

# LangGraph + Agent Chat-bot

발표 일시 : 2025.04.22(화)

발표자 : 강승현

목차

- 1. 프로젝트 개요
- 2. 프로젝트 구조
- 3. 코드 리뷰
- 4. 기능 시연
- 5. 보완 사항
- 6. 후기

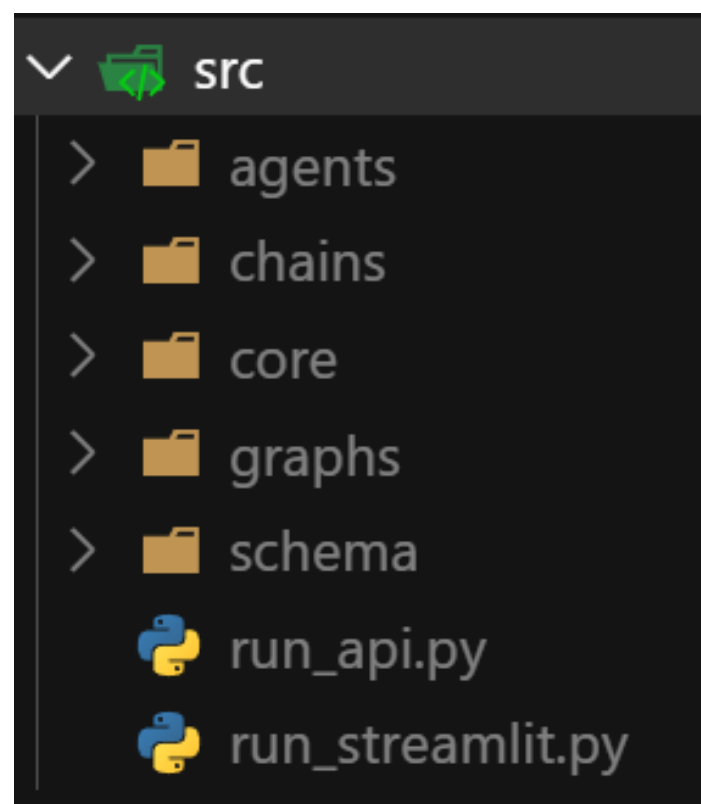
## 1. 프로젝트 개요

- 논문을 리뷰하여 공부해야하는 ai분야 특성상  
논문 리뷰를 도와줄 agent가 있으면 공부에 도움이 될거라 판단.  
해당 프로젝트를 시작함.
- 해당 프로젝트는 LangGraph 기반 대화형 chat-bot이며 몇몇 agent를 추가하였음.
- 추가한 agent는 'weather' / 'paper' 두가지 종류가 있음.  
weather agent: 서울의 날씨를 알려줌. (open weather map api 사용) \* 개인적인 사용 목적  
paper agent: arxiv의 논문을 로컬 db에 저장해주며 Document 객체로 변환하여 vector db에 저장해줌.

