

# 2024 전기 중간 보고서

머신러닝, 자연어 처리 기반 중고차 추천 플랫폼



팀명	도미노	담당교수	유영환 교수님
	202012142		이은진
학번	201824606	이름	최희웅
	201913543		최성빈

# Table

1. 요구조건 및 제약 사항 분석에 대한 수정사항.....	3
1.1. 요구 조건.....	3
1.2. 제약 사항 분석.....	3
2. 설계 상세화 및 변경 내역.....	4
2.1. 중고차 데이터셋 크롤링.....	4
2.2. 모델 학습 및 서버.....	5
3. 갱신된 과제 추진 계획.....	6
4. 구성원별 진척도.....	7
5. 보고 시점까지의 과제 수행 내용 및 중간 결과.....	7
5.1. 언어 모델 선정 및 테스트.....	7
5.2. DB 디자인 설계.....	10
5.3. 챗봇 웹 UI 디자인.....	11
5.4. 학습 데이터 모으기 - crawling, 전처리.....	13
5.5. 학습 데이터 모으기 - openAI.....	16
5.6. 백엔드 서버 개발 환경 구축.....	20

# 1. 요구조건 및 제약 사항 분석에 대한 수정사항

## 1.1. 요구 조건

- 연식, 주행거리, 연료, 배기량, 가격, 옵션 등의 정보를 전처리과정을 거쳐 정제한 대량의 중고차 데이터셋을 수집한다.
- 수집한 데이터셋을 pre trained model인 KULLM3에 학습시키고 fine tuning을 진행하여 모델의 개선 정도를 평가한다.
- 모델을 챗봇 형태로 웹어플리케이션을 통해 이용가능하게 만든다. 이를 통해 사용자들의 요구사항과 선호도를 입력하면 실시간으로 중고차 차량을 추천해주는 모델을 배포한다. 또한 사용자가 필요한 중고차 구매시 필요한 정보를 제공할 수 있다.
- 로그인 기능을 통해서 사용자들의 정보를 백엔드에서 관리하고 저장할 수 있다.

## 1.2. 제약 사항 분석

1.2.1. 프로젝트의 초기 계획에서는 최신 언어 모델인 KULLM3를 채택하여 개발을 진행할 예정이었다. 그러나 대용량의 파라미터를 가지는 KULLM3를 언어 모델로 사용하는것은 현재 보유하고있는 컴퓨팅 성능으로는 진행하기 어려웠다. 거기에 더해 예상과 달리 경량화 모델인 KULLM2 5.8b 모델조차도 Google Colab의 T4 GPU 무료 환경에서는 처리하기 버거운 상황이었다.

-> 이러한 제약 사항을 해결하기 위해, KULLM3의 이전 버전인 KULLM2의 5.8b 모델을 대안으로 선정하고 Google Colab Pro의 유료 버전을 사용하여 A100 GPU 환경에서 언어 모델을 처리하기로 결정하였다.

1.2.2. 대화를 통해 받은 사용자의 정보를 토대로 차량을 추천해주고 사용자가 적은 량의 정보만을 제공해 준 경우 추가적인 정보 제공을 요청하는 기능을 정확하게 pre - trained model인 KULLM 3에 구현하는 것은 google의 유료 챗봇 서비스인 dialogflow의 내부 기능인 slot filling을 이용하는 방법 외에는 기술적 어려움이 생겼다.

-> 크롤링 된 데이터 안의 모든 정보들을 이용하는 것이 아닌 주행거리, 연식, 브랜드등과 같이 중요한 부분만 집어서 이러한 정보들이 빠진 예시를 input과 output 형식으로 모델에 학습시킨다.

1.2.3. react.js 개발 전 웹페이지에서 버튼 및 상호작용 동작 확인이 어렵다.

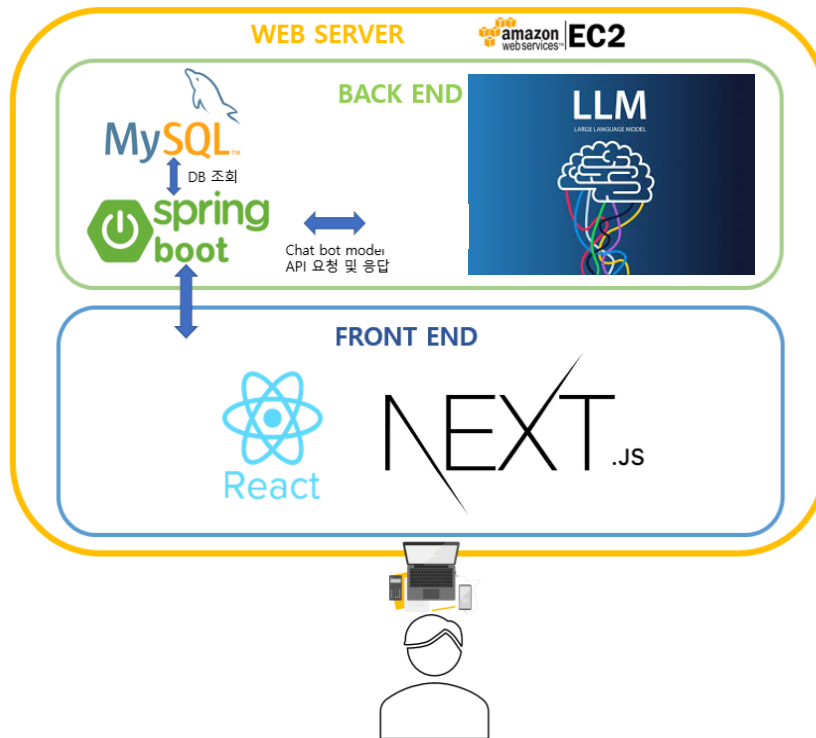
-> 우선적으로 html 파일 생성 후 페이지간 링크 및 동작 확인 진행, 차후 해당 디자인을 바탕으로 tsx 파일 생성 예정이다.

## 2. 설계 상세화 및 변경 내역

### 2.1. 중고차 데이터셋 크롤링

- 보배드림 사이트에서 크롤링을 통해서 제조사별로 중고차 데이터셋을 확보하였으나 결측값, 부정확한 값, 자동차의 총 가격 대신 월 납입금이 기재되어있거나 상세 가격 대신 상담이 기재되어있는 등의 문제가 있는 데이터가 다수 존재한다. 이러한 데이터들은 모델 fine tuning시 성능을 저하시키는 요인이 되므로 계속해서 꼼꼼히 확인하면서 결측값이 너무 많은 경우에는 열 삭제를 하고 가격이 부정확한 경우에는 비슷한 연식과 키로수를 가진 동일한 매물들의 가격의 평균이나 중앙값으로의 대체를 통한 수정을 계속해 나갈 예정이다.
- 중고차 매물 추천 서비스 외에도 사용자가 중고차 구매 과정에서 생기는 여러가지 문제(중고차 구매 시 절차, 취득등록세와 공채매입비 등의 수수료, 중고차 구매 시 필요 서류)에 대한 답변을 위해 관련 정보들을 200토큰 이하의 질문과 답변 형식으로 쪼갬다. 그리고 이를 시드데이터로 하여 OpenAI의 API를 통해 gpt 3.5-turbo 모델로 self instruct를 진행하여 데이터 2151개를 만들어 내었다. 추후 중고차 관련 데이터들을 계속 수집해서 추가적인 데이터를 만들어 낼 예정이다.

