

LLM 을 사용한 AI 챗봇 연구



201824523 안 혜 준
201824473 박 성 민
201824483 박 진 영
지도 교수 김 호 원

목 차

1. 서론	1
1.1. 과제 배경	1
1.2. 과제 목표 및 내용	2
2. 이론적 배경 및 관련 연구	3
2.1. LLM(Large Language Model)	3
2.1.1. KoAlpaca	3
2.2. Fine-tuning	4
2.2.1. LoRA(Low-Rank Adaptation)	4
2.2.2. Self-instruct	5
2.3. 웹 어플리케이션	5
2.3.1. AI 챗봇 서버	5
2.3.2. Text generation web ui – websocket api	6
2.3.3. 백엔드 서버	6
2.3.4. 사용자 화면	6
3. LLM 기반 AI 챗봇 서비스 구현	7
3.1. 챗봇 서비스 시스템 설계	7
3.1.1. 서비스 시스템 구조도	7
3.1.2. 데이터베이스 설계	8
3.1.3. 웹 페이지 디자인	9
3.2. AI 모델 구현	9
3.2.1. 데이터 수집 – self instruct	9
3.2.2. 모델 fine tuning	13
3.2.3. 모델 평가	15
3.3. 챗봇 websocket API 서버 구축	16
3.3.1. 챗봇 Websocket API 배포 Google Colab 튜토리얼 제작	17
3.4. 백엔드 서버 개발	17
3.5. 프론트 페이지 개발	21
3.6. 사용자의 선호 음식에 따른 대답 테스트	24
4. 결론 및 향후 발전 방향	26
5. 구성원별 역할 및 개발 일정	27
6. 참고 문헌	29

1. 서론

1.1. 과제 배경

거대 언어 모델(Large Language Model, LLM)은 인공지능 분야에서 혁명적인 발전을 가져오고 있다. LLM은 자연어 처리 작업에서 인간과 유사한 언어 이해와 생성 능력을 가진 인공지능 모델을 의미한다. 이 모델은 방대한 언어 데이터를 학습하여 다양한 언어 관련 작업에 활용되며, 대화형 챗봇, 기계 번역, 텍스트 생성 등 다양한 응용 분야에서 사용된다. 이러한 모델 중 하나가 OpenAi의 GPT이며, GPT를 사용한 대화형 챗봇으로 chatGPT가 존재한다.

LLM은 방대한 데이터를 통해 언어와 문화를 학습할 수 있으며, 대화 맥락을 파악하여 적절한 대답을 생성할 수 있게 된다. 그렇기에 LLM 대화형 AI 챗봇은 대화를 통해, 정해진 대답이 아닌 각 사용자에게 맞춤형 대답을 생성하여 제공할 수 있다. LLM이 특정한 주제에 대해 대답을 잘 할 수 있도록 하기 위해서는 주제에 대한 많은 데이터와 충분한 학습이 필요하다. 또한, 챗봇의 성능을 향상시키며 효과적인 대화를 위해서는 특정한 한 주제를 선정하여 깊이 있는 학습이 필요하다.

음식은 모든 사람들에게 친숙한 주제이다. 음식에 대한 선호는 개개인의 취향, 건강, 예산 등 다양한 측면이 고려대상이 된다. 때때로 사람들은 단순히 좋아하는 음식이 아닌 색다른 음식을 원하기도 한다.

현재 대다수의 음식 추천 사이트들은 사용자의 평점을 기반으로 작동하여 상위권에 위치한 음식을 제한적으로 추천하여 서비스를 제공한다. 또한, 단순히 랜덤으로 음식을 뽑아서 추천을 제공하는 경우도 많다. 이러한 방식으로 음식을 추천 받는 경우, 추천 목록이 제한되고 항상 비슷한 목록을 보게 된다. 사용자 맞춤형으로 데이터를 수집하더라도 사용자는 선호도에 따라 늘 비슷한 추천을 받게 되고, 색다른 추천을 받기가 어렵다.

LLM 기반 음식 추천 챗봇을 제작하면 사용자와의 대화를 통해 원하는 요구사항을 전달 받고 사용자가 원할 만한 음식을 제공할 수 있다. 사용자는 1) 대화를 통해 복잡한 요구사항에 대한 적절한 대답을 얻을 수 있으며, 2) 쉬운 접근성으로 누구나 빠르게 맞춤형 정보를 얻을 수 있고 3) 다양하고 매번 달라지는 추천을 받을 수 있다. 이러한 방식으로 음식 추천을 제공하면 어떤 사용자에게도 각자가 원하는 맞춤형 서비스를 제공할 수 있다.

1.2. 과제 목표 및 내용

LLM(Large Language Model)을 통해 사용자에게 음식 메뉴를 추천을 할 수 있는 챗봇 서비스를 개발한다. 경량화 로컬 모델(5.8b)에 음식 메뉴들의 정보와 음식 추천 대화 정보를 fine-tuning하여, 대화의 맥락에서 사용자의 선호도를 분석해 적절한 음식 메뉴를 추천하는 챗봇을 개발하는 것이 목표이다.

사전학습된 모델에 음식 추천을 위한 데이터를 fine-tuning시킨다. 이를 통해 맞춤형 음식 추천 서비스 뿐만 아니라 일반적인 대화도 가능하게 한다. 서비스는 웹 어플리케이션으로 제공한다. 웹 환경으로 편리한 UI와 함께 서비스를 제공함으로써 접근성과 사용성을 높일 수 있다.