### ※ LangGraph란?

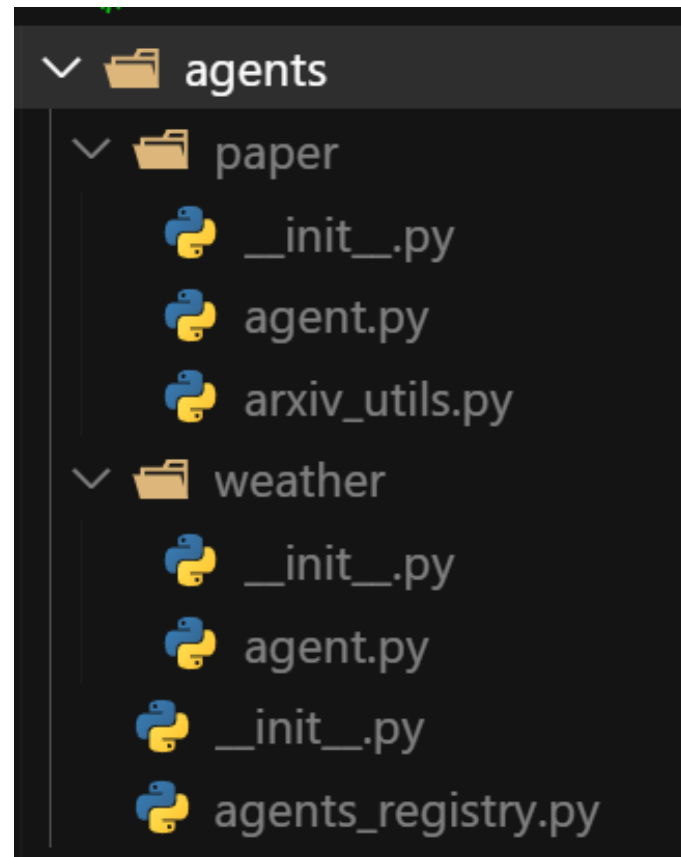
언어 모델과의 상호작용을 효율적이고 구조화된 방식으로 구현하여  
복합한 워크플로우와 정교한 의사 결정 프로세스를 구현하는 데 특화된 ai 개발  
프레임워크.

## 2. 프로젝트 구조



- agent/ : agent tool과 agent가 선언되는 폴더
- chains/ : LangChain runnable 객체의 chaining 관련 선언 폴더
- core/ : 재사용이 많은 프로젝트 기반 코드 폴더
- graphs/ : LangGraph node 분기와 workflow가 선언되는 폴더
- schema/ : agent 및 graph node에 argument 입력 데이터 타입 검증 관련 폴더
- run\_api.py : api server 실행 스크립트
- run\_streamlit.py : streamlit server 실행 스크립트

### 3. 코드 리뷰

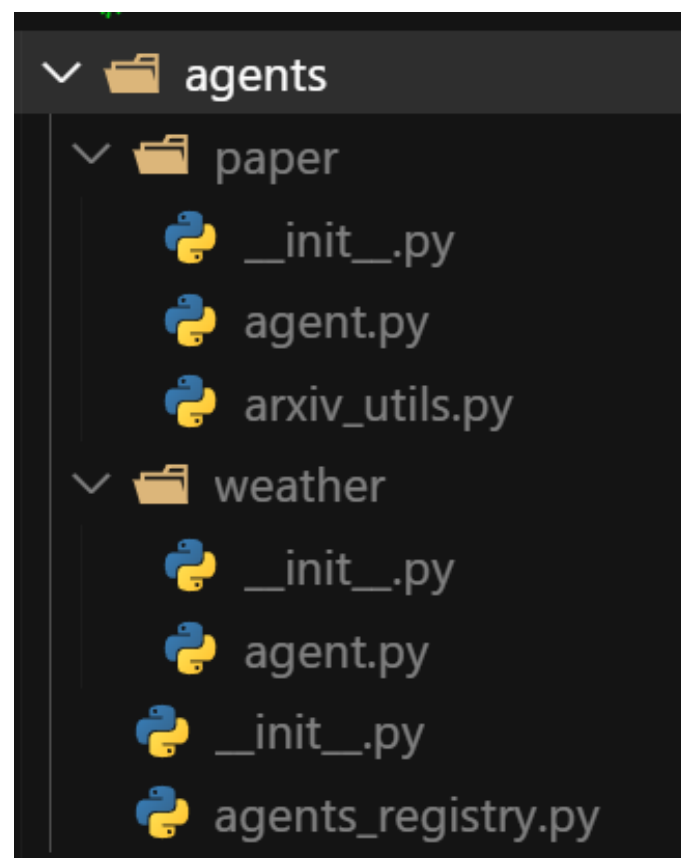


- weather/agent.py

개인적인 목적으로 만들어진 agent. 개인적으로 아침마다 늘 날씨를 체크하기에 해당 agent를 생성하였음.