## 2.2. 모델 학습 및 서버



LLM 모델	베이스 모델 KULLM3에 input, output 형식으로 만들어진 5000개 이상의 데이터를 fine tuning 하여 학습시킨 뒤 LoRA(Low Rank Adaption), RLHF(Reinforcement Learning with Human Feedback) 방식을 사용하여 모델의 성능을 강화한다.
프론트엔드(웹 어플리케이션)	웹 어플리케이션으로 서비스되며 접속했을 때 로그인 시에는 이전에 저장했던 정보를 토대로, 비로그인시에는 새로운 정보만을 토대로 채팅 형식으로 사용자에게 중고차 또는 중고차 구매에 필요한 정보를 알려준다.
챗봇API서버	모델을 오픈소스인text-generation-webui에서 제공하는 websocket api 서버 생성기능을 사용하여 챗봇API 서버를제작
클라우드 서비스	AWS EC2 서버를 사용하며 전체 시스템을 호스팅한다.
데이터베이스	DBMS로 MySQL을 사용하고 백엔드 서버에서는 spring JPA를 사용하여 접근한다. spring boot 프레임워크를 사용하고 하위 프레임워크로 spring JPA, spring security로 db와 인증을 관리한다. 사용자의 모든 대화와 로그인 정보는 db에 저장된다. 이전 대화를 불러올 수 있고 이어서 대화가 가능하다.

### 3. 갱신된 과제 추진 계획

7월					8월					9월					10월	
1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2
언어 모델 선정 및 테스트																
		DB 설계														
			UI 디자인													
				백엔드 환경 구축												
				데이터 크롤링 및 전처리												
					OpenAI 데이터 수집											
						중간 보고서										
							fine tuning 모델 학습									
									백엔드 개발							
										프론트 개발						
												배포 테스트 및 수정				
														최종 보고서 및 최종 발표		

## 4. 구성원별 진척도

이름	역할
이은진	<ul style="list-style-type: none"><li>언어 모델 선정 및 테스트</li><li>챗봇 웹페이지 UI 디자인 설계</li><li>OpenAI를 활용한 데이터 수집</li></ul>
최희웅	<ul style="list-style-type: none"><li>언어 모델 선정 및 테스트</li><li>데이터 크롤링 및 전처리</li><li>OpenAI를 활용한 데이터 수집</li></ul>
최성빈	<ul style="list-style-type: none"><li>언어 모델 선정 및 테스트</li><li>로컬에서 챗봇 웹페이지 UI 구현</li><li>백엔드(스프링부트) 개발환경구축</li><li>데이터베이스 설계</li></ul>

## 5. 보고 시점까지의 과제 수행 내용 및 중간 결과

### 5.1. 언어 모델 선정 및 테스트

#### 문제 배경

프로젝트의 초기 계획에서는 최신 언어 모델인 KULLM3를 채택하여 개발을 진행할 예정이었다. KULLM3는 GPT-4 Turbo와 비슷한 성능을 보여, 기존의 한국어 모델보다 우수한 생성 능력을 보인다. 그러나, 개발 환경의 한계로 인해 대용량의 파라미터를 가지는 KULLM3를 언어 모델로 사용하기 어려워졌다.

#### 대안 모델 선택

이러한 제약 사항을 해결하기 위해, KULLM3의 이전 버전인 KULLM2의 5.8b 모델을 대안으로 선정하였다. 다른 언어 모델을 사용할 수도 있었지만, 이전 버전이라도 원래 사용하려던 KULLM 언어 모델을 한번 사용해보는 것이 좋겠다고 판단하여 이 언어 모델을 선택하게 되었다.

#### 개발 환경 조성

예상과 달리 KULLM2 5.8b 모델조차도 Google Colab의 T4 GPU 무료 환경에서는 처리하기 버거운 상황이었다. 따라서, Google Colab Pro의 유료 버전을 사용하여 A100 GPU 환경에서 언어 모델을 처리하기로 결정하였다. 활용하기로 결정하였다.

```

1 #KULLM2_5.8b model
2
3 import locale
4 import torch
5 from transformers import AutoModelForCausalLM, AutoTokenizer, pipeline
6 from utils.prompter import Prompter
7
8 locale.getpreferredencoding = lambda: "UTF-8"
9
10 MODEL = "nlpai-lab/kullm-polyglot-5.8b-v2"
11
12 model = AutoModelForCausalLM.from_pretrained(
13     MODEL,
14     torch_dtype=torch.float16,
15     low_cpu_mem_usage=True,
16 ).to(device=f"cuda", non_blocking=True)
17
18 model.eval()
19
20 pipe = pipeline("text-generation", model=model, tokenizer=MODEL, device=0)
21 prompter = Prompter("kullm")
22
23 def infer(instruction="", input_text=""):
24     prompt = prompter.generate_prompt(instruction, input_text)
25     output = pipe(prompt, max_length=512, temperature=0.2, num_beams=5,
26 eos_token_id=2)
27     s = output[0]["generated_text"]
28     result = prompter.get_response(s)
29
30     return result
31
32 result = infer(input_text="0000 0000 000 000")
33 print(result)

```

[그림 - KULLM2 5.8b 실행 코드]

## 언어 모델 테스트

KULLM2 5.8b 언어 모델에 간단한 테스트를 진행하였다.

질문	한국에서 잘 팔리는 중고차 차종에는 무엇이 있어?
답변	<p>한국에서 잘 팔리는 중고차 차종에는 다음과 같은 것들이 있습니다:</p> <p>1. 아반떼 (Avante) 2. 소나타 (Sonata) 3. 그랜저 (Grandeur) 4. 싼타페 (Santa Fe)  5. 투싼 (Tucson) 6. 스포티지 (Sportage) 7. K3 (K3 Sedan) 8. K5 (K5 Sedan)  9. K7 (K7 Sedan) 10. K9 (K9 Sedan)</p>
질문	중고차 시세를 알고싶어.
답변	<p>중고차 시세를 확인하려면 다음 단계를 따르세요:</p> <p>1. Google에서 "cars.com" 또는 "cars.co.uk"과 같은 인기 있는 웹사이트를 검색합니다.</p>



	<p>2. 웹사이트의 검색창에 차량의 이름과 연식을 입력합니다.</p> <p>3. 검색 결과에서 해당 차량의 가격을 확인합니다.</p> <p>4. 가격이 마음에 들면 해당 웹사이트에서 차량을 찾습니다.</p> <p>5. 웹사이트에서 차량의 상태, 연식, 주행 거리 등 차량에 대한 자세한 정보를 확인합니다.</p> <p>6. 차량의 상태가 양호하고 가격이 합리적인지 확인합니다.</p> <p>중고차 시세는 시간이 지남에 따라 변경될 수 있으므로 항상 최신 정보를 확인하는 것이 좋습니다.</p>
--	--

질문	중고차를 구매하는 방법을 알려줘
답변	<p>중고차를 구매하는 방법에는 여러 가지가 있습니다. 다음은 몇 가지 일반적인 단계입니다:</p> <ol style="list-style-type: none"> <li>1. 예산 설정: 중고차를 구매하는 첫 번째 단계는 예산을 설정하는 것입니다. 중고차의 가격 범위를 알고 있어야 예산에 맞는 차량을 찾을 수 있습니다.</li> <li>2. 차량 검색하기: 예산을 설정했으면 다음 단계는 원하는 차량을 검색하는 것입니다. 여기에는 웹사이트, 앱 또는 자동차 딜러를 방문하는 것이 포함될 수 있습니다.</li> <li>3. 차량 평가하기: 차량을 검색한 후에는 차량의 상태와 성능을 평가해야 합니다. 여기에는 차량의 연식, 색상, 주행거리, 사고 여부 등이 포함될 수 있습니다.</li> <li>4. 판매자와 협상하기: 차량을 평가한 후에는 판매자와 협상을 시작할 수 있습니다. 판매자가 제시하는 가격이 합리적인지, 차량의 상태가 좋은지 확인하세요.</li> <li>5. 계약서에 서명하기: 판매자와 협상이 완료되면 계약서에 서명할 수 있습니다. 계약서에는 차량의 상태에 대한 정보와 보증이 포함되어 있어야 합니다.</li> <li>6. 차량 인수하기: 계약서에 서명한 후에는 차량을 인수하는 절차를 시작할 수 있습니다. 여기에는 차량의 상태를 확인하고, 필요한 모든 서류를 준비하고, 보험에 가입하는 것이 포함될 수 있습니다.</li> </ol> <p>중고차를 구매하는 것은 복잡한 과정일 수 있지만, 올바른 방법을 따르면 합리적인 가격에 좋은 차량을 찾을 수 있습니다.</p>

