나 다른 LLM 들에는 잘 적용되지 않을 수 있음. 작은 모델일수록 상세한 프롬프트를 잘 이해하지 못하는 경향

- LLM이 최종 답변을 내리기 전에 중간 추론 단계를 명시적으로 보여주는 프롬프팅 기법
- 복잡한 문제 해결력 향상, 추론 과정의 투명성 확보 등의 장점
- "Let's solve this step by step" 같은 명시적 지시어 사용, 중간 사고 과정 표현
- 수학 문제 해결, 논리 추론, 의사결정, 코딩 문제 등에 사용
- Zero-shot CoT, Few-shot CoT, Self-consistency CoT 등 다양한 변형 기법 개발
- 최근에는 LLM 자체를 CoT 형식의 데이터로 학습한 모델들도 나오고 있으며, o1-mini와 o1-preview가 대표적인 모델임. 이런 모델을 추론(reasoning) 모델이라고도 함
- CoT에 맞게 학습된 o1계열 모델들은 CoT 프롬프트를 쓰지 않는 편이 오히려 좋은 결과가 나올 가능성을 높임

- 명확하고 직접적이며 상세하게 표현하기
- 예시 사용하기 (멀티샷): 1-shot, few shot으로 예시를 보고 출력하게 유도
- Claude가 생각하게 하기 (CoT)
- XML 태그 사용하기: XML이나 JSON으로 출력을 제어하면 더 구조화된 출력을 얻을 수 있음
- Claude에게 역할 부여하기 (시스템 프롬프트): LLM에게 역할을 부여할 때 더 작업을 잘 수행
- Claude의 응답 미리 채우기(Prefill): Assistant의 입력을 미리 일부 채운 후 실행
- 복잡한 프롬프트 연결하기(프롬프트 체이닝): 긴 프롬프트를 여러 번에 나누어서 실행
- 긴 컨텍스트 팁: 긴 문서와 입력을 프롬프트 최상단에 배치, XML 태그로 문서내용과 메타데이터 구조화, 인용구로 응답 근거를 제시한 후 최종 답변을 하게 유도

sharebook-kr/pykrx: KRX 주식 정보 스크래핑

github.com/sharebook-kr/pykrx

Type to search

Code Issues Pull requests Actions Projects Security Insights

pykrx Public Watch 21 Fork 200 Star 546

master 6 branches 0 tags Go to file Add file Code

mr-yoo 수정 주가 조회시 Naver에서 데이터를 가져오도록 원복함 (#162) ... 2ef0fe2 on Aug 5 144 commits

pykrx 수정 주가 조회시 Naver에서 데이터를 가져오도록 원복함 (#162) last month

tests ETF의 get_trading_volume_and_value 함수의 수정 (#132) 4 months ago

.gitignore add gitignore for python distribution files last year

readme.MD ETF의 get_trading_volume_and_value 함수의 수정 (#132) 4 months ago

requirements.txt ETF의 get_trading_volume_and_value 함수의 수정 (#132) 4 months ago

setup.py pypi에 TTF 파일 추가 2 months ago

readme.MD

PyKrx

About

KRX 주식 정보 스크래핑

Readme

Activity

546 stars

21 watching

200 forks

Report repository

Releases

No releases published

Packages

 이 누리집은 대한민국 공식 전자정부 누리집입니다.

검색어를 입력해 주세요.



로그인 회원가입 사이트맵 ENGLISH

DATA . GO . KR

데이터찾기

국가데이터맵

데이터요청

데이터활용

정보공유

이용안내

오픈API 상세

[f](#) [t](#) [w](#) URL 복사

XML

JSON 한국관광공사_고캠핑 정보 조회서비스_GW

활용신청

한국관광공사 고캠핑 홈페이지에서 제공하는 캠핑장 정보를 제공합니다.
한국관광공사 고캠핑 홈페이지에서 제공하는 캠핑장 정보를 제공합니다.



7



2



관심

OpenAPI 정보

메타데이터 다운로드

▲ 오픈API 에러코드

데이터 개선요청

오류신고 및 문의

분류체계	문화체육관광 - 관광	제공기관	한국관광공사
관리부서명	디지털인프라팀	관리부서 전화번호	033-371-2733
API 유형	REST	데이터포맷	JSON+XML
활용신청	800	키워드	야영장, 카라반, 글램핑
등록	2022-06-30	수정	2022-12-13
비용부과유무	무료	신청가능 트래픽	개발계정 : 1,000 / 운영계정 : 활용사례 등록시 신청하면 트래픽 증가 가능
심의유형	개발단계 : 자동승인 / 운영단계 : 심의승인		

- Gradio: 사용자 친화적인 인터페이스를 통해 AI 모델 배포와 상호 작용을 단순화하는 Python 라이브러리
- 간편한 모델 배포: Gradio를 사용하면 딥러닝 및 기계 학습 모델을 간편하게 웹 애플리케이션으로 배포할 수 있음
- 다양한 입력 형식: 텍스트, 이미지, 오디오, 비디오 등 다양한 입력 형식을 모델에 제공할 수 있으며, 모델의 예측 결과를 즉시 확인 가능
- 웹 기반 사용자 인터페이스: 사용자 친화적인 웹 기반 인터페이스를 통해 모델에 입력을 제공하고 결과를 시각적으로 확인할 수 있음
- 간단한 코드 통합: 몇 줄의 코드로 Gradio를 사용하여 모델을 빠르게 웹 서비스로 변환할 수 있음
- 확장성: Gradio는 다양한 환경에서 사용할 수 있으며, 클라우드, 로컬 서버, 도커 컨테이너 등에서 확장 가능

- gr.interace는 Gradio 인터페이스와 python function을 연동
- Interface Class는 세가지 파라메터를 가짐
 - fn: UI에 적용되는 함수
 - inputs: 입력으로 사용되는 컴포넌트(복수개 가능)
 - outputs: 출력으로 사용되는 컴포넌트(복수개 가능)
- fn(inputs) 함수가 리턴한 결과는 outputs에 반영된다, 라는 매우 간단한 원칙이 있으며, 이 원칙은 더 복잡한 인터페이스, 복잡한 함수에도 동일하게 적용됨

- Jupyter notebook의 셀 내에서 실행 가능: 실행할 때마다 포트를 바꿔가면서 어플리케이션 구동
- python3 app.py로 어플리케이션 구동
- gradio app.py로 구동: 'python' 어플리케이션을 사용하므로 다음과 같이 symbolic link를 걸어둠

```
ln -sf /usr/bin/python3 /usr/bin/python
```

- gradio로 실행하면 소스의 변경을 감지하여 자동으로 반영해줌: 어플리케이션을 중단하고 다시 실행해야 하는 수고가 없음

- Streamlit은 파이썬 기반의 오픈 소스 라이브러리로, 데이터 과학자 및 개발자들이 빠르게 웹 기반의 대화형 데이터 애플리케이션을 만들 수 있게 도와주는 역할
- Gradio 보다 코드는 좀더 길어지나, 인터페이스에 발생하는 이벤트 처리 등을 더 섬세하게 처리할 수 있다는 장점이 있음
- Streamlit은 프론트엔드와 백엔드로 나누어져 있으며, 프론트엔드에서 뭔가 변경이 발생하게 되면(예: 사용자가 라디오 버튼을 선택, 텍스트를 입력) 백엔드의 코드는 처음부터 한번 '다시' 실행됨. 이 작업을 매번 반복
- 매번 백엔드 코드들이 실행될 때 기존의 데이터들을 잃지 않기 위해 'session'이라는 객체를 사용하는데, 이 객체에 Streamlit 프론트엔드 컴포넌트들의 상태도 저장되어 있음
- 따라서 개발자는 UI를 구성하고, 백엔드의 python 코드는 항상 처음부터 다시 실행된다고 가정하여 session으로부터 데이터를 다시 읽는 식으로 구성해야 함
- 한번만 초기화해야 하는 객체는 `st.cache_data`나 `st.cache_resource`로 캐쉬에 넣을 수 있음

- 설치

```
pip install streamlit
```

- 실행

```
streamlit run <실행할 python 파일명>
```

예) streamlit run plotting_demo.py

- streamlit은 실제로는 ~/.local/bin/streamlit이라는 이름으로 저장되는 python 스크립트로, streamlit.web.cli.main이라는 함수를 실행하도록 되어 있음. 따라서 배포시에도 동일하게 'streamlit run' 을 실행해도 되고, 아니면 이 함수를 사용해도 됨

```
#!/usr/bin/python3

# -*- coding: utf-8 -*-

import re

import sys

from streamlit.web.cli import main


if __name__ == '__main__':

    sys.argv[0] = re.sub(r'(-script\b|.pyw|\.exe)?$', "", sys.argv[0])

    sys.exit(main())
```

streamlit hello # Streamlit 데모를 실행

streamlit run plotting_demo.py # 기본적인 실행 방법, 웹은 8501 포트로 열림

streamlit run --server.port 8000 plotting_demo.py # 웹을 8000 포트로 구동

python -m streamlit run plotting_demo.py # python의 모듈로 실행하는 방식

streamlit run https://raw.githubusercontent.com/streamlit/demo-uber-nyc-pickups/master/streamlit_app.py # 원격지의 코드를 실행하는 방식

Streamlit 설정은 다음 네가지 방식으로 지정 가능

- ~/.streamlit/config.toml 파일에 저장
- 현재경로/.streamlit/config.toml 파일에 저장
- STREAMLIT_으로 시작하는 환경 변수를 지정