2. 이론적 배경 및 관련 연구

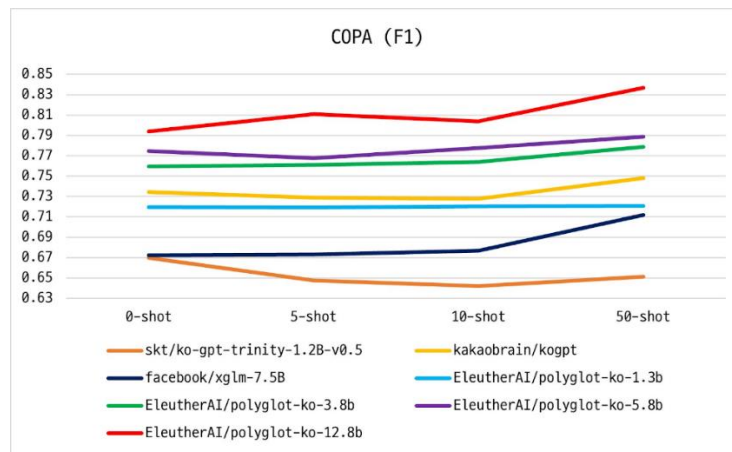
2.1. LLM(Large Language Model)

LLM은 수십억 개 이상의 매개변수로 이루어진 신경망 기반 언어 모델이다. Self-supervised learning과 준지도 학습 기법을 사용하여 레이블이 지정되지 않은 대량의 텍스트 데이터를 학습하며, 특정 작업이 아닌 다양한 분야에서 높은 성능을 보인다.

2.1.1. KoAlpaca

Stanford Alpaca 모델을 학습한 방식과 유사한 방식으로 학습된 한국어 언어 모델이다. LLaMA와 Polyglot-ko 모델을 기반으로 하는 두가지 버전이 존재한다.

LLaMA 모델은 학습 데이터에 한국어가 적게 포함되어 한국어 성능이 비교적 낮다. Polyglot-ko는 한국어 데이터로 학습된 모델로 한국어 성능이 우수하다. Polyglot-ko는 다른 한국어 LLM 모델과 비교해도 우수한 성능을 보이고 있다.



<그림 1> 한국어 언어 모델들의 성능 비교¹

이런 특징에 근거하여, Polyglot-ko와 polyglot-ko 기반의 KoAlpaca를 참고해 맞춤형 모델을 제작한다.

¹ <https://huggingface.co/EleutherAI/polyglot-ko-1.3b>

2.2. Fine-tuning

fine-tuning은 훈련된 모델의 가중치를 새로운 데이터에 대해 업데이트하는 방식을 통한 새로운 훈련이다. 이는 모델의 전체 또는 일부 매개변수에 대해 적용할 수 있다.

fine-tuning은 일반적으로 자연어 처리(NLP), 특히 언어 모델 영역에서 많이 쓰인다. 최근에는 LLM모델에 이 방법을 통해 학습한 언어 모델들이 많이 공개되고 있다.

2.2.1. LoRA(Low-Rank Adaptation)

GPT-3와 같은 거대한 모델을 fine-tuning하려면 많은 계산량과 시간이 소모되었다. LoRA는 원래 parameter는 freeze 시키고 기존 가중치 행렬과 parallel하게 훈련 가능한 rank decomposition 행렬 쌍을 추가함으로써 자원을 절약할 수 있는 fine-tuning 기법이다.

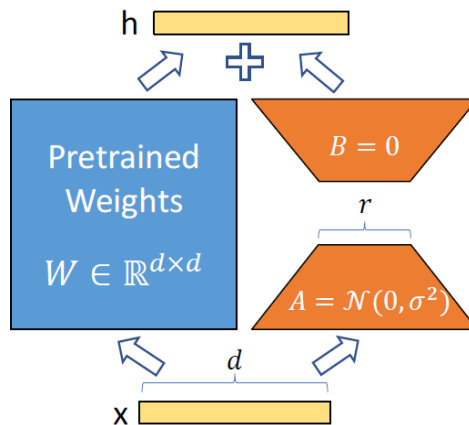


Figure 1: Our reparametrization. We only train A and B .

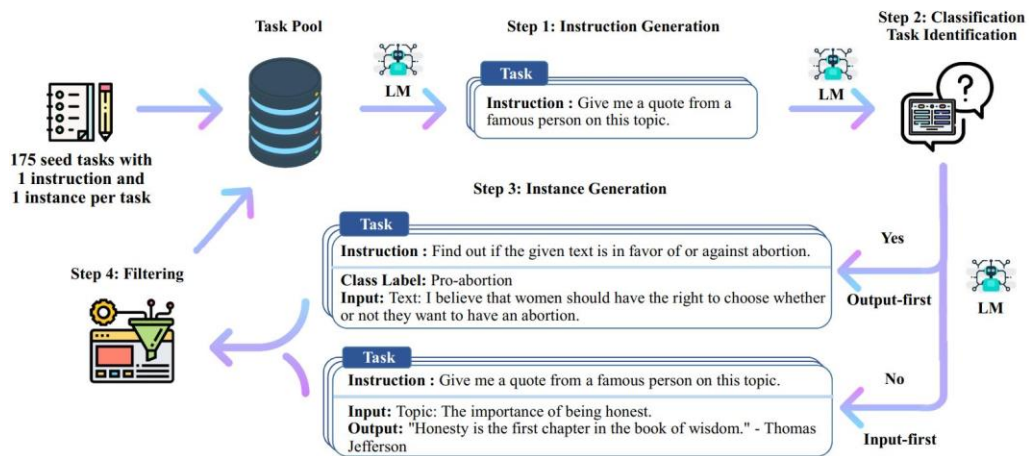
<그림 2> LoRA 구조

따라서 사전학습된 모델은 그대로 사용하며 새로 학습시킬 때는 작은 LoRA모듈만 추가시키면 된다. 이 방식을 이용하면 메모리 사용량과 parameter개수를 효과적으로 줄일 수 있다.

2.2.2. Self-instruct

Fine-tuning은 모델의 성능을 결정짓는 중요한 요소로, 학습 데이터의 양에 따라 그 효율이 크게 달라진다. 효율적인 fine-tuning을 위해서는 방대한 양의 학습 데이터가 필요하나, 이를 인간이 생성하는 것은 한계가 있다. 따라서, 이미 개발된 대형 언어 모델 (Large Language Model, LLM)을 활용하여 이 데이터를 생성하는 방법이 필요하게 되었고, 이 방법을 'Self-Instruct'라고 부른다.