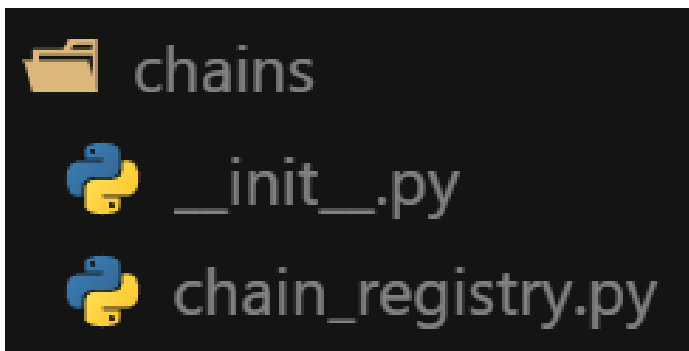
Openweathermap api를 사용하였으며 '서울'지역의 날씨를 가져올 수 있음.

### 3. 코드 리뷰



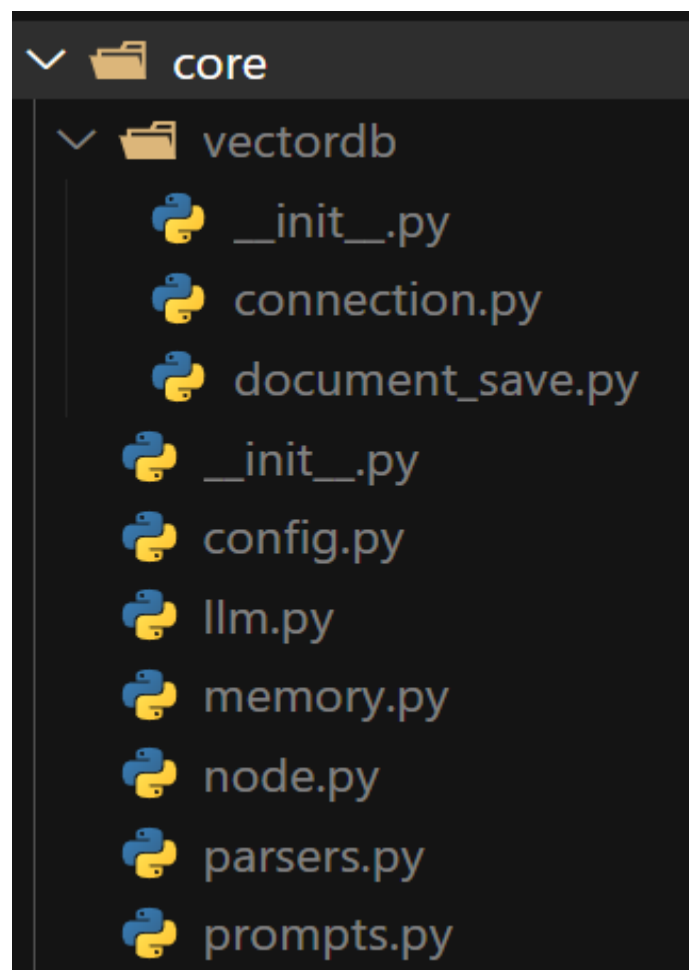
- `paper/agent.py`  
paper agent가 사용할 수 있는 tool이 선언되어 있는 스크립트.
- `paper/arxiv_utils.py`  
arxiv agent tool의 기반이 되는 함수들이 선언되어 있는 스크립트.  
'paper/agent.py'에 추가하지 않고 따로 분리해둔 이유는  
현재는 arxiv 라이브러리만을 사용하지만  
추후 arxiv 외 다른 라이브러리 사용도 고려하고 있기때문에  
목적과 역할을 구분할 목적으로 분리.
- `agents_registry.py`  
해당 프로젝트에서 사용하는 모든 agent들은 이 곳에서 선언됨.