예) export STREAMLIT_SERVER_PORT=80

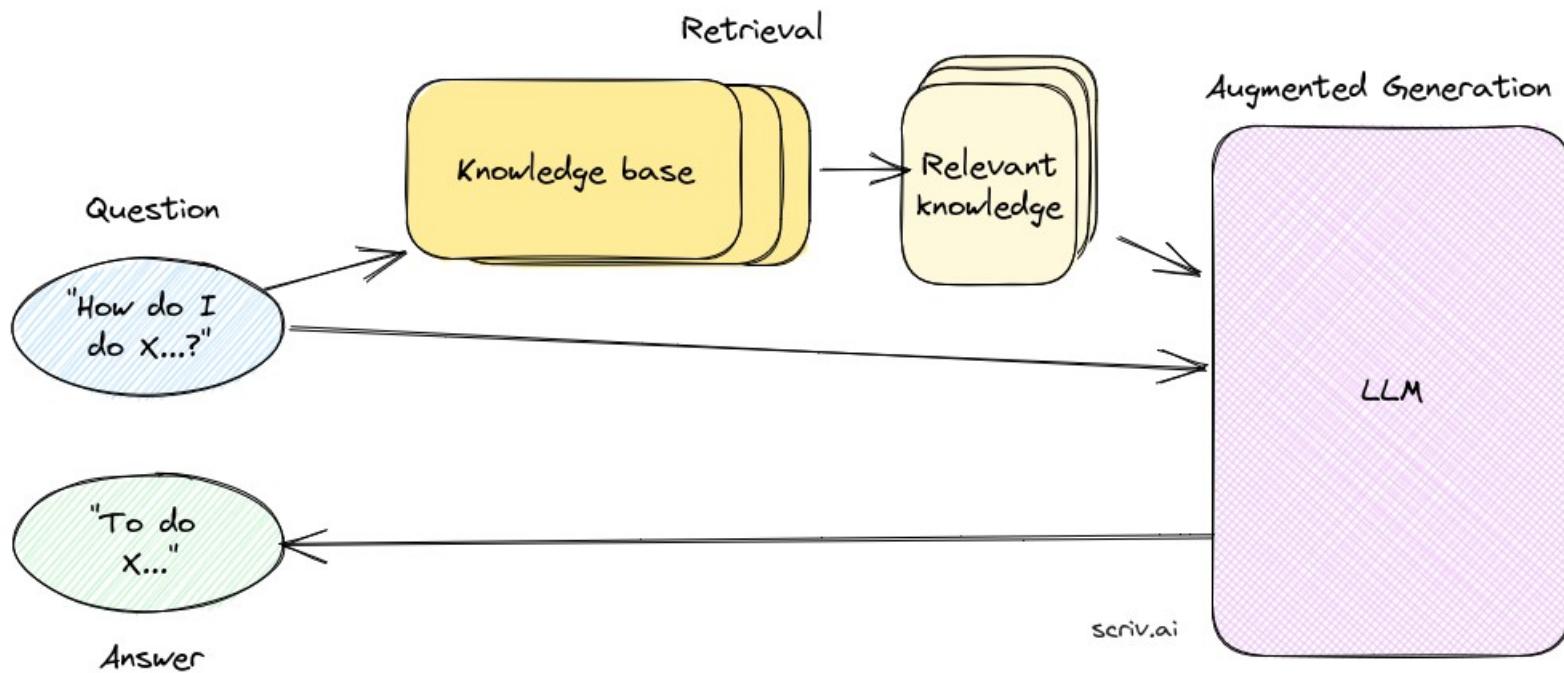
- streamlit run에 옵션으로 지정

<https://docs.streamlit.io/library/advanced-features/configuration>

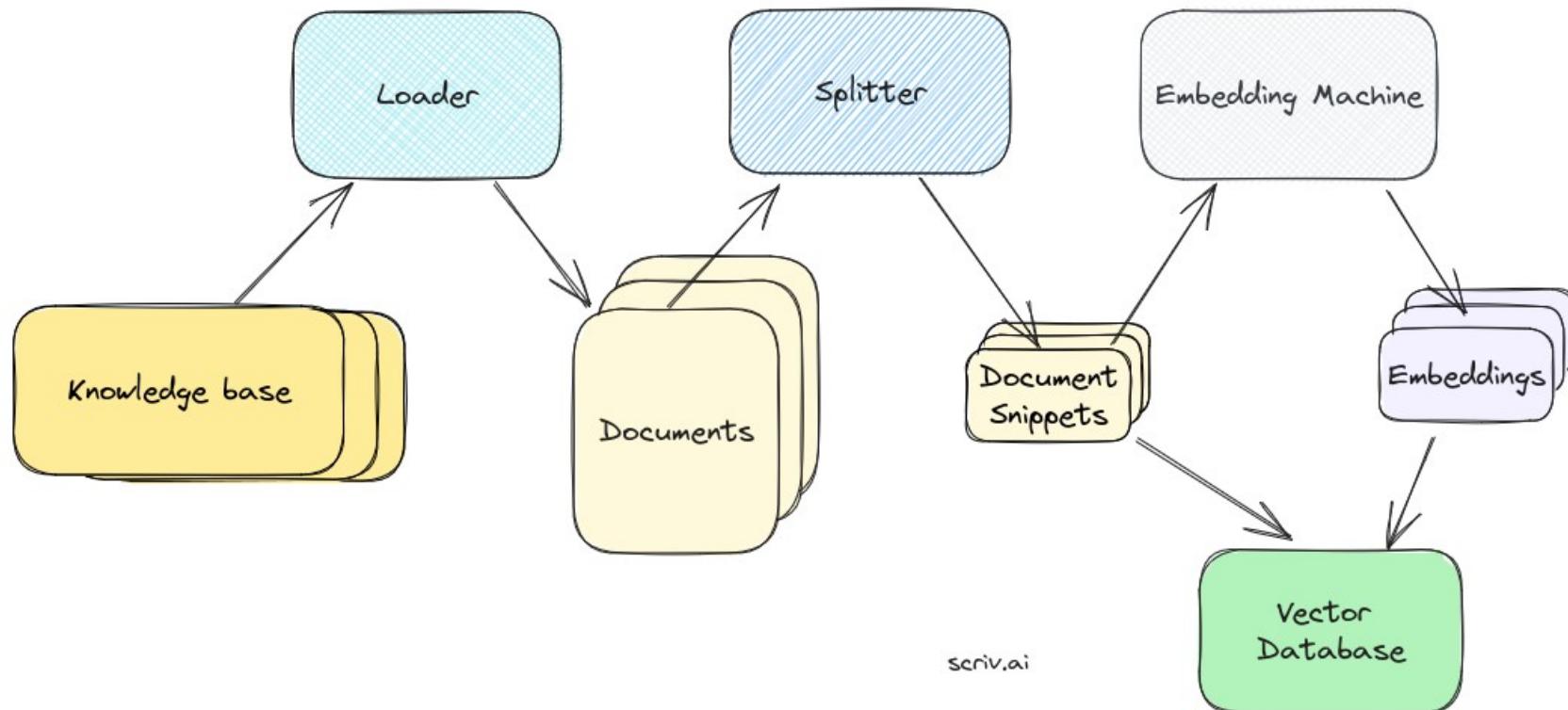
- 언어모델을 기반으로 한 어플리케이션을 개발하기 위한 프레임워크
- 언어모델 학습시 사용된 데이터 외 다른 데이터를 인식할 수 있게 함
- 여러가지 LLM 모델들을 선택적으로 사용할 수 있게 함
- 능동적으로 다른 기능들을 연동하여 추가적인 결과를 낼 수 있게 함
- 여러 기능 모듈을 체인으로 연결하여 파이프를 만들어서 기능적으로 확장할 수 있음
- 내부적으로는 LLM의 프롬프트 조작과 관련 라이브러리셋으로 이런 기능들을 구현

- Model I/O: LLM과 인터페이스 하는 모듈
- Retrieval: 어플리케이션과 연관된 데이터들을 검색하고 찾는 모듈
- Chains: 연속된 LLM 호출과 관계된 모듈
- Agents: 고수준 명령에 의해 적절한 도구를 사용하여 task를 수행하게 하는 모듈
- Memory: 어플리케이션의 상태를 저장하는 것과 관련된 모듈
- Callbacks: 각 체인간의 중간 단계를 저장하고 스트리밍하는 모듈
- Parsers: 대개 체인의 마지막에 존재하며, 이전 컴포넌트의 출력결과를 정해진 형식이나 객체로 구조화하는 모듈

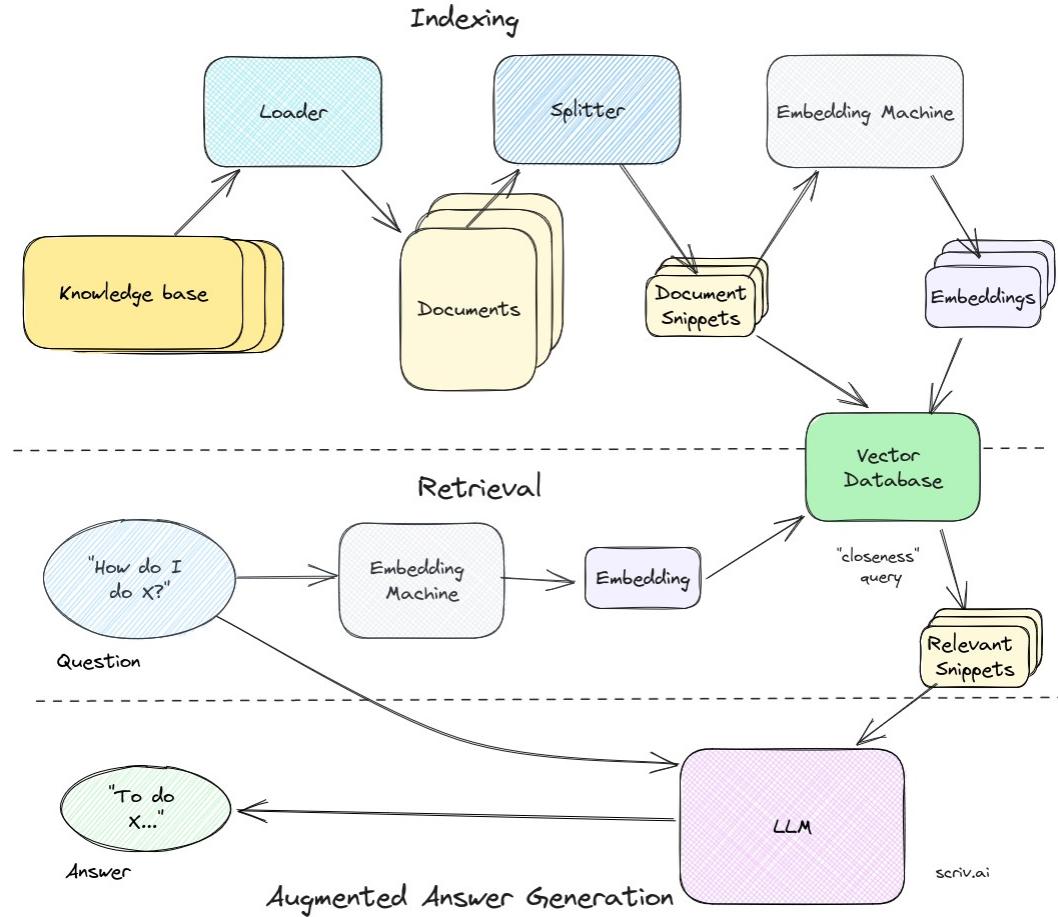
- Embedding은 다양한 데이터를 정해진 차원의 벡터로 변환한 것
- 이러한 벡터들을 저장하고 검색하여 주어진 벡터와 가장 가까운 것들을 추출할 수 있는데, 이렇게 벡터들을 저장하고 관리하는 일종의 Vector Store를 VectorDB라 부름
- 일반적인 데이터베이스에서는, 주로 쿼리와 정확히 일치하는 데이터베이스의 행을 찾는 반면, VectorDB에서는 유사성 메트릭을 적용하여 쿼리(사용자 질문)과 가장 유사한 벡터를 검색
- RAG(Retrieval Augmented Generation)에서 Retriever에 해당하는 것이 이 VectorDB
- 상용으로는 Pinecone이 유명하며, 오픈소스로 Qdrant, Weaviate, Milvus, Chroma, Vespa, FAISS 등이 있음. 어떤 것이나 제공하는 기본 기능은 비슷하지만, 검색의 품질에서는 미묘하게 차이가 있음



Knowledge Indexing



scriv.ai



Is Langchain really pointless? x +

medium.com/@wp237/is-langchain-really-pointless-2302dea10d6d

Medium Search

Write Sign up Sign in

Is Langchain really pointless?

Top Piriyakulwij · Follow
6 min read · Aug 28, 2023

21 Q



LangChain

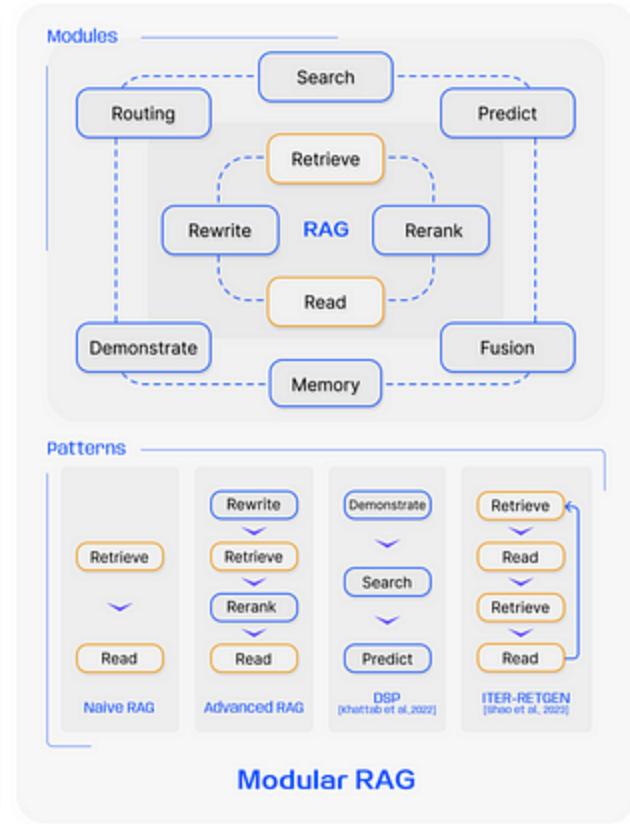
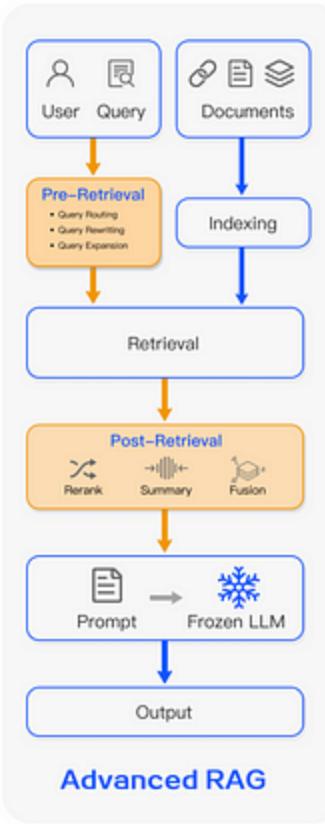
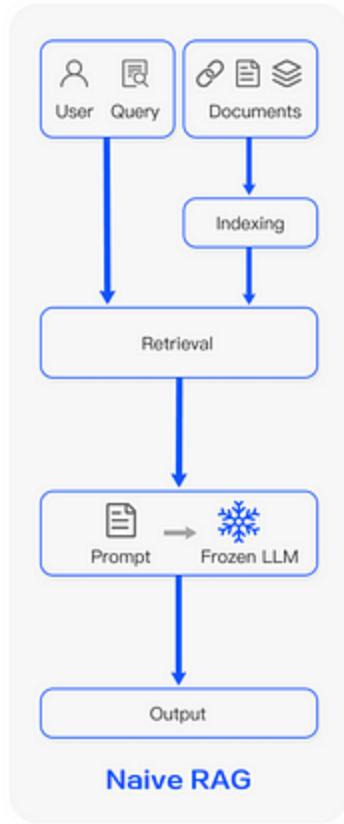
Credit: <https://www.langchain.com/>

In this article, I write about my personal thoughts on LangChain and its usability, in response to recent articles like [Langchain Is Pointless](#) and [The Problem With LangChain](#)

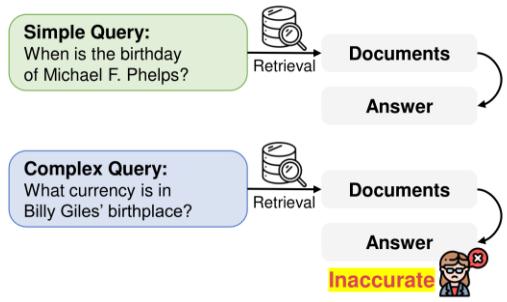
TL;DR - Not pointless for building quick cool demos BUT not worth learning

- LangChain에서 체인을 쉽게 구성하기 위해 도입된 형식
- {object} | {object} | {object} … 하는 식으로 각 객체들을 pipe로 연결하는 문법을 사용
- 자체적으로 Streaming을 지원하며, 병렬실행도 가능