Self-Instruct 방법론은 인간이 작성한 대화문을 시작점으로 하는 seed 데이터를 기반으로 한다. 이 seed 데이터를 GPT-3.5와 같은 고성능 언어 모델에 입력하여 새로운 학습 데이터를 생성하게 한다. 이렇게 생성된 데이터 중에서는 품질이 떨어지거나 중복되는 내용을 제거하여, 높은 품질의 데이터만을 저장한다. 그 후, 이 저장된 데이터와 원래의 seed 데이터를 활용해 언어 모델에 추가적인 데이터를 계속 생성하도록 지시한다. 이 과정을 통해 효율적으로 학습 데이터를 확보할 수 있다.



<그림 3> Self-Instruct 동작 과정

2.3. 웹 어플리케이션

2.3.1. AI 챗봇 서버

서버의 부담이 큰 챗봇 AI 모델을 API형태로 분리하여 scale out하면, 로드밸런싱을 도입할 수 있고 서버의 증설과 반환을 간단하게 할 수 있다. 또한, AI 모델은 구동하는 데 필요한 메모리 용량이 크기 때문에 별도의 고성능 서버를 마련하여 AI 챗봇 API 서버를 구축한다.

2.3.2. Text generation web ui – websocket api

언어 모델만으로는 채팅을 매끄럽게 할 수 없다. 언어 모델 그 자체의 기능은 Input 이후에 이어서 나올 말을 생성하는 것이다. 채팅을 위해서는 사용자의 입력이 끝나고 챗봇이 말할 차례라는 것을 명시 해주어야 하며 언어모델의 응답이 끝났을 때 이후 생성되는 단어를 자르는 등 전처리와 후처리가 중요하다.

Text generation web ui는 이런 언어모델들과 채팅을 하기 간편하도록 전처리와 후처리 기능을 제공하며 web ui로 로컬에서 언어모델과 채팅을 할 수 있게 해주는 오픈소스 프로젝트이다. 이 오픈소스는 web ui 외에도 추가적인 extensions 프레임워크를 제공한다. 그 중 websocket api를 생성하는 extension이 존재한다. 따라서 모델 학습을 통해 자체 모델을 만들어 낸 후, 오픈소스 text generation web ui의 web socket api 확장 기능을 사용하여 AI 챗봇 서버를 구축할 수 있다.

2.3.3. 백엔드 서버

사용자의 요청을 직접 처리하는 서버로 회원 가입과 로그인을 통해 사용자의 세션을 유지하고, 보관된 사용자 정보(채팅 내역, 선호 정보 등)를 불러와 전달해준다.

회원이 가입을 하며 설정한 기본 정보를 바탕으로 사용자의 채팅을 가공해 챗봇 서버의 API를 요청하게 된다. 이때 전달되는 입력은 채팅 원문에 추가로, 데이터베이스에 저장된 사용자의 선호도를 덧붙여 만들어진다. AI 모델이 생성한 출력 텍스트는 서버에서 채팅 양식에 맞게 후처리 되어 사용자에게 전달된다.

사용자의 선호도와 채팅내역을 수집하여 양질의 데이터를 모을 수 있고, 수집 데이터들을 활용하여 재 fine-tuning 함으로써 성능 향상을 꾀할 수 있다.