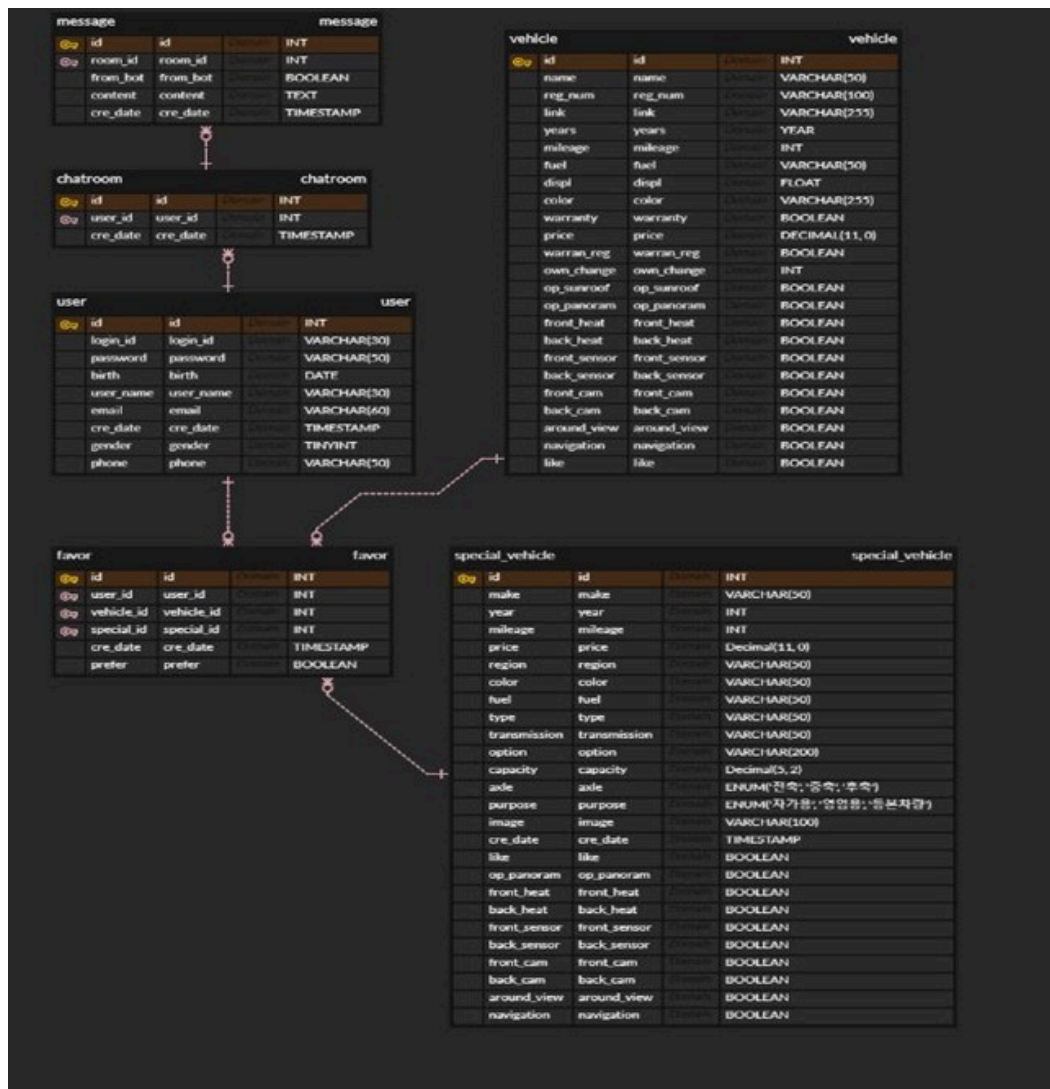
## 5.2. DB 디자인 설계

해당 서비스에서 DB에 저장하는 정보는 사용자의 개인정보, 채팅내역, 차량에 대한 관심도이다.

사용자는 여러 채팅방을 만들어서 이용할 수 있고 각 채팅방에는 사용자와 챗봇이 주고 받은 여러 메시지가 존재할 수 있다.

각각의 메시지는 사용자와 챗봇 중 어디에서 유래하였는지 구분 할 수 있으며, 생성된 시간 순서로 정렬된다.

사용자의 중고차 선호도는 경우 favor 테이블을 기준으로 M:N 관계를 형성한다. 사전 정의된 차량 정보들을 보관하는 vehicle, special\_vehicle 테이블이 존재하며 사용자는 선호하는 차량을 선택하여 선호도를 저장할 수 있다.



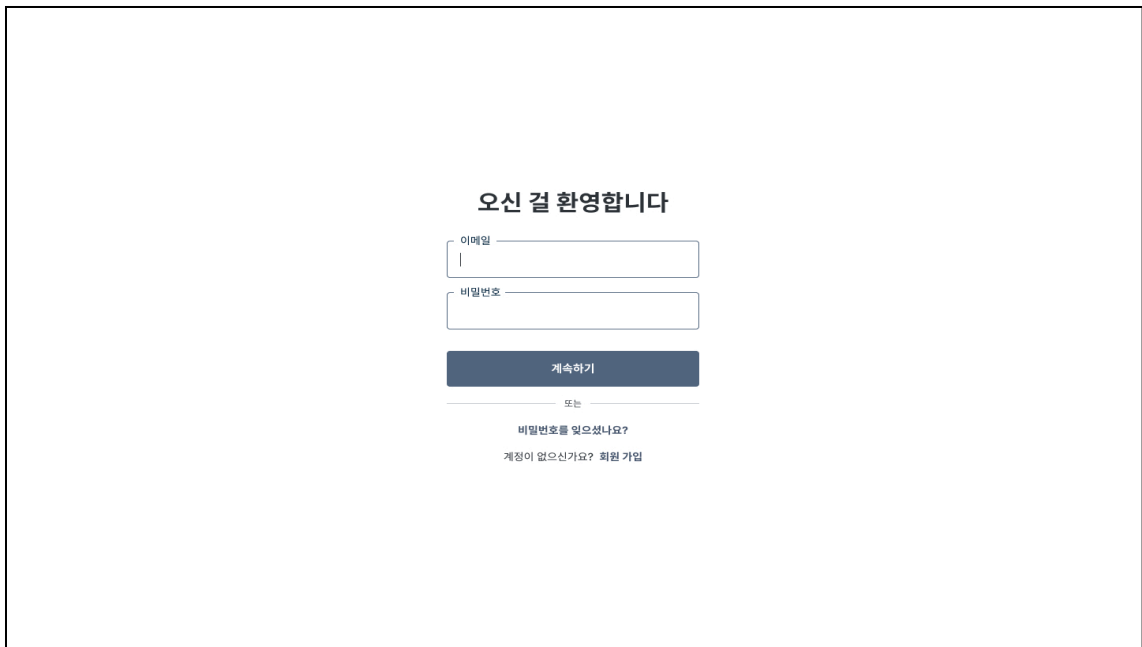
[그림 - 데이터베이스 디자인]

## 5.3. 챗봇 웹 UI 디자인

Figma 프로그램을 활용하여 챗봇의 로그인, 회원 가입, 메인 화면 UI를 디자인 하였다.

### 로그인 화면

- 구성 요소:
  - 이메일 입력 필드
  - 비밀번호 입력 필드
  - 로그인(계속하기) 버튼
  - 비밀번호 찾기 링크
  - 회원 가입 링크

A login form UI design centered on a white background. At the top, the text '오신 걸 환영합니다' (Welcome) is displayed. Below it are two input fields: '이메일' (Email) and '비밀번호' (Password). A dark blue button with the text '계속하기' (Continue) is positioned below the password field. Underneath the button is a horizontal line, followed by the text '또는' (or), and then two lines of smaller text: '비밀번호를 잊으셨나요?' (Forgot your password?) and '계정이 없으신가요? 회원 가입' (Don't have an account? Sign up).