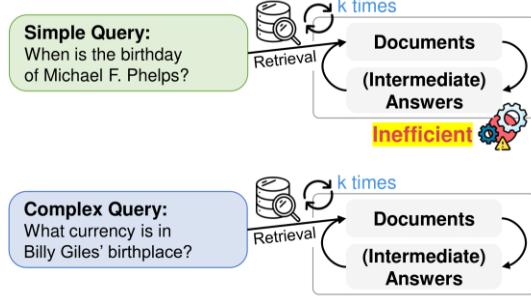
- Naive RAG: 가장 기본적인 구성 방식. Query가 들어오면 문서를 검색하고, 이를 context로 사용하여 응답을 생성. 간단하지만 검색된 문서의 관련성이나 신뢰성을 깊이 평가하지 않는다는 한계
- Advanced RAG: Pre-retrieval, Post-retrieval process를 통해 Retrieve되는 context의 질을 높이는 방식의 RAG. Query를 재작성하거나 Rerank를 사용하는 방법 등을 포함
- Modular RAG: 필요에 따라 여러 개의 모듈을 선택적으로 사용하여 결과물을 생성하는 방식의 RAG
- Adaptive RAG: 3단계의 복잡도에 따라 non-retrieval, single-step, multi-step으로 Retrieval 전략을 사용하여 답변하는 RAG. 복잡도 분류에 별도의 작은 모델을 사용
- Self RAG: 검색(retrieval)과 자기반성(self-reflection)을 통해 언어모델 응답의 품질과 사실성을 개선하는 방식의 RAG. 검색된 문서의 관련성에 따라 query를 재작성하거나 하는 방식을 취함
- Corrective RAG: Retrieval의 결과를 correct, ambiguous, incorrect로 분류하여 각각 다르게 처리하는 RAG. Self RAG에 CRAG을 더한 Self CRAG도 있음



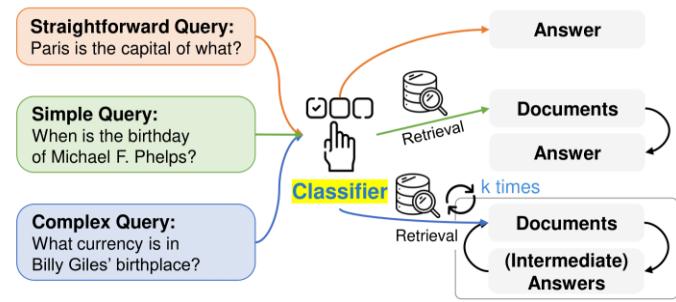
(A) Single-Step Approach



(B) Multi-Step Approach



(C) Our Adaptive Approach



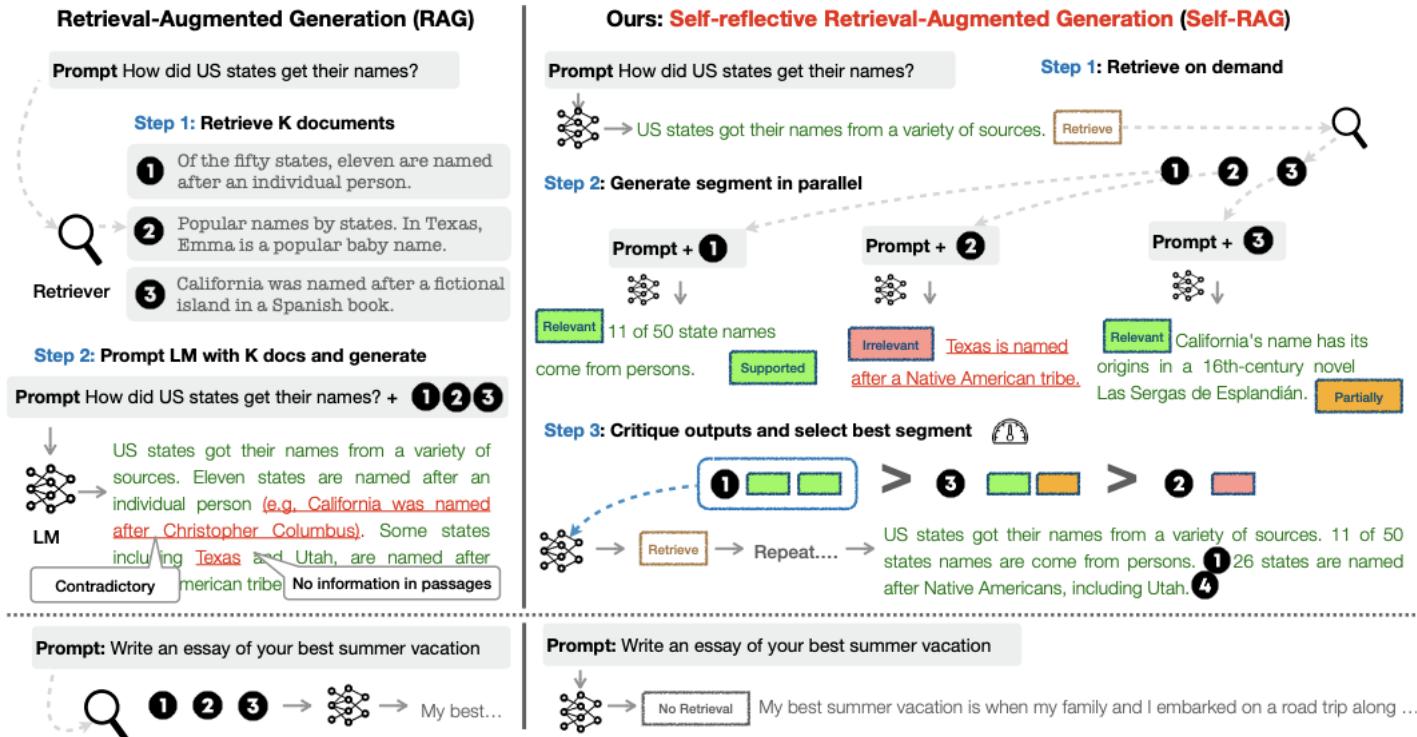


Figure 1: Overview of SELF-RAG. SELF-RAG learns to retrieve, critique, and generate text passages to enhance overall generation quality, factuality, and verifiability.

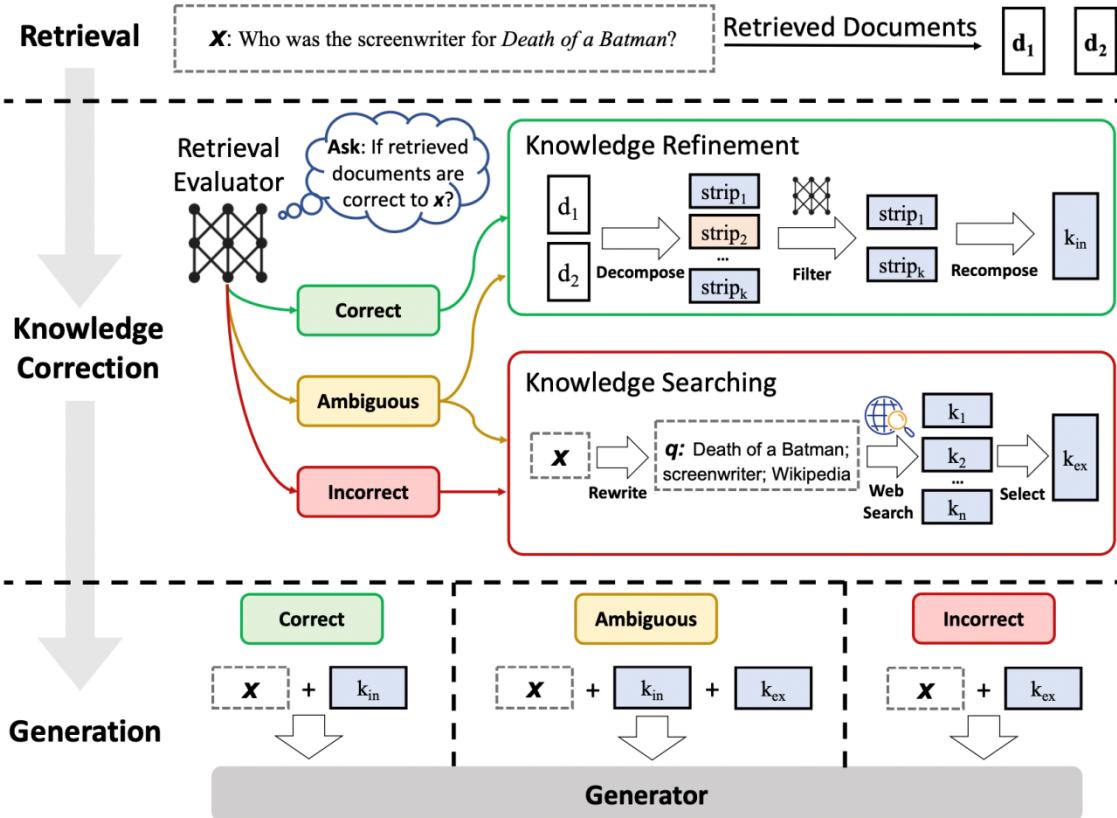
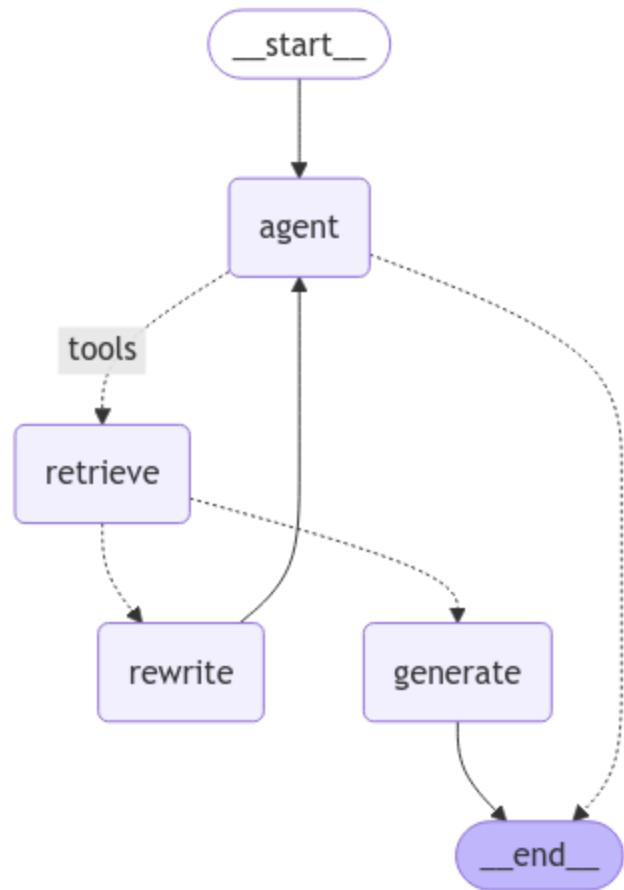
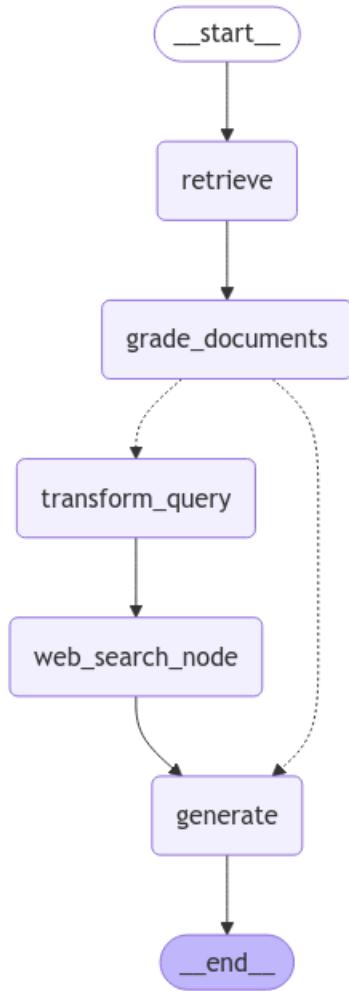


Figure 2: An overview of CRAG at inference. A retrieval evaluator is constructed to evaluate the relevance of the retrieved documents to the input, and estimate a confidence degree based on which different knowledge retrieval actions of {Correct, Incorrect, Ambiguous} can be triggered.