### 3. 코드 리뷰



- chain\_registry.py  
prompt / llm / memory / parser 등 해당 프로젝트에 사용할 모든 runnable 객체는 해당 스크립트에서 선언됨.

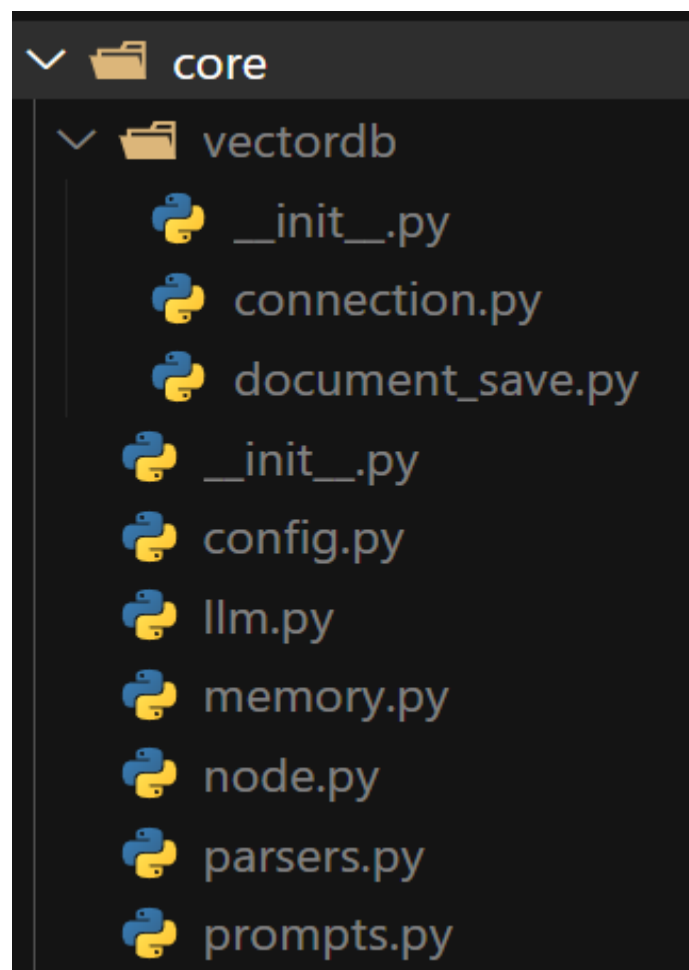
### 3. 코드 리뷰



- vectordb/connection.py  
vectordb 객체를 선언해주는 스크립트.
- vectordb/document\_save.py  
Document 객체를 vector db에 저장하는 스크립트.