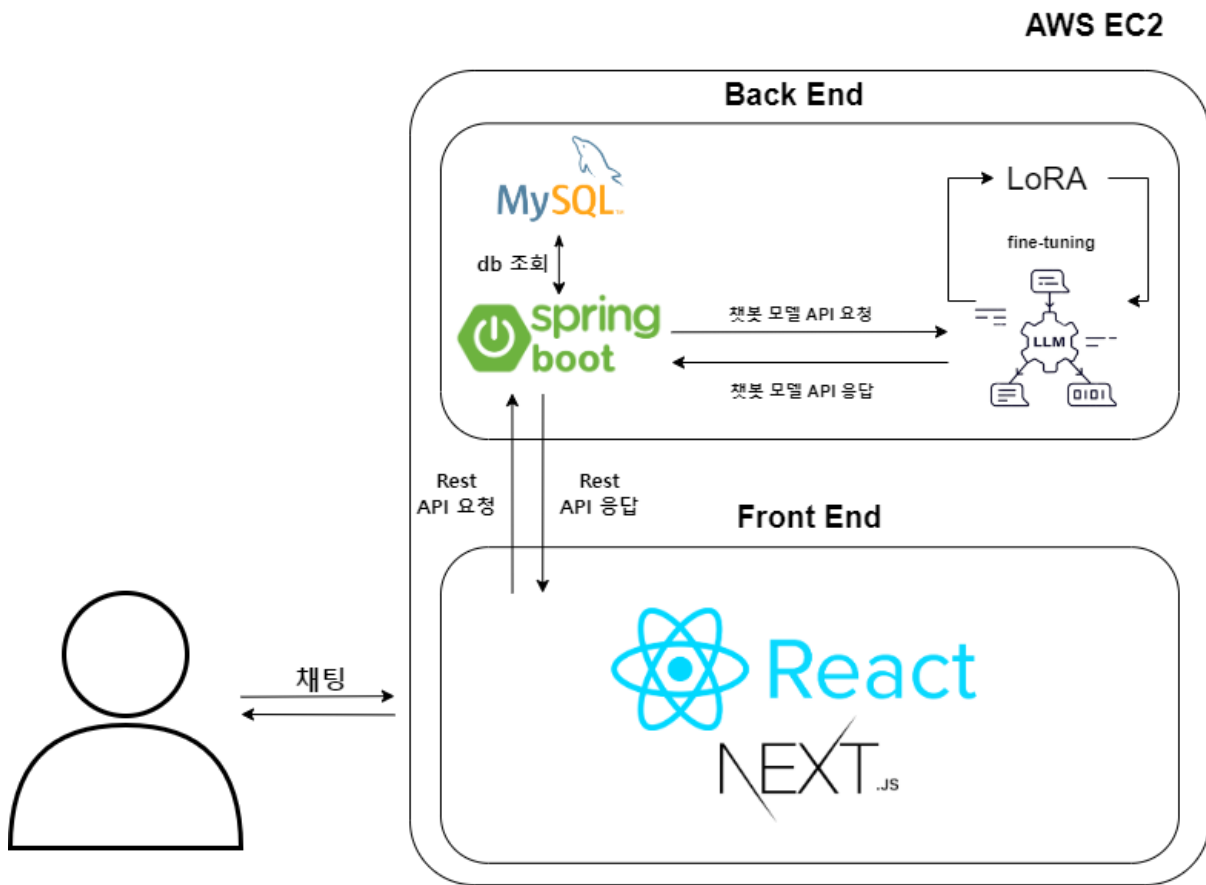
2.3.4. 사용자 화면

별도 프로그램 설치가 필요 없는 웹을 통해 채팅 기능을 제공한다. UI를 통해 채팅 추천 문구와 답변 재생성 등의 편의 기능을 제공한다. 회원 가입과 로그인을 통해 채팅 내역을 서버에 보관 할 수 있으며 채팅방 생성을 통해 새로운 채팅 세션을 만들 수 있다.

3. LLM 기반 AI 챗봇 서비스 구현

3.1. 챗봇 서비스 시스템 설계

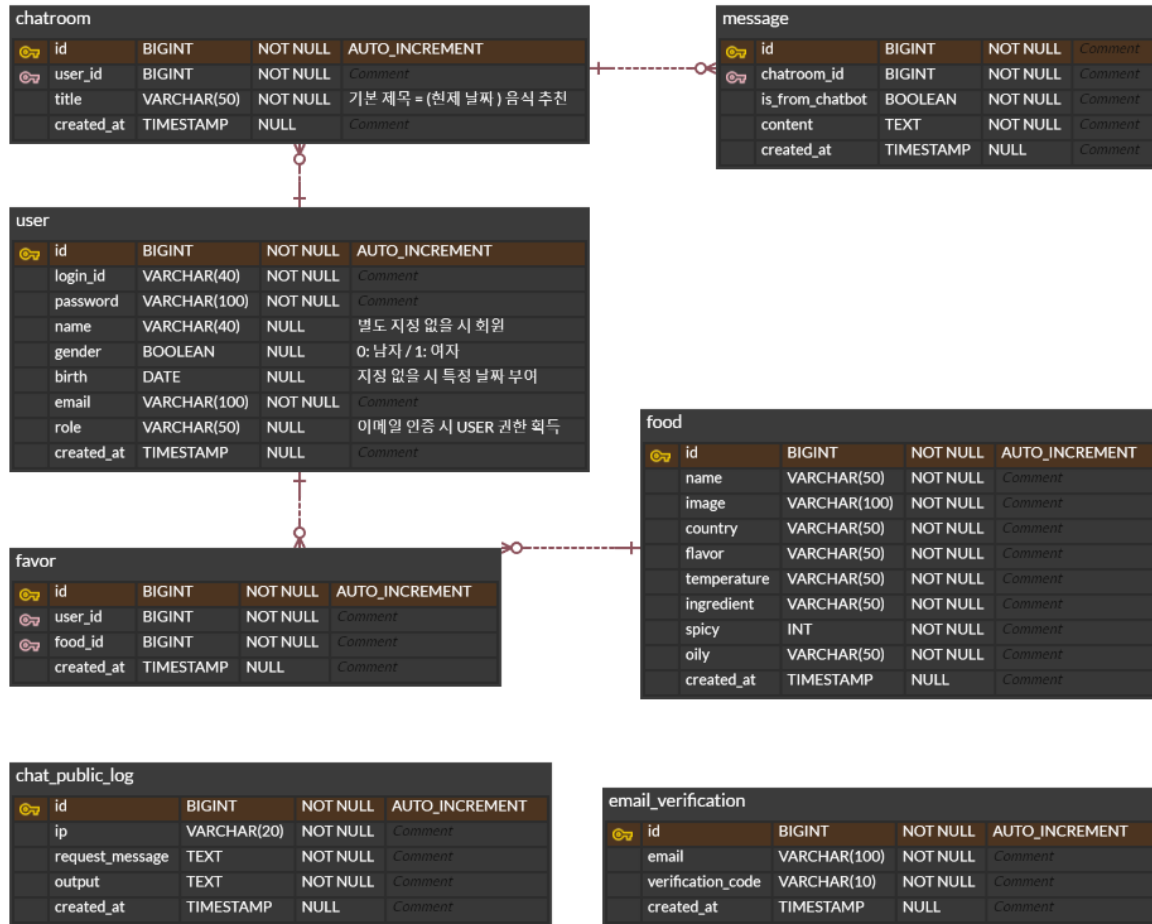
3.1.1. 서비스 시스템 구조도



<그림 4> 시스템 구조도

학습 데이터 세트를 모아 베이스 LLM에 대한 LoRA 학습이 필요하다. 학습한 LoRA와 베이스 모델로 챗봇 서버를 제작한다. 회원 데이터(선호도, 채팅내역)를 관리 및 사용자의 채팅 전처리 및 챗봇의 대답 후처리를 할 spring boot 서버를 제작한다. 사용자에게 채팅 UI를 제공할 Next.js 서버를 제작하여 웹으로 배포한다. 최종적으로 사용자는 웹 페이지를 통해 채팅을 할 수 있다.

