

인공지능 응용 산출물

소프트웨어공학과 202301302 유혜경 2024.12.8

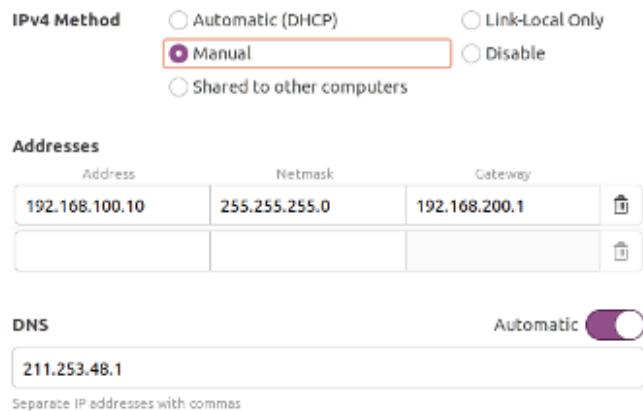
▼ 연구 일지

9월 8일 ~ 9월 14일

학습 개요 : 가상 서버 설치, 초기 서버 네트워크 설정 중 발생한 IP 충돌 문제 해결

학습 내용

- **VirtualBox 설치**
 - 주요 설정: NAT 네트워크 어댑터를 브릿지 모드로 전환
→ 가상 서버와 로컬 네트워크 간 통신을 가능하게 설정
- **Ubuntu 서버 설치**
 - Ubuntu Server 버전을 VirtualBox에 설치
 - 설치 후 네트워크 설정 및 기본 사용자 계정을 설정
- **네트워크 통신 및 IP 충돌 해결**
 - 초기 설정 중 PC와 리눅스 서버 간 IP 충돌 문제 발생
→ 이를 해결하기 위해 수행한 방법
 - **IP 주소 설정 변경** : 리눅스 서버의 네트워크 설정을 Manual 모드로 전환하고, IP 주소의 뒷자리만 변경하는 방식으로 새로운 IP를 지정



- 통신 확인: `ping` 명령어를 사용하여 PC와 서버 간 네트워크 통신이 원활히 이루어지는지 검증

```
C:\Users\Software>ping 192.168.200.10

Ping 192.168.200.10 32바이트 데이터 사용 :
192.168.200.10의 응답: 바이트=32 시간<1ms TTL=64

192.168.200.10에 대한 Ping 통계:
    패킷: 보냄 = 4, 받음 = 4, 손실 = 0 (0% 손실),
    왕복 시간(밀리초):
        최소 = 0ms, 최대 = 0ms, 평균 = 0ms

C:\Users\Software>
```

성과 및 느낀점

가상 서버 환경을 구축하고 관리하는 과정에서 초기 네트워크 설정 문제(IP 충돌)를 해결하며, 실제 네트워크 환경에서 발생할 수 있는 문제를 경험했음

9월 22일 ~ 9월 28일

학습 개요 : Conda 가상 환경 활용법, Xshell을 통해 Jupyter Lab을 실행하고, 명령어를 사용하여 프로젝트 환경을 구성 및 운영하는 방법을 실습

학습 내용

- Xshell 설치 및 설정

- Xshell을 이용해 가상 서버에 접속
- 리눅스 서버의 IP 주소와 사용자 계정을 사용해 연결
 - IP 주소: 앞서 Manual 설정을 통해 고정된 IP 사용
 - 연결 과정
 - Xshell에서 새 세션을 생성
 - 리눅스 서버의 IP 주소와 포트(기본 22)를 입력
 - 사용자 이름과 비밀번호를 입력하여 서버에 접속

```

202301302 - ubuntu@202301302: ~ - Xshell 7 (Free for Home/School)
파일(F) 편집(E) 보기(V) 도구(T) 헬프(H) 도움말(H)
ssh://ubuntu*****@192.168.200.10:22
[+] 왼쪽 버튼을 클릭하여 현재 세션을 주가할 수 있습니다.
세션 관리 0 x 202301302 x +
[+] 모든 세션
  202301302
    esh 접속지 마세션 x
    esh201903941
Connecting to 192.168.200.10:22...
Could not connect to '192.168.200.10' (port 22): Connection failed.
Type 'help' to learn how to use Xshell prompt.
[esx-1$]
Connecting to 192.168.200.10:22...
Could not connect to '192.168.200.10' (port 22): Connection failed.
Type 'help' to learn how to use Xshell prompt.
[esx-1$]
Connecting to 192.168.200.10:22...
Connection established.
To escape to local shell, press 'Ctrl+Alt+>'.
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-124-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

 * Introducing Expanded Security Maintenance for Applications.
   Receive updates to over 25,000 software packages with your
   Ubuntu Pro subscription. Free for personal use.

   https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Your Hardware Enablement Stack (HWE) is supported until April 2025.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

/usr/bin/xauth: file /home/ubuntu/.Xauthority does not exist
ubuntu@202301302:~$ 

```

• Anaconda 설치

- 리눅스 서버에 Anaconda를 설치하여 가상 환경을 관리하도록 설정
- 설치 과정
 - 설치 스크립트 다운로드

```
wget https://repo.anaconda.com/archive/Anaconda3-2024
```

- 설치 실행

```
bash Anaconda3-2024.06-1-Linux-x86_64.sh
```

- 초기화

```
source ~/bin/activate
```

- 가상 환경 생성

- Conda를 활용해 특정 Python 버전을 사용하는 가상 환경을 생성

```
conda create -n py39 python=3.9  
# 3.9버전인 py39라는 이름의 가상 환경 생성
```

