

지도교수 확인란
강승식 (인)

2024-2학기

알파프로젝트 최종보고서(팀유형)

구분	내용	비고
교과명	알파프로젝트 I (139940-01)	
신청학점/이수구분	9학점 (전공선택)	
지도교수 성명	강승식 교수님	
팀명	LLM_newbie	
팀원 정보	총 3명	

본 프로젝트팀은 위와 같이 알파프로젝트 최종보고서를 제출합니다.

2024. 12. 13.

(학과/전공) 소프트웨어학과
(학과/전공) 소프트웨어학과
(학과/전공) 소프트웨어학과

(학번) 20203146
(학번) 20203135
(학번) 20203148

(성명) 조현상
(성명) 정의현
(성명) 차민수

(안) 현상
(안) 정의현
(안) 차민수

- 팀원 전체의 정보 기입 후 서명 바랍니다. 서명은 필체로 직접 서명 바랍니다. -

I. 알파프로젝트 최종보고서

[팀 작성]

1. 프로젝트 개요(추진배경 및 목표)

오늘날 정보는 우리의 삶을 윤택하게 만드는 중요한 도구로 자리 잡았습니다. 사람들은 유용한 정보를 통해 더 나은 선택을 하고 삶의 질을 향상시키고 있습니다. 그러나 정보의 확산과 더불어 가짜 정보, 허위 광고 등으로 인해 피해를 입는 사례도 점차 증가하고 있습니다. 특히, 최근에는 유명인을 도용한 허위 광고와 거짓된 정보를 이용해 사람들을 선동하는 행위가 빈번히 발생하고 있습니다. 이러한 문제는 신뢰할 수 있는 정보 제공의 중요성을 더욱 부각시키고 있습니다.

저희 팀은 이러한 문제를 해결하기 위해 최신 데이터 분석 기술과 최근 주목받고 있는 대규모 언어 모델 기술을 활용하여, 정보를 신뢰성과 진위 여부에 따라 판별해주는 시스템을 개발하고자 합니다. 이를 통해 가짜 정보와 허위 광고로 인한 피해를 최소화하고, 신뢰할 수 있는 정보 환경을 구축하여 사람들이 보다 안전하고 윤택한 삶을 살 수 있도록 기여하는 것을 목표로 합니다.

2. 프로젝트 결과(주요 성과 및 산출물, 활동효과)

본 프로젝트는 가짜 정보를 판별하는 프로그램 개발을 목표로 하였습니다. 저희가 목표로 한 프로그램을 만들기 위한 핵심 기술로 LLM 활용 기술, RAG 시스템 활용, Langchain 활용이라고 판단하였습니다. 그리하여 LLM 및 RAG 시스템, Langchain에 대해 공부하고 활용해 볼 수 있는 '국회 회의록 기반 의정 활동 지원 및 대국민 알권리 보장 챗봇 서비스'를 제작하는 DATA AI 분석 경진 대회에 참여 하여 챗봇 서비스 구축을 완성 하였습니다. 이 공모전을 통해 LLM 기술과 RAG 기술을 공부하고 활용할 수 있었으며 공모전에서 사용한 기술을 수정 및 개선을 통해 알파프로젝트에 활용하였습니다. 네이버, 구글, 유튜브 등에서 수집한 공신력 있는 데이터를 바탕으로 구축된 데이터베이스로부터 LLM이 정보를 얻어 문제를 해결할 수 있게 하여 잘못된 답변이 나오지 않도록 하였으며 특히, LangGraph 기술을 사용해서 복잡하고 다층적인 정보 환경에서도 동적으로 문제를 해석, 해결하여 신뢰도 높은 결과를 도출할 수 있도록 설계되었습니다. 또한 이 시스템을 사용자들이 손쉽게 활용할 수 있도록 웹 플랫폼에 배포하였습니다. 직관적이고 간편한 웹 인터페이스를 제공하여, 누구나 쉽게 접속하여 필요한 정보를 검증하고 활용할 수 있도록 접근성을 강화하였습니다. 네이버, 구글, 유튜브 등에서 수집한 공신력 있는 데이터를 바탕으로 신뢰성과 허위성을 구분한 데이터베이스를 마련하고, 이를 활용하여 누구나 쉽게 접근할 수 있는 웹 플랫폼을 구축하였습니다. 이러한 활동을 통해 정보의 신뢰성을 높이고, 가짜 정보의 확산을 차단하며, 공신력 있는 정보의 가치를 재고하는 데 기여하였습니다. 사용자들은 직관적이고 간편한 웹 인터페이스를 통해 필요한 정보를 빠르게 확인하고 검증할 수 있으며, 이를 통해 보다 나은 의사결정을 내릴 수 있습니다. 나아가, 허위 정보로 인한 사회적 갈등과 혼란을 줄이고 정보 투명성을 확보함으로써 안정된 사회 환경 조성에 도움을 줄 것으로 기대됩니다.

2. 활동 소감

이번 프로젝트를 통해 저희 팀은 신뢰할 수 있는 정보 환경을 구축하는 일의 중요성을 다시금 깨닫게 되었습니다. 빠르게 변화하는 정보 환경 속에서 기술이 사람들에게 미치는 영향을 깊이 고민하고, 이를 해결하기 위해 데이터와 기술을 결합한 실질적인 솔루션을 개발할 수 있었습니다.

프로젝트를 진행하면서 예상치 못한 어려움도 많았지만, 이를 해결해 나가며 팀원 모두가 한층 더 성장 할 수 있었고, 협업의 중요성을 느끼는 소중한 경험이 되었습니다. 무엇보다도, 저희가 개발한 시스템이 실제로 사용자들의 삶에 긍정적인 변화를 가져올 수 있다는 가능성에 큰 보람을 느낍니다.

앞으로도 저희 팀은 사회에 긍정적인 영향을 미칠 수 있는 기술을 지속적으로 연구하고 발전시키며, 더욱 신뢰할 수 있는 정보 환경 조성에 기여하기 위해 노력하겠습니다. 감사합니다.

본 프로젝트팀은 위와 같이 알파프로젝트 최종 보고서를 제출합니다.

2024. 12. 13.

(학과/전공) 소프트웨어학과

(학번) 20203146

(성명) 조현상

(인) 현상

(학과/전공) 소프트웨어학과

(학번) 20203135

(성명) 정의현

(인) 정현

(학과/전공) 소프트웨어학과

(학번) 20203148

(성명) 차민수

(인) 민수

- 팀원 전체의 정보 기입 후 서명 바랍니다. 서명은 필체로 직접 서명 바랍니다. -

- 유의사항 -

- 1) 최종보고서는 I,II,III 모두 제출하여야 하며 팀당 1부 제출 (최종보고서를 제출해야 최종지원금이 지급됨)
- 2) 본 양식은 편집 가능 (이미지, 표 등은 첨부 가능하나 별첨으로 함)

II. 알파프로젝트 주차별 보고서

[팀단위 작성]

프로젝트명	뉴스 데이터를 기반으로 유튜브 영상의 가짜 정보 판별
-------	----------------------------------

주차별	활동내역	활동시간	지도교수 확인란
			강수석 (서명/날인)
1주차	알파프로젝트 개발과정 설립 및 공모전 주제 선정	18시간	
2주차	데이터 수집 시스템 구축 및 데이터 베이스 구축	18시간	
3주차	데이터 수집 시스템 구축 및 데이터 베이스 구축	18시간	
4주차	데이터 전처리 및 RAG 구현, 기본 인터페이스 구축	18시간	
5주차	비용 절감을 위한 소스코드 교체	18시간	
6주차	성능 향상을 위한 history 및 프롬프트 엔지니어링	18시간	
7주차	임베딩 성능 개선 및 버그 개선과 최적화	18시간	
8주차	공모전 제출 문서 작성, 알파프로젝트에 활용 방안 모색	18시간	
9주차	가짜 뉴스 판별 알고리즘 설계 및 langserve 학습	18시간	
10주차	LangGraph 학습 및 실습, Fast API, SQL Alchemy 학습	18시간	
11주차	LangGarph 기초 설계, youtube 음성 데이터 텍스트 변환	18시간	
12주차	뉴스 검색 tool 설계 및 LangGraph에 연결	18시간	
13주차	LangGraph 프롬프트 엔지니어링, 백엔드 서버 구축	18시간	
14주차	프로그램 최종 완성, 프론트엔드 및 로그인 기능 구현	18시간	
15주차	LangGraph와 서버 연결, 버그 수정 및 최적화	18시간	
합계		270시간	

본 프로젝트팀은 위와 같이 알파프로젝트 최종 보고서를 제출합니다.

20

(학과/전공) 소프트웨어학과
(학과/전공) 소프트웨어학과
(학과/전공) 소프트웨어학과

(학번) 20203146
(학번) 20203135
(학번) 20203148

(성명) 조현상
(성명) 정의현
(성명) 차민수

(인) 현상
(인) 정의현
(인) 차민수

III. 알파프로젝트 주차별 상세 보고서(활동사진 포함)

[팀 작성]

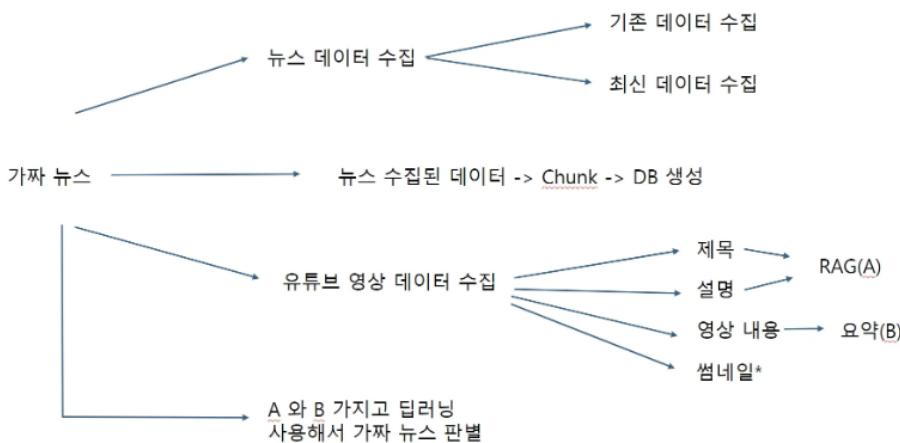
프로젝트명

뉴스 데이터를 기반으로 유튜브
영상의 가짜 정보 판별

<1주차>

활동시간	9/2 18:00 ~ 20:00 9/4 18:00 ~ 20:00 9/7 13:00 ~ 17:00	9/3 13:00 ~ 18:00 9/6 13:00 ~ 18:00	활동장소	온라인 화상 미팅(디스코드) 미래관 무한상상실 미래관 자유주행스튜디오
------	---	--	------	--

가짜 뉴스 시나리오



알파프로젝트 개발과정 설립 및 참가할 공모전 주제 선정

뉴스 데이터를 기반으로 하는 유튜브 영상의 가짜 정보를 판별하는 프로그램을 짜기 위해 전체적인 개발 흐름을 설립하고 각 파트별 개발에 필요한 기술 선정 및 스터디를 진행하였습니다.



또한 알파프로젝트를 진행하면서 사용되는 기술인 LLM을 이용할 수 있는 공모전 주제를 선정하여 참가하고 어떠한 방식으로 기술을 이용할지에 대하여 의견하였습니다.

문제 설명

데이터 설명

QnA

문제 제목

국회 회의록 기반 의정활동 지원 및 대국민 알권리 보장 챗봇 서비스

(경희지경) 국회도서관

일정 참가신청 마감 D - 2 (참가신청 : 2024-08-23~2024-10-25)

참여하기

국회도서관장상 상금 200만원

문제 개요

문제 제목

국회 회의록 기반 의정활동 지원 및 대국민 알권리 보장 챗봇 서비스

내용

국회는 법률안·예산안·의안 등 의심의 심의를 통해 법안의 요구하는 기능을 수행하며, 국민의 의사를 국경에 반영합니다. 국회 회의록에는 의사일정, 보고사항, 부의된 안건 등 의사에 관한 모든 발언이 기록됩니다. 최근 빠르게 발전하고 있는 언어모델 기술을 활용하여 국회 회의록을 기반으로 시각을 걸색하고 질의응답할 수 있는 인공지능 모델 및 서비스를 개발함으로써 국민의 알권리를 보장하고 국회의 일법 활동 투명성을 높일 수 있을 것입니다.

문제 설명

국회 회의록 기반의 자식 검색 데이터는 국회에서 다른 안건과 국회의원들의 발언을 질문과 답변 형식으로 구분하여 기록한 데이터입니다. 이를 활용한 첫봇 서비스는 두 가지 목적을 가지고 있습니다. 첫째, 국회의원이 과거 회의록을 검색하고 질의응답을 통해 더 나은 정책을 개발하는 등 의정활동을 지원하는 도구로 활용될 수 있습니다. 둘째, 국민이 정책 및 법안과 관련된 정보를 잘 이해할 수 있도록 도와 국민의 알권리를 보장하는 데 기여할 수 있습니다.