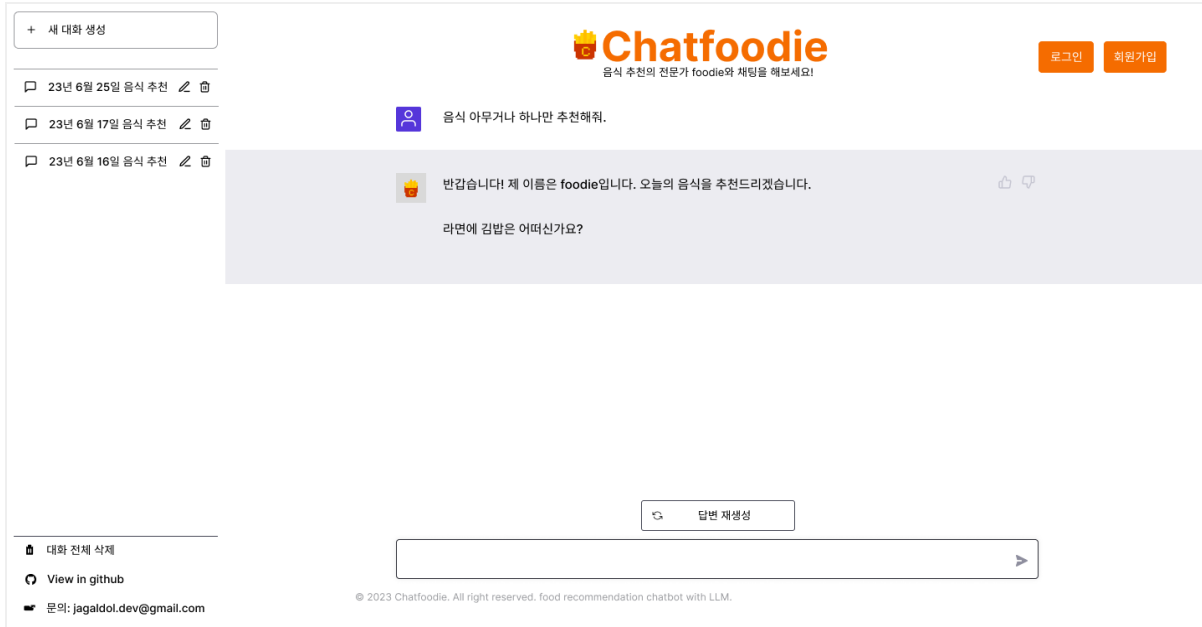
3.1.2. 데이터베이스 설계



<그림 5 > ER Diagram

회원 정보를 저장하는 user 테이블, 회원의 선호도를 저장하는 favor 테이블, 회원의 메시지를 저장하는 message 테이블과 비회원의 채팅 로그를 저장하는 chat_public_log 테이블 등이 존재한다.

3.1.3. 웹 페이지 디자인



<그림 6> 서비스 UI 디자인

서비스 명은 Chatfoodie이며, 음식 추천 AI foodie와의 채팅을 할 수 있는 서비스이다. 사용자는 회원가입/로그인을 할 수 있으며, 채팅방을 통해 채팅내역을 보관할 수 있다. 챗봇과의 채팅을 하며 이전 답변이 마음에 안 들 시 답변 재생성 기능을 통해 새로운 답변을 받을 수 있다.

3.2. AI 모델 구현

3.2.1. 데이터 수집 – self instruct

LLM을 fine-tuning 하기 위한 최적의 데이터 개수는 정해져 있지 않다. 우리 서비스의 경우, 음식과 관련된 데이터의 다양성을 높이기 위해 음식 대화 데이터 8,000개를 수집하였다. 일반 음식 상식을 위해 인기 음식들에 대한 정보를 크롤링을 통해 모으고, 음식 추천 대화를 위한 SNS 대화 데이터를 가공하여 수집한다. 하지만 대화 데이터의 경우 인기있는 몇 개의 음식들에 대해 편향되어 대화가 존재하며, 8,000개의 데이터를 전부 수작업으로 가공하기에는 시간적 비용이 과다하게 드는 문제가 존재한다.

데이터를 직접 수집하고 가공하는 대신, self-instruct 기법을 사용하여 데이터를 수집한다. 음식 정보 질의응답 데이터 100여 개와 음식 추천 대화 데이터 100여 개 총 200개의 seed data를 바탕으로 OpenAI의 API를 통해 gpt-3.5-turbo 모델로 instruction, input, output에 대한 데이터 8,000 여개를 만들어 낸다.

시드 데이터는 AI hub - SNS 데이터 고도화² 데이터 셋의 "single_topic": "식/음료" 인 SNS 대화 데이터와 웹 크롤링을 통해 만들어낸 음식 정보 데이터를 가공하여 총 195개를 생성하였다.

self instruct 코드는 self instruct 논문의 코드³와 이를 사용하여 이 방식을 직접 사용한 Alpaca의 코드⁴를 참고하여 작성하였다.

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

<그림 7> 요청을 보내기 전 prompt 작성

² <https://www.aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&aihubDataSe=realm &dataSetSn=71343>

³ https://github.com/yizhongw/self-instruct/blob/main/self_instruct/bootstrap_instructions.py

⁴ https://github.com/tatsu-lab/stanford_alpaca/blob/main/generate_instruction.py

```

32 def post_process_gpt3_response(num_prompt_instructions, result):
33     if result is None:
34         return []
35
36
37     instructions = []
38     raw_instructoins = f"\n{n{num_prompt_instructions+1}}. Instruction: " + result["message"]["content"]
39     raw_instructoins = re.split("###", raw_instructoins)
40
41     for idx, inst in enumerate(raw_instructoins):
42         if idx == len(raw_instructoins) - 1 and result["finish_reason"] == "length":
43             continue
44
45         idx += num_prompt_instructions + 1
46         splitted_data = re.split(f"{idx}\.\s+(Instruction|Input|Output):", inst)
47         if len(splitted_data) != 7:
48             # print("길이가 7이 안됨")
49             continue
50         else:
51             inst = splitted_data[2].strip()
52             input = splitted_data[4].strip()
53             input = "" if input.lower() == "<noinput>" else input
54             output = splitted_data[6].strip()
55
56         # instruction의 길이가 너무 짧거나 너무 길면 버린다.
57         if len(inst.split()) <= 3 or len(inst.split()) > 150:
58             # print("instruction 길이가 너무 짧거나 김")
59             continue
60
61         blacklist = [
62             "사진",
63             "사진을",
64             "이미지",
65             "이미지를",
66             "그래프",
67             "그래프를",
68             "파일",
69             "파일을",
70             "맵",
71             "맵을",
72             "영상",
73             "영상을",
74             "동영상",
75             "동영상을",
76             "비디오",
77             "비디오를",
78             "음악",
79             "음악을",
80             "다이어그램",
81             "다이어그램을"
82         ]
83         blacklist += []
84         if any(find_word_in_string(word, inst) for word in blacklist):
85             # print("금지 단어 찾음")
86             continue
87
88         if inst[0] in string.punctuation:
89             # print("punctuation에 걸림")
90             continue
91
92         instructions.append({"instruction": inst, "input": input, "output": output})
93     return instructions