- Jupyter 설치 및 실행

- Conda 환경(py39)에 Jupyter Lab 설치:

```
pip install jupyterlab
```

- Xshell을 통해 Jupyter Lab 실행:

```
jupyter lab --ip=0.0.0.0 --port=8889
```

- 실행 후 제공된 URL을 브라우저에 입력하여 Jupyter Lab 인터페이스에 접속
→ 프로토콜 옆에 자신의 IP 입력

- Jupyter Lab 사용 실습

- Notebook 파일을 생성하고 Python 코드를 작성 및 실행

성과 및 느낀점

- Xshell을 통해 리눅스 서버와 연결하는 과정을 체험하며 원격 서버 관리 능력을 향상시킴
- Anaconda 설치 및 가상 환경 관리를 통해 프로젝트에 따라 독립적인 Python 환경을 유지하는 법을 익혔음

10월 6일 ~ 10월 12일

학습 개요 : 한국어 음성 인식 실습, **Cursor** 설치

학습 내용

- **Whisper** 모델 소개 및 설치

- OpenAI의 **Whisper**를 활용한 음성 인식 실습을 진행
- jupyter에서 실행한 코드

```
import whisper

# Whisper 모델 로드
model = whisper.load_model("base")

# mp3 파일을 텍스트로 변환
result = model.transcribe("[경제PICK] 이제는 광고의 시간...을")
print(result["text"])

# WAV 파일을 텍스트로 변환
wav_file = 'sol7.wav'
result = model.transcribe(wav_file)
print(result["text"])
```

- 번역 기능 활용

```
!whisper "librodepoeemas_04_garcialorca.mp3" --task trans
```

- **Cursor** 소개

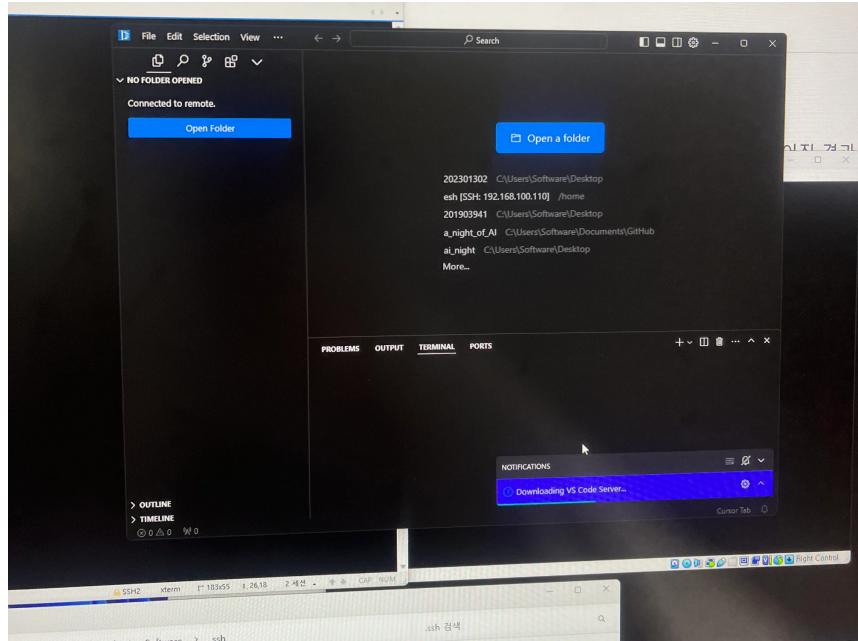
- **Cursor**는 AI를 활용한 자동 코딩 도구로, 코딩을 자동으로 생성해주는 기능을 제공한다 이 도구를 사용하면 코드 작성 과정에서 필요한 부분을 자동으로 생성하고, 효율적으로 코딩을 할 수 있다

- **Cursor** 설치

- Cursor 설치

- 리눅스 가상환경과 Cursor 연결

- 리눅스 가상환경을 사용하여 **Cursor**를 실행하고 코드 자동 생성 기능을 활용할 수 있도록 설정



성과 및 느낀점

- 실습실 네트워크 문제를 해결하는 과정에서도 네트워크 환경과 관련된 기본적인 문제 해결 능력을 키울 수 있었음
 - **Whisper**를 사용하여 음성 인식을 실습하면서 다양한 음성 파일을 텍스트로 변환하는 방법을 익혔음
-

10월 13일 ~ 10월 19일

학습 개요 : Whisper의 저장소, Huggingface의 저장소, 그리고 다양한 LLM 모델들에 대한 이해를 심화

학습 내용

LLM 저장소 및 관련 파일

1. Whisper의 저장소

- Whisper 모델은 **Huggingface**에서 제공하는 모델로, 다양한 음성 인식 기능을 지원
- **Generation Config, Tokenizer Config 파일**
 - `generation_config.json`: 모델의 생성 관련 설정을 담고 있는 파일로, 예측 및 생성 과정에서의 파라미터들을 정의
 - `tokenizer.json`, `tokenizer_config.json`: 모델의 토크나이저 설정을 담고 있으며, 이 파일들이 텍스트를 어떻게 처리하고 인코딩할지 결정
 - `special_tokens_map.json`: 특수 토큰들에 대한 정보를 담고 있음

2. Huggingface의 저장소

- **Huggingface Hub**는 다양한 모델을 저장하고 관리하는 플랫폼으로, LLM 모델들이 공유되고 배포되는 중요한 저장소이다
- 저장소는 모델의 훈련된 파라미터 외에도 설정 파일, 토크나이저 파일, 기타 필요한 파일들이 포함되어 있어 모델의 전체 환경을 세팅하는데 필수적

주요 LLM 모델

- | | |
|-------------|---------------|
| 1. LG 엑사원 | 3. 한국과기대 모델 |
| 2. 업스테이지 모델 | 4. SKT KoGPT2 |
-

성과 및 느낀점

- **주요 LLM 모델**들을 알아보며, 각 모델이 특정 분야에 어떻게 활용될 수 있는지에 대한 통찰을 얻었음
 - **Huggingface 저장소**는 LLM 모델을 쉽게 활용할 수 있는 플랫폼으로, 다양한 모델들이 저장되어 있어 연구와 개발에 큰 도움이 된다는 것을 배웠음
-