목표

문제 해결 목표 성능 또는 기대 결과물에 대한 설명

* 현안 히스토리 QA : 특정 정책/법률 현안에 관련된 논의 내용을 검색하고 시간순 및 이슈별로 정리/요약하여 답변

평가 방법 및 설명

* 답변으로 제시된 정보의 정확성(한국어 포함 여부)과 누락된 정보 없는지에 대한 재현성의 두 측면을 함께 고려하여 전문가가 평가

<2주차>

활동시간	9/9 18:00 ~ 20:00 9/10 13:00 ~ 18:00 9/11 18:00 ~ 20:00 9/12 13:00 ~ 18:00 9/13 13:00 ~ 17:00	활동장소	온라인 화상 미팅(디스코드) 미래관 무한상상실 미래관 자유주행스튜디오
-------------	---	-------------	--

Commits

History for fake-news-detection / Naver_api_news on [feat/data-collection](#)

Commits on Sep 10, 2024

- FEAT: 네이버 API를 이용한 입력 키워드로 뉴스 수집
Blue-Ladder committed 4 days ago 3c85878
- End of commit history for this file

Commits on Sep 10, 2024

- FEAT: create sqlite db UH3135 committed on Sep 10 a1d17e9

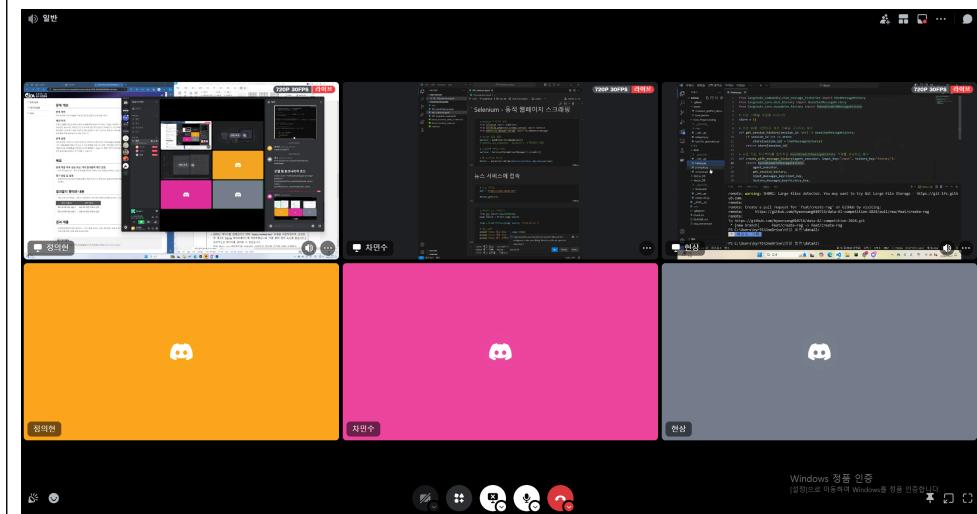
Commits on Sep 7, 2024

- Merge pull request #2 from hyeonsang010716/feat/create-db
hyeonsang010716 authored on Sep 7 9477929
- Merge branch 'main' of https://github.com/hyeonsang010716/data-AI-competition-2024 into feat/create-db
UH3135 committed on Sep 7 9c4dbdd
- Merge branch 'feat/create-db' of https://github.com/hyeonsang010716/data-AI-competition-2024 into feat/create-db
UH3135 committed on Sep 7 18da2eb
- Merge pull request #3 from hyeonsang010716/feat/create-rag
UH3135 authored on Sep 7 0dd4937f
- FEAT: init navie_rag
hyeonsang010716 committed on Sep 7 a85c7a9
- FEAT: 질문 답변 쌍으로 벡터 DB에 저장하는 방법 추가
UH3135 committed on Sep 7 658046c
- FEAT: add how to create vectorDB
UH3135 committed on Sep 7 feab36d
- FEAT:rename how_to_embedding
UH3135 committed on Sep 7 9dea4c2

[알파프로젝트] 네이버 API 활용 및 뉴스 수집 시스템 구축, [공모전] 데이터 베이스 구축 및 관리 방안 연구 Vector DB 사용 준비

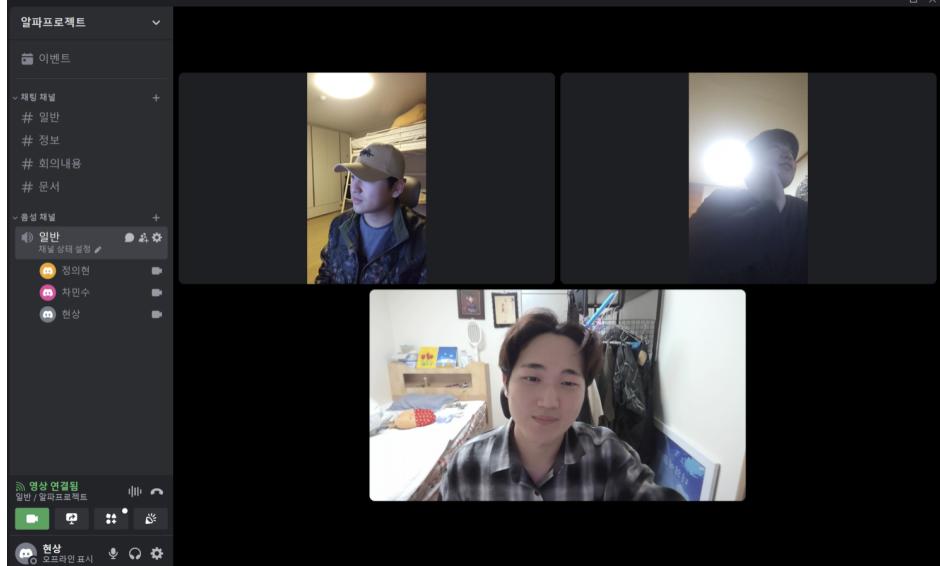
네이버 API로 뉴스 데이터를 가져오는 코드를 구현 했으나 한계가 있음을 깨닫고 웹크롤링을 통해 뉴스 데이터를 가져오기로 하였습니다.

VectorDB를 통한 데이터 저장 및 관리 시스템 구축하는 방식 학습 및 코드 구현하였고 일부 질문-답변 쌍 데이터 초기 테스트를 완료하였습니다.



<3주차>

활동시간	9/18 12:00 ~ 18:00 9/19 12:00 ~ 18:00 9/20 18:00 ~ 20:00 9/21 13:00 ~ 17:00	활동장소	온라인 화상 미팅(디스코드) 미래관 무한상상실 미래관 자유주행스튜디오
-------------	--	-------------	--



> Commits on Sep 21, 2024

- FEAT: 멀티쿼리 리트리버와 chroma DB 구현
hyeonsang010716 committed on Sep 21

> Commits on Sep 15, 2024

- Merge pull request #6 from hyeonsang010716/feat/create-db
hyeonsang010716 authored on Sep 15
- FEAT: delete wrong directory
UH3135 committed on Sep 15

> Commits on Sep 14, 2024

- Merge pull request #5 from hyeonsang010716/feat/create-db
hyeonsang010716 authored on Sep 14
- FEAT: add searching
UH3135 committed on Sep 14
- FEAT: add keywords to data
UH3135 committed on Sep 14
- Merge branch 'feat/create-db' of https://github.com/hyeonsang010716/data-AI-competition-2024 into feat/create-db
UH3135 committed on Sep 14

```
from langchain_llm import OllamaEmbeddings
from langchain_chroma import Chroma
import os
import pickle
from langchain.schema import Document
from langchain.core import Document as langchain.Document
from langchain.text_splitter import CharacterTextSplitter

def save_documents(documents, file_path):
    with open(file_path, "wb") as file:
        pickle.dump(documents, file)

def load_documents(file_path):
    with open(file_path, "rb") as file:
        return pickle.load(file)

def load_learning_materials():
    learning_materials = []
    base_path = "./assets"

    # TXT 파일 읽기
    txt_files = [f for f in os.listdir(base_path) if f.endswith(".txt")]
    for txt_file in txt_files:
        file_path = os.path.join(base_path, txt_file)
        if os.path.exists(file_path):
            with open(file_path, encoding="utf-8") as file:
                learning_materials += file.read() + "\n\n\n"
    return learning_materials

def create_index():
    embeddings = OllamaEmbeddings(model="nomic-embed-text") # 임베딩 모델
    directory_path = "./vector_06/chromaDB"
    if os.path.exists(directory_path):
        print("존재하는 임베딩 세팅")
        db = Chroma(
            embedding_function=embeddings, persist_directory=directory_path
        )
        recursive_split_document = load_documents("./assets/recursive_splitted_document.pkl")
        return db, recursive_split_document
    else:
        print("새로운 임베딩 세팅")
        learning_materials = load_learning_materials()
        texts = learning_materials.split("\n\n\n")
        # texts 리스트를 Document 객체로 변환
        documents = [Document(page_content=text) for text in texts]
```

[알파프로젝트] 네이버
뉴스 데이터 웹크롤링
수집 방식 구현,
[공모전] 데이터 베이스
구축 및 Chroma DB
적용, 데이터 모듈화
작업

2주차에 진행했던
네이버 api를 사용하지
않고 크롤링 방식으로
데이터를 수집하는 코드
구현하고 lm을 통한
뉴스 내용 요약 기능
추가하였습니다.

Chroma DB에 활용한
멀티쿼리 리트리버
기능을 구현하는데
초점을 두고 질문-답변
쌍의 저장 및 검색 성능
향상시켰습니다.

코드들을 모듈화하여
재사용성 및 유지보수성
향상 시켰습니다.

<4주차>

활동시간	9/23 18:00 ~ 20:00	9/24 13:00 ~ 18:00	활동장소	온라인 화상 미팅(디스코드)
	9/25 18:00 ~ 20:00	9/26 13:00 ~ 18:00		미래관 무한상상실
	9/27 13:00 ~ 17:00			미래관 자유주행스튜디오

Simple chat

국회일정

국회 입법과 관련된 여러 논의가 있습니다. 첫째, 원자력안전위원회의 역할에 대한 논의가 있습니다. 입법 단계에서부터 원자력안전위원회가 안전성과 품질보증에 대한 관리 감독에 적극적인 역할을 수행했는지에 대한 의문이 제기되고 있으며, 향후 입법 과정에서 이러한 관리 감독에 대한 적극적인 참여가 필요하다는 의견이 있습니다. 둘째, 규제 단체는 입법과 관련하여 규제 개혁 방향이나 위치를 흐름으로 인해 국회 입법부 간의 협력을 통해서 어려움이 예상된다라는 우려의 목소리가 있습니다. 이에 대해 국회 법안을 우선 추진하고, 추가적으로 행정 규제 분야에 포괄적인 조항을 포함하는 방향으로 진행할 계획입니다. 셋째, 개인정보 보호와 활용에 대한 법적 규정 및 관련 입법 고민을 종합적으로 정리한 자료를 마련할 계획이 있으며, 공공데이터 혁신과 개인정보 보호에 대한 법적 핵심 정점을 정리하고 관련 입법 방향에 대한 자료를 준비할 예정입니다. 넷째, 공공자치법에 대한 입법 형식의 조정이 강조되고 있으며, 모든 공공자치법을 통합 포함하여 결사가 수립할 수 있어야 한다는 입장이 제시되고 있습니다. 표지하자면, 국회 입법 과정에서는 다양한 분야에서의 안전성, 규제개혁, 개인정보 보호, 공공자치법 수사 등 여러 생활 영역에 대한 논의와 계획이 진행되고 있습니다. 추가적인 정보나 특정 분야에 대한 질문이 있으시면 말씀해 주세요.

| What is up? ▾

```

class Agent:
    def __init__(self):
        self._tool_init()

    def _tool_init(self):
        self.custom_multiquery_chain = (
            ("question", Runnable("passthrough")),
            | multi_query_prompt,
            | model,
            | StringOutputParser()
        )
        self.multiquery_retriever = MultiQueryRetriever.from_llm(
            retriever=retriever,
            llm=custom_multiquery_chain,
        )
        self.bm25_retriever = BM25Retriever.from_text(
            [doc.page_content for doc in documents]
        )

        self.ensemble_retriever = EnsembleRetriever(
            retrievers=[bm25_retriever, multiquery_retriever],
            weights=[0.6, 0.4],
        )
        self.tool = create_retriever_tool(
            ensemble_retriever,
            "Internal_Knowledge_Retriever",
            "Retrieval and Augmented Generation for NBC+ guide",
        )
        self.prompts = ChatPromptTemplate.from_messages(
            [
                ("system", sys_prompt),
                MessagePlaceholder(variable_name="history"),
                ("human", "({input})"),
                MessagePlaceholder(variable_name="agent_scratchpad"),
            ]
        )
        self.agent = create_tool_calling_agent(llm=llm, tools=self.tool, prompt=prompt_template)
        self.executor = AgentExecutor(
            agent=agent,
            tool=agent,
            verbose=True,
            max_iterations=10
        )

```

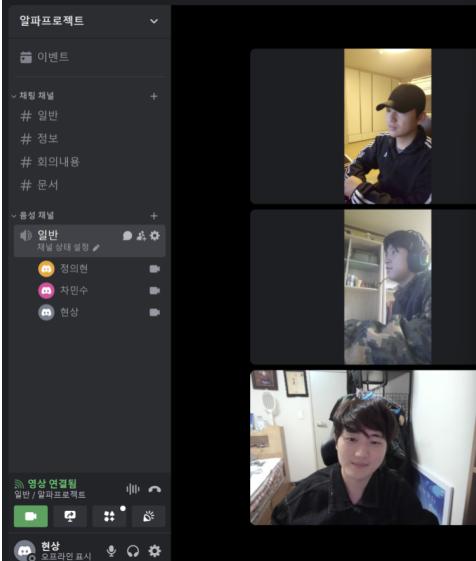
[공모전] 학습데이터 전처리 및 RAG 시스템 및 Streamlit 구현

텍스트 정보를 전부 모아서 하나의 파일로 정리하였습니다. 이후 이 텍스트 정보를 청크로 나누어서 임베딩 후 리트리버로 검색할 수 있게 하였습니다.