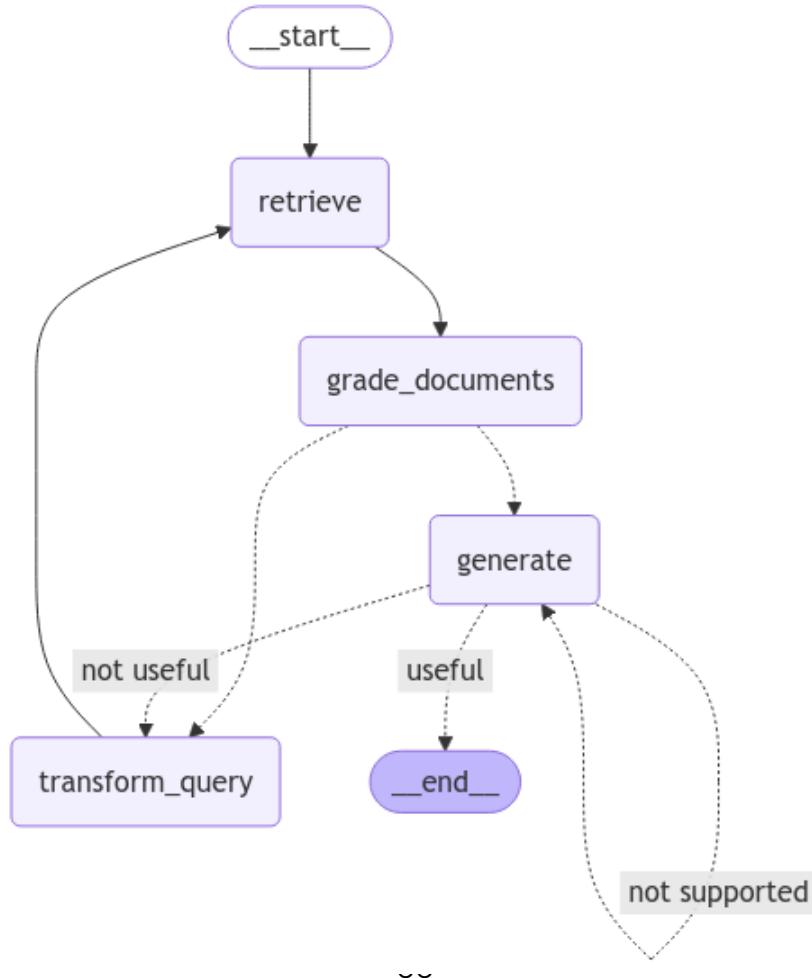
Agentic RAG



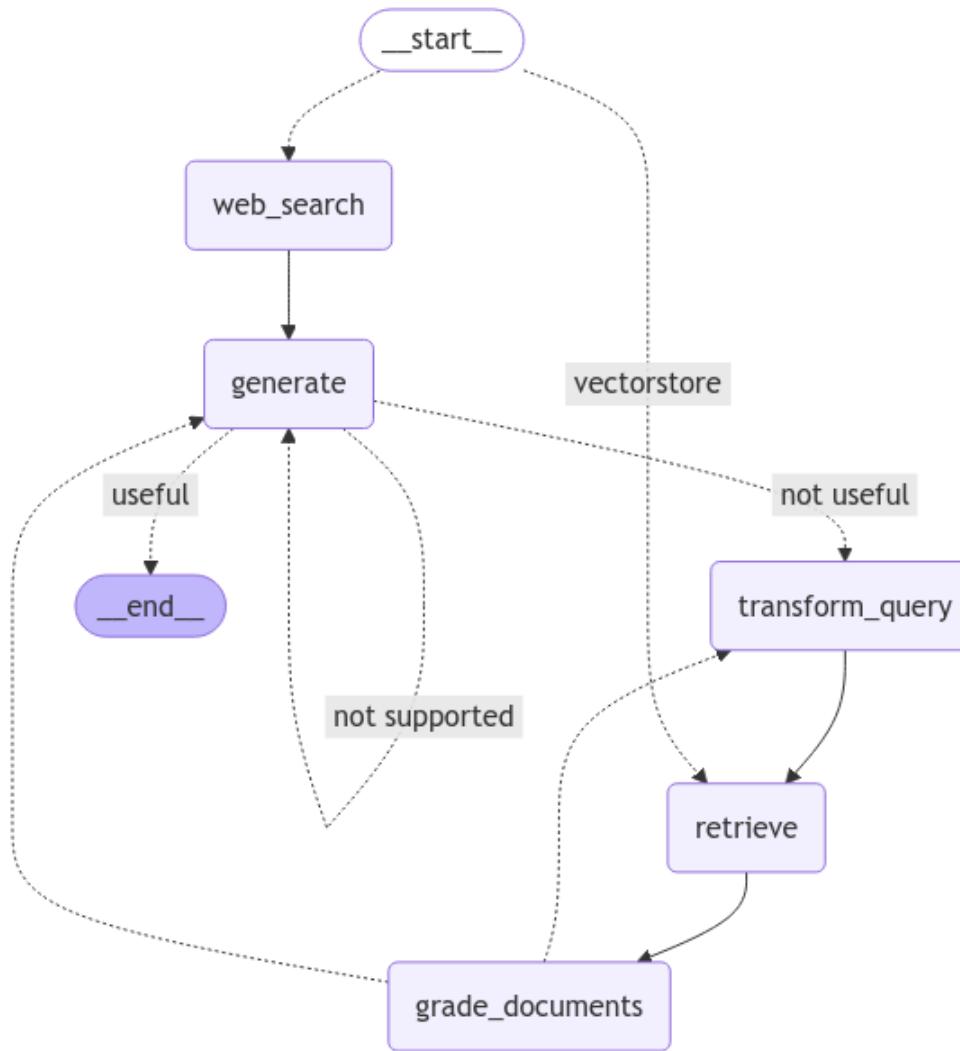
CRAG



Self RAG



Adaptive RAG



- LLM 기반 멀티에이전트 시스템을 위한 프레임워크
- LangChain 생태계의 일부로, LLM 기반 에이전트들의 상태 관리와 워크플로우 조정을 위한 프레임워크
- 상태 머신 기반의 워크플로우 관리, 다중 에이전트 간 협업 조율, 복잡한 태스크의 단계별 실행
- 주요 컴포넌트: State Managers, Channel Managers, Graph Execution Engine, Agent Coordinators
- Python 기반, 비동기 실행 지원, 모듈러 디자인, 확장 가능한 아키텍처
- 복잡한 워크플로우를 간단하게 구현할 수 있어서, Modular RAG이나 Self RAG, CoT, AI agent 등 반복적으로 생성하고 판단해야 하는 어플리케이션에 적합

- LangGraph에서 실행의 단위는 Node로, Node는 표준 python 함수
- START와 END는 시작과 끝을 나타내는 특별한 종류의 Node
- 각 Node들은 State를 입력으로 받고, State를 출력함
- State는 Node들 간에 공유하는 상태 클래스: TypeDict 타입으로, 일반적으로 messages(사용자와 챗봇이 주고 받은 메시지)를 포함하며, 그외 어플리케이션의 종류에 따라 다양한 속성을 추가할 수 있음
- Node 간의 연결은 Edge라고 하며, Node는 Edge가 연결된 방향과 순서에 따라 실행될 수 있음
- 특정 조건을 만족할 때만 연결이 진행되는 Edge를 정의할 수 있는데, 이를 Conditional Edge라 함
- Conditional Edge에서 정의한 함수의 리턴값에 따라 분기하는 Edge가 달라짐
- 각 Node는 모두 특정한 역할을 하며, Node 실행결과에 따라 실행경로를 다르게 하여 복잡한 기능을 수행하게 하는 것이 LangGraph의 요체
- RAG에서 사용시는 BaseModel을 상속한 클래스를 llm.with_structured_output() 메소드로 처리하면 LLM의 출력을 클래스의 인스턴스로 바꾸어서 사용할 수 있음

- ChatGPT 기반 어플리케이션을 구성하려면 LLM 외에도 많은 요소들이 필요
- Assistant API는, 이를 내부적으로 구현하여 이러한 어플리케이션들을 빠르게 구축할 수 있게 한 것: 현재 제공되는 도구는 Code Interpreter, Retrieval, Function calling임. 다시 말해 별도의 시스템 구축 없이 API만으로 python code를 실행하고, 특정 문서에 대해 질의하고, 함수를 실행할 수 있게 해줌
- 이와 더불어, 복수 개 사용자가 동시에 접속하는 경우의 세션 관리나, 대화의 메모리 관리, 입력 token이 overflow되지 않게 하는 기능도 기본적으로 갖추고 있음
- 기본적인 테스트는 Assistant Playground에서 모두 해볼 수 있으며, 여기서 만들어진 Assistant도 어플리케이션에서 사용할 수 있음
- 현재는 Streaming을 지원하지 않으며, completed 상태를 확인하기 위해서 polling해야 하고, DALL-E와 브라우징을 tool로 사용할 수 없으며, User Message에 이미지를 추가할 수 없는 등의 제한이 있음. 또한 계정별로 분당 60 req의 rate limit도 걸려 있음(베타 상태)

- Assistant API를 사용하는 순서는 다음과 같음
- 1) Assistant를 만든다.
 - 2) Thread를 만든다. Thread는 한 작업 세션으로, 사용자별로 주어지는 것이 보통이며, 대화 내역은 저장된다.
 - 3) Thread에 Message를 추가한다. 보통 사용자 Role인 'User'로 Message를 추가해준다.
 - 4) 1)의 Assistant id와 2)의 Thread id를 주어서 Assistant를 실행한다. 여기에 추가 Instruction을 줄 수도 있다.
 - 5) 실행된 작업은 queued state가 된다. 이 상태가 completed가 되면 Assistant의 작업이 끝난 것이다.
 - 6) Thread에 추가된 Assistant의 Message를 확인한다.

- Assistant API를 사용하면 기본 요금 외에 추가로 붙는 요금들이 있음
- File Search를 위해 파일을 업로드할 경우, GB당 \$0.10이 하루에 과금됨. 하나의 파일을 여러 개의 Assistant 가 같이 사용한다면, Assistant 수만큼 요금이 늘어남. 파일을 삭제하지 않으면 계속 과금(1G는 무료)
- User Message의 일부로 파일이 추가되었다면 해당 파일에도 동일하게, 각각 GB당 \$0.10이 하루에 청구됨
- 파일의 크기는 512MB를 넘을 수 없으며, token으로 환산하면 200만 token이 됨
- 파일 종류는 일반 텍스트, 개발용 소스, pdf, md, docx, html, jpg, png 등 업무에서 사용되는 많은 종류의 파일 포맷을 지원하고 있음: 파일 종류에 따라 Code Interpreter나 File Search에서의 사용 가능 여부가 달라지니 확인해야 함
- Code Interpreter는 세션당 \$0.03이며, 각 세션은 1시간 동안 활성화 됨. 그리고 이 요금은 1시간 동안 1회만 과금되는 형태

- AI agent란, 인공지능 기술을 활용하여 특정한 작업을 수행하거나 의사결정을 내릴 수 있는 소프트웨어 시스템: 주변 환경을 인식하고 해석할 수 있으며, 그에 따라 자율적으로 행동하고 목표를 달성하기 위해 최선의 행동 계획을 수립
- 단순히 사용자의 질문에 답하는 것에 그치지 않고, 주어진 도구들을 이용하여 스스로 프롬프트를 만들고 추론하면서 주어진 목표를 달성하려고 노력
- AI agent의 핵심은 역시 LLM이지만, 이 LLM에 어떤 식으로 목표를 주고 작동하게 하는가에 대해 최근 연구가 활발함: 단순 챗봇이라면 효용은 떨어짐, AI agent가 실제 사람이 하는 작업의 일부 또는 전부를 수행할 수 있어야 생산성을 끌어올릴 수 있음

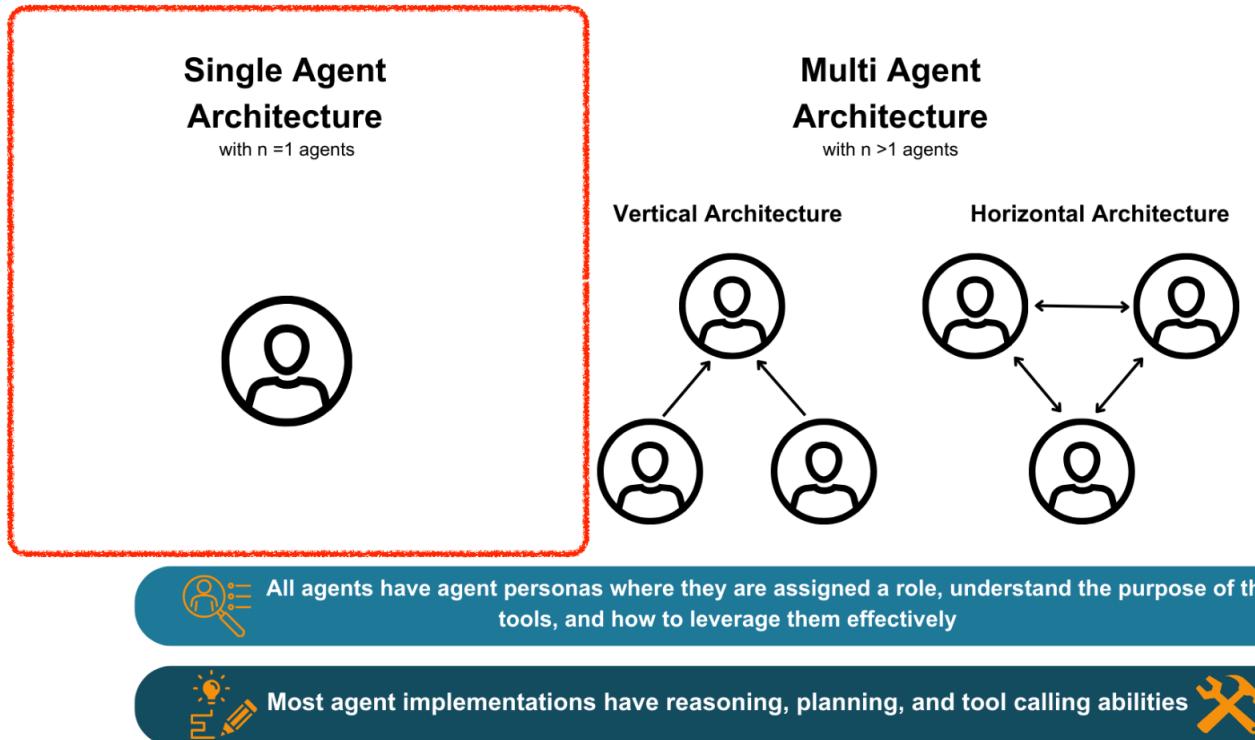
- AI 에이전트(Agent): 목표를 달성하기 위해 여러 번에 걸쳐 반복적으로 계획을 세우고 **행동**을 취할 수 있는 언어 모델(LM, Language Model) 기반의 개체. 각각의 에이전트는 서로 다른 역할을 수행하도록 별도의 페르소나(Persona)가 주어짐. 이 페르소나는 일종의 역할(role)과 같은 것으로, 에이전트에 대한 설명을 비롯하여 접근 가능한 도구들을 정의하는 역할 등을 포함하고 있음
- 단일 에이전트 아키텍처(Single-Agent Architecture): 하나의 언어 모델이 모든 추론(reasoning), 계획(planning), 실행(execution)을 담당하는 구조. 이 경우, 에이전트는 다른 에이전트와의 상호작용 없이 독립적으로 작업을 수행
- 다중 에이전트 아키텍처(Multi-Agent Architecture): 두 개 이상의 에이전트가 협업하여 작업을 수행하는 구조. 각 에이전트는 고유한 페르소나를 가지며, 동일하거나 서로 다른 언어 모델과 도구를 사용할 수 있음. 다중 에이전트 시스템은 다시 수직형(Vertical)과 수평형(Horizontal)으로 나눌 수 있음