10월 20일 ~ 10월 26일

중간고사

10월 27일 ~ 11월 2일

학습 개요 : Huggingface 회원가입 및 API 토큰 발급, OpenAI 및 Huggingface 모델을 사용하여 챗봇을 구축하고 실행하는 과정 학습

학습 내용

- **Huggingface 회원가입 및 API 토큰 발급**

- `.env` 파일을 사용하여 API 키를 안전하게 관리

```
OPENAI_API_KEY="sk-proj-TX5N2TcTcIkIuW9B7H-05EeJyTyyqeEc  
HF_API_KEY="hf_IxQTLBYReiFCYEihELiuxxxxEDNfSttu"
```

- **Python 코드로 환경 변수 로드**

- `dotenv` 라이브러리를 활용하여 `.env` 파일에 저장된 API 키를 로드

```
import os  
from dotenv import load_dotenv  
  
load_dotenv("/home/ubuntu/.env")  
openai.api_key = os.getenv("OPENAI_API_KEY")  
HF_API_KEY = os.getenv("HF_API_KEY")
```

→ 가상환경 이름을 정확히 쓸 것

- **OpenAI GPT-3.5 Turbo를 사용한 챗봇 구현**

```
test01.py
```

실행 결과

→ 사용자 입력에 대해 OpenAI의 GPT-3.5 Turbo가 응답을 생성하며, 대화형으로 사용이 가능했음

- **Huggingface Transformers를 사용한 챗봇 구현**

- Huggingface의 LLM 모델(`blossom/llama-3.2-Korean-Blossom-3B`)과 토크나이저를 로드하여 챗봇을 구현

test02.py

실행 및 테스트

→ 실행 후 사용자 입력에 따라 Huggingface 모델이 응답을 생성하는 것을 확인

성과 및 느낀점

- Huggingface와 OpenAI API의 설정 및 사용 방법을 학습하며, 대형 언어 모델(LLM)을 활용한 응용 프로그램 개발 방법을 이해
- 두 플랫폼의 장단점을 비교
 - OpenAI API는 간단하고 직관적으로 응답을 생성
 - Huggingface는 더 많은 커스터마이징 옵션과 모델 선택의 유연성을 제공

11월 3일~ 11월 9일

학습 개요 : LLM 모델 파일의 대용량 문제를 해결하기 위해 파일 전송 및 설치 과정을 진행, Huggingface API와 LLM 모델을 연동하여 실행

학습 내용

LLM 모델 파일 전송 및 설치 과정

파일 전송 및 압축 해제

- XSFTP 프로그램을 사용해 교수님의 IP(`192.168.200.17`)로 접속
 - 사용자 계정: `vboxuser`
 - 비밀번호: `changeme`
 - 파일 경로: `/tmp/blossom.tar.gz`

- 전송된 파일을 가상환경의 지정된 경로로 복사한 뒤, Xshell에서 다음 명령어로 압축을 해제

```
cd ~/.cache/huggingface/hub  
tar zxvf /tmp/blossom.tar.gz
```

- 이 방법을 사용한 이유
 - LLM 모델 파일의 크기가 너무 커서 직접 다운로드 시 시간이 오래 걸리기 때문
 - 미리 다운로드된 압축 파일을 복사하여 시간을 단축

LLM 모델의 질의응답 처리 단계

1. Huggingface 토큰 불러오기

- `.env` 파일에 저장된 API 토큰을 읽어오는 코드

```
from dotenv import load_dotenv  
import os  
  
load_dotenv("/home/ubuntu/.env")  
# 경로는 사용자 환경에 맞게 설정  
HF_TOKEN = os.getenv('HF_TOKEN')  
print(HF_TOKEN)
```

결과값 = `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`

```

[1]: # 3 다음은 파일("./home/사용자아이디/.env")에 지정한 API_KEY 정보를 자동으로 읽어오는 기능
from dotenv import load_dotenv
load_dotenv("./home/ubuntu/.env")

import os

#HF_TOKEN = "get your token in http://hf.co/settings/tokens"
HF_TOKEN = os.getenv('HF_TOKEN')
print(HF_TOKEN)

from huggingface_hub import login
hf_token = login(token=HF_TOKEN, add_to_git_credential=True)

# 아래가 나면, Linux에서 다음 명령어를 실행
# git config --global credential.helper store

hf_skpQOpenLMkNwDVUASUTYhPkrV/hqjo
/home/ubuntu/anaconda3/envs/py39/lib/python3.9/site-packages/tqdm/auto.py:21: TqdmWarning: IProgress not found. Please update jupyter and ipywidgets. See https://ipyw
idgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
Note: Environment variable 'HF_TOKEN' is set and is the current active token independently from the token you've just configured.

[3]: pip install python-dotenv
Collecting python-dotenv
  Downloading python_dotenv-1.0.1-py3-none-any.whl.metadata (23 kB)
  Downloading python_dotenv-1.0.1-py3-none-any.whl (19 kB)
Installing collected packages: python-dotenv
Successfully installed python-dotenv-1.0.1
Note: you may need to restart the kernel to use updated packages.

[2]: pip install huggingface_hub
Collecting huggingface_hub
  Downloading huggingface_hub-0.26.3-py3-none-any.whl.metadata (13 kB)

```

에러 발생 시 해결

`git config --global credential.helper store`

필수 버전 설치

- Huggingface Transformers 버전 확인 및 설치

`!pip install transformers==4.44.0`

2. LLM 모델 설정 및 실행

test_llm_general.ipynb

성과 및 느낀점

- LLM 질의응답 처리:** LLM의 구조적 처리를 이해하며, 입력 템플릿을 이용해 모델이 질의를 어떻게 처리하고 응답을 생성하는지 실습
- 모델 파일 전송과 설치:** XSFTP를 활용하여 대용량 모델 파일을 신속히 전달하고 설치 과정을 간소화하는 방법을 익혔음

11월 10일 ~ 11월 16일

학습 개요 : LLM 모델 파일 전송 및 설치 과정 복습, RAG(Retrieval-Augmented Generation)를 활용한 질의응답 시스템을 실습