[그림 - 로그인 UI]

### 회원 가입 화면

- 구성 요소:
  - 이메일 입력 필드
  - 비밀번호 입력 필드
  - 이름 입력 필드
  - 생일 입력 필드
  - 전화 번호 입력 필드
  - 회원 가입(계속하기) 버튼
  - 로그인 링크

### 계정 만들기

이메일

비밀 번호

이름

생일  
년도

월

일

전화 번호

계속하기

[또는](#)  
 이미 계정이 있으신가요? [로그인](#)

[그림 - 회원 가입 UI]

## 메인 화면

- 구성 요소:
  - 챗봇과의 이전 채팅방
  - 질문 입력 필드
  - 질문 전송 버튼
  - 로그인/로그아웃 버튼
  - 마이페이지 버튼

📅
오늘

8월 13일의 채팅...
✎ 🗑

📅
지난 3일

8월 12일의 채팅...
✎ 🗑

8월 11일의 채팅...
✎ 🗑

📅
지난 7일

8월 9일의 채팅...
✎ 🗑

8월 8일의 채팅...
✎ 🗑

📅
지난 30일

8월 1일의 채팅...
✎ 🗑

7월 30일의 채팅...
✎ 🗑

7월 25일의 채팅...
✎ 🗑

🗑
채팅 비우기

✎
문의하기

➡
로그아웃

CarTalk

[로그인](#)
[회원 가입](#)

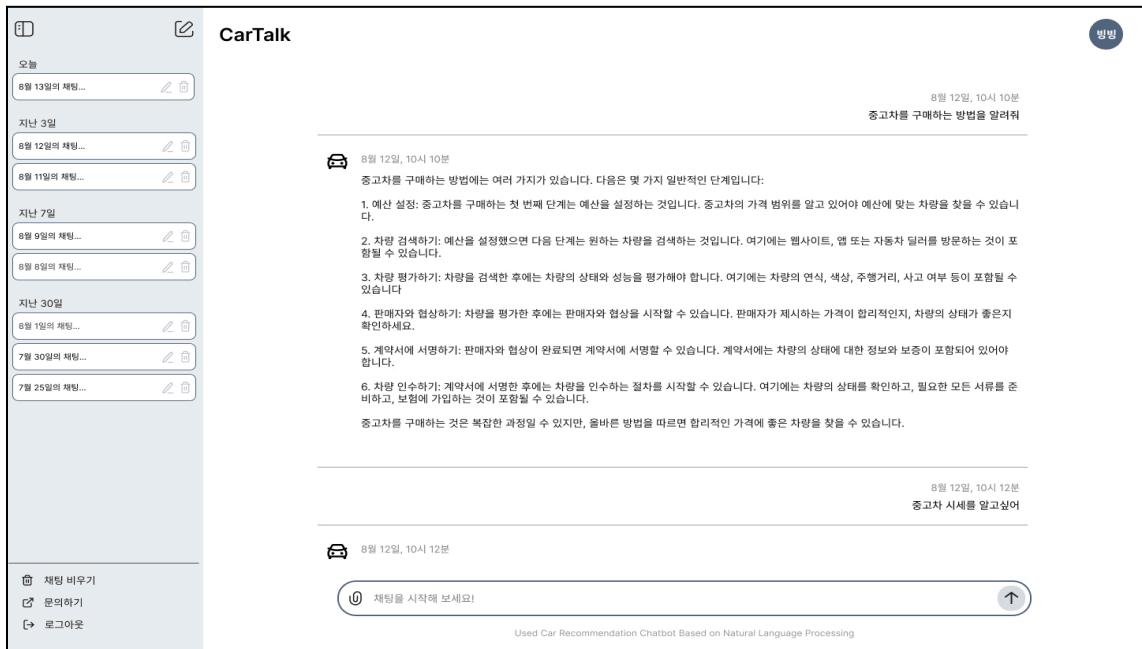
## Discover the best cars for you & your lifestyle

Find your perfect car with our smart recommendations.

🗑 채팅을 시작해 보세요!

Used Car Recommendation Chatbot Based on Natural Language Processing

[그림 - main UI]



[그림 - main UI]

## 5.4. 학습 데이터 모으기 - crawling, 전처리

### 5.4.1 crawling

Python의 'requests'와 'Beautifulsoup' 라이브러리를 사용하여 웹 페이지의 HTML을 파싱하고, 중고차정보를 추출한 뒤, 이를 Pandas라이브러리를 통해 DataFrame으로 변환하여 CSV 파일로 출력했다.

```

import requests
from bs4 import BeautifulSoup
import pandas as pd
totalurl=[
    ("https://www.bobaedream.co.kr/mycar/mycar_list.php?gubun=i&maker_no=13&group_no=201&page=1&order=S11&view_size=20",1)]

df_cars=[]

urls=[]
for i in totalurl:
    pagenum=i[1]
    for j in range(pagenum):
        url=i[0].format(str(j))
        urls.append(url)

info=["이름", "차량번호", "링크", "연식", "주행거리", "연료", "배기량", "색상", "보증정보", "가격", "신차대비가격"]

coloptions=["음션_선루프", "음션_파노라마선루프", "음션_열선앞", "음션_열선뒤", "음션_전방센서",
            "음션_후방센서", "음션_전방캠", "음션_후방캠", "음션_여라운드뷰", "음션_내비순정"]

findoptions=["선루프", "파노라마선루프", "열선시트(앞좌석)", "열선시트(뒷좌석)", "전방센서",
            "후방센서", "전방카메라", "후방카메라", "여라운드뷰", "네비게이션(순정)"]
colacci_info=["보험이력등록", "소유자변경횟수", "사고상세_전순", "사고상세_침수전순", "사고상세_침수분순", "사고상세_도난", "보험_내차피해(횟수)", "보험_내차피해(가

cols=info+coloptions+colacci_info
#링크추가: 아웃라이어인지 확인하기 위해

#음션이름 받아서 있는지 있는지 확인
def option_check(soupobject,option_name):
    check = soupobject.find("button", text=option_name).find_parent().find_previous_sibling().get_attribute_list('checked')

    if check[0]=='':
        return '유'
    else:
        return '무'

```

[그림 - crawling 코드1]

option\_check 함수는 HTML 구조를 기반으로 특정 옵션의 존재여부를 '유' 또는 '무'로 반환한다.

```
for url in urls:
    res=requests.get(url)
    res.raise_for_status()
    requests.adapters.DEFAULT_RETRIES = 100000
    soup=BeautifulSoup(res.text,"lxml")

    cars=soup.find_all("li",attrs={"class":"product-item"})
    links=[]
    #한 url마다 들어있는 모든 차들에 대해 실행
    for car in cars:
        link = "https://www.bobaedream.co.kr" + car.a["href"]
        links.append(link)
    for link in links:
        print(link)

    res2=requests.get(link,timeout=5)
    res2.raise_for_status()
    soup2 = BeautifulSoup(res2.text, "lxml")
    infobox = soup2.find("div", attrs={"class": "info-util box"})
    try:
        ratiopr = infobox.find("b")
    except:
        continue

    name=soup2.find("h3",attrs={"class":"tit"})
    state=soup2.find("div",attrs={"class":"tbl-01 st-low"})
    galdata=soup2.find("div",attrs={"class":"gallery-data"})
    carnumber=galdata.find("b")

    year=state.find("th",text='연식').find_next_sibling("td")
    km=state.find("th",text='주행거리').find_next_sibling("td")
    fuel=state.find("th",text='연료').find_next_sibling("td")
    amount=state.find("th",text='배기량').find_next_sibling("td")
    color=state.find("th",text='색상').find_next_sibling("td")
    guarn=state.find("b",text='보증정보').find_next("td")
    price=soup2.find("span",attrs={"class":"price"})
```

[그림 - crawling 코드2]

urls 리스트의 각 URL에 대해 웹페이지 요청을 보내 해당 페이지에 있는 모든 차량의 상세 페이지 링크를 추출한다. 추출한 정보는 다음과 같다.