- 수직형 아키텍처(Vertical Architecture): 하나의 리드 에이전트(Lead Agent)가 다른 에이전트들을 관리하고 지시하는 구조. 이 경우, 주도 에이전트를 중심으로 의사소통이 이루어짐
- 수평형 아키텍처(Horizontal Architecture): 모든 에이전트가 동등한 지위를 가지며, 그룹 토론을 통해 정보를 공유하고 협업하는 구조
- 추론(Reasoning): 에이전트가 주어진 정보를 바탕으로 논리적인 결론을 도출하거나 문제를 해결하는 과정. 이는 에이전트가 복잡한 상황을 이해하고 적절한 행동을 선택하는 데 필수적인 능력임
- 계획 수립(Planning): 에이전트가 목표를 달성하기 위해 일련의 행동을 순서대로 정리하는 과정. 이는 문제를 더 작은 하위 문제로 분해하고, 각 단계에서 필요한 행동을 결정하는 것을 포함
- 도구 사용(Tool Use): 에이전트가 외부 데이터 소스, API, 기능 등과 상호작용할 수 있는 능력. 도구 사용을 통해 에이전트는 더 많은 정보를 수집하고 활용할 수 있으며, 실제 환경에 변화를 일으킬 수 있음

- 언어 모델(Language Model, LM): 자연어 데이터를 기반으로 학습된 AI 모델로, 인간의 언어 사용 패턴을 모방할 수 있음. 언어 모델은 텍스트 생성, 질의응답, 번역 등 다양한 자연어 처리(NLP, Natural Language Processing) 작업에 활용됨
- 강화 학습(Reinforcement Learning, RL): 에이전트가 환경과 상호작용하면서 시행착오를 통해 학습하는 방법. 에이전트는 행동(action)을 선택하고, 그에 따른 보상(reward)을 받는데, 이 과정을 반복하면서 에이전트는 장기적인 관점에서 최적의 행동 정책(policy)을 학습하게 됨. 강화 학습은 게임 플레이, 로봇 제어 등 다양한 분야에서 활용됨
- 벤치마크(Benchmark): AI 모델이나 에이전트의 성능을 평가하기 위한 표준화된 테스트 또는 데이터셋. 벤치마크를 통해 서로 다른 모델 간의 성능을 객관적으로 비교할 수 있으며, 연구의 발전 방향을 제시할 수 있음. 대표적인 벤치마크는 ImageNet(이미지 분류), SQuAD(질의응답), GLUE(자연어 이해) 등

- 에이전트가 실제 문제를 효과적으로 해결하기 위해서는 추론(reasoning), 계획 수립(planning), 그리고 도구 사용(tool use) 능력이 필수적
- 추론은 인간의 인지 능력 중 가장 기본적이면서도 핵심적인 부분
- 계획 수립은 추론 능력과 밀접히 연관: 에이전트는 주어진 목표를 달성하기 위해 효과적인 행동 시퀀스를 생성해야 하는데, 이는 복잡한 추론 과정을 필요. 일반적으로 계획 수립 과정은 아래 다섯가지 중 하나를 선택
 1. 작업 분해(Task Decomposition): 주어진 문제를 더 작고 다루기 쉬운 하위 문제로 나누는 방법. 에이전트는 각 하위 문제를 순차적으로 해결함으로써 전체 목표에 도달할 수 있음
 2. 다중 계획 선택(Multi-Plan Selection): 에이전트가 여러 개의 대안적인 계획을 생성한 뒤, 그 중 가장 적합한 계획을 선택하는 방법. 이를 통해 에이전트는 다양한 가능성을 고려하고, 상황에 맞는 최적의 계획을 수립할 수 있음

3. 외부 모듈 기반 계획 수립(External Module-Aided Planning): 에이전트가 미리 정의된 외부 모듈이나 알고리즘을 활용하여 계획을 생성하는 방법. 복잡한 계획 수립 과정을 단순화할 수 있지만, 모듈의 성능과 범용성에 크게 의존하게 됨
4. 반영과 개선(Reflection and Refinement): 에이전트가 수립한 계획을 실행하면서 얻은 경험과 피드백을 바탕으로 계획을 지속적으로 개선하는 방법. 이는 에이전트가 환경의 변화에 적응하고, 더 나은 성능을 달성하는 데 도움이 됨
5. 메모리 증강 계획 수립(Memory-Augmented Planning): 에이전트가 과거의 경험이나 지식을 활용하여 계획을 수립하는 방법. 이를 위해서는 에이전트가 중요한 정보를 효과적으로 저장하고 검색할 수 있는 메모리 메커니즘을 갖추어야 함



Source: Neudesic, an IBM Company

Figure 1: A visualization of single and multi-agent architectures with their underlying features and abilities

- ReAct(Reason + Act) 는 "이성적으로 사고한 다음 행동한다(Reason + Act)"는 컨셉의 방식으로, AI 에이전트가 작업에 대해 무엇인가를 수행하기 전, 먼저 '생각'을 하고 그 다음에 '행동'하도록 설계한 구조: LangChain의 경우 Thought/Action/Action Input/Observation을 반복하면서 최종 답안을 얻을 때까지 실행을 계속함
- RAISE 방식은 ReAct를 기반으로 하면서 인간의 단기 및 장기 기억과 유사한 메모리 구성 요소를 추가한 방식
- Reflexion은 언어적 피드백을 통한 자기 성찰(self-reflection)을 사용하는 단일 에이전트 패턴
- AutoGPT+P(Planning) 는 기존의 AutoGPT 방식에 계획(Planning) 기능을 추가하여 자연어를 이해하고 로봇 작업을 수행하는 AI 에이전트의 능력을 강화한 방식
- LATs(Language Agent Tree Search, 언어 에이전트 트리 탐색) 는 계획 수립 및 행동, 추론을 통합하는 트리(Tree) 검색 기법을 적용한 것이 특징인 단일 에이전트 동작 방식

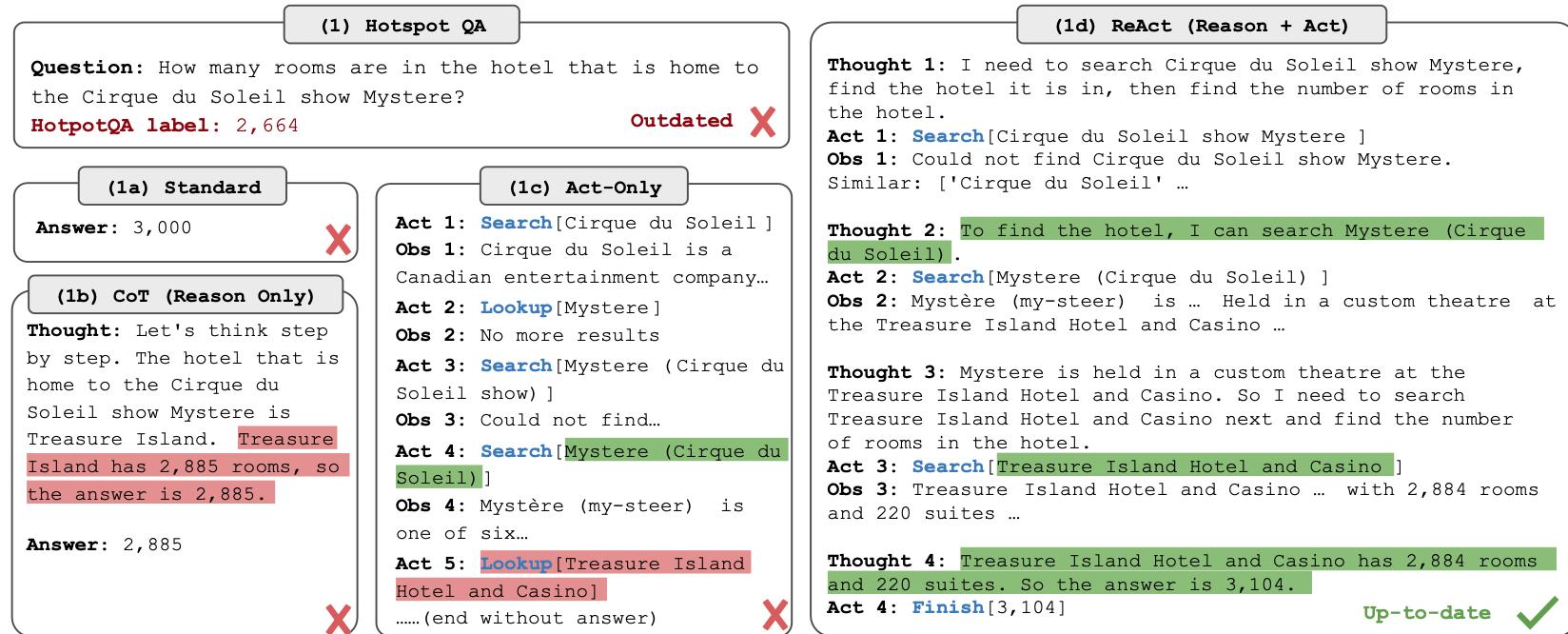


Figure 2: An example of the ReAct method compared to other methods [32]

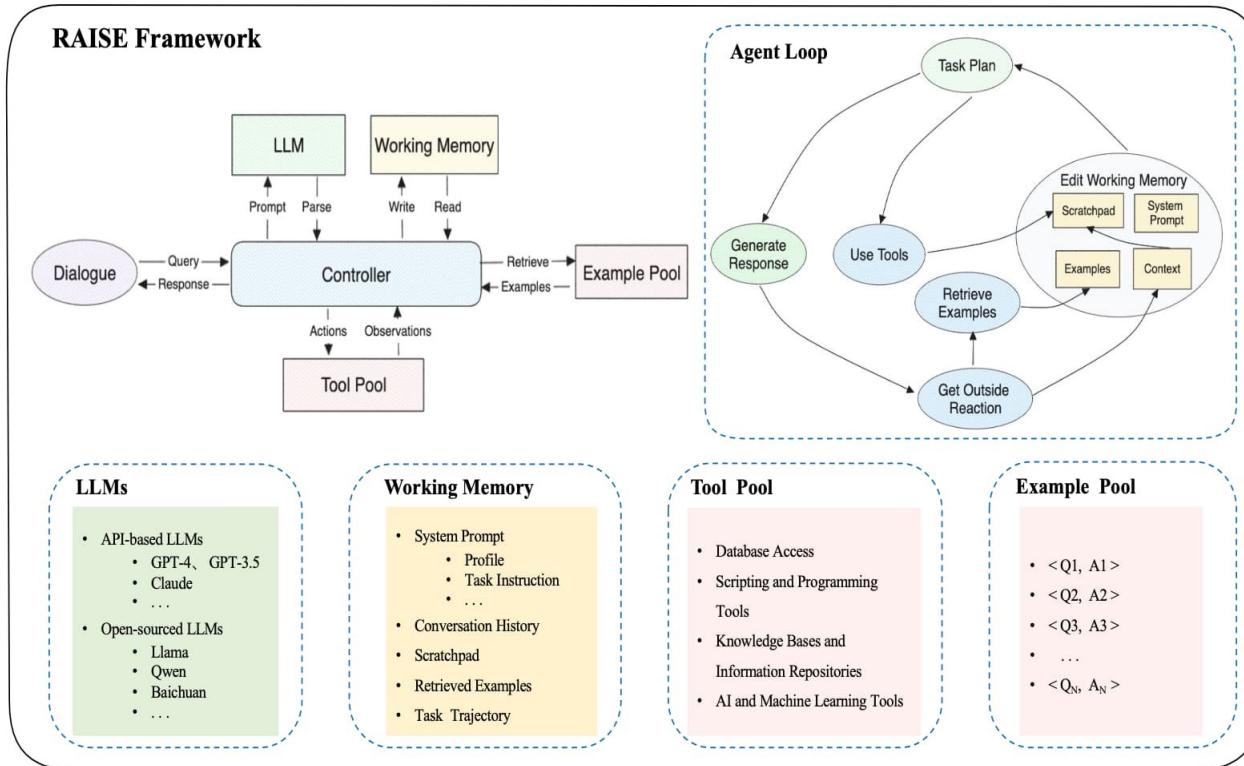
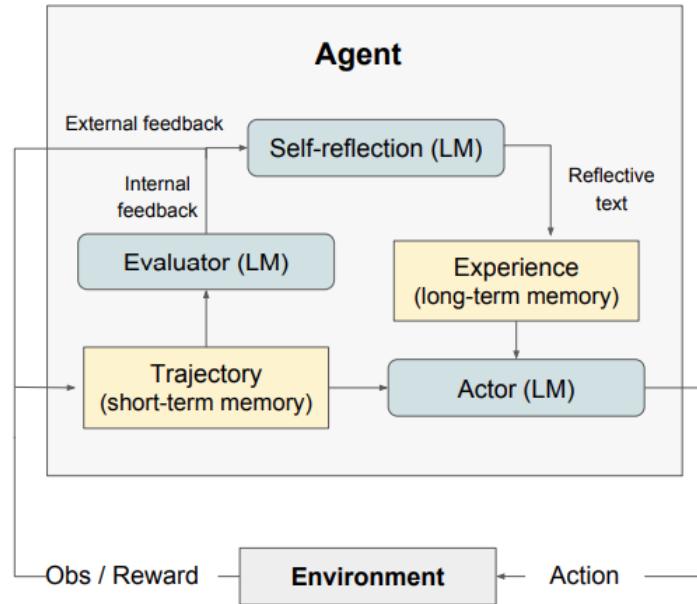