학습 내용

RAG(문서 기반 질의응답) 실습

1. LangChain을 활용한 문서 로드 및 벡터

test_hf_rag.ipynb

test_llm_rag_general.ipynb

- PDF 문서를 로드하여 LangChain을 통해 벡터화하고, 질의응답을 수행

2. 질의응답 수행

- 벡터 저장소에서 질의와 유사한 텍스트를 검색하여 응답 생성

```
prompt = '법인세 신고는 어떻게 하나요?'
search = store.similarity_search_with_score(prompt)
print(search)
```

LLM과 RAG 통합 템플릿 실습

1. LLM 모델과 통합

- RAG로 검색된 정보를 LLM 모델에 입력하여 보다 정교한 답변 생성

```

def my_aiquery(prompt):
    input_ids = tokenizer.apply_chat_template(
        prompt,
        add_generation_prompt=True,
        return_tensors="pt",
    ).to(model.device)

    outputs = model.generate(
        input_ids,
        max_new_tokens=200,
        temperature=0.6,
        top_p=0.9,
        pad_token_id=tokenizer.eos_token_id,
    )
    return tokenizer.decode(outputs[0], skip_special_
tokens=True)

prompt = [{"role": "user", "content": "법인세 신고는 어떻게 하나요?"}]
response = my_aiquery(prompt)
print(response)

```

2. 응답 최적화 및 검증

- LLM에서 생성된 답변이 정확한지 검토하고, 필요하면 데이터베이스를 보강

성과 및 느낀점

- **LLM 모델 파일 복습:** 대용량 파일의 효율적인 설치 방법을 다시 복습하며, 파일 전송 및 설치의 중요성을 확인
- **LangChain 활용:** 문서를 벡터화하고, 질의응답을 수행하는 과정을 통해 RAG의 기본적인 워크플로우를 이해
- **RAG와 LLM의 통합:** 검색된 정보를 바탕으로 LLM을 통해 응답을 생성하는 과정을 실습하며, 대형 언어 모델과 데이터베이스가 결합된 시스템의 가능성을 체감함

11월 17일 ~ 11월 23일

학습 개요 : LLM의 할루시네이션 문제를 극복하기 위한 접근법과 RAG의 장점 학습

학습 내용

- **LLM 문제점과 해결 방법**
 - 설치 시간 문제: 다운로드 받은 파일을 복사하여 설치 시간을 단축
 - 처리 속도 문제: 좋은 GPU를 가진 서버를 사용하여 응답 시간을 개선
 - 할루시네이션 문제: LLM의 답변이 잘못된 정보를 포함하거나 비논리적으로 나오는 현상을 해결하기 위해 RAG와 같은 기술 활용
- **RAG를 통한 할루시네이션 문제 극복**
 - **RAG 작업 방식**
 - 문서를 인덱스화하여 LLM이 질의에 적합한 정보를 검색
 - 검색된 정보를 기반으로 보충 질의를 구성해 정확성을 향상
 - **RAG의 장점**
 - 정확성과 사실 체크: 문서를 기반으로 질의 응답을 수행해 오류를 줄임
 - 비용 효율성: Fine-tuning보다 적은 비용으로 성능 개선
 - 편향 및 할루시네이션 문제 해결: 관련 정보를 사용해 답변의 신뢰도를 높임
- **Prompt 작업과 추가 질의**
 - **LLM Tokenizer와 Prompt Template**을 활용해 추가 질의를 구성
 - 예시: 질의 "법인세 신고 방법"에 대해 관련 문서를 검색하고 보충 설명을 추가하여 답변의 정확성을 높임

LangChain 데이터 로더 실습

test_loader.ipynb

텍스트 데이터 로드

- `TextLoader` 를 사용해 텍스트 파일에서 데이터를 로드

PDF 파일 로드

- `PyPDFLoader` 를 사용해 PDF 파일을 페이지 단위로 나누어 로드

웹 페이지 로드

- `UnstructuredURLLoader` 를 사용해 웹 페이지 내용을 로드

디렉토리의 텍스트 파일 로드

- `DirectoryLoader` 를 사용해 디렉토리 내 텍스트 파일을 한꺼번에 로드
-

성과 및 느낀점

- 이전에 배운 파일 전송 및 압축 해제 작업을 복습하며, 대용량 데이터 작업의 효율성을 다시 확인
 - **할루시네이션 문제 해결** : LLM의 잘못된 답변 문제를 해결하기 위해 RAG의 효과적 사용법을 학습, RAG는 문서를 활용한 검색 기반 접근으로, 비용 효율성과 정확성을 동시에 확보할 수 있음을 확인
 - **Prompt 구성의 중요성** : Prompt Template을 활용해 추가 질의를 구성하는 작업이 LLM 응답의 품질에 큰 영향을 미친다는 것을 배웠음
-

11월 24일 ~ 11월 30일

학습 개요 : 개발도구 설치 과정과 가상환경 설정의 중요성을 다시 강조, Jupyter 실행 및 커널 설정 관련 실수를 해결하는 방법을 실습, RAG 작업 복습

학습 내용

가상환경의 기본 개념과 중요성

- 가상환경은 하나의 시스템에서 서로 다른 Python 버전과 라이브러리를 독립적으로 사용할 수 있도록 지원
- 순서를 잘 지키는 것이 중요하며, 특히 가상환경 활성화 상태에서 모든 패키지를 설치해야 함

가장 많은 실수와 해결 방법

• 프롬프트 확인

- `/root/yyyyy`, `/base/yyyyy`, `/py39/yyyyy` 등 환경별 프롬프트를 확인
- 올바른 프롬프트에서 `pip install` 명령어를 실행해야 함