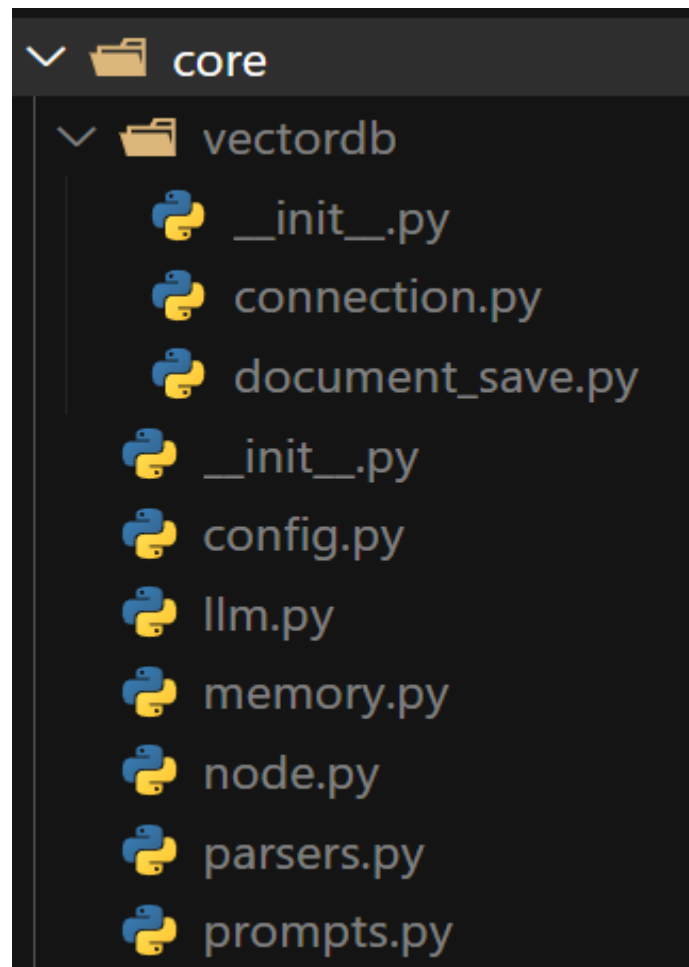


### 3. 코드 리뷰



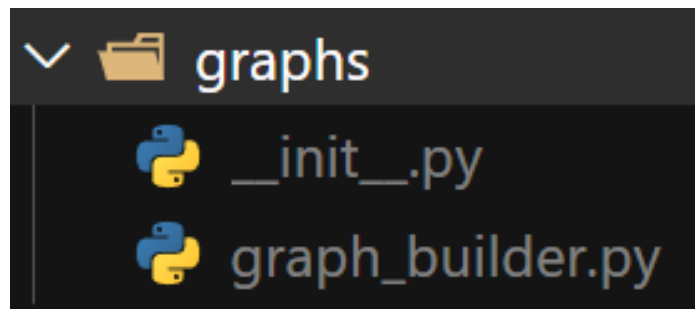
- config.py  
.env파일로부터 api key, db path 등 프로젝트에 사용될 설정 값들을 선언하는 스크립트
- llm.py  
runnable 객체의 chaining에 사용될 llm 객체 선언 스크립트. (현재는 openai만 선언 가능)
- memory.py  
llm이 대화를 기억할 수 있도록 만들어주는 memory 기능.  
runnable 객체를 상속 받아서 직접 class로 선언하였음. 단, agent에는 적용되지 않음.
- node.py  
LangGraph workflow에 사용되는 node 생성 함수 스크립트.

### 3. 코드 리뷰



- parsers.py  
runnable 객체의 chaining에 사용될 output parser 선언 스크립트.
- prompts.py  
각 LLM에 전달될 message와 prompt template 선언 스크립트.  
human message와 system message로 구분되어 있음.

### 3. 코드 리뷰

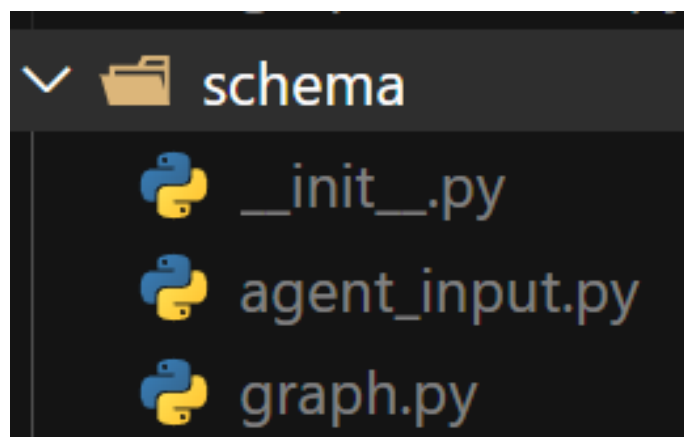


- graph\_builder.py

LangGraph workflow가 compile되어 선언되는 스크립트.



workflow 내 node 또한 해당 스크립트에서 선언됨.

### 3. 코드 리뷰




- agent\_input.py  
agent tool 내 입력(argument)에 대한 schema 스크립트.  
입력 데이터 검증이 가능한 pydantic 라이브러리를 사용함.
- graph.py  
LangGraph workflo의 node에 입력될 state에 대한 schema 스크립트.  
타입 힌트를 명시할 수 있는 Typedict 라이브러리를 사용함.