Figure 3: A diagram showing the RAISE method [16]

**Algorithm 1** Reinforcement via self-reflection

```

Initialize Actor, Evaluator, Self-Reflection:  

 $M_a, M_e, M_{sr}$   

Initialize policy  $\pi_\theta(a_i|s_i)$ ,  $\theta = \{M_a, mem\}$   

Generate initial trajectory using  $\pi_\theta$   

Evaluate  $\tau_0$  using  $M_e$   

Generate initial self-reflection  $sr_0$  using  $M_{sr}$   

Set  $mem \leftarrow [sr_0]$   

Set  $t = 0$   

while  $M_e$  not pass or  $t < \text{max trials}$  do  

    Generate  $\tau_t = [a_0, o_0, \dots, a_i, o_i]$  using  $\pi_\theta$   

    Evaluate  $\tau_t$  using  $M_e$   

    Generate self-reflection  $sr_t$  using  $M_{sr}$   

    Append  $sr_t$  to  $mem$   

    Increment  $t$   

end while  

return

```

Figure 2: (a) Diagram of Reflexion. (b) Reflexion reinforcement algorithm

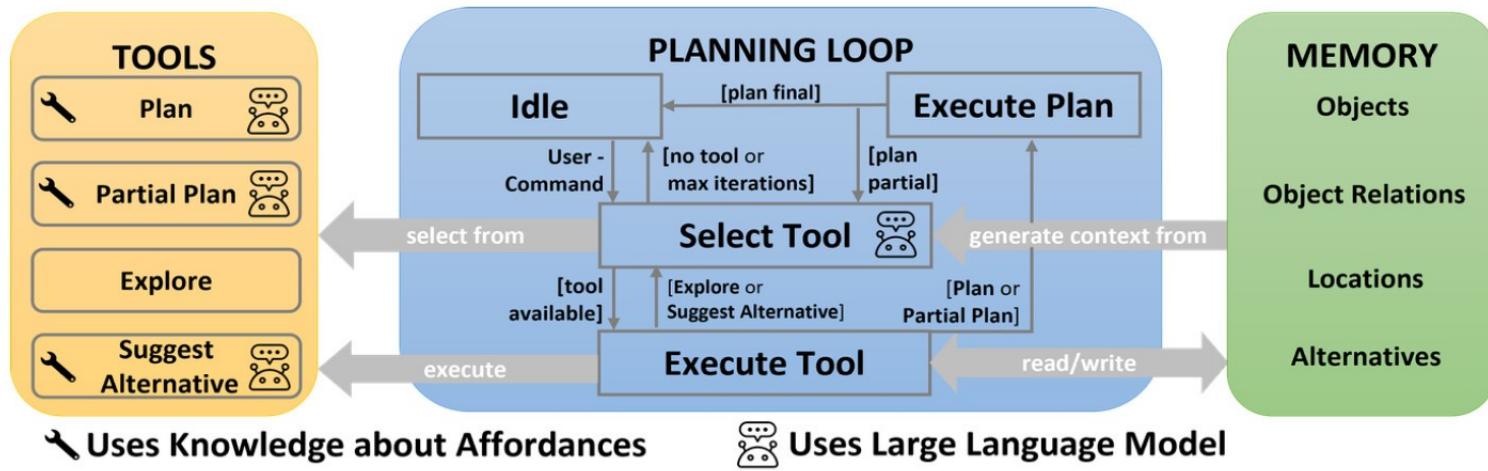
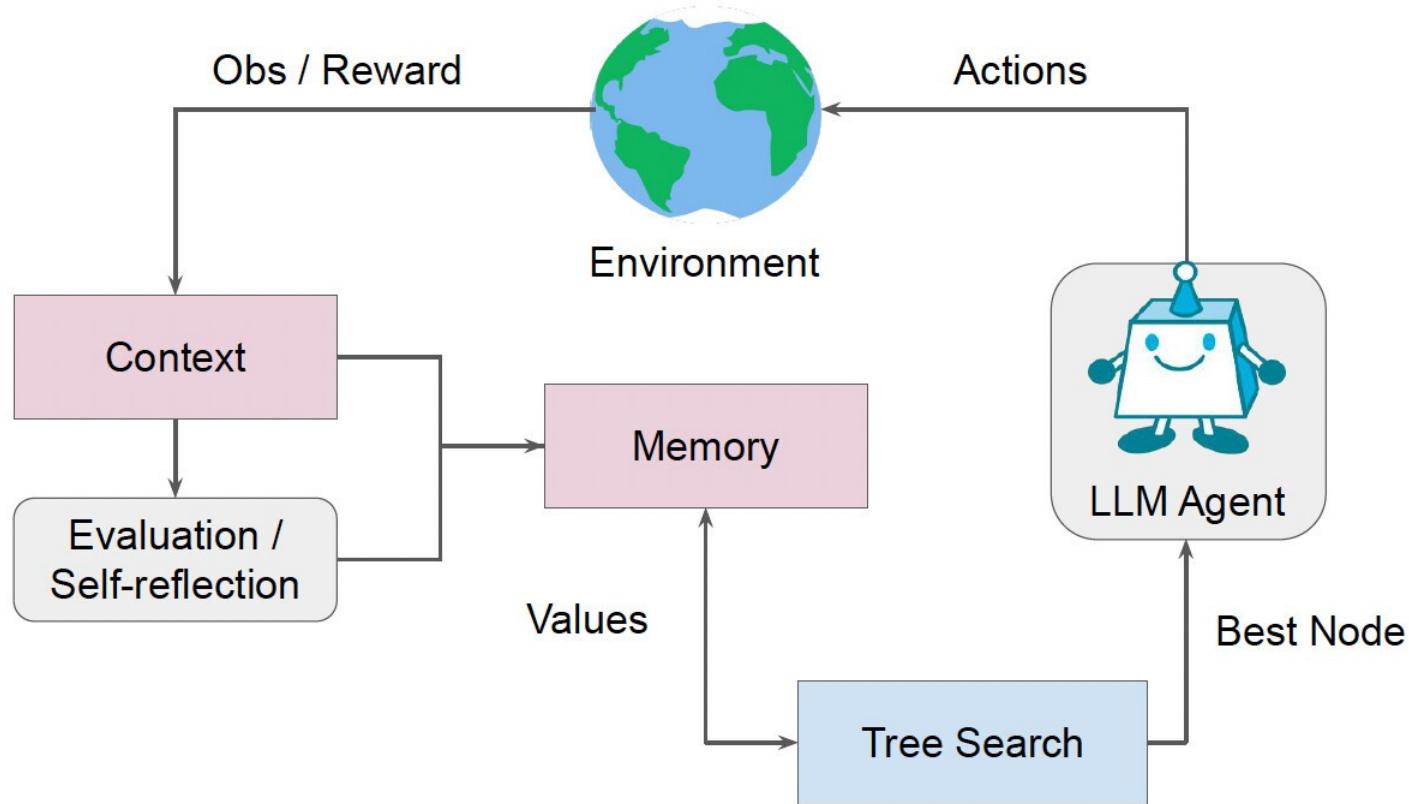


Figure 4: A diagram of the AutoGPT+P method [1]



Single Agent Architecture

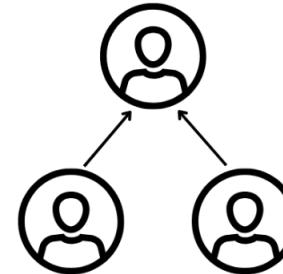
with n = 1 agents



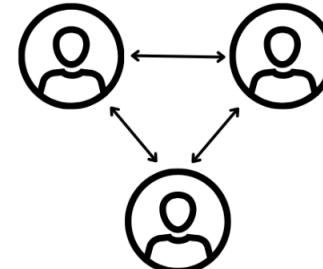
Multi Agent Architecture

with n > 1 agents

Vertical Architecture



Horizontal Architecture



All agents have agent personas where they are assigned a role, understand the purpose of their tools, and how to leverage them effectively



Most agent implementations have reasoning, planning, and tool calling abilities



Source: Neudesic, an IBM Company

Figure 1: A visualization of single and multi-agent architectures with their underlying features and abilities

- 실체화된 LLM 에이전트들의 조직화된 팀 협력 학습(Embodied LLM Agents Learn to Cooperate in Organized Teams)
- DyLAN(Dynamic LLM-Agent Network, 동적 LLM 에이전트 네트워크)은 복잡한 작업 수행에 특화된 프레임워크로, 동적인 상황에 대응할 수 있는 LLM 기반의 다중 에이전트 네트워크
- AgentVerse는 각각의 AI 에이전트에게 명확한 역할을 부여하고, 임무 수행을 위한 단계별 계획을 제공하는 방식으로 추론 및 문제 해결 능력을 향상하는 다중 에이전트 아키텍처 방식 중 하나
- MetaGPT는 다수의 AI 에이전트들이 서로 의사소통을 하며 공통의 문제를 해결할 수 있는 기능을 제공하는 다중 에이전트 아키텍처 방식