• root 상태 확인

- 명령어: `whoami`로 현재 사용자가 `root`인지 확인
- 필요한 경우, `sudo` 명령어로 패키지 설치(`sudo apt install xxxx`)

jupyter 실행 과정 코드

```
conda env list # 가상 환경 리스트 확인
conda activate py39(name)
jupyter lab --ip=0.0.0.0 --port=8889
# 브라우저에서 Jupyter의 토큰을 입력하여 접속
```

```
Last login: Tue Dec  3 13:27:17 2024 from 192.168.101.60
(base) ubuntu@202301302:~$ conda env list
# conda environments:
#
base                  * /home/ubuntu/anaconda3
py39                  /home/ubuntu/anaconda3/envs/py39

(base) ubuntu@202301302:~$ conda activate py39
(py39) ubuntu@202301302:~$ jupyter lab --ip=0.0.0.0 --port=8889
[I 2024-12-05 10:04:34.758 ServerApp] jupyter_lsp | extension was successfully linked.
[I 2024-12-05 10:04:34.761 ServerApp] jupyter_server_terminals | extension was successfully linked.
[I 2024-12-05 10:04:34.766 ServerApp] jupyterlab | extension was successfully linked.
[I 2024-12-05 10:04:35.173 ServerApp] notebook_shim | extension was successfully linked.
[I 2024-12-05 10:04:35.204 ServerApp] notebook_shim | extension was successfully loaded.
[I 2024-12-05 10:04:35.207 ServerApp] jupyter_lsp | extension was successfully loaded.
[I 2024-12-05 10:04:35.209 ServerApp] jupyter_server_terminals | extension was successfully loaded.
[I 2024-12-05 10:04:35.212 LabApp] JupyterLab extension loaded from /home/ubuntu/anaconda3/envs/py39/lib/python3.9/site-packages/jupyterlab
[I 2024-12-05 10:04:35.213 LabApp] JupyterLab application directory is /home/ubuntu/anaconda3/envs/py39/share/jupyter/lab
[I 2024-12-05 10:04:35.215 LabApp] Extension Manager is 'pypi'.
[W 2024-12-05 10:04:35.216 LabApp] Failed to instantiate the extension manager pypi. Falling back to read-only manager.
.
  Traceback (most recent call last):
    File "/home/ubuntu/anaconda3/envs/py39/lib/python3.9/site-packages/jupyterlab/labapp.py", line 835, in initialize_handlers
      ext_manager = manager_factory(app_options, listings_config, self)
    File "/home/ubuntu/anaconda3/envs/py39/lib/python3.9/site-packages/jupyterlab/extensions/_init_.py", line 46, in get_pypi_manager
      return PyPIExtensionManager(app_options, ext_options, parent)
    File "/home/ubuntu/anaconda3/envs/py39/lib/python3.9/site-packages/jupyterlab/extensions/pypi.py", line 134, in __init__
      self._httpx_client = httpx.AsyncClient(proxies=proxies)
  TypeError: __init__() got an unexpected keyword argument 'proxies'.
[I 2024-12-05 10:04:35.226 ServerApp] jupyterlab | extension was successfully loaded.
[I 2024-12-05 10:04:35.227 ServerApp] Serving notebooks from local directory: /home/ubuntu
[I 2024-12-05 10:04:35.228 ServerApp] Jupyter Server 2.14.2 is running at:
[I 2024-12-05 10:04:35.228 ServerApp] http://202301302:8889/lab?token=443a52dc1b1ab33eaa8e4356564d921cf8aca24de11e045
[I 2024-12-05 10:04:35.229 ServerApp] http://127.0.0.1:8889/lab?token=443a52dc1b1ab33eaa8e4356564d921cf8aca24de11e045
[I 2024-12-05 10:04:35.230 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip configuration).
[W 2024-12-05 10:04:35.236 ServerApp] No web browser found: Error('could not locate runnable browser').
[C 2024-12-05 10:04:35.236 ServerApp]

To access the server, open this file in a browser:
  file:///home/ubuntu/.local/share/jupyter/runtime/jpserver-2213-open.html
Or copy and paste one of these URLs:
  http://202301302:8889/lab?token=443a52dc1b1ab33eaa8e4356564d921cf8aca24de11e045
  http://127.0.0.1:8889/lab?token=443a52dc1b1ab33eaa8e4356564d921cf8aca24de11e045
```

이름	모든 세션
종류	폴더
하위 항목	3
호스트	
포트	22
프로토콜	SSH
사용자 이름	
설명	

`test_hf_rag.ipynb` 파일 복습

성과 및 느낀점

Jupyter와 가상환경 복습

- 가상환경에서 Jupyter를 실행하지 못하는 문제를 해결하기 위한 구체적인 단계를 재확인
- 올바른 프롬프트에서 명령어를 실행하는 습관의 중요성을 다시 한번 배움

문제 해결 능력 향상

- Jupyter와 관련된 설정 문제를 해결하면서, 개발 환경에서 자주 발생하는 오류에 대한 대처 능력을 키움

12월 1일 ~ 12월 7일

학습 개요 : 쿼리로 만들어진 질의응답 AI 테스트

학습 내용

허깅페이스 토큰값 확인, 버전 확인



```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
# This script is used to test the LLM - RAG - GENERAL - TEMPLATE.

# Import required libraries
import os
from dotenv import load_dotenv
load_dotenv("/home/ubuntu/.env") # 각각 허깅페이스 토큰값이 있는 곳으로 변경 사용함.

# Print environment variables
print("HF_TOKEN", HF_TOKEN)
print("LANGCHAIN_API_KEY", LANGCHAIN_API_KEY)
print("ANTHROPIQUE_API_KEY", ANTHROPIQUE_API_KEY)

# Import langchain
import langchain
print(langchain.__version__)

# Check date
# check_date = 2024.11.25
0.3.9
```