- 차량 기본 정보 : 이름, 차량번호, 연식, 주행거리, 연료, 배기량, 색상, 보증정보, 가격, 신차대비가격, 옵션, 선루프, 옵션 파노라마선루프, 옵션 열선 시트(앞좌석 및 뒷좌석), 전방/후방 센서, 전방/후방 카메라, 어라운드뷰, 네비게이션(순정)
- 보험 이력 정보: 보험 등록 여부, 소유자 변경 횟수, 사고 이력, 보험료 액수

```
이름, 차량번호, 링크, 연식, 주행거리, 연료, 배기량, 색상, 보증정보, 가격, 신차대비가격, 옵션, 선루프, 옵션 파노라마선루프, 옵션 열선 시트, 옵션 열선 시트, 옵션 전방센서, 옵션 후방센서,
9, 현대 그랜스타렉스 2.5 디젤 벤 3인승 2WD CVX 럭셔리, 차량번호 84고5130, https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2233564&gubun=K, 2011.08.
1, 현대 그랜스타렉스 2.5 디젤 벤 5인승 2WD CVX 럭셔리, 차량번호 82가9617, https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2230653&gubun=K, 2011.04.
2, 현대 더 뉴 그랜스타렉스 2.5 디젤 벤 5인승 모던, 차량번호 837부1201, https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2231580&gubun=K, 2018.10.
3, 현대 스타렉스 캠핑카, 차량번호 76무3495, https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2226233&gubun=K, 2006.06, "190,000 km", 디젤, "2,476 cc ()"
4, 현대 그랜스타렉스 캠핑카, 차량번호 76가6251, https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2227392&gubun=K, 2008.03, "87,458 km", 디젤, "2,497 cc"
5, 현대 뉴스타렉스 캠핑카, 차량번호 75부4997, https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2227727&gubun=K, 2005.04, "190,000 km", 디젤, "2,476 cc ()"
6, 현대 그랜스타렉스 어린이보호차 디젤 12인승 2WD, 차량번호 73저6252, https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2221524&gubun=K, 2012.06, "147,
7, 현대 그랜스타렉스 2.5 디젤 벤 5인승 2WD CVX 프리미엄, 차량번호 97소5633, https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2230576&gubun=K, 2012.
8, 현대 더 뉴 그랜스타렉스 2.5 디젤 벤 3인승 스타일, 차량번호 82타6421, https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2228156&gubun=K, 2018.08.
9, 현대 그랜스타렉스 2.5 디젤 벤 3인승 2WD CVX 디럭스, 차량번호 81우9883, https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2230147&gubun=K, 2015.07.
10, 현대 더 뉴 그랜스타렉스 2.5 디젤 벤 5인승 스마트, 차량번호 91무1474, https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2231253&gubun=K, 2018.04,
11, 현대 그랜스타렉스 2.5 디젤 벤 5인승 2WD CVX 프리미엄, 차량번호 87로3258, https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2231928&gubun=K, 2013.
12, 현대 더 뉴 그랜스타렉스 2.5 디젤 3인승 스마트, 차량번호 87마0370, https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2232228&gubun=K, 2018.09.
13, 현대 그랜스타렉스 어린이보호차 디젤 12인승 2WD, 차량번호 702부7870, https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2226408&gubun=K, 2015.08, "15
14, 현대 그랜스타렉스 캠핑카, 차량번호 981가2396, https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2207385&gubun=K, 2012.06, "94,157 km", 디젤,
15, 현대 그랜스타렉스 2.5 디젤 왜건 12인승 2WD 스마트, 차량번호 72무9086, https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2227722&gubun=K, 2017.6
16, 현대 그랜스타렉스 2.5 디젤 왜건 12인승 2WD 스마트, 차량번호 71서8695, https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2232109&gubun=K, 2017.6
17, 현대 그랜스타렉스 2.5 디젤 벤 3인승 2WD CVX 럭셔리, 차량번호 83누8035, https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2231812&gubun=K, 2013.0
18, 현대 그랜스타렉스 2.5 디젤 왜건 12인승 2WD 스마트, 차량번호 70타8754, https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2232050&gubun=K, 2017.01
```

[그림 - crawling된 데이터]

## 5.4.2 전처리

전처리 과정에서는 데이터의 일관성을 높이고 모델 학습에 적합한 형식으로 데이터를 변환하였다. 차량의 이름과 배기량 거리 등의 데이터에서 불필요한 공백 및 문자를 제거하여 학습에 쉬운 형식으로 변환하고 연료 타입, 옵션, 보험 정보 등에 매핑 딕셔너리를 적용하여 모델이 학습하기 좋은 숫자 값으로 변환하였다.

```
# 함수 정의
def changemodel(str1):
    index1 = str1.find(' ')
    index2 = str1.find(' ', index1 + 1)
    index3 = str1.find(' ', index2 + 1)
    return str1[:index3]

def engine(str1):
    index = str1.find('cc')
    return str1[:index].replace(',', '').rstrip()

def guar(str1):
    if str1 == '만료' or str1 == '불가':
        return pd.Series([0, 0, 0])
    elif str1 == '정보없음':
        return pd.Series([np.nan, np.nan, np.nan])
    else:
        index = str1.find('/')
        if index == -1:
            time = 0
        else:
            time = int(str1[:index-3].strip())
        km = str1[index+1:].replace(",", "")[:2].strip()
        return pd.Series([1, time, km])

# 매핑 딕셔너리
matchop = {'무': 0, '유': 1}
insurinfo = {'미등록': 0, '등록': 1}
matchfuel = {'디젤': 0, 'LP6': 1, '가솔린': 2, '전기': 3}

# 전처리
df_cars['이름'] = df_cars['이름'].apply(changemodel)
df_cars['연식'] = df_cars['연식'].apply(lambda x: x[:4] + x[5:7])
df_cars['주행거리'] = df_cars['주행거리'].apply(lambda x: x.replace(",","")[:-2].rstrip())
df_cars['연료'] = df_cars['연료'].replace(matchfuel)
df_cars['배기량'] = df_cars['배기량'].apply(engine)

# 보증정보 처리 및 병합
guartable = df_cars['보증정보'].apply(guar)
guartable.columns = ['보증여부', '보증기간', '보증거리']
df_cars = pd.concat([df_cars, guartable], axis=1)
df_cars = df_cars.drop(['보증정보'], axis=1)

# 매핑 적용
df_cars = df_cars.replace(matchop)
df_cars['보험이력등록'] = df_cars['보험이력등록'].replace(insurinfo)
df_cars['보험_내차피해(가격)'] = df_cars['보험_내차피해(가격)'].apply(lambda x: x.replace(",","") if isinstance(x, str) else np.nan)
```

[그림 - 전처리 코드1]

```
# 매핑 딕셔너리
matchop = {'무': 0, '유': 1}
insurinfo = {'미등록': 0, '등록': 1}
matchfuel = {'디젤': 0, 'LP6': 1, '가솔린': 2, '전기': 3}

# 전처리
df_cars['이름'] = df_cars['이름'].apply(changemodel)
df_cars['연식'] = df_cars['연식'].apply(lambda x: x[:4] + x[5:7])
df_cars['주행거리'] = df_cars['주행거리'].apply(lambda x: x.replace(",","")[:-2].rstrip())
df_cars['연료'] = df_cars['연료'].replace(matchfuel)
df_cars['배기량'] = df_cars['배기량'].apply(engine)

# 보증정보 처리 및 병합
guartable = df_cars['보증정보'].apply(guar)
guartable.columns = ['보증여부', '보증기간', '보증거리']
df_cars = pd.concat([df_cars, guartable], axis=1)
df_cars = df_cars.drop(['보증정보'], axis=1)

# 매핑 적용
df_cars = df_cars.replace(matchop)
df_cars['보험이력등록'] = df_cars['보험이력등록'].replace(insurinfo)
df_cars['보험_내차피해(가격)'] = df_cars['보험_내차피해(가격)'].apply(lambda x: x.replace(",","") if isinstance(x, str) else np.nan)
```