Rag 시스템을 구축하여
임베딩 된 데이터를
리트리버와 RAG에 통합
작업을 완료하여 데이터
검색을 통해 응답을
생성하도록 하였습니다.

Streamlit 통해
프로그램의 기본적인
인터페이스 구축에
성공하였습니다.

활동시간 9/30 18:00 ~ 20:00 10/1 13:00 ~ 18:00 10/2 18:00 ~ 20:00 10/3 13:00 ~ 18:00 10/4 13:00 ~ 17:00	활동장소 온라인 화상 미팅(디스코드) 미래관 무한상상실 미래관 자유주행스튜디오
 <p>Commits on Oct 6, 2024</p> <ul style="list-style-type: none"> Merge pull request #7 from hyeonsang010716/feat/create-rag Fix: __main__.py answer Update README.md version 0.1.0 	<p>[공모전] 임베딩에 사용되는 LLM 비용 문제 해결을 위한 코드 변경, [알파프로젝트] 뉴스데이터 전처리시 오픈 소스 사용 방안 고안</p> <p>유료 LLM이 아닌 오픈 소스를 이용하여 로컬에서 LLM을 사용할 수 있도록 llama를 통해 rag 시스템을 구축하는 방법을 모색하였습니다. 해당 방식을 통해 새롭게 임베딩 코드를 짜고 기존 streamlit 코드 문제를 해결하고 새롭게 만들어 버전 0.1.0을 완성하였습니다.</p> <p>오픈소스를 이용한 뉴스 데이터 전처리 방식은 수집 시간 및 GPU 사용량 문제로 성능의 문제가 있다는 것을 확인하여 사용하지 않기로 하였습니다.</p>

활동시간	10/8 13:00 ~ 18:00 10/9 13:00 ~ 18:00 10/10 13:00 ~ 18:00 10/11 13:00 ~ 16:00	활동장소	온라인 화상 미팅(디스코드) 미래관 무한상상실 미래관 자유주행스튜디오
	 <pre>from RAG.ensemble_retriever import Agent import torch import time from dotenv import load_dotenv load_dotenv() # lm 안스터스를 세션 상태에 저장하고 재사용 if "lm" not in st.session_state: agent = Agent() st.session_state.lm = agent.get_agent() # lm 안스터스 저장 lm = st.session_state.lm # 대답 방식 def response_generator(response): for word in response.split("\n"): print(word) yield word + "\n" time.sleep(0.05) # 세팅 서비스 이름 설정 st.title("Simple chat") # 세팅 기록 초기화 if "messages" not in st.session_state: st.session_state.messages = [] # 새로운 고침 시 아래에 있던 세팅 기록 지우기 for message in st.session_state.messages: with st.chat_message(message["role"]): st.markdown(message["content"]) st.session_state.messages = [] # 유저 입력 = 변수 prompt </pre> <p>Commits on Oct 13, 2024</p> <p>Merge pull request #9 from hyeonsang010716/feat/history</p> <p>FEAT: 프롬프트 엔지니어링</p> <p>Commits on Oct 10, 2024</p> <p>Merge pull request #8 from hyeonsang010716/feat/history</p> <p>FEAT: 대화 내용을 기억하는 history 기능 구현</p> <p>sys_prompt = ***</p> <p>You are an AI assistant designed to support parliamentary activities and ensure the public's right to information based on the minutes of the National Assembly. Your role is to answer questions using only the context provided in the retrieved document. The internal knowledge retriever provides multi-relevant contexts from the parliamentary minutes through a combination of multi-query retrieval and BM25-based searches. Your task is to use this information to answer user queries in detail, but strictly avoid providing any information that is not found in the retrieved context. If the provided context does not contain sufficient information to answer the question, you must clearly state that you do not know and ask the user for more specific details or clarify the question.. Under no circumstances should you hallucinate or guess.</p> <p>You must follow these rules:</p> <ul style="list-style-type: none"> - Do not use pre-trained data or external web searches for generating responses. - Only use the context provided by the Internal_Knowledge_Retriever, which retrieves documents using multi-query and BM25-based searches. - If no relevant information is found in the retrieved context, state that you do not know. - Both the provided context and the user's question will always be in Korean, and your answer must be in Korean. <p>Response style:</p> <ul style="list-style-type: none"> - Provide detailed and factual explanations based on the parliamentary minutes: Focus solely on the content provided by the Internal_Knowledge_Retriever. - Avoid speculation or adding information that is not present in the context. - Use clear and formal language: Answer in a professional tone, ensuring clarity when explaining legislative matters. For example, "해당 법안은 국회 본회의에서 논의되었으며, 그 주요 내용은 A입니다." - Summarize key points: After providing a detailed explanation, conclude with a summary that highlights the main points. For example, "요약하자면, 이 논의는 교육 예산에 관한 것입니다." - Handle complex questions step-by-step: For complicated questions, break down your explanation into clear steps. For example, "첫 번째로, 이 안건은 국회 법제사법위원회에서 검토되었습니다. 두 번째로, 본회의에서 투표되었습니다." - Offer follow-up opportunities: Encourage users to ask further questions if they need more details. For example, "혹시 더 자세히 물어보세요." - When information is unavailable: If you cannot find the enough information to answer the question, state without referencing the retrieved context that you do not know and ask the user for more specific information. For example, "죄송하지만, 해당 질문에 대한 내용을 찾지 못하였습니다. 혹시 더 구체적인 상황이나 세부 내용을 알려주시면 추가적인 정보를 제공하도록 노력하겠습니다." - Use clear, concise language with proper sentence spacing and line breaks to improve readability. - Separate different ideas or key points by using line breaks between sentences. - Avoid long blocks of text by keeping sentences short and well-spaced. - Keep the overall tone formal and professional. <p>Limitations:</p> <ul style="list-style-type: none"> - Answer only based on the retrieved context from the Internal_Knowledge_Retriever. - If the context does not provide enough information, state that you do not know. - Answer only in Korean. <p>## Few-shot examples:</p> <p>## Example 1: Context: 신도시 지역의 평생학습교육에 대한 논의가 있으며, 배령문화지역과 청운문화지역 간의 학업 성취도 차이가 낮다는 내용이 있습니다. Question: 신도시 평생학습 교육, 청운문화지역 간의 학업 성취도 차이가 있는지 여부</p>	<p>[공모전] 응답 성능 향상을 위한 history 구현 및 프롬프트 엔지니어링</p> <p>프로그램의 성능향상을 위해 과거의 질문과 응답 정보를 기억하여 답변의 성능을 높여주는 history 기능을 구현하였고 langsmith를 통해 history가 정상 작동하는 것 까지 확인하였습니다.</p> <p>LLM의 성능을 높여주기 위해 시스템 프롬프트, 멀티쿼리 프롬프트를 프롬프트 엔지니어링을 진행하였고 모델이 답변을 할 때, 환각을 줄이고 특정 상황에서 일정한 답변을 할 수 있도록 규칙, 제한, 응답방식등을 설정하였으며 실제로 응답의 변화를 이끌어내는데 성공하였습니다.</p>	

활동시간 10/14 18:00 ~ 20:00 10/15 13:00 ~ 18:00 10/16 18:00 ~ 20:00 10/17 13:00 ~ 18:00 10/18 13:00 ~ 17:00	활동장소 온라인 화상 미팅(디스코드) 미래관 무한상상실 미래관 자유주행스튜디오	<p>[공모전] 임베딩 성능 향상 방법 고안 및 임베딩 모델 평가 지표 작성, 프로그램의 버그 및 최적화</p> <p>임베딩 모델 성능 향상을 위하여 다른 임베딩 모델을 사용해도록 하고 가 임베딩 모델 별로 우리의 프로그램의 성능을 얼마나 끌어 올려주는지 평가 지표를 작성하였습니다.</p> <p>해당 지표를 바탕으로 text-embedding-3-large 모델이 가장 성능이 잘 나오는 것으로 판단하여 임베딩 모델을 변경하고 그에 따라 VectorDB도 변경하였습니다.</p> <p>streamlit에서 출력이 불어서 나오는 버그 해결 및 코드의 전반적인 최적화를 진행하였습니다.</p>																														
<p>각 임베딩 모델 평가 및 지표 확인</p> <p>사용된 모듈은 ragas이며 faithfulness와 answer_relevancy를 사용하였습니다.</p> <p>faithfulness</p> <ul style="list-style-type: none"> 생성된 답변이 얼마나 사실(Context)에 근거한 정확한 답변인가요? answer와 context에서 계산됩니다. Faithfulness 측정값은 값은 (0.1) 범위로 스케일링 되며 높을수록 좋습니다. <p>answer_relevancy</p> <ul style="list-style-type: none"> 생성된 답변이 질문(공식문서: 주어진 프롬프트)에 얼마나 관련성이 있나요? 이 지표는 question, answer, context 사용하여 계산됩니다. 불완전하거나 중복된 정보를 포함하는 답변에는 낮은 점수가 부여되고, 높은 점수는 더 나은 관련성을 나타냅니다. <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th>임베딩 모델</th> <th>faithfulness</th> <th>answer_relevancy</th> </tr> </thead> <tbody> <tr> <td>text-embedding-3-small</td> <td>0.5871</td> <td>0.4452</td> </tr> <tr> <td>intfloat/multilingual-e5-large-instruct</td> <td>0.5865</td> <td>0.5619</td> </tr> <tr> <td>solar-embedding-1-large-passage</td> <td>0.5839</td> <td>0.3795</td> </tr> <tr> <td>nomic-embed-text</td> <td>0.6729</td> <td>0.431</td> </tr> <tr> <td>BAAI/bge-m3</td> <td>0.5863</td> <td>0.4152</td> </tr> <tr> <td>BM-K/KoSimCSE-roberta-multitask</td> <td>0.7569</td> <td>0.43</td> </tr> <tr> <td>intfloat/multilingual-e5-large</td> <td>0.6721</td> <td>0.2518</td> </tr> <tr> <td>snunlp/KR-SBERT-V40K-klueNLI-augSTS</td> <td>0.6737</td> <td>0.5728</td> </tr> <tr> <td>gpt4all Embedding</td> <td>0.6743</td> <td>0.3225</td> </tr> </tbody> </table>			임베딩 모델	faithfulness	answer_relevancy	text-embedding-3-small	0.5871	0.4452	intfloat/multilingual-e5-large-instruct	0.5865	0.5619	solar-embedding-1-large-passage	0.5839	0.3795	nomic-embed-text	0.6729	0.431	BAAI/bge-m3	0.5863	0.4152	BM-K/KoSimCSE-roberta-multitask	0.7569	0.43	intfloat/multilingual-e5-large	0.6721	0.2518	snunlp/KR-SBERT-V40K-klueNLI-augSTS	0.6737	0.5728	gpt4all Embedding	0.6743	0.3225
임베딩 모델	faithfulness	answer_relevancy																														
text-embedding-3-small	0.5871	0.4452																														
intfloat/multilingual-e5-large-instruct	0.5865	0.5619																														
solar-embedding-1-large-passage	0.5839	0.3795																														
nomic-embed-text	0.6729	0.431																														
BAAI/bge-m3	0.5863	0.4152																														
BM-K/KoSimCSE-roberta-multitask	0.7569	0.43																														
intfloat/multilingual-e5-large	0.6721	0.2518																														
snunlp/KR-SBERT-V40K-klueNLI-augSTS	0.6737	0.5728																														
gpt4all Embedding	0.6743	0.3225																														

Commits on Oct 20, 2024

```
vectorDB 변경 & front history FIX
hyeonsang010716 committed 3 days ago
177b0d4 ↗ ↘

Merge pull request #10 from hyeonsang010716/fix_ui
hyeonsang010716 authored 3 days ago
Verified dd27e9d ↗ ↘
```

Commits on Oct 16, 2024

```
FIX: 텍스트가 전부 불어서 나오는 현상 해결
UHB135 committed last week
ef49cb4 ↗ ↘
```

활동시간	10/21 18:00 ~ 20:00	10/22 13:00 ~ 18:00	활동장소	온라인 화상 미팅(디스코드)
	10/23 18:00 ~ 20:00	10/24 13:00 ~ 18:00		미래관 무한상상실
	10/25 13:00 ~ 17:00			미래관 자유주행스튜디오

[공모전] 공모전에 결과제출을 위해 참가 신청서 및 결과물과 문제 해결 방법에 대한 설명 문서 작성, 발표자료 ppt로 작성,

[알파프로젝트]
공모전에서 사용한 기술
알파프로젝트에서 사용할 방법 모색

결과물을 제출을 위해 개요, 활용 데이터, 모델 개발 방법, 실험 및 평가, 기대효과등이 담긴 문제 해결방법에 대한 설명 문서를 작성하고 발표자료로 ppt를 준비하였습니다.