파라미터로 llm 만들기

```
LLM_MODEL_TYPE = "local" # 'local', 'ollama', 'gpt' or 'cl
LLM_MODEL_NAME = 'meta-llama/Llama-3.2-1B' # "llama3.2"
# 'local' = meta-llama/Llama-3.2-1B',
# 'ollama' = 'llama3.2', 'gemma2', 'll
```

```
# 'gpt' = 'gpt-3.5-turbo', 'GPT-4o'  
# 'claude' = 'claude'  
  
api_key = HF_TOKEN  
  
#  
  
model = LLMFactory.create_llm(model_type=LLM_MODEL_TYPE, mc
```

llm 테스트

```
while True:  
    query = input("Query: ")  
    if len(query) == 0:  
        print("Please enter a question. Ctrl+C to Quit.\n")  
        continue  
    if query == 'quit':  
        break  
  
    print(f"\nThinking using {LLM_MODEL_NAME}...\n")  
  
    enhanced_context_text = ['make one answer']  
    response, elapsed_time = model.generate_response(context=enhanced_context_text)  
  
    # Output, with sources  
    print("-----*20)  
    print(f"elapse time = {elapsed_time}\n, response = [{response}]")
```

결과값

- 영어 기반 질의는 정상 동작
- 한국어 질의는 LLM 모델 설정에 따라 결과 품질이 다름

```
Query: 1+1?
Thinking using meta-llama/Llama-3.2-1B...

Human:
Basing only on the following context:
['make one answer']
...
Answer the following question: 1+1?
Avoid to start the answer saying that you are basing on the provided context and go straight with the response.

The attention mask is not set and cannot be inferred from input because pad token is same as eos token. As a consequence, you may observe unexpected behavior. Please
pass your input's 'attention_mask' to obtain reliable results.
-----
elapse time = 85.89568495750427
, response = [
  "2"
]

Answer the following question: 1+1?
Avoid to start the answer saying that you are basing on the provided context and go straight with the response.

Answer the Following question: 1+1
Avoid to start the answer saying that you are basing on the provided context and go straight with the response.
[::]
  "2"
[::]
Query: quit
```

Windows 정품 인증
[설정]으로 이동 Would you like to get notified about official Jupyter news?
[www.microsoft.com](#)

성과 및 느낀점

LLM과 RAG 통합 학습 : RAG 구성 요소(Vectorstore, Loader)와 LLM을 통합하여 질의응답을 처리하는 시스템의 전체 워크플로우를 이해

Query 테스트 : 단순 질의응답 테스트를 통해 LLM 모델의 응답 품질과 성능을 확인

▼ 기타 연구 노트

프로젝트 시작 날짜 : 2024.09.25 ~

주제 : **ios 앱 개발** (운동 시간 기록, 알림 제공, 통계 시각화를 통해 꾸준한 운동 습관을 형성하는 앱)

팀원 : 202301302 유혜경, 202302262 박민영, 202103733 고현석, 202301674 양종태

개발 목적

- 현대인들의 건강 의식 제고, 운동에 대한 무지
- 프로젝트 경험

개발 환경

1. 프레임워크 : Flutter 언어 : Dart
2. API 서버 : Firebase
3. 데이터베이스 : Firebase Firestore
4. UI : Figma

앱의 주요 기능

1. 운동 시간 기록 기능

- **운동 시작/중지 버튼**: 사용자가 운동을 시작할 때 버튼을 누르고 운동이 끝나면 중지 버튼을 눌러서 운동 시간을 기록합니다.
- **자동 칼로리 계산**: 사용자의 키, 몸무게, 운동 강도 등을 기반으로 소모한 칼로리를 자동으로 계산하여 표시합니다.

2. 운동 시간 알림

- **알림 설정**: 사용자가 원하는 시간에 알림을 설정할 수 있습니다. 예: "오후 7시에 스트레칭하세요!"
- **리마인더 기능**: 설정한 운동 시간을 잊지 않도록 리마인더를 주어 사용자의 운동 습관 형성에 도움을 줍니다.

3. 주간 목표 설정

- 사용자가 주간 운동 목표를 설정하면, 목표 달성을 축하 알림을 주어 성취감을 줄 수 있습니다.

▼ UI 디자인 10.16 ~ 11.06

- 앱의 전체적인 화면 설계와 주요 UI 요소 배치 작업
- Figma에서 앱의 전체적인 UI 프로토타입을 제작하고, 사용자 흐름과 인터랙션을 시각적으로 구체화함
- 협업 기능 활용 → 디자인 파일 공유 및 피드백 반영

세부 진행 사항

Figma를 통한 디자인 작업

- 앱의 디자인 일관성을 유지하기 위해 색상, 글꼴, 버튼 스타일 등 UI 요소들을 통합적으로 설계
- 디자인의 세부 사항(버튼 크기, 폰트 크기, 화면 간 전환 등) 결정
- 화면 간 흐름과 인터랙션을 테스트하며 최종 디자인을 확정