### 3. 코드 리뷰

 run\_api.py  
 run\_streamlit.py

- run\_api.py  
api server 실행 스크립트.  
FastAPI와 langserve를 통해 api 서버를 구축함.
- run\_streamlit.py  
frontend server 실행 스크립트.  
대화 내용 저장 및 대화형 UI가 구현되어 있음.

### 3. 코드 리뷰



Dockerfile  
run.sh  
requirements.txt

- Dockerfile  
Docker container 설정 파일
- run.sh  
Docker build시 해당 프로젝트 실행 관련된 bash command를 작성해둔 스크립트.  
api serve와 streamlit serve를 실행함.
- requierments.txt  
해당 프로젝트 실행에 필요한 python 라이브러리 목록.

4. 기능 시연

● Memory 기능

### Lang-Secretary



 안녕

 안녕하세요! 무엇을 도와드릴까요? 😊

 우리집 강아지 이름은 '철수'야


 네, 기억할게요! 앞으로 '철수'에 대해 이야기할 때 참고할게요. 철수에 대해 더 궁금한 점이나 이야기 해주고 싶은 게 있으면 언제든지 말씀해 주세요!


 우리집 강아지 이름이 뭐라고?

 너희 집 강아지 이름은 '철수'야!

4. 기능 시연

● Weather agent

 오늘 날씨 알려줄래?

 오늘 서울의 날씨는 맑고, 기온은 약 11.8도입니다.

```
> Entering new AgentExecutor chain...
Thought: 사용자가 오늘의 날씨를 알고 싶어 합니다. 별도의 도시명을 언급하지 않았으므로, 기본값인 Seoul의 날씨 정보를 확인해야 합니다
Action: weather_tool
Action Input: Seoul
<<<<<<<<<<<<<<<< Seoul의 날씨를 조회합니다. >>>>>>>>>>>>>>

Observation: 현재 Seoul의 온도는 16.76도이고, 날씨는 Mist입니다.
Thought:Thought: 이제 사용자가 요청한 오늘의 날씨 정보를 알게 되었습니다.
Final Answer: 오늘 서울의 날씨는 16.76도이고, 안개(Mist)가 끼어 있습니다.


> Finished chain.
INFO:      127.0.0.1:61789 - "POST /graphbot/invoke HTTP/1.1" 200 OK
```




4. 기능 시연

● Paper agent

<https://arxiv.org/abs/2210.03629> 이 논문 요약해줘





논문을 요약하려면 먼저 논문을 vector db에 저장해야 합니다. 논문 정보를 입력해 주세요.

<https://arxiv.org/abs/2210.03629> 이 논문 vector db에 저장해



vector db 저장에 성공했습니다. 이 논문의 요약본이 필요하신가요?