| 실체화된 LLM 에이전트들의 조직화된 팀 협력 학습

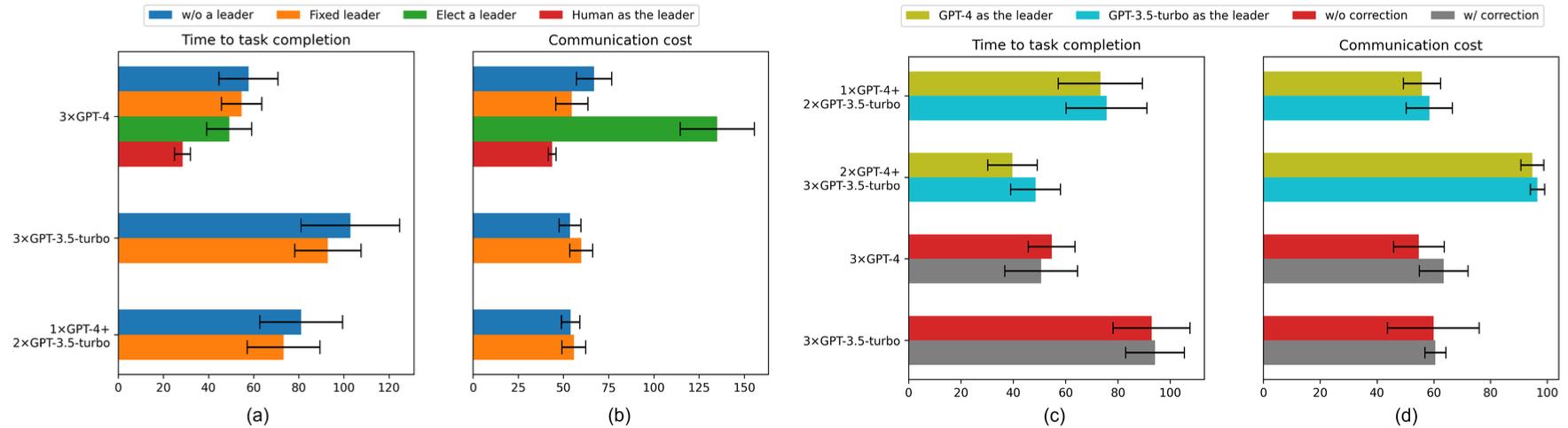
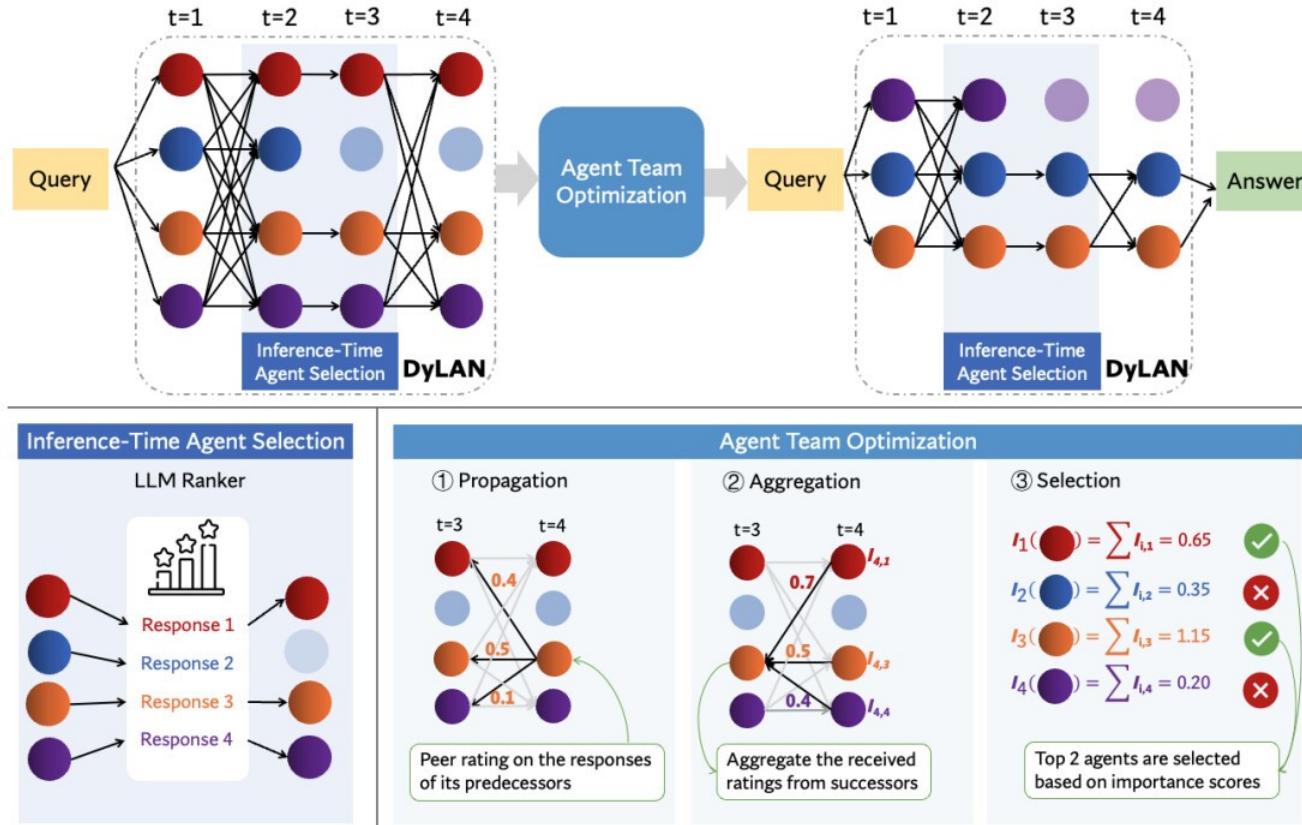


Figure 5: Agent teams with a designated leader achieve superior performance [9]

| DyLAN (Dynamic LLM-Agent Network)



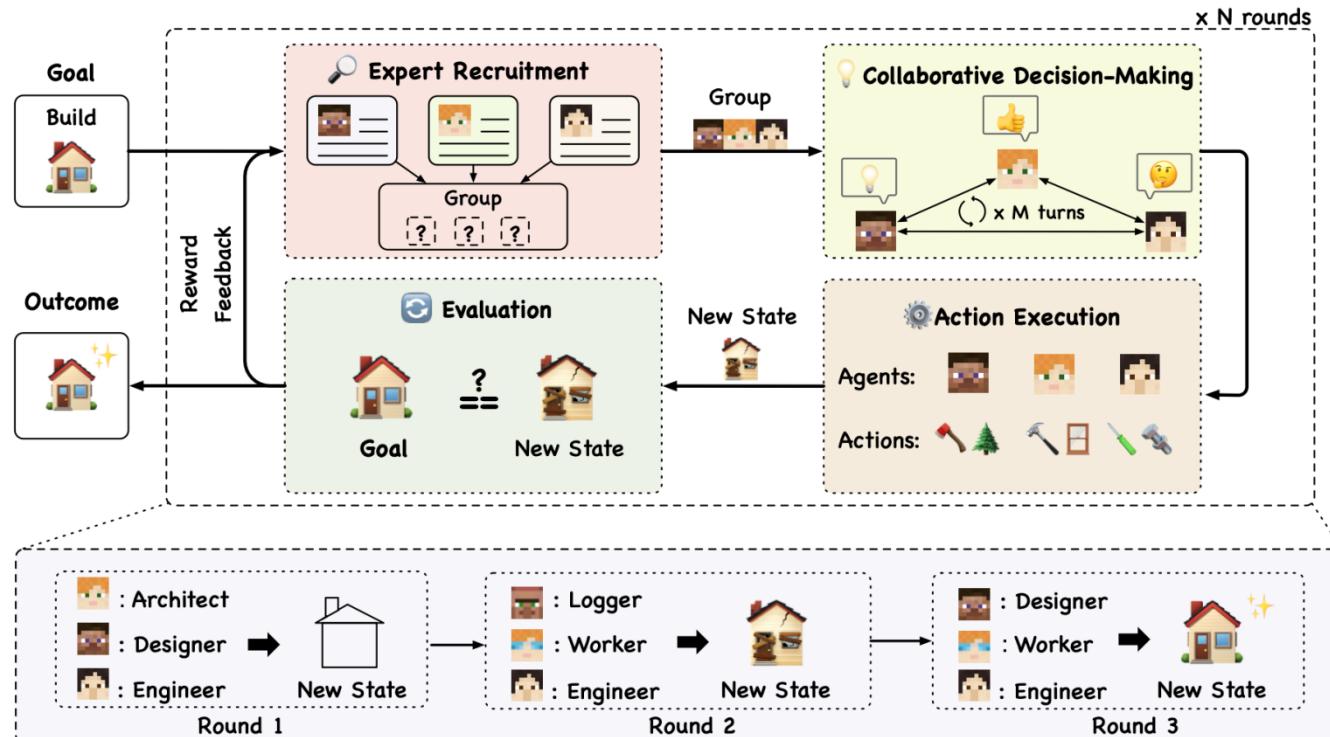
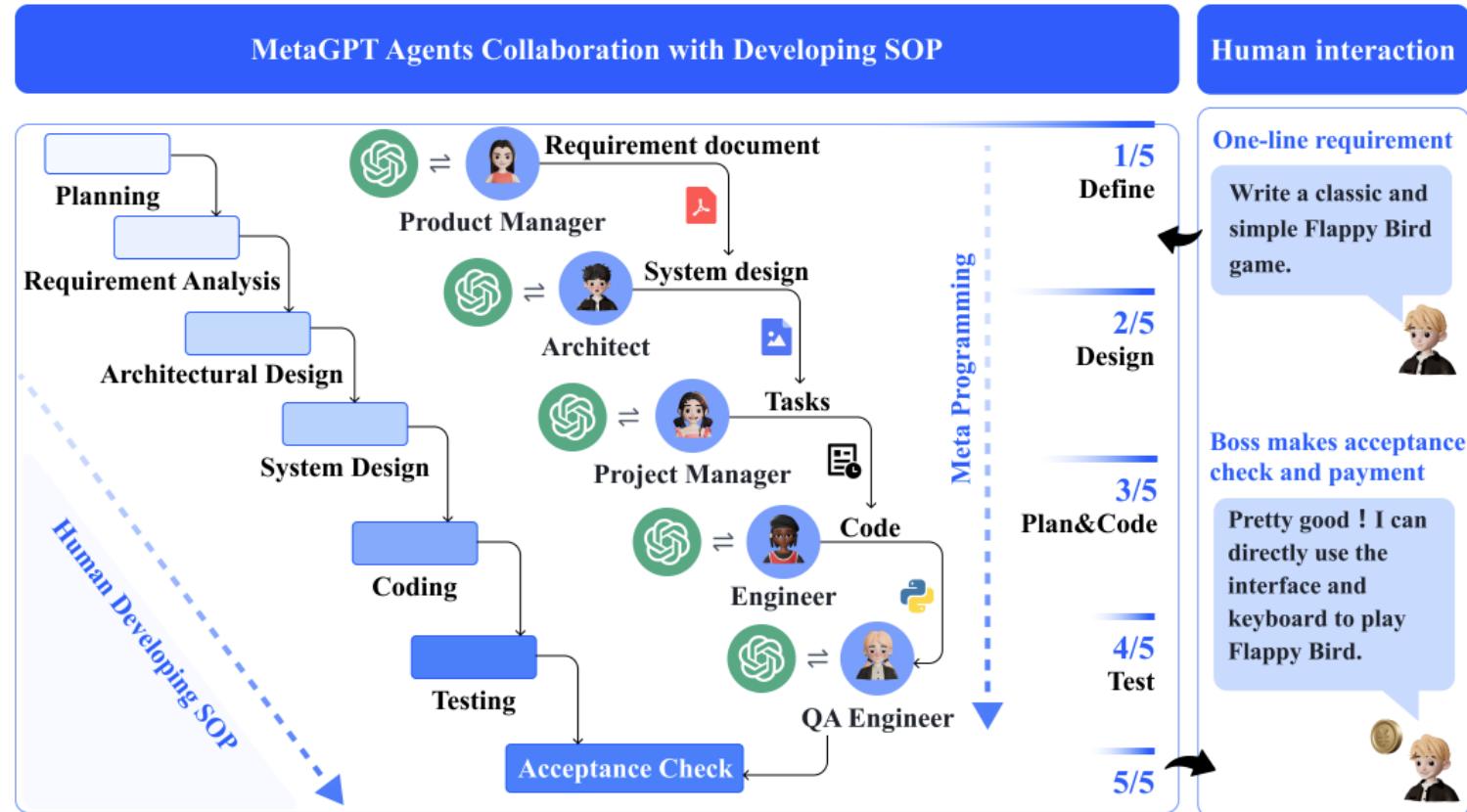


Figure 6: A diagram of the AgentVerse method [2]



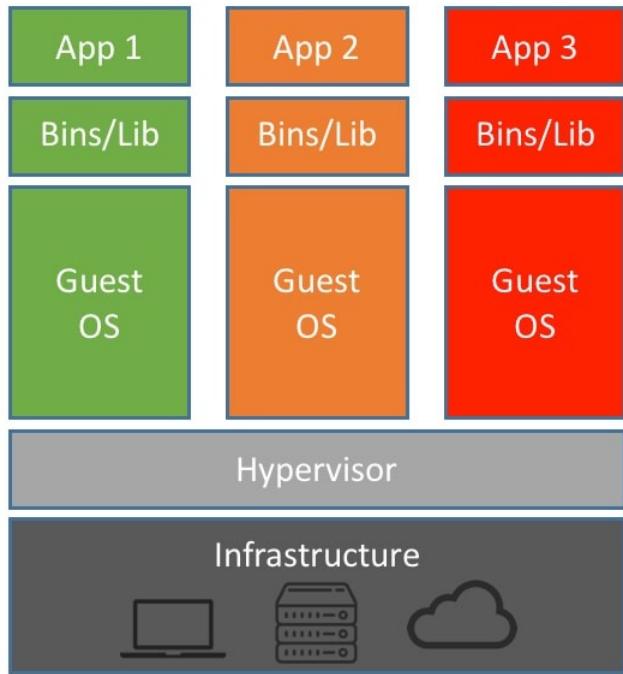
| 단일 vs 다중 에이전트 아키텍처

- 단일 에이전트 패턴은 일반적으로 도구 목록이 좁게 정의되어 있고 프로세스가 명확한 작업에 적합
- 단일 에이전트는 하나의 에이전트와 도구 세트만 정의하면 되므로 구현도 일반적으로 더 쉬움
- 단일 에이전트 아키텍처는 다른 에이전트의 부정확한 피드백이나 관련 없는 잡담으로 인한 산만함 등의 제약을 받지 않음, 그러나 추론 및 개선 능력이 강건하지 않으면 실행 루프에 빠져 목표 달성을 이루지 못할 수 있음
- 다중 에이전트 아키텍처는 작업 수행에 있어 다양한 페르소나의 피드백이 도움이 되는 경우에 적합
- 다중 에이전트 시스템은 서로 다른 작업이나 워크플로우에 걸친 병렬 처리가 필요한 경우에도 유용
- 다중 에이전트 시스템은 본질적으로 더 복잡하며 견고한 대화 관리와 명확한 리더십의 이점을 누릴 수 있음
- 에이전트에 제공되는 프롬프트가 충분히 강력하다면 다중 에이전트 토론이 반드시 추론을 향상시키는 것은 아니다: 즉 추론능력 향상을 위해 다중 에이전트를 쓴다는 건 적합하지 않음