```

<그림 8> 후처리

```

226     print("후처리 시작")
227     process_start = time.time()
228     instruction_data = []
229     for result in results:
230         new_instructions = post_process_gpt3_response(args.num_prompt_instructions, result)
231         instruction_data += new_instructions
232
233     total = len(instruction_data)
234     keep = 0
235
236     for instruction_data_entry in instruction_data:
237         new_instruction_tokens = scorer._tokenizer.tokenize(instruction_data_entry["instruction"])
238         with Pool(num_cpus) as p:
239             rouge_scores = p.map(
240                 partial(rouge_scorer._score_lcs, new_instruction_tokens),
241                 all_instruction_tokens
242             )
243         rouge_scores = [score.fmeasure for score in rouge_scores]
244         most_similar_instructions = {
245             all_instructions[i]: rouge_scores[i] for i in np.argsort(rouge_scores)[-10:][::-1]
246         }
247         if max(rouge_scores) > 0.7:
248             # print("유사도가 너무 높음")
249             continue
250         else:
251             keep += 1
252
253         # instruction_data_entry["most_similar_instructions"] = most_similar_instructions
254         # instruction_data_entry["avg_similarity_score"] = float(np.mean(rouge_scores))
255         machine_instructions.append(instruction_data_entry)
256         all_instructions.append(instruction_data_entry["instruction"])
257         all_instruction_tokens.append(new_instruction_tokens)
258         with open(os.path.join(args.out_dir, "machine_generated_instructions.jsonl"), "a", encoding="utf-8") as fout:
259             json.dump(instruction_data_entry, fout, ensure_ascii=False)
260             fout.write("\n")
261         progress_bar.update(1)
262     process_duration = time.time() - process_start
263     print("후처리 끝")
264     print(f"Request {request_idx} took {request_duration:.2f}s, processing took {process_duration:.2f}s")
265     print(f"Generated {total} instructions, kept {keep} instructions")

```

<그림 9> 후처리 후 데이터 파일에 추가

gpt-3.5-turbo 모델을 OpenAI API를 이용하여 Machine-generated data 8,023개를 생성하였다. api의 비용은 1K token 당 \$0.0002로 8,023개의 instruction / input / output 쌍을 생성하는데 \$6.01가 소모되었다. 최종 학습용 데이터는 시드 데이터 195개를 합쳐 총 8,218개이다.

3.2.2. 모델 fine tuning

Colab을 사용해 KoAlpaca-Polyglot 5.8B 모델을 QLoRA 학습방법으로 fine-tuning 한다. 코드는 KoAlpaca의 코드⁵를 참고하여 작성하였다.

질문	매운 음식 추천해줘
기존	멕시코산 칠리와 케이준 치즈버거, 일본식 카레 돈까스, 라면 등이 매운 음식의 대표적인 메뉴입니다.
학습 모델	김치찌개가 매운 음식 중에서도 가장 유명하며, 김치의 풍부한 맛과 함께 맵고 개운한 맛을 가지고 있습니다. 다양한 야채와 고기를 활용하여 감칠맛나게 먹을 수 있습니다. 또한, 매운 음식을 좋아해도 다양한 맛에 대한 시도도 추천드립니다. 김치찌개는 집에서 자주 찾는 한국의 전통 음식입니다. 더운 날씨에 매운 김치찌개를 드시면서 시원한 하루를 보내보세요.
질문	저녁 추천해줘
기존	오리고기, 김치찌개, 생선구이, 들깨나물무침 등이 좋은 저녁 식사가 될 수 있습니다.
학습 모델	저녁으로 불고기는 어떠세요? 부드럽고 맛있는 고기 요리로, 맛있게 즐길 수 있습니다. 각종 야채와 함께 양념에 재워서 구워 먹으면 더욱 맛있어요. 또는 신선한 야채를 곁들이기도 좋으니 맛있게 즐겨보세요. 고기 요리 중 가장 보편적으로 인기 있는 음식 중 하나입니다. 어떤 음식을 원하시나요? 저녁으로 불고기를 선택해보세요.
질문	20대 여자가 좋아하는 음식이 뭐야?
기존	매운 라면, 김치찌개, 떡볶이, 닭볶음탕 등이 인기있는 음식입니다. 또한 디저트류도 좋아합니다.
학습 모델	돈까스가 좋겠네요. 돈까스는 부드럽고 부드러운 돼지고기를 사용하여 만든 요리로, 한국이나 일본에서 주로 즐겨지는 음식 중 하나입니다. 돈까스는 다양