[그림 - 전처리 코드2]

```
이름,차량번호,링크,연식,주행거리,연료,배기량,색상,가격,옵션_전부,옵션_파노라마선루프,옵션_열선앞,옵션_열선뒤,옵션_전방센서,옵션_후방센서,옵션_전방캠,옵션_후
현대 그랜저스타렉스 2.5,차량번호 84고5130,https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2233564&gubun=K,201108,195981,0,2497,은색,499만원,
현대 그랜저스타렉스 2.5,차량번호 82가9617,https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2230653&gubun=K,201104,184671,0,2497,검정색,379만원,
현대 더 뉴,차량번호 837부1201,https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2231580&gubun=K,201810,101079,0,2497,남색,1,550만원,0.0,0.0,1.0
현대 스타렉스 캠핑카,차량번호 76우3495,https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2226233&gubun=K,200606,190000,0,2476,흰색투톤,1,399만원
현대 그랜저스타렉스 캠핑카,차량번호 76기6251,https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2227392&gubun=K,200803,87458,0,2497,은색,1,900만원
현대 뉴스타렉스 캠핑카,차량번호 75비4997,https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2227727&gubun=K,200504,190000,0,2476,흰색투톤,1,399만원
현대 그랜저스타렉스 어린이보호차,차량번호 73저6252,https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2221524&gubun=K,201206,147000,0,2497,노란색,53
현대 그랜저스타렉스 2.5,차량번호 97소5633,https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2230576&gubun=K,201211,154000,0,2497,은색,529만원,0.0,
현대 더 뉴,차량번호 82터6421,https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2228156&gubun=K,201808,122286,0,2497,청색,1,250만원,1,
현대 그랜저스타렉스 2.5,차량번호 81우9883,https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2230147&gubun=K,201507,263566,0,2497,은색,660만원,
현대 더 뉴,차량번호 91무1474,https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2231253&gubun=K,201804,73005,0,2497,은색,1,490만원,1,3
현대 그랜저스타렉스 2.5,차량번호 87로3258,https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2231928&gubun=K,201310,253000,0,2497,은색,550만원,
현대 더 뉴,차량번호 87마0370,https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2232228&gubun=K,201809,105326,0,2497,은색,1,360만원,1,
현대 그랜저스타렉스 어린이보호차,차량번호 702누7870,https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2226408&gubun=K,201508,156190,0,2497,노란색,6
현대 그랜저스타렉스 ,차량번호 981가2396,https://www.bobaedream.co.kr/mycar/mycar_view.php?no=2207385&gubun=K,201206,94157,0,2497,회색,2,380만원,0.0
```

[그림 - 전처리된 데이터]







## GPT-3.5 Turbo API를 활용한 데이터 생성

시드 데이터를 기반으로 더 많은 학습 데이터를 생성하기 위해 OpenAI의 GPT-3.5 Turbo API를 활용하여 학습에 사용할 질문과 답변 쌍을 자동으로 생성하였다. GPT-3.5 Turbo는 간단한 작업에 특화되어 있고, 가격이 저렴하기에 이를 택하였다. 이 API를 통해, 151개의 시드 데이터에 약 10달러의 비용으로 총 2000개의 데이터를 생성하였다.

### 프롬프트 구성

사용된 프롬프트 파일(prompt.txt)은 다양한 지침을 포함하여 GPT-3.5 Turbo가 학습 데이터 생성 작업을 수행할 수 있도록 설계되었다.

```
2 You are asked to come up with a set of 20 diverse task instructions. These task instructions will be given to a GPT model and we will evaluate the GPT model.
3
4 Here are the requirements:
5 1. Try not to repeat the verb for each instruction to maximize diversity.
6 2. The language used for the instruction also should be diverse. For example, you should combine questions with imperative instructions.
7 3. The type of instructions should be diverse. The list should include diverse types of tasks like open-ended generation, classification, editing, etc.
8 4. A GPT language model should be able to complete the instruction. For example, do not ask the assistant to create any visual or audio output. For another example, do not ask the assistant to perform a task that requires a specific tool or API.
9 5. The instructions should be in Korean.
10 6. The instructions should be 1 to 2 sentences long. Either an imperative sentence or a question is permitted.
11 7. You should generate an appropriate input to the instruction. The input field should contain a specific example provided for the instruction. It should involve the instruction.
12 8. Not all instructions require input. For example, when a instruction asks about some general information, "what is the highest peak in the world", it is not necessary to provide an input.
13 9. The output should be an appropriate response to the instruction and the input. Make sure the output is less than 1000 words.
14 11. The instructions and outputs should include stories related to cars (especially used cars). For example, information on used cars, recommendations for used cars, etc.
15
16 List of 20 tasks:
```

[그림 - prompt 구성]

사용된 주요 코드 일부는 다음과 같다.

### encode\_prompt 함수

```
19 def encode_prompt(prompt_instructions):
20     prompt = open("generate_instance/prompt.txt").read() + "\n"
21
22     for idx, task_dict in enumerate(prompt_instructions):
23         (instruction, input, output) = task_dict["instruction"], task_dict["input"], task_dict["output"]
24         instruction = re.sub(r"\s+", " ", instruction).strip().rstrip(":")
25         input = "<noinput>" if input.lower() == "" else input
26         prompt += f"###\n"
27         prompt += f"{idx + 1}. Instruction: {instruction}\n"
28         prompt += f"{idx + 1}. Input:\n{input}\n"
29         prompt += f"{idx + 1}. Output:\n{output}\n"
30     prompt += f"###\n"
31     prompt += f"{idx + 2}. Instruction: "
32     return prompt
```

[그림 - encode\_prompt 함수 코드]

encode\_prompt 함수는 GPT-3.5 Turbo 모델에 전달할 프롬프트를 구성하는 역할을 한다. 이 함수는 여러 개의 지침(instructions), 입력(input), 출력(output) 데이터가 포함된 리스트를 받아 이를 하나의 텍스트 프롬프트로 인코딩한다. 이 함수는 데이터 생성을 위한 핵심적인 역할을 하며, GPT 모델이 이해하고 수행할 수 있는 형식의 프롬프트를 생성한다.

## make\_request 함수

```
5 def make_requests(  
6     model, messages, max_tokens, temperature, top_p,  
7     frequency_penalty, presence_penalty, stop,  
8     retries=3, api_key=None, organization=None  
9 ):  
10  
11     client = OpenAI(api_key=api_key)  
12  
13     retry_cnt = 0  
14  
15     formatted_messages = [{"role": "system", "content": "you're the best making {instruction, input, output} data set and the best assistant"}] + \  
16     [{"role": "user", "content": message} for message in messages]  
17     results = []  
18     while retry_cnt <= retries:  
19         try:  
20             response = client.chat.completions.create(  
21                 model=model,  
22                 messages=formatted_messages,  
23                 max_tokens=max_tokens,  
24                 temperature=temperature,  
25                 top_p=top_p,  
26                 frequency_penalty=frequency_penalty,  
27                 presence_penalty=presence_penalty,  
28                 stop=stop  
29             )  
30             results = [choice.message.content for choice in response.choices]  
31             break  
32         except Exception as e:  
33             print("Unexpected error occurred")  
34             print(e)  
35             break  
36     return results
```

[그림 - make\_request 함수 코드]

make\_requests 함수는 OpenAI의 GPT-3.5 Turbo API에 프롬프트를 보내고 응답을 받는 역할을 한다. 이 함수는 모델과의 상호작용을 담당하며, 주어진 프롬프트에 따라 다양한 자연어 텍스트를 생성한다. 성공적으로 응답을 받으면, 함수는 생성된 텍스트를 리스트로 반환한다.

## main 함수

main은 전체 데이터 생성 과정을 관리하는 역할을 하며, 다음과 같은 단계로 구성된다.

### 1)데이터 로딩 및 준비

먼저, 시드 데이터가 로드되고 초기 프롬프트 지침이 준비된다. 시드 데이터는 고유한 학습 데이터로, 모델이 새로운 데이터를 생성하는 데 기준이 되는 정보이다.

### 2)요청 수행

make\_requests 함수를 호출하여 GPT-3.5 Turbo 모델에 프롬프트를 전달하고, 새로운 데이터를 생성한다. 각 요청에 대해 모델이 생성한 응답은 후처리 과정을 통해 필터링된다. 이 과정에서 유효하지 않거나 잘못된 형식의 응답은 제거되고, 최종적으로 남은 텍스트만이 학습 데이터로 사용된다.