<https://arxiv.org/abs/2210.03629> 이 논문 요약해줘



대형 언어 모델(LLM)은 언어 이해와 상호작용적 의사결정 과제에서 인상적인 성능을 보여주고 있지만, 추론(예: chain-of-thought prompting)과 행동(예: action plan generation) 능력은 주로 별개의 주제로 연구되어 왔습니다. 본 논문에서는 LLM이 추론 과정(reasoning traces)과 과제별 행동(task-specific actions)을 교차적으로 생성하도록 하여, 두 요소 간의 시너지를 극대화하는 방법을 탐구합니다. 추론 과정은 모델이 행동 계획을 유도, 추적, 갱신하고 예외 상황을 처리하는 데 도움을 주며, 행동은 외부 지식베이스나 환경과 상호작용하여 추가 정보를 수집할 수 있게 합니다. 저자들은 이러한 접근법을 ReAct라고 명명하고, 다양한 언어 및 의사결정 과제에 적용하여 기존 최첨단 기법 대비 효과적임을 입증했습니다. 또한, reasoning이나 acting 요소가 없는 방법에 비해 인간의 해석 가능성과 신뢰성이 향상됨을 보였습니다. 구체적으로, 질문 응답(HotpotQA)과 사실 검증(Fever) 과제에서 ReAct는 단순 Wikipedia API와 상호작용함으로써 chain-of-thought reasoning에서 흔히 발생하는 환각(hallucination)과 오류 전파 문제를 극복하고, reasoning trace가 없는 baseline보다 더 해석 가능한 인간 유사 문제 해결 과정을 생성합니다. 두 가지 상호작용적 의사결정 벤치마크(ALFWorld, WebShop)에서는 ReAct가 모방 학습(imitation learning) 및 강화 학습(reinforcement learning) 방법 대비 각각 34%와 10%의 절대 성공률 향상을 보였으며, 단 1~2개의 in-context 예시만으로도 효과를 보였습니다. 프로젝트 사이트 및 코드는 <https://react-lm.github.io> 에서 확인할 수 있습니다.

## 5. 보완 사항

- 불완전한 Memory 기능

현재 LLM이 ReAct를 사용할 경우 관련 prompt까지 전부 Memory에 저장하는 이슈가 있고

이로 인해 LLM output parser 이슈가 있음.

(Action / Action input / Observation / Thought 등을 input에 다시 사용.)

user input과 Final answer만 저장하도록 보완 필요.

```
> Entering new AgentExecutor chain...
Action: save_paper_in_vector_db
Action Input: {"url": "https://arxiv.org/abs/2210.03629"}
Observation:
  "status": save_success,
  "arxiv_id": 2210.03629,
  "message": vector db에 저장을 완료하였습니다.,
  "system_message":
    1. vector db 저장에 성공했다면 요약본이 필요냐고 사용자에게 물어봐.
    2. 논문의 정보를 찾을 수 없거나 vector db 저장에 실패했다면 사용자에게 논문 url을 다시 확인해달라고 말해
    3. 이미 논문이 vector db에 저장되어있다면 이미 저장되어있음을 알려주면서 요약본이 필요냐고 사용자에게 물어봐
    4. 만약 이미 사용자가 요약본을 요청했다면 요약본을 출력해줘

    - 하나의 Thought/Action만 수행하거나
    - Final Answer를 작성할 때는 Action을 사용하지 마세요.
    - Observation 이후 Final Answer를 직접 작성하지 말고, 다음 Thought를 통해 진행하세요.


    <CHAT_HISTORY_START>에서 <CHAT_HISTORY_END> 사이의 대화는 전부 무시해


Thought:Thought: 논문이 vector db에 성공적으로 저장되었으니, 사용자에게 요약본이 필요한지 물어봐야 한다.
Final Answer: 논문이 vector db에 성공적으로 저장되었습니다. 해당 논문의 요약본이 필요하신가요?

> Finished chain.
INFO: 127.0.0.1:61067 - "POST /graphbot/invoke HTTP/1.1" 200 OK
```

5. 보완 사항

- prompt engineering을 통한 출력물 개선
- 아직까지는 chat-gpt의 출력물이 훨씬 더 효과적임.
- Prompt를 통해 출력물에 차별화를 두면 더욱 효율적일 것으로 판단.