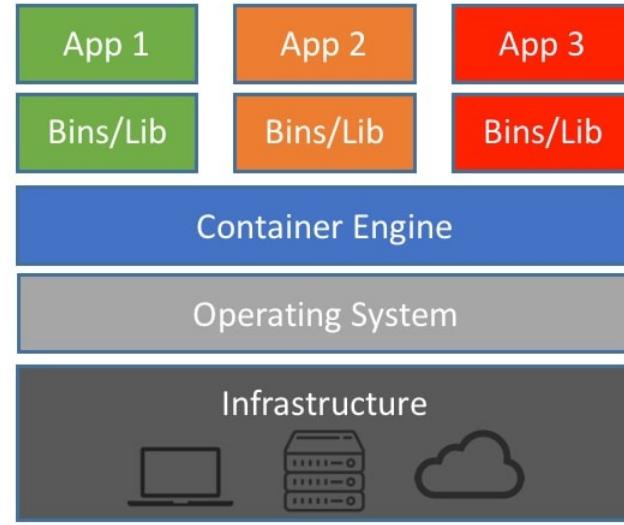
- LLM 기반의 멀티에이전트 시스템을 그래프 형태로 구성
- LangChain과 연계하여 활용 가능
- 복잡한 에이전트 간 상호작용 설계 지원
- 상태 기반(State-based) 워크플로우: 워크플로우의 각 단계를 명확하게 정의 가능
- 그래프 기반 모델링: 에이전트 간 관계 및 실행 흐름을 그래프 형태로 시각화
- 유연한 확장성: 다양한 에이전트 및 도구와 통합 가능
- LangChain 통합: LangChain의 강력한 기능을 활용한 워크플로우 구축 가능
- Graph는 상태를 유지하지 않으며, StateGraph는 상태를 유지한다는 차이점: StateGraph는 전체 노드가 공유하는 상태를 유지하면서 실행이 거듭되면서 업데이트되나, Graph는 Stateless하며 매번 실행의 input에 대해 output을 출력하는 역할

- Microsoft에서 개발한 LLM 기반 협업 에이전트 프레임워크
- 여러 에이전트 간의 역할 분담 및 자동 협업 기능 제공
- 자동 협업 시스템: 에이전트들이 상호작용하며 문제 해결
- 멀티턴 대화 관리: 긴 대화 흐름을 유지하며 자연스럽게 진행
- 프롬프트 최적화: 실행 결과에 따라 프롬프트를 동적으로 조정
- 다양한 작업 지원: 코드 생성, 데이터 분석, 문서 작성 등
- LocalCommandLineCodeExecutor와 DockerCommandLineCodeExecutor를 통해 생성된 코드를 실행할 수 있음: 실 구동시는 DockerCommandLineCodeExecutor 사용이 권장됨
- 현재 지원되는 언어는 shell script(bash, powershell 포함), javascript, python 만이며, 이 언어들로 코드를 작성하고 구동할 수 있음: css, html는 저장만 가능

- docker는 VM보다 더 light-weight한 컨테이너의 한 종류로 Linux, MacOS, Windows 등 다양한 OS를 지원함(Windows는 WSL2를 통해 지원)
- VM은 실제 가상의 서버를 부트업 과정부터 실행하는 것이지만, 컨테이너는 이미 부트업된 서버의 커널만 사용하는 가상환경
- 컨테이너가 서버에서 사용하는 것은 커널(+드라이버) 뿐이며, 나머지 필요한 파일과 라이브러리, 설정 등은 모두 컨테이너가 가지고 있음: 따라서 커널만 컨테이너 구동에 필요한 최소 조건을 갖춘다면 컨테이너는 반드시 정상적으로 구동할 수 있음
- 결국 컨테이너가 패키징하는 것은 어플리케이션(+실행에 필요한 구성요소들)로, 어플리케이션을 컨테이너 규격으로 감싼 형태라 할 수 있음



Machine Virtualization



Containers

- docker -v
- docker run -i -t centos:7
- 바로 CentOS 콘솔로 들어감, apt-get은 실행 안됨, rpm이나 yum은 실행됨
- exit
- docker pull ubuntu:20.04
- docker images
- docker create -i -t --name myubuntu ubuntu:20.04 (실행 안함)
- docker start myubuntu
- docker attach myubuntu
- Ctrl + P, Q
- docker attach myubuntu

- docker ps
- docker ps -a (docker의 모든 컨테이너 id와 이름은 동일, 줄여서 써도 됨)
- docker run -i -t ubuntu:20.04 echo hello world!
- docker rename myubuntu renamed
- docker rename renamed myubuntu
- docker ps -a
- docker rm myubuntu
- docker rm -f myubuntu
- docker container prune

- docker ps -a -q
- docker stop \$(docker ps -a -q)
- docker rm \$(docker ps -a -q)
- docker run -i -t --name network_test ubuntu:14.04
- ifconfig
- Ctrl + P, Q
- docker run -it --name mywebserver -p 80:80 ubuntu:14.04
- apt-get update
- apt-get install apache2 -y
- service apache2 start

| AssistantAgent vs UserProxyAgent

분류	AssistantAgent	UserProxyAgent
역할과 목적	AI 보조자 역할을 수행하도록 설계되었으며, 주로 LLM(Large Language Model)을 사용하여 질문에 답변하고 작업을 수행	인간 사용자를 대신하여 작동하며, 사용자 입력을 처리하고 다른 에이전트와 상호작용
입력 처리	기본적으로 인간의 입력이 필요하지 않도록 설계됨	인간 사용자로부터 입력을 받아 처리할 수 있음
코드 실행 능력	일반적으로 코드를 직접 실행하지 않음	코드를 실행할 수 있는 능력이 있어, 프로그래밍 작업을 수행할 수 있음
자율성	주어진 지시에 따라 작업을 수행하며, 상대적으로 수동적	자율적으로 작동하며, 사용자를 대신하여 결정을 내리고 작업을 수행할 수 있음
상호작용 방식	주로 다른 에이전트나 시스템과 상호작용	사용자와 직접 상호작용하며, 사용자의 의도를 해석하고 실행
용도	정보 제공과 작업 수행	사용자 인터페이스 역할을 하며 더 복잡한 작업과 의사결정 과정에 관여

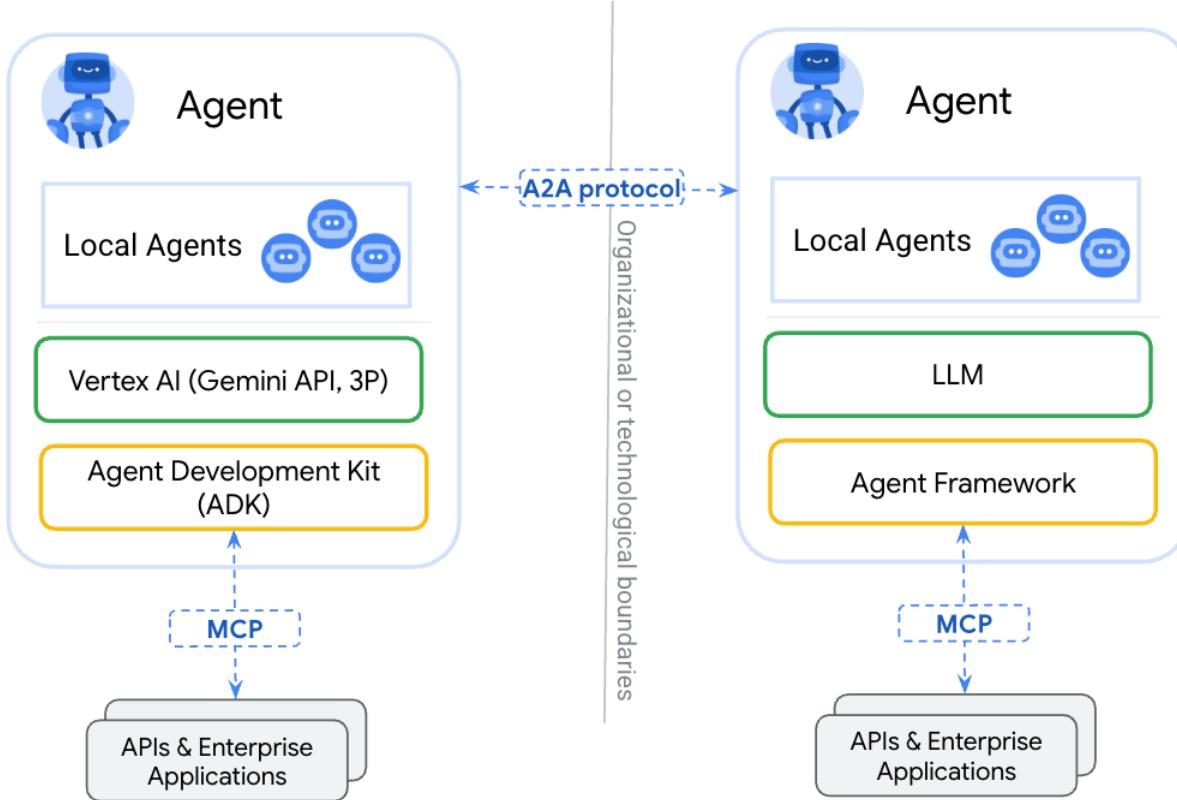
| register_for_llm vs register_function

- `assistant.register_for_llm`은 특정 에이전트에 도구를 등록하는 메서드. 도구의 시그니처와 설명을 LLM(Large Language Model)에 제공하여 도구 사용을 제안할 수 있게 함
- `register_function`은 두 개의 에이전트에 동시에 도구를 등록하는 함수, 도구를 제안할 수 있는 에이전트 (caller)와 실행할 수 있는 에이전트(executor)를 모두 지정. `assistant.register_for_llm`과 `ConversableAgent.register_for_execution`을 동시에 수행하는 효과가 있음
- `register_for_llm`은 단일 에이전트에 도구를 등록하는 반면, `register_function`은 두 에이전트에 동시에 등록
- `register_for_llm`은 더 세밀한 제어가 가능하며, `register_function`은 간편한 등록 방법을 제공
- `register_for_llm`은 특정 에이전트의 기능을 확장할 때 유용하며, `register_function`은 도구의 제안과 실행을 동시에 설정할 때 편리

- LLM 에이전트 팀을 구성하여 협업하는 프레임워크
- 역할(Role) 기반 태스크 분배 및 실행 지원
- 에이전트 역할 정의: 각 에이전트가 특정 역할을 수행하도록 설정 가능
- 병렬 및 순차 작업 수행: 복잡한 프로젝트를 다중 에이전트가 병렬로 처리 가능
- 유연한 커스터마이징: 다양한 조합으로 에이전트 설정 가능
- 사용자 친화적 구성: 간단한 설정으로 팀워크 자동화

- LLM 기반 AI 에이전트 간 표준화된 메시징 프로토콜: HTTPS를 이용한 JSON-RPC 2.0을 통신에 사용하며, SSE(Server Side Event)를 이용하여 업데이트를 스트리밍함
- 한 에이전트는 요청이나 작업을 시작하는 “클라이언트” 에이전트 역할이며, 다른 에이전트는 요청을 수신하고 이를 수행하는 “원격” 에이전트 역할
- 각 에이전트는 자신의 에이전트 카드를 게시할 수 있으며, 이 문서는 에이전트의 기능, 기술, API 엔드포인트 및 인증 요구사항을 설명: 클라이언트 에이전트는 이를 보고 작업에 적합한 파트너 에이전트를 식별
- 목표 에이전트가 선택되면, 클라이언트 에이전트는 **작업(Task)** 객체를 만들어 전송

카테고리	A2A(Agent-to-Agent)	MCP(Model Context Protocol)
주요 목표	에이전트 간 작업 교환 가능	LLM이 외부 도구나 컨텍스트에 접근할 수 있도록 함
설계 대상	자율 에이전트 간의 통신	추론 시 단일 에이전트 능력 강화
초점	다중 에이전트 워크플로우, 조정, 위임	동적 도구 사용, 컨텍스트 증가
실행 모델	에이전트가 작업 및 아티팩트를 주고받음	LLM이 툴을 선택하여 추론 중 실행함
보안	OAuth 2.0, API 키, 선언적 범위	애플리케이션 통합 레이어에서 처리됨
개발자 역할	엔드포인트를 통해 작업 및 아티팩트를 노출하는 에이전트 구축	모델이 사용할 수 있는 구조화된 도구와 컨텍스트 정의
생태계 파트너	구글, 세일즈포스, SAP, LangChain 등	Anthropic, 툴 기반 LLM UI에서의 새로운 채택



```
{  
    "jsonrpc": "2.0",  
    "method": "tasks/send",  
    "params": {  
        "taskId": "20250615123456",  
        "message": {  
            "role": "user",  
            "parts": [  
                {  
                    "text": "Find flights from New York to Miami on 2025-06-15"  
                }  
            ]  
        }  
    },  
    "id": "12345"  
}
```

```
{  
    "jsonrpc": "2.0",  
    "id": "12345",  
    "result": {  
        "taskId": "20250615123456",  
        "state": "completed",  
        "messages": [  
            {  
                "role": "user",  
                "parts": [  
                    {  
                        "text": "Find flights from New York to Miami on 2025-06-15"  
                    }  
                ]  
            },  
            {  
                "role": "agent",  
                "parts": [  
                    {  
                        "text": "I found the following flights from New York to Miami on June 15, 2025:  
1. Delta Airlines DL1234:  
Departs JFK 08:00, Arrives MIA 11:00, $320  
2. American Airlines AA5678: Departs LGA 10:30, Arrives MIA 13:30,  
$290  
3. JetBlue B9101: Departs JFK 14:45, Arrives MIA 17:45, $275"  
                    }  
                ]  
            }  
        ],  
        "artifacts": []  
    }  
}
```