### 3) 유사도 검증

생성된 데이터가 기존 데이터와 지나치게 유사하지 않도록 하기 위해 유사도 검증 과정을 거친다. 이를 통해 데이터의 다양성과 유효성을 보장한다. 유사도 검증은 ROUGE 스코어를 기반으로 하며, 새롭게 생성된 텍스트와 기존 텍스트 간의 유사도를 계산한다.

만약 유사도가 너무 높으면, 해당 데이터는 버려지고, 새로운 데이터가 생성되거나 추가 요청을 통해 더 다양한 데이터를 확보하게 된다. 이 과정을 통해 모델이 생성한 데이터가 중복되지 않고, 새로운 학습 데이터를 풍부하게 확보할 수 있게 된다.

```
137 if __name__ == "__main__":
138     num_cpus = 4
139     num_machine_instructions = 2
140
141     args = parse_args()
142     seed_tasks = [json.loads(l) for l in open(args.seed_tasks_path, "r", encoding="utf-8")]
143     seed_instructions = [
144         {"instruction": t["instruction"], "input": t["instances"][0]["input"], "output": t["instances"][0]["output"]}
145         for t in seed_tasks
146     ]
147     print(f"Loaded {len(seed_instructions)} human-written seed instructions")
148
149     os.makedirs(args.batch_dir, exist_ok=True)
150     request_idx = 0
151
152     machine_instructions = []
153
154     if os.path.exists(os.path.join(args.batch_dir, "machine_generated.jsonl")):
155         with open(os.path.join(args.batch_dir, "machine_generated.jsonl"), "r", encoding="utf-8") as fin:
156             for l in fin:
157                 tmp = json.loads(l)
158                 machine_instructions.append(tmp)
159             print(f"Loaded {len(machine_instructions)} machine-generated instructions")
160
161     scorer = rouge_scorer.RougeScorer(["rougeL"], use_stemmer=False)
162
```

[그림 - main 함수 코드1]

```
163 progress_bar = tqdm.tqdm(total=args.num_instructions_to_generate)
164 if machine_instructions:
165     progress_bar.update(len(machine_instructions))
166
167 all_instructions = [d["instruction"] for d in seed_instructions] + \
168     [d["instruction"] for d in machine_instructions]
169 all_instruction_tokens = [scorer.tokenizer.tokenize(inst) for inst in all_instructions]
170
171 while len(machine_instructions) < args.num_instructions_to_generate:
172     request_idx += 1
173
174     batch_inputs = []
175     for _ in range(args.request_batch_size):
176         if len(machine_instructions) > num_machine_instructions:
177             prompt_instructions = random.sample(seed_instructions, args.num_prompt_instructions - num_machine_instructions)
178             prompt_instructions += random.sample(machine_instructions, num_machine_instructions)
179         else:
180             prompt_instructions = random.sample(seed_instructions, args.num_prompt_instructions)
181         prompt = encode_prompt(prompt_instructions)
182         batch_inputs.append(prompt)
183
184     request_start = time.time()
185     results = make_requests(
186         model=args.model,
187         messages=batch_inputs,
188         max_tokens=3000,
189         temperature=1.0,
190         top_p=0.95,
191         frequency_penalty=0,
192         presence_penalty=0,
193         stop=["\n20", "20.", "20."],
194         retries=3,
195         api_key=args.api_key,
196         organization=args.organization
197     )
198     request_duration = time.time() - request_start
199
```

[그림 - main 함수 코드2]

```

200 process_start = time.time()
201 instruction_data = []
202 for result in results:
203     new_instructions = post_process_gpt_response(args.num_prompt_instructions, result)
204     instruction_data += new_instructions
205
206 total = len(instruction_data)
207 keep = 0
208
209 for instruction_data_entry in instruction_data:
210     new_instruction_tokens = scorer._tokenizer.tokenize(instruction_data_entry["instruction"])
211     with Pool(num_cpus) as p:
212         rouge_scores = p.map(
213             partial(rouge_scorer._score_lcs, new_instruction_tokens),
214             all_instruction_tokens
215         )
216     rouge_scores = [score.fmeasure for score in rouge_scores]
217     most_similar_instructions = {
218         all_instructions[i]: rouge_scores[i] for i in np.argsort(rouge_scores)[-10:][::-1]
219     }
220     if max(rouge_scores) > 0.4:
221         continue
222     else:
223         keep += 1
224
225     machine_instructions.append(instruction_data_entry)
226     all_instructions.append(instruction_data_entry["instruction"])
227     all_instruction_tokens.append(new_instruction_tokens)
228     with open(os.path.join(args.batch_dir, "machine_generated.jsonl"), "a", encoding="utf-8") as fout:
229         json.dump(instruction_data_entry, fout, ensure_ascii=False)
230         fout.write("\n")
231     progress_bar.update(1)
232 process_duration = time.time() - process_start
233
234 print(f"Request {request_idx} took {request_duration:.2f}s, processing took {process_duration:.2f}s")
235 print(f"Generated {total} instructions, kept {keep} instructions")

```

[그림 - main 함수 코드3]

## 결과 학습 데이터

```

1 {"instruction": "차량 구매 시 중요한 고려사항은 무엇인가요?", "input": "", "output": "차량을 구매할 때 고려해야 하는 중요한 사항은 차량의 주행거리, 연식, 사
2 {"instruction": "자동차 보험 가입 시 고려해야 할 사항은 무엇인가요?", "input": "", "output": "자동차 보험 가입 시 고려해야 할 사항은 보장 범위, 보험료, 자기
3 {"instruction": "카니발과 쏘렌토 중 어떤 차종이 가족 여행에 더 적합한가요?", "input": "", "output": "카니발은 대형 RV 차량으로 공간 활용이 뛰어나 가족 여행에
4 {"instruction": "중고차 구매 시 차량의 외부상태를 어떻게 확인해야 하나요?", "input": "", "output": "차량의 외부상태를 확인할 때는 차체 흠집, 부식, 도색 여부
5 {"instruction": "중고차 구매 시 꼭 확인해야 할 서류는 어떤 것이 있나요?", "input": "", "output": "중고차 구매 시 필요한 서류로는 차량등록증, 소유자증명서, 차
6 {"instruction": "차량의 연비 향상을 위한 궁극적인 방법은 무엇인가요?", "input": "", "output": "차량의 연비를 향상시키기 위한 궁극적인 방법은 정기적인 엔진 오
7 {"instruction": "중고차 구매 시 안전을 위해 확인해야 할 부분은 무엇인가요?", "input": "", "output": "중고차 구매 시 안전을 위해 반드시 확인해야 할 부분으로는
8 {"instruction": "중고차 구매 시 꼭 알아두어야 할 중요한 팀은 무엇인가요?", "input": "", "output": "중고차 구매 시 꼭 알아두어야 할 중요한 팀으로는 정확한 차량
9 {"instruction": "자동차 보유 시 유지보수 비용을 절감하는 방법은 무엇이 있나요?", "input": "", "output": "자동차 보유 시 유지보수 비용을 절감하는 방법으로는
10 {"instruction": "신차와 중고차의 장단점에 대해 설명해주세요.", "input": "", "output": "신차의 장점은 최신 기술과 디자인, 보증 기간 및 무상 정비 등이 있으며,
11 {"instruction": "중고차 구매 시 고려해야 할 중요한 부분은 무엇인가요?", "input": "", "output": "중고차 구매 시 고려해야 할 중요한 부분으로는 주행거리, 차량
12 {"instruction": "차량 유지 관리 비용을 절감하는 방법은 무엇이 있나요?", "input": "", "output": "차량 유지 관리 비용을 절감하는 방법으로는 정기적인 오일 교환
13 {"instruction": "차량 구매 시 중요한 고려사항은 무엇인가요?", "input": "", "output": "차량을 구매할 때 고려해야 하는 중요한 사항은 차량의 외부 내부 상태, 주

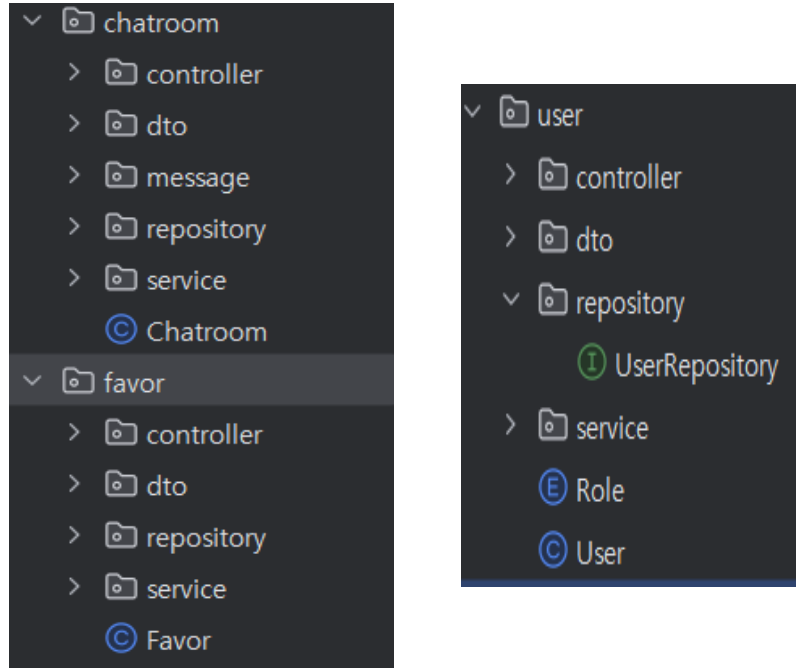
```