결과물은 제출을 위해 개요, 활용 데이터, 모델 개발 방법, 실험 및 평가, 기대효과등이 담긴 문제 해결방법에 대한 설명 문서를 작성하고 발표자료로 ppt를 준비하였습니다.

결과물의 형태와 내용

- 첫 번째 서비스의 형태로, 사용자가 질문하면 이에 대해 적절한 답변을 제시하는 구조로, 질문-답변 쌍으로 구성

질문(사용자)	답변(챗봇)
국회 회의록 대상 질문1	질문1에 대한 첫번째 답변
국회 회의록 대상 질문2	질문2에 대한 첫번째 답변

결과 제출

- 문제해결 방법에 대한 설명 문서 : 개요, 활용 데이터, 모델 개발 방법, 실험 및 평가, 기대효과 등
- 모델 개발 결과(모델 실행 매뉴얼 포함)

<9주차>

활동시간	10/28 18:00 ~ 20:00 10/29 13:00 ~ 18:00 10/30 18:00 ~ 20:00 10/31 13:00 ~ 18:00 11/01 13:00 ~ 17:00	활동장소	온라인 화상 미팅(디스코드) 미래관 무한상상실 미래관 자유주행스튜디오
 <pre>agent.py from langserve import RemoteRunnable import asyncio async def main(): remote_runnable = RemoteRunnable("http://localhost:8000") chat_history = [] while True: human = input("Human (Q/q to quit): ") if human.lower() == "q": print("AI: Bye bye human") break ai = await remote_runnable.invoke({ "input": human, "chat_history": chat_history }) response = ai['output'] print(f"AI: {response}") if __name__ == "__main__": agent.main()</pre> <p>Langserve 구현</p> <p>Chain 구성하기</p> <ul style="list-style-type: none"> Chain Prompt <ul style="list-style-type: none"> ChatPromptTemplate는 System에 전달할 플랫폼, 그리고 유저의 입력을 담을 부분을 지정하게 됩니다. MessagePlaceholder는 메시지의 기록을 담게 되며 chatting 형식의 Chain을 구성하도록 도와줍니다. <pre>prompt = ChatPromptTemplate.from_messages([("system", "You are a helpful, smart, kind, and efficient AI assistant. You always ful ("user", "User Input"), "MessagesPlaceholder(variable_name="messages")</pre> <p>LLM</p> <ul style="list-style-type: none"> LLM은 자신이 사용하고 싶은 어떤 모델을 가져와도 됩니다. <pre>llm = ChatUpstage()</pre> <p>Output Parser는 langchain_core의 StrOutputParser를 사용합니다.</p> <pre>chain = prompt llm StrOutputParser()</pre> <p>로컬 서버 실행</p> <ul style="list-style-type: none"> main.py 구성 방법 <ul style="list-style-type: none"> langserve는 Fast api 기반으로 만들어졌습니다. prompt 형태로해서 사용할 수 있고 Chatting 형식으로도 사용할 수 있습니다. 기본적인 형태는 app으로 FastAPI 클래스를 흐릅니다. <pre>app = FastAPI()</pre> <ul style="list-style-type: none"> CORS를 통해 외부 HTTP 요청을 허용하도록 작성합니다. CORS는 외부의 cross-origin 요청을 확인 하도록 해줍니다. CORS는 사이트가 사용자의 공격에 대비할 수 있게 해주고 다른 사이트가 모방하는 것을 막습니다. <pre>1 langgraph==0.2.50 2 langchain==0.3.7 3 langchain-core==0.3.19 4 langchain-community==0.3.7 5 langchain-openai==0.2.8 6 langgraph_sdk==0.1.36 7 langgraph-checkpoint-sqlite==2.0.1 8 langchain_chroma==0.1.4 9 langchain-text-splitters==0.3.2 10 unstructured==0.16.5 11 uvicorn==0.32.0 12 streamlit==1.40.1 13 fastapi==0.114.0 14 Requests==2.32.3 15 python-dotenv==1.0.1 16 pydantic==2.9.2 17 rank-bm25==0.2.2 18 python-multipart==0.0.17 19 bs4 20 pytube 21 pytubefix 22 moviepy==1.0.3</pre>			

<10주차>

```

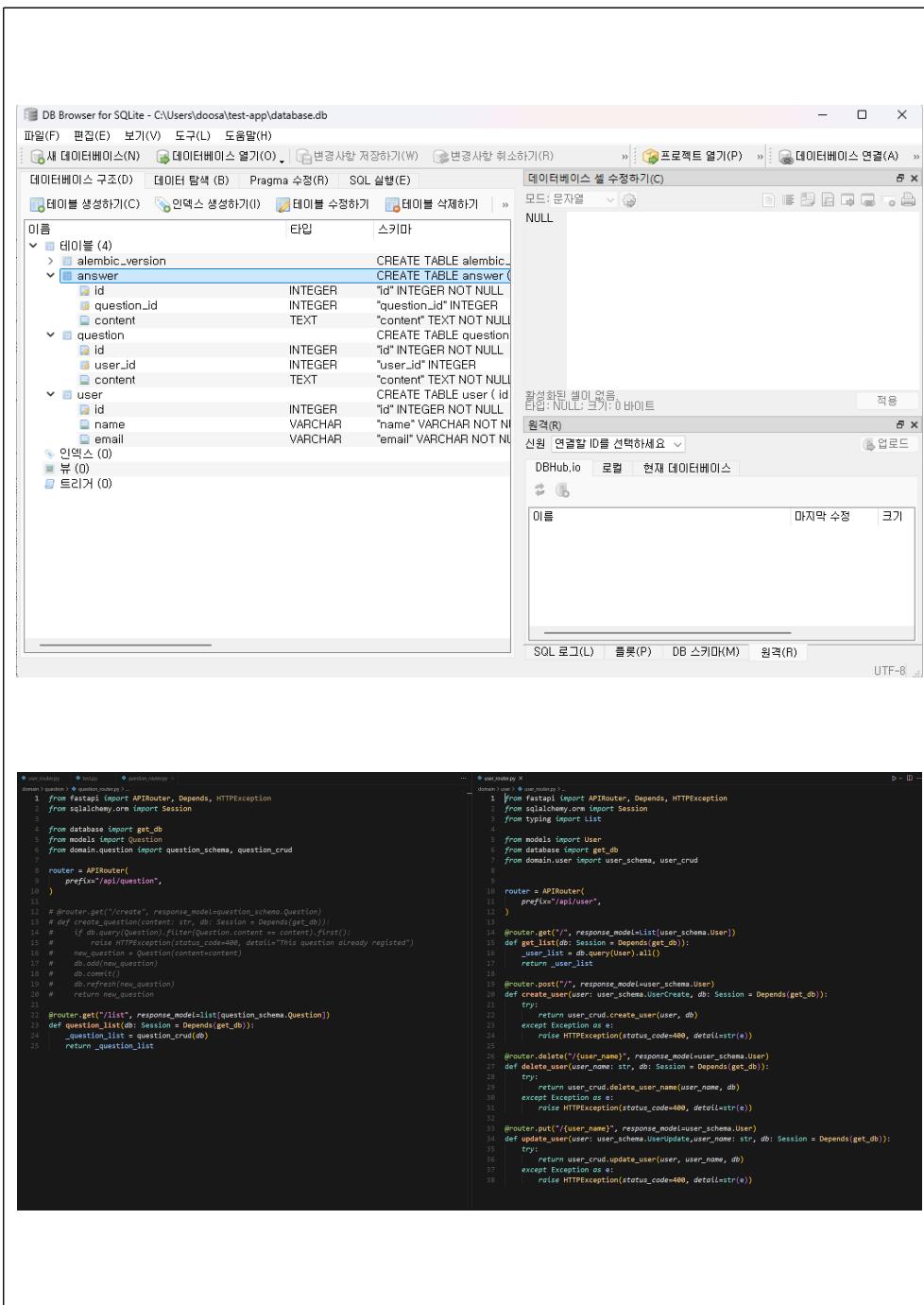
1 from typing import Literal, Optional, TypedDict
2 from langchain.graph import StateGraph, STATEID, END
3 from Pygments.display import Image, display
4
5 # 아래에서 템플릿을 사용하는 경우
6 # def extract_location(location: str) -> Optional[str]:
7 #     """ 위치에서 'in' 다음에 오는 단어를 위치로 추출 """
8 #     words = query.split()
9 #     if len(words) > 1:
10 #         index = words.index("in")
11 #         if index + 1 < len(words):
12 #             return words[index+1]
13 #     return None
14
15 def is_ambiguous(location: str) -> bool:
16     """ 위치가 2개 이상인 경우에 대한 여부를 판별 """
17     return len(location) > 2
18
19 def fetch_weather_data(location: str) -> str:
20     """ 각 위치에 맞는 날씨 데이터 """
21     weather_data = {
22         "New York": "Sunny, 25°C",
23         "London": "Rainy, 15°C",
24         "Paris": "Cloudy, 20°C",
25         "Paris": "Partly cloudy, 22°C"
26     }
27     return weather_data.get(location, "Weather data not available")
28
29 def generate_location_clarification(location: str) -> str:
30     """ 위치에 대해서만 예상되는 정보를 제공 """
31     return f"Would you please provide more details about the location '{location}'?"
32
33 def format_weather_response(location: str, forecast: str) -> str:
34     """ 위치와 예보를 결합 """
35     return f"The weather in {location} is {forecast}"
36
37 class WeatherState(TypedDict):
38     location: Optional[str]
39     forecast: Optional[str]
40
41 def parse_query(state: WeatherState):
42     location = extract_location(state["query"])
43     return {"location": location}
44
45 def get_forecast(state: WeatherState):
46     forecast = Fetch_weather_data[state["location"]]
47     return {"forecast": forecast}
48
49 def check_location(state: WeatherState) -> Literal["valid", "invalid", "ambiguous"]:
50     """ 위치가 유효한지 검증 """
51     if state["location"] == END:
52         return "invalid"
53     elif is_ambiguous(state["location"]):
54         return "ambiguous"
55     else:
56         return "valid"
57
58
59 graph = StateGraph(WeatherState)
60 graph.add_conditional_edges(
61     [{"location": "Paris", "forecast": "Cloudy", "clarification": "clarify_location"}],
62     check_location,
63     [
64         ("valid", "get_forecast"),
65         ("invalid", END),
66         ("ambiguous", "clarify_location"),
67     ],
68 )
69
70 graph.add_node("parse_query", parse_query)
71 graph.add_node("get_forecast", get_forecast)
72 graph.add_node("clarify_location", clarify_location)
73 graph.add_edge("parse_query", "get_forecast", generate_response)
74 graph.add_edge("STATEID", "parse_query", generate_response)
75 graph.add_edge("get_forecast", "generate_response", generate_response)
76 graph.add_edge("clarify_location", "generate_response", generate_response)
77 graph.add_edge("generate_response", END)
78
79 # 그래프를 만드는 앱
80 app = graph.compile()
81
82 graph_image = app.get_graph().draw_mermaid_png()
83 display(graph_image)

```

활동시간 11/04 18:00 ~ 20:00 11/05 13:00 ~ 18:00 11/06 18:00 ~ 20:00 11/07 13:00 ~ 18:00 11/08 13:00 ~ 17:00	활동장소 온라인 화상 미팅(디스코드) 미래관 무한상상실 미래관 자유주행스튜디오
<pre> from typing import Annotated, List from typing_extensions import TypeDict import operator from langchain_core.messages import AnyMessage, HumanMessage, AIMessage from langchain.graph import StateGraph, START, END class SubgraphState(TypeDict): """서브그래프의 상태를 정의하는 클래스""" messages: List[AnyMessage] = operator.add # 메시지 미스터리 context: str = '' # 컨텍스트 정보 def process_node(state: SubgraphState): """임의 메시지를 처리하는 노드""" context = state['messages'][-1] # 메시지 처리 로직 processed_result = f"Processed: {context.message.content}" return {'context': processed_result} def respond_node(state: SubgraphState): """응답을 생성하는 노드""" context = state['context'] # 응답 생성 로직 response = AIMessage(content=f"Response based on: {context}") return {'messages': response} @property def subgraph(self) -> StateGraph[SubgraphState]: """노드 추가""" subgraph = StateGraph(SubgraphState) subgraph.add_node("process", process_node) subgraph.add_node("respond", respond_node) # 초기 상태 설정 subgraph.add_edge(START, "process") # 시작 -> 처리 subgraph.add_edge("process", "respond") # 처리 -> 응답 subgraph.add_edge("respond", END) # 응답 -> 종료 # 그래프 컴파일 compiled_subgraph = subgraph.compile() from IPython.display import Image, display subgraph_image = compiled_subgraph.get_graph().draw_mermaid_png() display(Image(subgraph_image)) initial_state = { "messages": [HumanMessage(content="Hello!")], "context": "" } # 서브그래프 실행 result = compiled_subgraph.invoke(initial_state) # 결과 확인 for message in result["messages"]: print(f"Message: {message}") ... </pre> <p>LangGraph를 사용하면서 langchain-core를 0.3 버전을 사용해야 합니다. Langchain-core를 사용하면 pydantic v2 버전을 사용해야 합니다.</p>	<h3>1. Langgarph 학습 및 프로젝트의 적용 방안 제시</h3> <h3>2. Fast API의 router와 SQL Alchemy 사용법 학습</h3> <p>1. 기존의 Langchain 사용에는 여러 한계가 존재하여 이를 보완하기 위해 최신 기술인 LangGraph를 학습하였습니다. LangGraph의 주요 기능과 활용 사례를 분석하고 프로젝트에 효과적으로 적용할 수 있는 방안을 제시하였습니다. 이를 통해 가짜뉴스 판별 프로그램의 성능 향상과 시스템 통합 가능성을 검토했습니다.</p>

차이점

특징	PydanticOutputParser	with_structured_output
출력 생성 방식	LLM 출력 문자열을 받아서 Pydantic 모델로 파싱	LLM이 출력 자체를 Pydantic 모델 구조에 맞춰 생성
에러 발생 가능성	출력이 구조화되지 않으면 파싱 실패 가능	LLM이 정확히 형식에 맞춰 출력하지 않으면 에러 발생
유연성	출력이 Pydantic 형식과 완전히 일치하지 않아도 파싱 가능	LLM 출력이 Pydantic 형식과 반드시 일치해야 함



2. FastAPI의 주요 기능
학습에 집중하였으며,
특히 서버 호스팅과 API
Router 생성 방법을
중심으로
학습하였습니다.
데이터베이스 연동을
위해 SQLAlchemy
사용법도 학습하여
가짜뉴스 판별
프로그램의 백엔드 구조
설계에 반영할 준비를
마쳤습니다.