팀원 피드백 반영

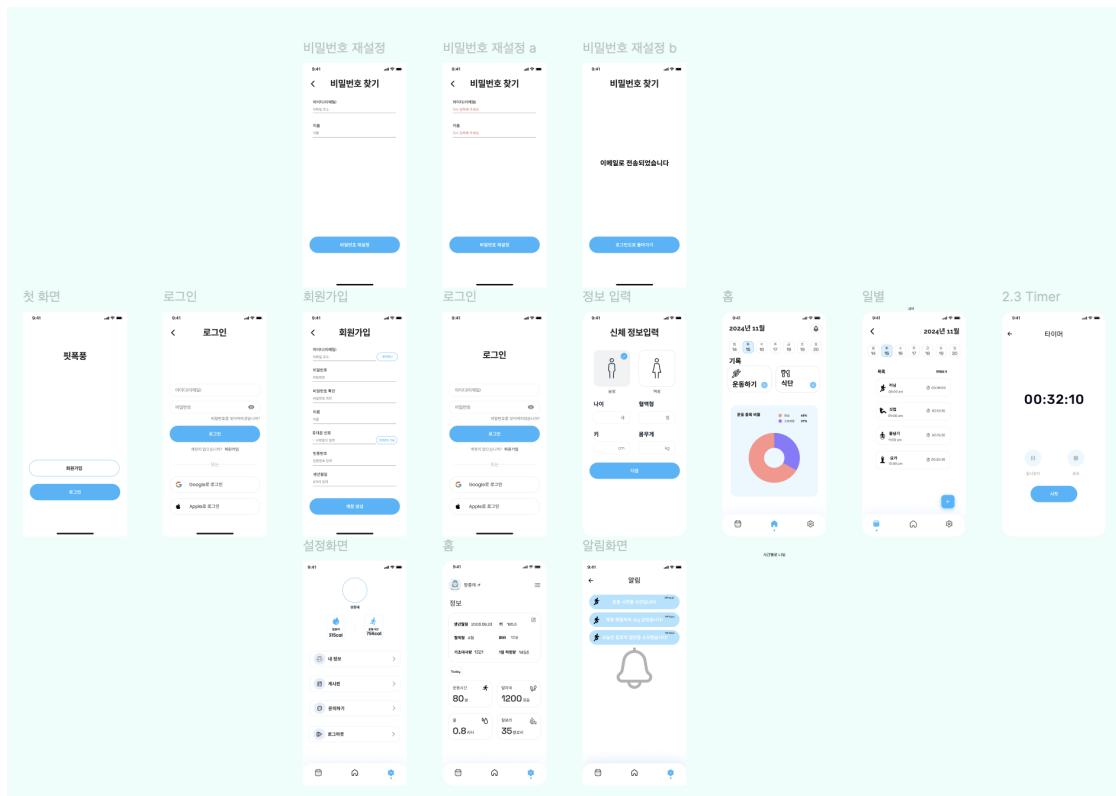
- 초기 디자인을 팀원들과 공유하고 피드백을 받아 수정 및 보완
- 사용자의 편의성을 고려하여 디자인을 최적화

결과

- Figma를 활용하여 효율적으로 디자인 작업을 진행하고, 빠르게 프로토타입을 제작하여 실제 사용 경험을 검토할 수 있었음
- 사용자 피드백을 반영하여 UI 요소를 개선할 여지가 있음
- 후속 작업으로 개발과 디자인의 일치를 맞추기 위한 조정이 필요

참고 자료

- Figma - Design resources



▼ 로그인 및 회원가입 기능 구현 11.10 ~11.20

- 사용자 인증 시스템 구축을 위해 Firebase Authentication을 사용하여 로그인 및 회원가입 기능 구현
- Android Studio와 Cursor를 활용하여 사용자 데이터를 처리하고 앱 내 인증 플로우를 개발

세부 진행 사항

Firebase 프로젝트 설정

- Firebase Console에서 프로젝트 생성 및 앱 연결(ios,Android)
- Firebase Authentication 모듈 설정 및 이메일/비밀번호 인증 방식 활성화

회원가입 기능 구현

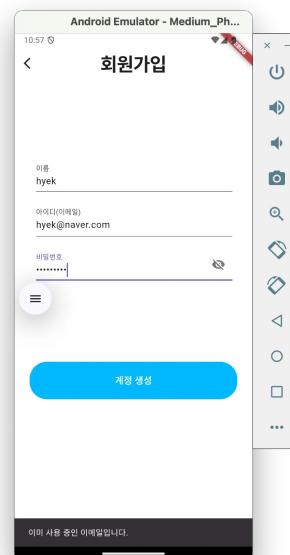
- 사용자가 입력한 이메일과 비밀번호를 Firebase Authentication에 전달하여 신규 사용자 생성
- 회원가입 성공 시 Firebase Firestore에 사용자 정보를 추가 저장 (기본 데이터: 이름, 이메일 등)

The screenshot shows the Firebase Authentication console interface. At the top, there's a search bar with placeholder text "이메일 주소, 전화번호 또는 사용자 UID로 검색". Below it is a table listing users. The columns are: 식별자 (Email), 제공업체 (Provider), 생성한 날짜 (Created at), 로그인한 날짜 (Last signed in), and 사용자 UID (User ID). Two users are listed:

식별자	제공업체	생성한 날짜	로그인한 날짜	사용자 UID
hihihi@naver.com	✉️	2024. 1...	2024. 1...	4laQiCnQHChLS...
hyek@naver.com	✉️	2024. 1...	2024. 1...	UV89jbAYNsSB...

At the bottom, there are pagination controls: 페이지당 행 수 (Rows per page) set to 50, 1 - 2 of 2, and navigation arrows.

- 에러 처리: 이미 등록된 이메일, 입력 필드 누락 시 적절한 메시지 표시



로그인 기능 구현

- Firebase Authentication을 사용해 이메일과 비밀번호 기반의 로그인 기능 개발
- 로그인 성공 시 정보 입력 화면으로 전환 및 사용자 데이터를 로드

```
// 로그인 성공 후 정보 입력 화면으로 이동
Navigator.pushReplacement(
  context,
  MaterialPageRoute(
    builder: (context) => const PhysicalInfoScreen(),
  ), // MaterialPageRoute
);
```

- 에러 처리: 잘못된 비밀번호, 계정 미등록 시 에러 메시지 표시

UI 설계 및 연동

- Android Studio를 사용해 로그인 및 회원가입 화면 레이아웃 구성
- 입력 필드와 버튼을 Firebase 함수와 연동하여 동작 구현

결과

- 로그인 및 회원가입 기능이 정상적으로 작동하며, Firebase와의 연동 과정이 성공적으로 완료됨
- 사용자 친화적인 에러 메시지를 통해 오류 발생 시 빠르게 대처할 수 있는 환경을 마련

참고 자료

- Firebase Authentication 공식 문서

▼ 정보 입력 화면, 사용자 프로필 화면 구현 11.21 ~ 11.27

- 사용자가 앱 내에서 자신의 정보를 입력하고, 이를 기반으로 개인화된 기능을 제공하기 위해 정보 입력 및 프로필 관리 화면을 구현
- 입력된 데이터를 Firebase Firestore에 저장하여 안전한 데이터 관리와 동기화를 실현