[그림 - 결과 학습 데이터]

## 5.6. 백엔드 서버 개발 환경 구축

- 회원가입, 로그인, 해당 사용자와 관련된 정보를 불러오기 위한 메서드 등 사용자에게 대한 정보 처리를 위한 User package.
- 사용자가 생성한 채팅방에 대한 정보를 저장하는 chatroom package.
- 하위 package로 채팅방에서 생성된 메시지를 저장하기 위한 message package.
- 선호하는 중고차 정보를 저장하기 위한 favor package.

- 중고차 정보를 저장하기 위한 vehicle package를 생성하였다.
- 특수차량 정보를 저장하는 special\_vehicle package는 차후 데이터를 추출하면 설계할 예정이다.
- 모든 패키지는 MVC 패턴을 기반으로 하위 항목으로 controller, dto, message, repository, service, entity로 구성되어 있다.



[그림 - 패키지 구조]

각 패키지마다 controller에서의 주요한 기능은 다음과 같다.

### 5.6.1 UserController

```
@GetMapping("/{id}")
public ResponseEntity<> getUser(@PathVariable Long id, @AuthenticationPrincipal CustomUserDetails userDetails) {
    if (!Objects.equals(userDetails.getId(), id)) {
        throw new Exception403("권한이 없습니다.");
    }
    //userdetails 정보와 id를 비교해서 일치하지 않으면 403 리턴
    UserResponse.GetUserDto getUserDto = userService.getUser(id);
    ApiUtils.Response<> response = ApiUtils.success(getUserDto);
    return ResponseEntity.ok().body(response);
    //생성한 응답 본문 반환
}
```

[그림 - UserController 코드1]

- 인증된 사용자만 자신의 정보에 접근할 수 있도록 권한을 체크한다.

- 요청된 사용자 ID와 현재 인증된 사용자의 ID를 비교하여 일치하지 않으면 403 Forbidden exception을 발시킨다.
- ID가 일치하는 경우, 사용자 정보를 조회하고 성공적인 응답을 반환한다.

```
@PostMapping(Ⓜ"/join")
public ResponseEntity<?> join(@RequestBody @Valid UserRequest.JoinDto requestDto, Errors errors) {
    validateBirthForm(requestDto.birth());

    var responseDto = userService.join(requestDto);

    var responseCookie = createRefreshTokenCookie(responseDto.refresh(), JwtProvider.REFRESH_EXP_SEC);

    return ResponseEntity.ok()
        .header(JwtProvider.HEADER, responseDto.access())
        .header(HttpHeaders.SET_COOKIE, responseCookie.toString())
        .body(ApiUtils.success());
}
```

[그림 - UserController 코드2]

- 클라이언트에서 전달된 사용자 가입 정보를 검증한다.
- 생년월일에 대한 유효성 검사를 수행한다.
- 사용자를 가입시키고, 생성된 정보(토큰 포함)를 담은 DTO를 반환받는다.
- access token을 응답 헤더에 추가하고, refresh token을 쿠키로 설정하여 응답한다.
- 성공적인 가입 처리를 클라이언트에 알리는 응답을 반환한다.

이 외에도 로그인, 로그아웃, user정보 삭제, 생성, 비밀번호 초기화 등의 기능이 있다.

## 5.6.2 FavorController

```
@PostMapping(Ⓜ"/favors")
public ResponseEntity<?> saveUserVehiclePreference(@AuthenticationPrincipal CustomUserDetails userDetails,
    @RequestBody @Valid FavorRequest.SaveUserVehiclePreferenceDto requestDto,
    Errors errors){
    favorService.saveUserVehiclePreference(userDetails.getId(), requestDto);
    ApiUtils.Response<?> response = ApiUtils.success(requestDto);
    return ResponseEntity.ok().body(response);
}
```

[그림 - FavorController 코드1]

- 현재 인증된 사용자의 정보를 기반으로 차량 선호도를 저장한다.
- 클라이언트에서 전달된 차량 선호도 정보에 대해 유효성 검사를 수행한다.

- 차량 선호도가 성공적으로 저장된 후, 요청 본문을 포함한 성공적인 응답을 반환한다.

### 5.6.3 ChatroomController

```
@PostMapping(Ⓜ"/chatrooms")
public ResponseEntity<?> create(@AuthenticationPrincipal CustomUserDetails userDetails) {
    ChatroomResponse.CreateChatroomDto createChatroomDto = chatroomService.create(userDetails.getId());
    ApiUtils.Response<?> response = ApiUtils.success(createChatroomDto);
    return ResponseEntity.ok().body(response);
}
```

[그림 - ChatroomController 코드1]

- 현재 인증된 사용자의 정보를 기반으로 새로운 채팅방을 생성한다.
- 채팅방 생성 요청이 성공적으로 처리된 후, 생성된 채팅방에 대한 정보를 포함한 응답을 반환한다.

```
@GetMapping(Ⓜ"/chatrooms")
public ResponseEntity<?> get(@AuthenticationPrincipal CustomUserDetails userDetails) {
    ChatroomResponse.GetChatroomDto getChatroomDto = chatroomService.get(userDetails.getId());
    ApiUtils.Response<?> response = ApiUtils.success(getChatroomDto);
    return ResponseEntity.ok().body(response);
}
```

[그림 -ChatroomController 코드2]

- 현재 인증된 사용자의 정보를 기반으로 해당 사용자의 채팅방 정보를 가져온다.
- 채팅방 조회 요청이 성공적으로 처리된 후, 조회된 채팅방에 대한 정보를 포함한 응답을 반환한다.

```
@DeleteMapping(Ⓜ"/chatrooms/{chatroomId}")
public ResponseEntity<?> deleteChatroom(@PathVariable Long chatroomId,
                                         @AuthenticationPrincipal CustomUserDetails userDetails) {
    chatroomService.delete(userDetails.getId(), chatroomId);
    ApiUtils.Response<?> response = ApiUtils.success();
    return ResponseEntity.ok(response);
}
```

[그림 -ChatroomController 코드3]

- 특정 채팅방 ID와 현재 인증된 사용자의 정보를 기반으로 해당 채팅방을 삭제한다.
- 채팅방 삭제 요청이 성공적으로 처리된 후, 성공 응답을 반환한다.

추후 백엔드 기능 추가 예정은 다음과 같다.

1. JavaScript, WebSocket을 통해 사용자가 UI를 통해 입력하는 정보를 실시간으로 웹페이지에 반영하는 기능
2. CORS를 통한 회원가입 및 로그인 후 사용자 정보를 이용하는 모든 기능에 대한 보안