<11주차>

활동시간 11/11 18:00 ~ 20:00 11/12 13:00 ~ 18:00 11/13 18:00 ~ 20:00 11/14 13:00 ~ 18:00 11/15 13:00 ~ 17:00	활동장소 온라인 화상 미팅(디스코드) 미래관 무한상상실 미래관 자유주행스튜디오
<pre> graph LR START[START] --> YOUTUBE[유튜브 영상 내용] YOUTUBE --> LLM1[유튜브 영상 요약 LLM] LLM1 --> LLM2[영상 내용을 가지고 키워드 생성 LLM] LLM2 --> LLM3[키워드 가지고 네이버 뉴스 검색 (중복된 뉴스 제거)] LLM3 --> LLM4[유튜브 영상과 뉴스가 연관성이 있는지 판단 LLM] LLM4 --> LLM5[뉴스를 가지고 유튜브 영상이 가짜인지 판단 LLM] LLM5 --> END[END] </pre> <p>LangGraph 시나리오</p>	<p>1. 가짜정보 판별 프로그램 Langgraph 기초 설계 및 구현</p> <p>2. youtube 영상에서 텍스트 추출 및 전처리</p> <p>3. koyeb을 사용하여 실제 웹 사이트 구현</p> <p>1. 가짜뉴스 판별 프로그램의 LangGraph 기반 기초 설계 및 초기 구현을 진행하였습니다.</p> <p>가짜 뉴스 판별 프로그램 구현을 위한 Langgraph 시나리오를 작성하고 주요 기능 구조를 설계하고 데이터 흐름 및 알고리즘 동작 원리를 반영하여 LangGraph를 이용한 프로그램의 뼈대를 완성하였습니다.</p>
<pre> graph TD start__([__start__]) --> youtube_download[youtube_download] youtube_download --> summary_news[summary_news] summary_news --> generation_news_core_sentence[generation_news_core_sentence] generation_news_core_sentence --> search_Tavily[search_Tavily] search_Tavily --> fake_news_detection[fake_news_detection] fake_news_detection --> end__([__end__]) </pre> <p>Commits on Nov 15, 2024</p> <ul style="list-style-type: none"> FEAT: streamlit client 추가 UH3135 committed last month <p>Commits on Nov 14, 2024</p> <ul style="list-style-type: none"> FEAT: 형태소 분석기 추가한 retriever UH3135 committed last month FEAT: agent 관련 내용 추가 UH3135 committed last month <p>Merge pull request #17 from heonsang010716/minsu</p> <p>Commits on Nov 13, 2024</p> <ul style="list-style-type: none"> FEAT : pytube를 통한 유튜브 영상 다운로드와 whisper를 통한 텍스트 변환 Blue-Ladder committed last month <p>Commits on Nov 12, 2024</p> <ul style="list-style-type: none"> FEAT: 간단한 Agent ipynb 파일 추가 UH3135 committed last month Merge remote-tracking branch 'refs/remotes/origin/study_uihyun' into study_uihyun UH3135 committed last month FEAT: add agent UH3135 committed last month 	<p>1. 가짜뉴스 판별 프로그램의 LangGraph 기반 기초 설계 및 초기 구현을 진행하였습니다.</p> <p>가짜 뉴스 판별 프로그램 구현을 위한 Langgraph 시나리오를 작성하고 주요 기능 구조를 설계하고 데이터 흐름 및 알고리즘 동작 원리를 반영하여 LangGraph를 이용한 프로그램의 뼈대를 완성하였습니다.</p>
<p>Terminal Screenshot:</p> <pre> \$ ls whisper.ipynb audio.mp3 whisper.ipynb .env audio.mp3 audio.mp4 whisper.ipynb whisper.py whisper2.ipynb whisper2.py </pre> <p>Code Snippet 1 (Python):</p> <pre> #video_url = "https://www.youtube.com/watch?v=bN8XdhQulRVU" #audio_path = download_audio(video_url) #transcribed_text = transcribe_audio(audio_path) #transcribed_text = "이번엔 전세 물건 보증금을 약 300만 원을 소식입니다. 앞으로 전세대출이 더 까다로워질 것 같습니다. 가격부채 줄이라는 정책 때문에 아닙니다. 당시 온통 입장에서 생각해 볼까요. 주택담보대출은 원금抵押로 카드, 즉 드론 벌리주소, 대출금을 안 갚더라도 집을 저버려면 되니까 움직은 안심입니다. 그런 입장에서 시장가격에 따라 월세를 돌리려는 권리, 세입자의 그 권리가 일종의 당고인데 사실 저주인의 예언으로 그들이 어제 매우 불안한 달보입니다. 그래서 보완장치가 필요합니다. 주택도시기금공사, 주택금융공사, 서울보증보험, 보증회사들이 믿고 유통하는 전세보증금을 받고주는 겁니다. 위장아래 보증여 모재가 생기면 전세대출 전세대출의 40% 보증하는 주택도시기금공사가 가장 심각한데 대신 갚아주고 있습니다. 나중이라도 사기꾼들에게 받아야겠지만 사기꾼들이 순수년리가 있었습니다. 전세대출의 보증금은 2018년 500원이었지만 지난해 3억 5천원 원을 넘겼고 올해는 최소 4.2% 이 정도면 아무도 인상승가는 보증에 반대해 줄 수밖에 없습니다. 앞으로는 전세보증금 전액이 아닌 일부분 대체로 80% 이하로 보증금 청탁이 이루어져야 합니다. 이에 실행되면 집주인 신용 좋은 세입자도 전세대출" </pre> <p>Code Snippet 2 (Python):</p> <pre> # 키워드 추출 keywords = extract_keywords_with_llm(transcribed_text) print("추출된 키워드:", keywords) # 키워드 충실 사실 검증 keyword_verification = verify_with_llm.keywords(transcribed_text, keywords) print("키워드 충실 검증 결과:", keyword_verification + "\n") </pre> <p>... (Continues)</p>	<p>1. 가짜뉴스 판별 프로그램의 LangGraph 기반 기초 설계 및 초기 구현을 진행하였습니다.</p> <p>가짜 뉴스 판별 프로그램 구현을 위한 Langgraph 시나리오를 작성하고 주요 기능 구조를 설계하고 데이터 흐름 및 알고리즘 동작 원리를 반영하여 LangGraph를 이용한 프로그램의 뼈대를 완성하였습니다.</p>

2. YouTube 영상 다운로드 및 요약 기능을 구현하였습니다.

Whisper 모델을 활용하여 동영상의 오디오 내용을 텍스트로 변환하고, 해당 텍스트를 요약하는 기능을 설계하고 테스트하였습니다. 이를 통해 가짜뉴스 판별 시스템과 연계할 수 있는 멀티미디어 콘텐츠 분석 기능 개발의 기초를 마련하였습니다.

3. 웹 플랫폼 서비스 구현을 위한 다양한 방법을 탐색하였으며, 그 일환으로 Koyeb을 사용하여 프로그램을 실제 웹 사이트에 배포하는 실습과 구현을 수행하였습니다. 이를 통해 배포 프로세스를 이해하고, 서비스 운영의 가능성을 확인하였습니다. 무거운 모델을 올릴 경우 비용문제와 저장 공간 문제가 발생할 수 있기 때문에 공모전에서 사용한 Streamlit을 이용한 추가 배포 방법을 고안하였습니다.

[Home](#) > [progressive-fionna/test-app](#)

test-app Web service | progressive-fionna-uh3135-d5472958.koyeb.app/

[Overview](#) [Metrics](#) [Console](#) [Settings](#)

[View active deployment](#) [View latest deployment](#)

b0520c04 Healthy 27d ago

Triggered because service has been updated or redeployed

Overview

Public URL: progressive-fionna-uh3135-d5472958.koyeb.app/ (forwarded to port 8000)

Repository	Branch	Commit	Builder type	Auto deploy	Privileged
UH3135/test-app	main	Buildpack	True	False	

Instance type	Scaling	Regions	Environment variables	Volumes
Free	1 instance	Washington, D.C.	2 variables	0 volumes attached

[View more](#)

Koyeb 배포

배포 준비

- GitHub 업로드
 - 현재까지의 코드들을 GitHub에 업로드 합니다.
- Koyeb 세팅
 - <https://www.koyeb.com/> 회원가입을 합니다.
 - Overview로 가서 Web Service를 생성합니다.
- 프로젝트 구성
 - Setting에 Builder/Run command에 아래 명령어를 입력합니다

```
uvicorn main:app --host 0.0.0.0 --port 8000
```

 - Exposed ports의 포트 번호 설정이 명령어 포트와 동일한지 확인합니다.
 - Environments variables에 LLM API Key를 추가합니다.

주의사항

- API 키 관리
 - api key는 악용될 가능성이 높기 때문에 GitHub에 올리지 말고 로컬에서 관리합니다.
 - Koyeb에서도 Secret으로 설정해서 악용되지 않게 사용합니다.
- 패키지 호환성
 - pywin32와 같이 특정 운영체제에서만 사용되는 모듈이 requirements.txt에 포함되지 않게 관리합니다.
 - pydantic.v1처럼 각 모듈의 버전을 확인하고 충돌이 나지 않게 관리 합니다.
- 이미지 크기 최적화
 - Koyeb은 자동으로 github 코드를 Docker img로 바꾸어 줍니다. 이때 img의 크기가 너무 크면 호스팅이 안되므로 이를 확인해 주어야 합니다.

<12주차>

활동시간	11/18 18:00 ~ 20:00 11/20 18:00 ~ 20:00 11/22 13:00 ~ 17:00	11/19 13:00 ~ 18:00 11/21 13:00 ~ 18:00	활동장소	온라인 화상 미팅(디스코드) 미래관 무한상상실 미래관 자유주행스튜디오
------	---	--	------	--



1. NLP를 사용해서 문장 요약
 2. Youtube API 사용방법 학습
 3. 뉴스 검색 함수 구현 및 뉴스 검색 tool Lang graph에 연결

1. NLP 기술을 활용하여 문장을 요약기능 강화를 위해 학습 및 코드 구현을 진행하였고 이를 이용하여 유튜브 영상의 음성 데이터에서 가져온 텍스트 데이터의 전처리, 문장 분리, 핵심 문장 추출 등에 사용 방법을 찾아보았습니다.

```
agentipy_M agentipy_client_M mappyx om summary_llm.py summary_skpt.py
youtube_summary > summary_llm.py ...
1 from langchain.tools import tool
2 from langchain_core.prompts import ChatPromptTemplate
3 from langchain_core.output_parsers import StrOutputParser
4 from langchain_upstage import ChatUpstage
5 from typing import List, Dict, Annotated
6 from dotenv import load_dotenv
7
8 from get_news import get_news_context
9
10 print(f"🦙 헤이惆타스 분석기 요청: {summary_llm.summary}")
11 print(f"🦙 LLM 요청: {summary_llm.search_news_keyword}")
12
13 llm = ChatUpstage()
14
15 prompt = ChatPromptTemplate.from_messages([
16     ("system", "You are a helpful assistant specialized in"),
17     ("system", "Please provide a concise summary of the given"),
18     ("system", "Make sure to include the main points and key"),
19     ("system", "단행은 한국어로 해줘"),
20     ("human", "다음 뉴스 기사를 요약해줘:\n\n(context"),
21     ("human", "1)"),
22
23     chain = prompt | llm | StrOutputParser()
24
25 def search_news_keyword(keyword: str):
26     text = get_news_context(keyword)
27     return chain.invoke({"context": text})
28
29 if __name__ == "__main__":
30     print(summary_llm.summary())
31
32 youtube_summary > summary_skpt.py ...
1 from konpy.tag import Okt
2 from collections import Counter
3 import numpy as np
4 from get_news import get_news_context
5
6 def summarize_text_korean(text):
7     # 필요한 모듈 가져오기
8     okt = Okt()
9
10     nouns = okt.nouns(text)
11
12     from collections import Counter
13     word_freq = Counter(nouns)
14
15     sentences = text.split('.')
16     sentence_scores = []
17     for sentence in sentences:
18         score = sum(word_freq[noun] for noun in okt.nouns(sentence))
19         sentence_scores.append((sentence, score))
20
21     top_sentences = sorted(sentence_scores, key=lambda x: x[1], reverse=True)
22     summary = '\n'.join([sentence for sentence, _ in top_sentences])
23     return summary
24
25 def summary_news_with_keyword(keyword: str):
26     text = get_news_context(keyword)
27     summary = summarize_text_korean(text)
28     return summary
29
30 if __name__ == "__main__":
31     print(summary_news_with_keyword("폭설"))
```

서울 한역에 내린 큰 눈에 시민들은 출근길에 불편을 겪기도 했습니다. 한 시민은 지역주민과 사용들이 특별하게 들어온 사연을 KBS에 제보했습니다. 제보자는 “얼굴 긁고 9호선에서 갈려 끌어 놓은 차운데 암사고자 날 뻗쳤다”면서 “회사는 제 계약을 저지른다”라고 말했습니다.