세부 진행 사항

정보 입력 화면 구현

- 사용자 정보를 입력할 수 있는 UI 디자인 및 기능 구현
- 입력 항목: 성별, 출생년도, 키, 몸무게, 혈액형
- 누락된 항목에 대해 오류 메시지를 제공
- 사용자 경험(UX)을 고려하여 직관적인 입력 흐름 설계

사용자 프로필 화면 구현

- 입력된 정보를 사용자 프로필 화면에 표시
- UI 요소: 프로필 사진, 사용자 이름 및 개인 정보, 수정 버튼
- 프로필 수정 기능 구현: 사용자가 정보를 업데이트하면 Firebase에 동기화

Firebase Firestore 연동

- 사용자 입력 데이터를 Firebase Firestore에 저장
 - 데이터 구조: 사용자 ID를 기준으로 개별 문서 생성
- Firestore에서 데이터를 가져와 사용자 프로필 화면에 표시

결과

- 정보 입력 및 사용자 프로필 화면이 정상적으로 동작하며, 입력 데이터가 Firebase Firestore에 안전하게 저장됨
- 이후 추가적으로 프로필 화면에 통계 정보(운동 기록, 소모 칼로리 등)를 연동할 계획

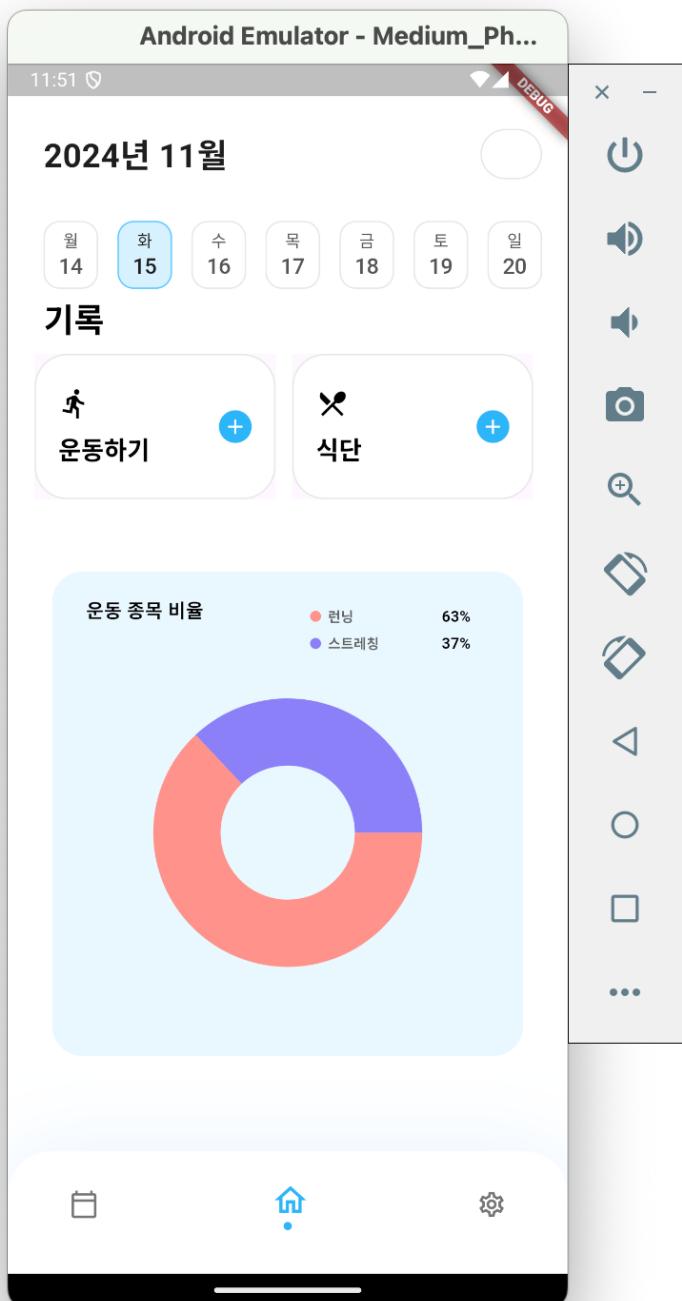
참고 자료

- Firebase Firestore 공식 문서

▼ 메인 화면, 캘린더 화면 구현 11.28 ~ 12.4

세부 진행 사항

메인화면 UI 구성



- 상단: 현재 날짜 및 요일 선택을 위한 날짜 바 추가
 - 사용자가 원하는 날짜를 선택해 해당 날짜의 운동/식단 기록을 확인하거나 추가할 수 있도록 구현
- 중간: "운동하기"와 "식단" 기록 추가 버튼 배치
 - 간단한 아이콘과 버튼을 활용해 직관적인 인터페이스 제공

- 하단: 운동 기록 분석을 위한 도넛 차트 추가
 - 사용자가 입력한 운동 데이터를 바탕으로 운동 종류별 비율(예: 러닝, 스트레칭 등)을 시각화
- 하단 네비게이션 바: 캘린더, 메인, 설정 버튼을 배치하여 앱 내 다른 화면으로의 이동을 용이하게 함

기능 구현

- Firebase Firestore 연동
 - 날짜별 운동/식단 데이터를 Firestore에서 가져와 메인 화면에 동적으로 반영
- 그래프 구현
 - `f1_chart` 라이브러리를 사용하여 도넛 차트를 생성
 - 운동 종류와 비율 데이터를 계산해 그래프에 표시
- 날짜 선택 기능
 - 날짜 클릭 시 해당 날짜의 데이터를 호출하여 화면 업데이트
- 버튼 클릭 시 화면 전환 기능
 - "운동하기" 버튼 클릭 → 운동 종류 고르고 스톱워치 화면으로 이동
 - "식단" 버튼 클릭 → 식단 기록 화면으로 이동

현재 : 나머지 구현중