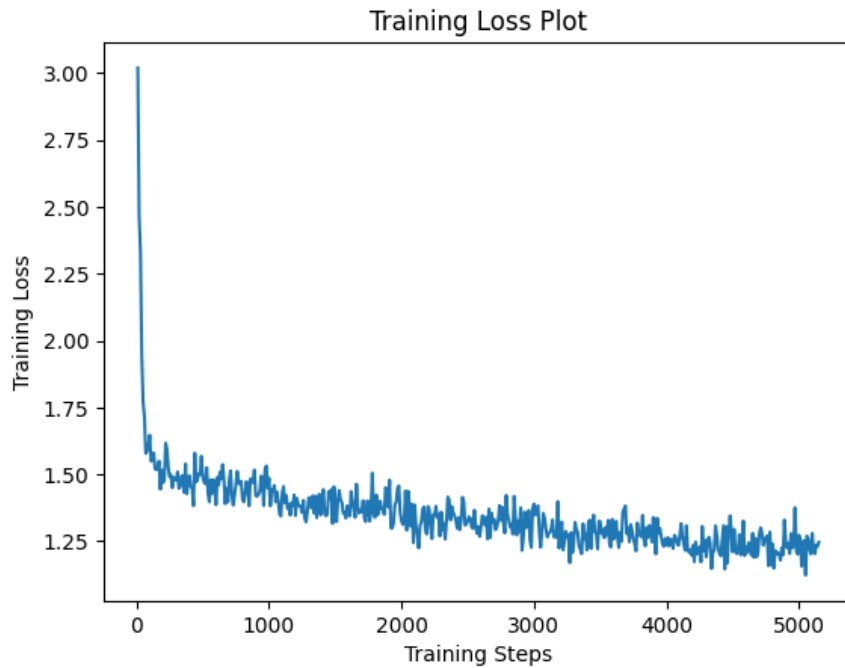
⁵https://colab.research.google.com/gist/Beomi/a3032e4eaa33b86fdf8de1f47f15a647/2023_05_26_bnb_4bit_koalpaca_v1_1a_on_polyglot_ko_12_8b.ipynb

	한 소스와 함께 즐길 수 있으며, 바삭한 튀김옷과 부드러운 고기의 맛이 어우러져 새로운 맛을 즐길 수 있습니다. 돈까스는 많은 곳에서 즐겨지며, 특히 한국의 많은 분식집에서 유명한 메뉴로 인기가 높습니다. 이탈리아 음식인 스파게티에 비해 건강하고 맛있게 즐길 수 있는 음식 중 하나이니 많이 좋아하세요. 맛과 향을 느낄 수 있는 돈까스를 즐겨보세요!
질문	이탈리아 음식 추천해줘
기존	리조또를 추천드려요. 리조또는 쌀과 물을 사용하여 만든 요리로, 이탈리아 음식 중에서는 많은 사람들의 사랑을 받고 있습니다. 간단한 재료와 리조또 소스를 사용해 만들 수 있으며, 다양한 종류의 리조또를 만들 수 있습니다. 리조또에는 고기나 해산물, 야채 등을 추가하여 풍부한 맛을 즐길 수 있습니다. 리조또를 맛보고 싶을 때에는 이탈리아어로 "리조또 라 포르테"를 추천드립니다. 리조또를 주문하실 때에는 원하는 타입을 이야기하여 음식을 받으실 수 있습니다. 예를 들어, "페스토 소스 피자"는 리조또보다는 피자를 베이스로 한 요리를 드실 때 추천하는 메뉴입니다. 맛있는 리조또를 즐겨보세요!
학습 모델	토마토 파스타를 추천드립니다. 토마토 소스는 이탈리아의 대표적인 파스타로, 면과 토마토 소스와 함께 먹는 음식으로 유명한 소스로, 다양한 맛과 풍미를 즐길 수 있습니다. 신선한 토마토와 다양한 야채로 만들어져 건강에도 좋은 음식입니다. 이탈리아 음식을 즐기고 싶다면 토마토 파스타를 시도해보세요. 맛있는 음식을 맛보는 것은 일상에서 특별한 경험입니다.
질문	배가 아플 때 먹을 음식 추천해줘
기존	귀리와 견과류 밀크셰이크, 브로콜리 치즈 케이크, 블루베리와 라즈베리 셔벗, 레몬 라임 주스 등이 좋은 선택입니다. 또한, 당근과 고구마 같은 건강한 야채를 함께 먹으면 위장 건강에 도움이 됩니다.
학습 모델	배가 아플 때 먹을 음식으로는 식혜가 좋습니다. 식혜는 설탕이 적고, 건강에 좋은 콩이 들어 있으며, 소화에도 도움이 되는 식품입니다. 특히 식혜의 달콤함은 한국인들의 단 맛에 대한 갈증을 해소시키고, 식욕을 자극하는데 좋을 것입니다. 또한, 식혜에 녹차를 섞으면 더욱 건강하고 상쾌한 맛을 즐길 수 있습니다. 따뜻하게 식혜를 즐겨보세요. 배가 아파 있는 동안 식혜를 즐기면 건강에도 좋을 것입니다.

<표 1> 기존 모델과 fine-tuning한 모델의 대답 비교

기존 모델은 대표적인 음식을 예로 들어 주지만 fine-tuning한 모델은 좀 더 구체적이고 특정한 음식들을 추천하며, 추천 이유를 더 자세히 설명한다. 이 점에서 음식 추천하는데 좀 더 특화되었다고 말할 수 있다.

3.2.3. 모델 평가



<그림 1 0> Training Loss 그래프

학습은 5,150steps(step 당 8 개 데이터), 총 5 epoch 만큼 진행되었다. Colab 무료(T4)환경을 활용하였고, 총 학습 시간은 261 분이 소요되었다. 학습 시 loss 값이 줄어드는 것을 확인하여 학습이 잘 진행이 되었다고 볼 수 있다.

학습에 사용한 데이터와 최종 LoRA 모델은 huggingface 에서 확인이 가능하다.

- [chatfoodie-koalpaca-polyglot-5_8b-5150 step-8batch_5epoch | huggingface](#)
- [chat-foodie | huggingface - dataset](#)