버스도 눈이 많이 쌓여 도로를 닦았습니다.

동이 티 아워, 시민들이 차운데 번역해 청원도장을 조심스럽게 건くなりました. 뛰어난 여성들은 도로에 친환경 냄새를 만드는 속도를 늦춰줍니다.

3. 사회·경관

3. 실험 결과

제작자: 김민경
제작일: 2024-01-15
제작장소: 서울특별시 강남구 테헤란로 123
제작설명: 본 문서는 경기장 규모 및 관중 수에 대한 조사 결과입니다.
주출판 문장:

- 경기장은 앞으로 내일까지 강원 신지에 최대 30cm 이상, 경기 내륙과 강원 내륙, 경북과 전북 등부위는 최대 15에서 20cm의 많은 눈이 더 내리겠고 내다닙니다.
- 서울교통공사는 오늘(27일) 오전 폭설에 1~8호선 '러시아워(열차 집중 투입 시간대)' 운행을 9시 30분까지 깊은 연장했습니다.

LLM을 통한 요약 결과

- 생성된 요약:

서울에 큰 눈이 내리 도로가 눈으로 뒤덮였고, 이로 인해 시민들이 출근길에 불편을 겪었습니다. 한 시민은 905번 지역에서 여사를 페스티벌에 빠져들어 사전을 서신을 제보하였습니다. 사내가 사고를 막았다고 전했습니다. 비단들은 눈이 쌓인 차도를 뚫고, 사람들은 길로 변한 횡단보도를 조심스럽게 건넙니다. 대낮주의와가 발령되었으며, 서울교통공사는 오늘 오전 출동 1~8호선 「리사야운」 운행을 9시 30분까지 연장했습니다. 앞으로 더 많은 눈이 예상되며, 시민들은 차량을 기울여야 합니다.

4. 분석 및 비교

1. 형태소 분석기의 요약 특징:

- 핵심적인 문장을 추출하였기 때문에 본문에 더 충실한 요약이었습니다.
- 그러나 원문 내용에 치중되어 있어서 전체내용을 압축하는 요약이 불가능했습니다.

3. NUM91-091

2. “LUM의 효과 특징”
- 음악을 고려하여 저연스럽고 유팽한 음장을 선보히겠습니다.

- 본책을 고려하여 자연스럽고 유창한 문장을 생성하였습니다

- 형태소 분석기 방식은 빠르고 단순한 요약에 적합하다는 것을 알 수 있습니다.

- 110 -

- LLM은 간단한 프롬프트 작성만으로 고급질의 묘역이 가능합니다. 특히 이는 BERT 사용과는 달리 짧은 프롬프트만으로 가능하다는 점이 큰 장점입니다.
- 또한 형태소 분석기로 요약, 추출한 문장을 LLM에게 전달함으로써 요약의 성능을 향상시킬 수 있습니다.

올릴 수 있습니다.

2. Youtube 영상의
음성 데이터를 텍스트로
가져올 수 있는 더
효율적인 방식을
구현하기 위해 Youtube
API를 사용방법에 대한
학습을 진행하였습니다.
youtube api도 좋지만
whisper의 성능을 넘기
힘들다는 결론을 얻을
수 있었습니다.

add_conditional_edge를 사용할 때 next 노드를 항상 명시 해줘야합니다!

```
def check_class_cancellation(self, state: operation_state) -> Literal["Node_A", "Node_B"]:
    print("---- this node is check_class_cancellation ----")

    prompt = PromptTemplate(
        template=prompts.check_class_cancellation_prompt,
        input_variables=["question"],
    )

    chain = prompt | self.model.with_structured_output(ClassCancellationCheck)
    response = chain.invoke({"question": state["question"]})

    if response.relevance_to_class_cancellation == "yes":
        return "Node_A"
    else:
        return "Node_B"
```

```
import time
import random
import requests
from bs4 import BeautifulSoup
from fake_useragent import UserAgent

def fetch_news_content(keyword):
    if keyword == "None": # 네이버 뉴스 검색 (리스트화 - 표 편성)
        query = "날씨"
        search_url = "https://search/naver.com/search/naver?where=news&ie=utf8&sm=ne_wrti&query=%query"
    else: # 검색 키워드
        query = keyword
        search_url = "https://search/naver.com/search/naver?where=news&ie=utf8&sm=ne_wrti&query=%query"

    # HTTP 헤더 설정
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.557.36"
    }

    # 검색 결과 요청
    response = requests.get(search_url, headers=headers)
    if response.status_code != 200:
        print(f"Error: {response.status_code}")
        return []

    # 결과 HTML 문서 추출
    soup = BeautifulSoup(response.text, "html.parser")
    link_tags = soup.select('a[href]')[:10] # 링크 태그 10개 추출
    links = [tag['href'] for tag in link_tags] # 링크 가져오기
    news_contents = [] # 뉴스 내용 초기화

    selected_links = links[1]
    news_contents = []

    for news_url in selected_links:
        # 뉴스 페이지 요청
        news_response = requests.get(news_url, headers=headers)
        if news_response.status_code != 200:
            print(f"Error: {news_response.status_code}")
            continue

        news_contents.append(f"뉴스 제목(상태): {news_response.status_code}\n")
        news_contents.append(f"\n{news_response.text}\n")

        article_tag = news_response.find('article', {'id': 'article_area'})
        if article_tag:
            content = article_tag.get_text(separator=' ')
            news_contents.append(f"Content: {content}\n")
            news_contents.append(f"\n(자사 본문을 찾을 수 없습니다.)\n")

        news_contents.append(f"\n(한국 기사가 끝나니.)\n")

    return news_contents[0]

if __name__ == "__main__":
    keywords = ["전세대출", "부동산", "보증금", "전세 계약"]
    news_results = fetch_news_content(keywords)
    print(news_results)
    tasks = []
    for keyword in keywords:
        tasks.append(fetch_news_content(keyword))
    print(tasks)

# 전세대출, 부동산, 보증금, 전세 계약
```

```
def search_Tavily(self, state: State) -> State:
    print("---- this node is search_Tavily ----")

    tavily_tool = TavilySearch()

    tasks = []
    for keyword in state["keywords"]:
        search_result = tavily_tool.search(
            query=keyword, # 검색 키워드
            topic="news", # 뉴스 주제
            days=100, # 최대 검색 결과
            max_results=1, # 최대 결과 개수
            format_output=True, # 결과 포맷팅
        )
        tasks.append("\n".join(search_result))

    #print(tasks)

    outputs = "\n".join(tasks)

    # return ("naver_news": outputs)
    return State(naver_news=outputs)

def fake_news_detection(self, state: State) -> State:
    print("---- this node is generation_question ----")

    prompt = PromptTemplate(
        template=prompts.fake_news_detection_prompt,
        input_variables=[ "content", "news" ],
    )

    chain = prompt | self.model | StrOutputParser()
    output = chain.invoke({"content": state["youtube_content"], "news": state["naver_news"]})

    # return ("response", output)
    return State(response=output)

def __graph_init__(self):
    graph_builder = StateGraph(state)
    graph_builder.add_node("youtube_download", self.youtube_download)
    graph_builder.add_node("generation_question", self.generation_question)
    graph_builder.add_node("search_naver_news", self.search_naver_news)
    graph_builder.add_node("search_Tavily", self.search_Tavily)
    graph_builder.add_node("fake_news_detection", self.Fake_news_detection)

    graph_builder.add_edge("youtube_download", "generation_question")
    graph_builder.add_edge("generation_question", "search_naver_news")
    graph_builder.add_edge("search_naver_news", "fake_news_detection")
    graph_builder.add_edge("search_Tavily", "fake_news_detection")
    graph_builder.addEdge("fake_news_detection", END)

    graph_builder.set_entry_point("youtube_download")
    graph = graph_builder.compile()

    return graph
```

```
.class Agent:
    def __init__(self):
        load_dotenv()
        self.model = ChatOpenAI(model="gpt-4o", temperature=0)
        self.graph = self.__graph_init__()

    def youtube_download(self, state: State) -> State:
        print("---- this node is youtube_download ----")

        audio = download_audio(state["youtube_link"])

        text = transcribe_audio(audio)

        return State(youtube_content=text)

    def generation_question(self, state: State) -> State:
        print("---- this node is generation_question ----")

        prompt = PromptTemplate(
            template=prompts.generation_question_prompt,
            input_variables= [ "content" ],
        )

        chain = prompt | self.model.with_structured_output(Sentence)

        output = chain.invoke({ "content": state["youtube_content"] })

        print(output.text)

        #return ("keywords": output.text)
        return State(keywords=output.text)

    def search_naver_news(self, state: State) -> State:
        print("---- this node is search_naver_news ----")

        tasks = [fetch_news_content(keyword) for keyword in state["keywords"]]
        print(tasks)

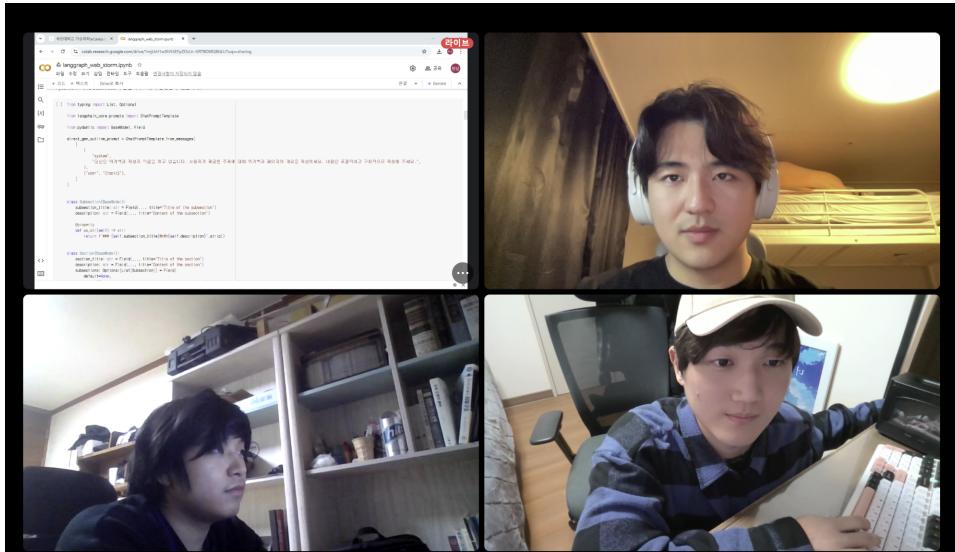
        outputs = "\n".join(tasks)

        # return ("naver_news": outputs)
        return State(naver_news=outputs)
```

3. 키워드를 가지고
네이버 뉴스를 검색하는
함수를 직접 구현하여
사용자가 제공한 유튜브
영상의 관련 네이버
뉴스를 수집할 수 있게
하였습니다. 이를 tavyly
API tool과 함께
LangGraph 뉴스 검색
도구에 통합했습니다.
이를 통해 프로그램은
빠르게 뉴스 데이터를
가져올 수 있고 가져온
뉴스 데이터를 가지고
가짜 정보에 대한 진위
여부 판별을 할 수 있게
되었습니다.