<https://arxiv.org/abs/2210.03629> 이 논문 요약해줘



대형 언어 모델(LLM)은 언어 이해와 상호작용적 의사결정 과제에서 인상적인 성능을 보여주고 있지만, 추론(예: chain-of-thought prompting)과 행동(예: action plan generation) 능력은 주로 별개의 주제로 연구되어 왔습니다. 본 논문에서는 LLM이 추론 과정(reasoning traces)과 과제별 행동(task-specific actions)을 교차적으로 생성하도록 하여, 두 요소 간의 시너지를 극대화하는 방법을 탐구합니다. 추론 과정은 모델이 행동 계획을 유도, 추적, 갱신하고 예외 상황을 처리하는 데 도움을 주며, 행동은 외부 지식베이스나 환경과 상호작용하여 추가 정보를 수집할 수 있게 합니다. 저자들은 이러한 접근법을 ReAct라고 명명하고, 다양한 언어 및 의사결정 과제에 적용하여 기존 최첨단 기법 대비 효과적임을 입증했습니다. 또한, reasoning이나 acting 요소가 없는 방법에 비해 인간의 해석 가능성과 신뢰성이 향상됨을 보였습니다. 구체적으로, 질문 응답(HotpotQA)과 사실 검증(Fever) 과제에서 ReAct는 단순 Wikipedia API와 상호작용함으로써 chain-of-thought reasoning에서 흔히 발생하는 환각(hallucination)과 오류 전파 문제를 극복하고, reasoning trace가 없는 baseline보다 더 해석 가능한 인간 유사 문제 해결 과정을 생성합니다. 두 가지 상호작용적 의사결정 벤치마크(ALFWorld, WebShop)에서는 ReAct가 모방 학습(imitation learning) 및 강화 학습(reinforcement learning) 방법 대비 각각 34%와 10%의 절대 성공률 향상을 보였으며, 단 1~2개의 in-context 예시만으로도 효과를 보였습니다. 프로젝트 사이트 및 코드는 <https://react-lm.github.io> 에서 확인할 수 있습니다.

<https://arxiv.org/abs/2210.03629> 이 논문 요약해줘

좋아, 요청한 논문  
"Self-Instruct: Aligning Language Models with Self-Generated Instructions"  
를 요약해줄게.

### 논문 요약:

Self-Instruct: Aligning Language Models with Self-Generated Instructions

#### 1. 연구 배경

- 기존 **Instruction Tuning** (예: FLAN, T5)은 사람이 직접 만든 **명령어-응답 데이터**로 LLM을 fine-tune한다.
- 하지만 이 방법은 **대규모 인간 작업**에 의존해서 **비용이 많이 들고** 확장성이 떨어진다.
- 그래서 "사람 없이도 모델 스스로 Instruction 데이터를 만들어 fine-tuning할 수 있을까?"라는 질문을 한다.

#### 2. 제안 방법: Self-Instruct

Self-Instruct 방법은 크게 두 단계를 거친다:

1. 초기 **Seed Instruction** 몇 개(= 사람이 만들어준 기본 질문/명령어들)를 준비.
2. LLM이 스스로:
  - 새로운 Instruction을 생성하고
  - 그에 대한 답변도 스스로 만들어낸다.
3. 이렇게 생성한 대규모 데이터셋으로 다시 LLM을 fine-tuning한다.

✅ 즉, 적은 인간 개입으로 수십만 개 Instruction-Answer pair를 얻어냄.

## 6. 후기

- 지금까지 'prompt는 그냥 적당히 하면 되는 거 아닌가?' 이러한 생각을 하였으나 LangChain 객체를 chaining 하는 과정, 그리고 agent tool에서 prompt가 사용되는 것을 보며 생각하던 것 이상으로 중요한 부분을 차지한다고 느꼈음.
- 입력한 prompt에 따라 agent가 tool을 사용할 때가 있고 안 할 때가 있다. agent가 조금 더 효과적으로 tool을 사용하게 하기 위해서는 tool description을 정확히 적어야 한다는 생각을 하였으며 이로 인해 개발자가 고려할 부분이 더 늘어난 거 같다.
- 기존에는 LLM을 아무리 fine-tuning을 하더라도 결국 "chat-gpt 보다 좋아?" 라는 물음에 답변할 수 없었으나 LangChain/LangGraph를 활용한다면 철저히 개인화된 chat-bot service를 만들 수 있을다는 가능성을 확인하였으며 이는 다른 사람에게 chat-gpt와 견줄 선택지를 줄 수 있을거라 판단한다.