<13주차>

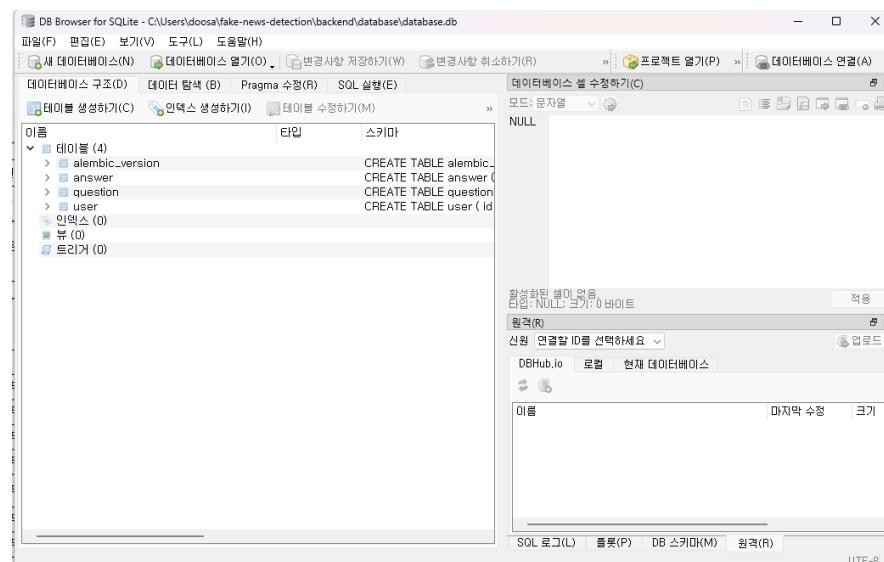
활동시간 11/25 18:00 ~ 20:00 11/26 13:00 ~ 18:00 11/27 18:00 ~ 20:00 11/28 13:00 ~ 18:00 11/29 13:00 ~ 17:00	활동장소 온라인 화상 미팅(디스코드) 미래관 무한상상실 미래관 자유주행스튜디오
--	---



```

14 fake_news_detection_prompt = """당신은 "가짜뉴스 탐지 AI" 역할을 수행합니다.
15 다음의 두 가지 정보가 주어집니다:
16
17 사용자가 제공한 글(기사 또는 주장)
18 해당 글과 관련해 외부 뉴스 소스(검색 결과)에서 얻은 뉴스 내용
19 당신의 임무는 다음과 같습니다:
20
21 주어진 글(“글: {query}”)에 제기된 정보나 주장과 “뉴스 내용: {context}”을 비교, 검증하십시오.
22 뉴스 내용에서 확인할 수 있는 사실과 일치하는지, 혹은 명백히 반박되는지 분석하십시오.
23 글이 가짜뉴스인지 아닌지를 판단하십시오.
24 만약 가짜뉴스라면, 어떤 부분이 잘못되었는지, 왜 그것이 잘못되었는지 구체적으로 설명하십시오.
25 분석 시 유의사항:
26
27 뉴스 내용은 관련 정보의 사실적 근거로 사용하십시오.
28 만약 뉴스 내용에서 글이 주장하는 핵심 정보와 정면으로 배치되는 사실이 확인된다면 가짜뉴스로 판단하십시오.
29 정보가 불명확하거나 뉴스 내용과 글의 정보가 정확히 일치하지 않더라도, 명백한 반증이 없다면 쉽게 가짜뉴스로 단정하지 마십시오. 대신 추가적으로 신중히 검토하십시오.
30 가짜뉴스로 판단할 경우, 해당 주장 중 어느 부분이 실제 뉴스 내용과 상충하는지, 그리고 그 차이가 무엇인지 명확히 제시하십시오.
31 최종 출력 형식 예:
32
33 결과 판정: “진짜 뉴스” 또는 “가짜 뉴스”
34 분석 근거: 뉴스 내용과 글을 비교하여 어떤 대목에서 불일치 또는 오류가 있는지 구체적으로 제시하십시오."""

```



1. Langgraph 성능개선을 위한 프롬프트 엔지니어링

2. 데이터 베이스 생성

3. 백엔드 서버 구축

1.LangGraph 기반 시스템의 성능을 개선하기 위해 프롬프트 엔지니어링을 수행하였습니다. 다양한 입력 프롬프트 구조를 실험하여 응답 정확성과 처리 속도를 최적화하였으며, 프로그램의 신뢰성과 유연성을 강화하였습니다. 이를 통해 가짜뉴스 판별 결과의 정확도를 향상시켰습니다.

```
backed 3� routers.py 刹游 > @answer question
16     @router.get("/", response_model=List[question.Question])
17     def get_all_question(db: Session = Depends(get_db)):
18         _questions_lists = db.query(Question).all()
19         return [question for question in _questions_lists]
20
21     @router.post("/", response_model=question.Question)
22     def answer_question(question: question.QuestionCreate, user: user.UserCreate, db: Session = Depends(get_db)):
23         _existing_question = db.query(Question).filter(Question.content == question.content).first()
24         if _existing_question is not None:
25             return _existing_question
26
27         try:
28             curr_user = db.query(User).filter(User.name == user.name).first()
29             new_question = Question(content=question.content, user=curr_user)
30
31             # agent 단백 코드
32             # agent = Agent()
33             # graph = agent.graph
34
35             # model_answer = graph.invoke("youtube_content": question.content)
36             # if not model_answer:
37             #     model_answer = "죄송합니다. 질문에 대한 답변이 어렵습니다."
38             model_answer = answer_answer
39             new_answer = Answer(content=model_answer, question=new_question)
40
41             db.add(new_question)
42             db.add(new_answer)
43             db.commit()
44             db.refresh(new_question)
45             db.refresh(new_answer)
46
47             return new_question
48         except Exception as e:
49             raise HTTPException(status_code=400, detail=str(e))
50
51
52
53
54
55
56
57
58
59
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
487
488
489
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
517
518
519
519
520
521
522
523
524
525
526
527
527
528
529
529
530
531
532
533
534
535
535
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1620
1621
1621
1622
1622
1623
1623
1624
1624
1625
1625
1626
1626
1627
1627
1628
1628
1629
162
```

FastAPI 0.1.0 OAS 3.1

/openapi.json

default

^

GET /api/chat/ Get All Question

POST /api/chat/ Answer Question

GET /api/login/{username} Get User By Username

POST /api/login/ Create User

^

Schemas

^

AnswerBase > Expand all object

Body_answer_question_api_chat__post > Expand all object

HTTPValidationError > Expand all object

Question > Expand all object

QuestionBase > Expand all object

QuestionCreate > Expand all object

User > Expand all object

UserCreate > Expand all object

ValidationError > Expand all object

<14주차>

활동시간	12/02 18:00 ~ 20:00 12/04 18:00 ~ 20:00 12/06 13:00 ~ 17:00	활동장소	온라인 화상 미팅(디스코드) 미래관 무한상상실 미래관 자유주행스튜디오
------	---	------	--

AzureChatOpenAI

Run Feedback Metadata

00:01:56,620 -> 00:02:02,120 현재 진행 중인 광고 속 이미지 문제, 특히 뉴진스라는 이름은 물론 모든 음악

21 00:02:02,860 -> 00:02:07,260 저작권 문제 등 뉴진스가 새로운 활동을 독자적으로 분석 진행하는 그 순간부터

22 00:02:07,500 -> 00:02:11,139 성질 다양한 법적 다툼을 피해가는 쉽지 않을 것으로 보입니다.

23 00:02:11,160 -> 00:02:12,160 SBS 심호흡입니다.

24 00:02:13,899 -> 00:02:16,399 --I(앵커, 환호 소리입니다.)

Rendered Output

AI

그룹 뉴진스가 소속사 어도어와의 전속 계약을 해지하고 오늘부터 자유롭게 활동을 이어나가겠다고 밝혔습니다. 어도어는 전속 계약이 2029년까지 유효하다고 주장하며 강경한 입장은 보이고 있습니다. 뉴진스는 법적 절차 없이 기존 스케줄을 예정대로 진행할 계획이지만, 계약 해지 과정에서 법적 다툼이 예상됩니다.

AzureChatOpenAI

Run Feedback Metadata

contracts with Ador Thursday, citing breaches of contract by the agency. "The agency could file an injunction to prohibit NewJeans from engaging in entertainment activities with another company, or pursue claims for penalties and damages for breach of contract," according to Bae, Friday. "The legality of the termination (by NewJeans) due to

Rendered Output

AI

결과 판정: "진짜 뉴스"

분석 근거:

사용자가 제공한 글과 외부 뉴스 소스에서 얻은 뉴스 내용을 비교한 결과, 두 정보는 대체로 일치합니다. 다음은 그 근거입니다:

1. **계약 해지 이유**:
 - 글: "그룹 뉴진스가 소속사 어도어와의 전속 계약을 해지하고 오늘부터 자유롭게 활동을 이어나가겠다고 밝혔습니다."

- 뉴스 내용: 미국 뉴스 소스에서 뉴진스가 어도어와의 전속 계약을 해지했다고 밝힌 사실이 확인됩니다. 예를 들어, The Korea Herald와 코리아타임스는 뉴진스가 계약 해지를 발표했으며, 어도어와의 법적 분쟁이 예상된다고 보도했습니다.

2. **계약 해지 이유**:
 - 글: "구체적인 이유는 언급되지 않았지만, 뉴스 내용에서는 뉴진스가 어도어의 계약 위반을 이유로 계약을 해지했다고 밝히고 있습니다. 이는 글의 내용과 일치합니다."

3. **법적 대응**:
 - 뉴스 내용에서는 어도어가 뉴진스의 계약 해지에 대해 법적 대응을 준비하고 있다고 보도하

1. 가짜 뉴스 판별하는
프로젝트 기능 구현
완성 및 테스트

2. streamlit을 이용한
프론트엔드 구현과 연결

3. 로그인 기능 완성

1. 가짜뉴스 판별
프로젝트의 주요 기능이
모두 구현되었으며,
초기 테스트가
완료되었습니다.
유튜브 영상 데이터
요약 및 진위여부
판별에 대해서 모델의
동작을 langsmith를
통해 잘 작동하는 것을
확인하였습니다.

```

183     def search_Tavily(self, state: State) -> State:
184         print(".... this node is search_Tavily ....")
185
186         tavily_tool = TavilySearch()
187
188         search_results = []
189
190         for keyword in state["Keywords"]:
191
192             result = tavily_tool.search(
193                 query=keyword, # 검색 키워드
194                 topic="news", # 뉴스 주제
195                 days=100,
196                 max_results=5, # 최대 검색 결과
197                 format_output=True, # 결과 포맷
198             )
199
200             search_results.append("\n".join(result))
201
202         return State(search_results=search_results)
203
204     def fake_news_detection(self, state: State) -> State:
205         print(".... this node is generation_question ....")
206
207         prompt = PromptTemplate(
208             template=prompts.Fake_news_detection_prompt,
209             input_variables=["context", "query"],
210         )
211
212         chain = prompt | self.model | StrOutputParser()
213
214         works = [
215             {"context": context, "query": query}
216             for context, query in zip(state["search_results"], state["Keywords"])
217         ]
218
219         results = chain.batch(works)
220
221         return State(response=results)
222
223     def __graph__(self):
224
225         graph_builder = StateGraph(State)
226         graph_builder.add_node("youtube_download", self.youtube_download)
227         graph_builder.add_node("summary_news", self.summary_news)
228         graph_builder.add_node("generation_news_core_sentence", self.generation_news_core_sentence)
229         graph_builder.add_node("search_naver_news", self.search_naver_news)
230         graph_builder.add_node("search_Tavily", self.search_Tavily)
231         graph_builder.add_node("fake_news_detection", self.fake_news_detection)
232
233         graph_builder.add_edge(START, "youtube_download")
234         graph_builder.add_edge("youtube_download", "summary_news")
235         graph_builder.add_edge("summary_news", "generation_news_core_sentence")
236         graph_builder.add_edge("generation_news_core_sentence", "search_Tavily")
237         graph_builder.add_edge("search_Tavily", "Fake_news_detection")
238         graph_builder.add_edge("fake_news_detection", END)
239
240         graph = graph_builder.compile()
241
242         return graph

```

↳ Commits on Dec 2, 2024

Merge pull request #10 from hyeonsang010716/feat/graph

hyeonsang010716 authored last week

Merge branch 'main' into feat/graph

hyeonsang010716 authored last week

↳ Commits on Dec 1, 2024

FEAT : graph update

Blue-Ladder committed last week

Merge pull request #9 from hyeonsang010716/feat/api

hyeonsang010716 authored last week

FEAT: login 기능추가

UH3135 committed 2 weeks ago

FEAT: 유저 정보 전달

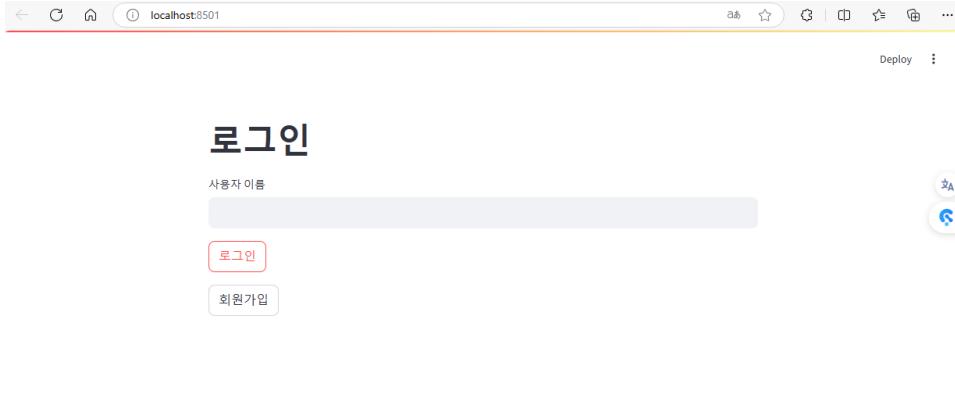
UH3135 committed 2 weeks ago

Merge branch 'feat/api' of https://github.com/hyeonsang010716/fake-news-detection into feat/api

UH3135 committed 2 weeks ago

FEAT: user login 기능 추가

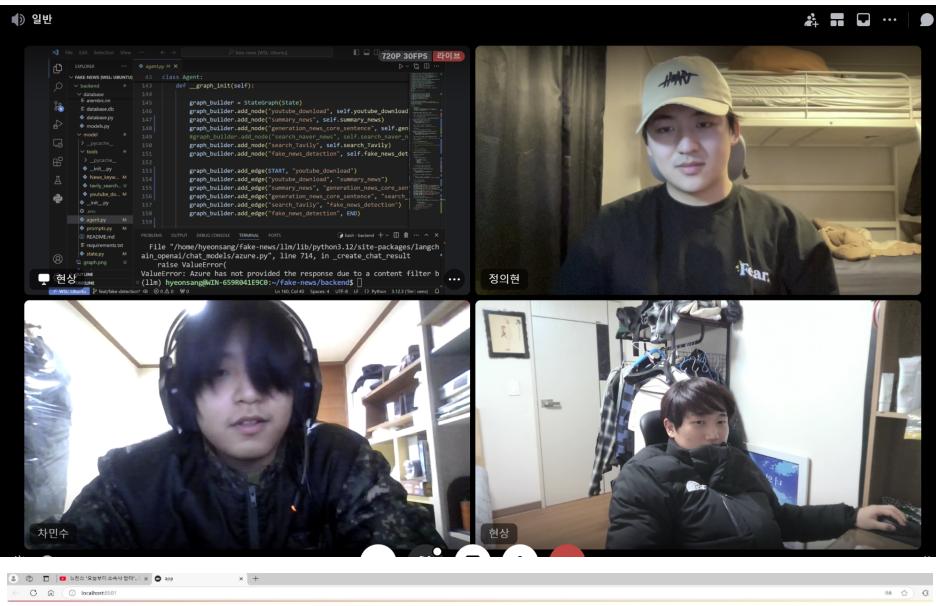
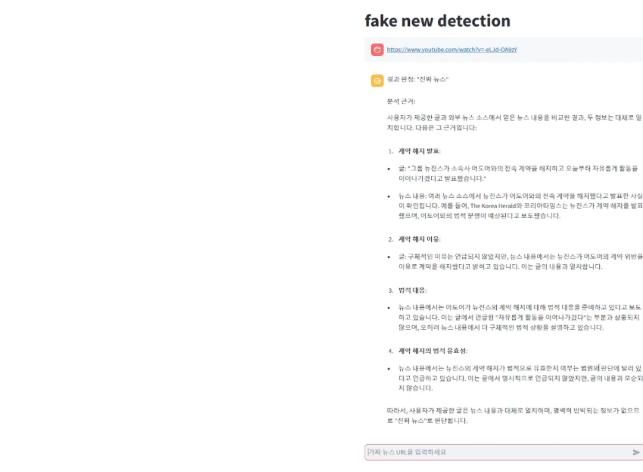
UH3135 committed 2 weeks ago



2. Streamlit을 이용해 프론트엔드를 구현하고 백엔드 서버와 성공적으로 연결하였습니다. 이를 통해 사용자 인터페이스가 완성되었으며, 사용자는 웹 플랫폼을 통해 가짜뉴스 판별 요청과 결과 확인을 쉽게 할 수 있게 하였습니다.

3. 로그인 기능이 완성되어 사용자의 접근 권한을 관리할 수 있는 시스템이 구축되었습니다. 이를 통해 사용자 인증과 개인 맞춤형 서비스를 지원할 수 있는 기반을 마련하였습니다.

<15주차>

활동시간 12/09 20:00 ~ 22:00 12/10 13:00 ~ 20:00 12/11 18:00 ~ 20:00 12/12 13:00 ~ 20:00	활동장소 온라인 화상 미팅(디스코드) 미래관 무한상상실 미래관 자유주행스튜디오
 	<p>1. 버그 수정 및 최적화</p> <p>2. 서버와 LangGraph 모델 연결</p> <p>1. 프로젝트의 주요 기능을 점검하여 발생한 버그를 수정하고 시스템을 최적화하였습니다. 코드 성능 개선, 응답 속도 향상, 시스템 안정성 확보 등을 중점적으로 작업하여 마무리 하였습니다.</p> <p>2. 백엔드 서버와 LangGraph 모델을 성공적으로 연결하였습니다. 이를 통해 사용자의 입력이 서버로 전달되고 LangGraph 모델이 적절히 응답하는 구조가 완성되었습니다. 전체 시스템이 유기적으로 작동하도록 API 호출과 데이터 처리 과정을 통합하였습니다.</p>

- 1주~8주차의 내용은 중간보고서 내용과 동일하게 작성하여도 무관
- 팀의 경우 회의록 작성시 팀원의 50% 이상 인원이 참석하여야 함.
- 15주차 활동 모두 촬영하여 보고서 1장에 사진 1장씩 15주 보고서 작성

양식7)

IV. 알파프로젝트 개인 활동 보고서

-주차별계획-

지도교수 확인란
강승식 (인)

[개인별 작성]

프로젝트명	내용	성명	차민수 (인)
	뉴스 데이터를 기반으로 유튜브 영상의 가짜 정보 판별		

주차별	활동내용 -팀내 개인의 역할-	소요시간 합계	장소
1주차	프로젝트 회의(2h), LLM 스터디(8h), 공모전 개발 설계(8h),	18시간	
2주차	프로젝트 회의(2h), 크롤링 및 스크래핑 공부(8h), 네이버 API를 이용한 뉴스 데이터 수집(8h)	18시간	
3주차	프로젝트 회의(2h), 크롤링으로 네이버 뉴스 수집 방식 설계(8h), 뉴스 수집 코드 구현(8h)	18시간	
4주차	프로젝트 회의(2h), streamlit 사용 방식 공부(6h), rag 시스템 이론과 실습 공부(10h)	18시간	
5주차	프로젝트 회의(2h), 뉴스데이터 전처리 방식 고안 및 설계(8h), 비용 문제 해결을 위한 ollama 공부(8h)	18시간	
6주차	프로젝트 회의(2h), langsmith 공부(2h), 공모전 rag 시스템 history 기능 공부 및 구현(12h), 공모전 프롬프트 엔지니어링(4h)	18시간	온라인 화상 미팅(디스코드)
7주차	프로젝트 회의(2h), 공모전 성능 개선 방안 고안(4h), 프롬프트 성능 개선(6h), 버그 개선 및 코드 최적화(6h)	18시간	
8주차	프로젝트 회의(2h), 공모전 제출 보고서 작성(8h), 중간 보고서 작성(2h), 공모전 제출 영상 제작(2h), 공모전 결과물 알파 프로젝트 활용 방안 고안 및 설계(4h)	18시간	미래관 무한상상실
9주차	프로젝트 회의(2h), 데이터 수집 비용 문제 해결 방안 고안(8h), 전처리 방식 고안 및 설계(8h)	18시간	미래관
10주차	프로젝트 회의(2h), langgraph 학습(12h), langgraph 실습(4h)	18시간	자유주행스튜디오
11주차	프로젝트 회의(2h), 유튜브 영상 다운 및 전처리 방식 공부 및 설계(8h), whisper 사용 방식 설계(8h)	18시간	
12주차	프로젝트 회의(2h), 뉴스 크롤링 방식 개선 및 구현(6h) 유튜브 텍스트 데이터 전처리 및 활용 설계(10h)	18시간	
13주차	프로젝트 회의(2h), langgraph 기능 구현 및 성능 개선(10h), 뉴스 크롤링 함수 개선(2h), 프롬프트 개선(2h), tavy api 활용(2h)	18시간	
14주차	프로젝트 회의(2h), langgraph 수정 및 성능 계선(16h)	18시간	
15주차	최적화 및 버그 수정 보고서 작성(18h)	18시간	
합계		270시간	

본인은 위와 같이 알파프로젝트 개인 활동 보고서를 제출합니다.

2024. 12. 13.

(학과/전공) 소프트웨어학과

(학번) 20203148

(성명) 차민수

(인)

양식7)

IV. 알파프로젝트 개인 활동 보고서

-주차별계획-

지도교수 확인란
강승식 (인)

프로젝트명	뉴스 데이터를 기반으로 유튜브 영상의 가짜 정보 판별하는 프로그램	성명	조현상 조(인)
-------	--------------------------------------	----	----------

주차별	활동내용 -팀내 개인의 역할-	소요시간 합계	장소
1주차	공모전 주제 회의 2시간, LLM 기술 결정 8시간, LangChain 문법 공부 8시간	18시간	온라인
2주차	DATA AI 분석 경진 대회 국회 회의록 데이터 데이터베이스 구축 16시간, 개발 방향성 회의 2시간	18시간	온라인
3주차	DATA AI 분석 경진대회 국회 회의록 데이터 전처리 8시간, LangChain 문법 공부 6시간, 개발 방향성 회의 2시간	18시간	온라인
4주차	LangChain RAG 공부 및 DATA AI 분석 경진 대회에 적용 16시간, 개발 방향성 회의 2시간	18시간	온라인
5주차	LangChain Agent 공부 8시간, Agent Executor를 대회 LLM에 적용 8시간, 개발 방향성 회의 2시간	18시간	온라인
6주차	LLM 성능 향상을 위해 프롬프트 앤지니어링 10시간, 히스토리 최적화 6시간, 개발 방향성 회의 2시간	18시간	온라인
7주차	임베딩 최적의 모델 탐구 10시간, 모델 적용 및 검증 6시간, 개발 방향성 회의 2시간	18시간	온라인
8주차	공모전 문서 작성 및 대회 마무리 16시간, 개발 방향성 회의 2시간	18시간	온라인
9주차	가짜 뉴스 탐지 알고리즘 설계 회의 16시간, 개발 방향성 회의 2시간	18시간	온라인
10주차	LangGraph 기술 공부 10시간, LangGraph로 가짜뉴스탐지 플로우 설계 6시간, 개발 방향성 회의 2시간	18시간	온라인
11주차	LangGraph로 뉴스 검색 Tool 설계 10시간, LangGraph Tool 연결 및 검증 6시간, 개발 방향성 회의 2시간	18시간	온라인
12주차	네이버 뉴스 검색 Tool과 Tavily API Tool 연결 10시간, 버그 수정 6시간, 개발 방향성 회의 2시간	18시간	온라인
13주차	가짜뉴스 탐지 고도화를 위한 프롬프트 앤지니어링 16시간, 개발 방향성 회의 2시간	18시간	온라인
14주차	LangGraph State 수정 8시간, LangGraph Chain 수정 8시간, 개발 방향성 회의 2시간	18시간	온라인
15주차	LangGraph 모델과 연결 및 보고서 작성 18시간	18시간	온라인
합계		270시간	

본인은 위와 같이 알파프로젝트 개인 활동 보고서를 제출합니다.

2024. 12. 13.

학과/전공 소프트웨어학과 학번 20203146 성명 조현상 조(인)

IV. 알파프로젝트 개인 활동 보고서

-주차별계획-

지도교수 확인란

강승식 (인)

[개인별 작성]

프로젝트명	뉴스 데이터를 기반으로 유튜브 영상의 가짜 정보 판별	성명	정의현 (인)
-------	-------------------------------	----	---------

주차별	활동내용 -팀내 개인의 역할-	소요시간 합계	장소
매주	프로젝트 회의	2시간	ZOOM
1주차	알파 프로젝트 공모전 조사	18시간(회의포함)	무한상상실
2주차	Embedding 모델 호출 코드 작성	18시간(회의포함)	무한상상실
3주차	FAISS를 이용한 VectorDB 코드 작성	18시간(회의포함)	무한상상실
4주차	Streamlit을 활용한 인터페이스 작성	18시간(회의포함)	무한상상실
5주차	오픈소스 Embedding 모델들 체크 및 ollama 관련 API 조사	18시간(회의포함)	무한상상실
6주차	Langsmith를 활용하여 현재 코드 동작 분석	18시간(회의포함)	무한상상실
7주차	각 Embedding 모델들 성능 지표 분석	18시간(회의포함)	무한상상실
8주차	공모전 참가를 위한 설명 문서 작성	18시간(회의포함)	무한상상실
9주차	Langserve를 이용한 서버 코딩 학습	18시간(회의포함)	무한상상실
10주차	Fast API의 router와 SQL Alchemy 사용법 학습	18시간(회의포함)	무한상상실
11주차	Koyeb을 사용해서 온라인에 서버 호스팅 진행	18시간(회의포함)	무한상상실
12주차	NLP를 사용해서 문장 요약하는 방법 정리 및 Youtube API 사용방법 학습	18시간(회의포함)	무한상상실
13주차	실제 백엔드 서버 구축 및 Database 생성	18시간(회의포함)	무한상상실
14주차	Streamlit으로 Frontend 작성한 후 서버와 연결, 로그인 기능까지 구현	18시간(회의포함)	무한상상실
15주차	LangGraph 모델과 연결 후 온라인에 배포 이후 RAG 시스템에 대해 정리 및 보고서 작성	18시간(회의포함)	무한상상실
합계		270시간	

본인은 위와 같이 알파프로젝트 개인 활동 보고서를 제출합니다.

2024. 12. 13.

학과/전공 소프트웨어학과

학번 20203135

성명 정의현

정의현
(인)