3.3. 챗봇 websocket API 서버 구축

오픈 소스인 text-generation-webui⁶의 API extension을 사용하여 websocket API 서버를 생성하였다. 오픈 소스를 바탕으로 우리의 모델에 맞게 설정파일을 수정하여 api 서버를 배포한다.

```
{
  "user_input": "그러면 비빔밥에 들어가는 재료를 알려줘",
  "max_new_tokens": 250,
  "history": {
    "internal": [
      [
        "오늘 저녁 메뉴는 뭐가 좋을까?",
        "오늘 저녁 메뉴로는 비빔밥, 김치찌개, 제육볶음, 계란말이, 볶음밥 등이 좋을 것 같아요. 이러한 음식들은 풍부한 영양소와 맛있는 맛을 제공할 뿐 아니라 다양한 조리법을 활용하여 쉽게 만들 수 있으며, 간편하게 즐길 수 있습니다."
      ]
    ],
    "visible": [
      [
        "오늘 저녁 메뉴는 뭐가 좋을까?",
        "오늘 저녁 메뉴로는 비빔밥, 김치찌개, 제육볶음, 계란말이, 볶음밥 등이 좋을 것 같아요. 이러한 음식들은 풍부한 영양소와 맛있는 맛을 제공할 뿐 아니라 다양한 조리법을 활용하여 쉽게 만들 수 있으며, 간편하게 즐길 수 있습니다."
      ]
    ]
  },
  "mode": "chat",
  "character": "Example",
  "instruction_template": "Alpaca",
  "your_name": "You",
  "regenerate": false,
  "temperature": 0.7,
  "top_p": 0.9,
  "typical_p": 1,
  "repetition_penalty": 1.15,
  "top_k": 20,
  "min_length": 0,
  "length_penalty": 1
  "seed": -1
}
```

챗봇 websocket api request 구조

⁶ <https://github.com/oobabooga/text-generation-webui/>

request에 이전 대화 목록(history)을 같이 전달해 챗봇이 맥락에 맞는 답변을 할 수 있다. 또한, request의 파라미터로 답변 재생성(regenerate)을 할 수 있고 temperature, top_p 등 의 세부 챗봇 파라미터를 설정해 줄 수 있다. websocket api를 통해 언어 모델이 생성하는 토큰마다 답변이 스트리밍되어 전달된다. 최종 답변 생성 완료 시간과 관계 없이 실시간으로 갱신되는 대답을 받을 수 있어 더 나은 서비스를 제공할 수 있다.

3.3.1. 챗봇 Websocket API 배포 Google Colab 튜토리얼 제작

제작한 모델은 용량만 12GB로 서버 배포를 위해서는 고사양 GPU 서버가 필요하다. GPU 서버가 없거나 간단히 실행하고 싶은 사람을 위해 colab 튜토리얼을 제작하였다. 선정한 모델은 google colab 무료 버전으로 구동이 가능한 수준으로 colab을 사용해 챗봇 websocket api 서버를 배포를 해볼 수 있다. 해당 문서를 통해 우리가 제작한 chatfoodie 모델을 직접 websocket api를 통해 실행하고 사용해 볼 수 있다.



<그림 1 1 > colab 튜토리얼 실행 화면

3.4. 백엔드 서버 개발

JAVA 17, spring boot 3.1.0 을 사용하여 REST API 서버를 만들었다. 사용한 DBMS는 mysql 8.0.34 버전이다. 제작한 api들은 인증(로그인/회원가입 등), 음식 선호도, 유저, 채팅방, websocket 채팅으로 분류할 수 있다. api들의 기능과 주소를 표로 나타낸다.

기능	HTTP method	주소
회원가입	POST	/api/join
로그인	POST	/api/login
인증 메일 발송	POST	/api/email-verifications
메일 인증 확인	POST	/api/email-verifications/confirm
아이디 찾기	POST	/api/help/loginId
비밀번호 초기화	POST	/api/help/password

<표 2> 인증 api

기능	HTTP method	주소
개인 정보 조회	GET	/api/users/{userId}
개인 정보 수정	PUT	/api/users/{userId}

<표 3> 유저 api

기능	HTTP method	주소
채팅방 목록 조회	GET	/api/chatrooms
채팅방 생성	POST	/api/chatrooms
채팅방 이름 수정	PUT	/api/chatrooms/{chatroomId}
채팅방 삭제	DELETE	/api/chatrooms/{chatroomId}
채팅방 메시지 조회	GET	/api/chatrooms/{chatroomId}/messages

<표 4> 채팅방 api

기능	HTTP method	주소
랜덤 음식 목록 조회	GET	/api/foods/random?size=30
선호도 저장	POST	/api/favors

<표 5> 선호도 api

기능	HTTP method	주소
비회원 채팅	WebSocket	/api/public-chat
회원 전용 채팅	WebSocket	/api/chat

<표 6> WebSocket 채팅 api

이 중 websocket 채팅 api만을 설명한다.

비회원 채팅의 경우 서버 리소스의 한계로 사용자 당 일일 채팅 가능 횟수를 20회로 제한한다. Request Message에 대한 양식은 아래와 같다. 채팅 내역 전체와 사용자의 새로운 input 그리고 답변 재생성할지에 대한 여부를 JSON 형태로 요청한다.

```
{
  "input": "피자 때문에 배가 아픈데 속이 편한 음식을 알려줘",
  "history": [
    ["안녕하세요!", "안녕하세요! 반갑습니다."],
    [
      "저녁 메뉴 추천해줘",
      "오리고기, 김치찌개, 생선구이, 들깨나물무침 등이 좋은 저녁 식사가 될 수 있습니다."
    ]
  ],
  "regenerate": false
}
```

request message 예시

이에 대한 서버의 메시지는 event와 response의 형태로 전달된다. Event의 종류에는 text_stream/stream_end/error가 존재하며, 각각 답변 생성 중/답변 생성 완료/오류 발생을 가르킨다.

```
{
  "event": "text_stream",
  "response": "부드러운"
}
```

response message 예시 - text_stream 1

```
{
  "event": "text_stream",
  "response": "부드러운 계란말이나, 매콤한 김치찌개"
}
```

response message 예시 - text_stream 2

```
{
  "event": "text_stream",
  "response": "부드러운 계란말이나, 매콤한 김치찌개, 콩나물국, 그리고 야채볶음밥
등이 속을 편안하게 해 줄 수 있는 음식입니다. 이러한 음식들은 소화가 잘 되고 건강에도
좋으며, 간단하면서도 맛있어서 야식으로 많이 즐겨먹는 음식입니다"
}
```

response message 예시 - text_stream 완료

```
{
  "event": "stream_end",
  "response": ""
}
```

response message 예시 - stream_end

```
{
  "event": "error",
  "response": "일일 최대 횟수에 도달했습니다."
}
```

response message 예시 - error

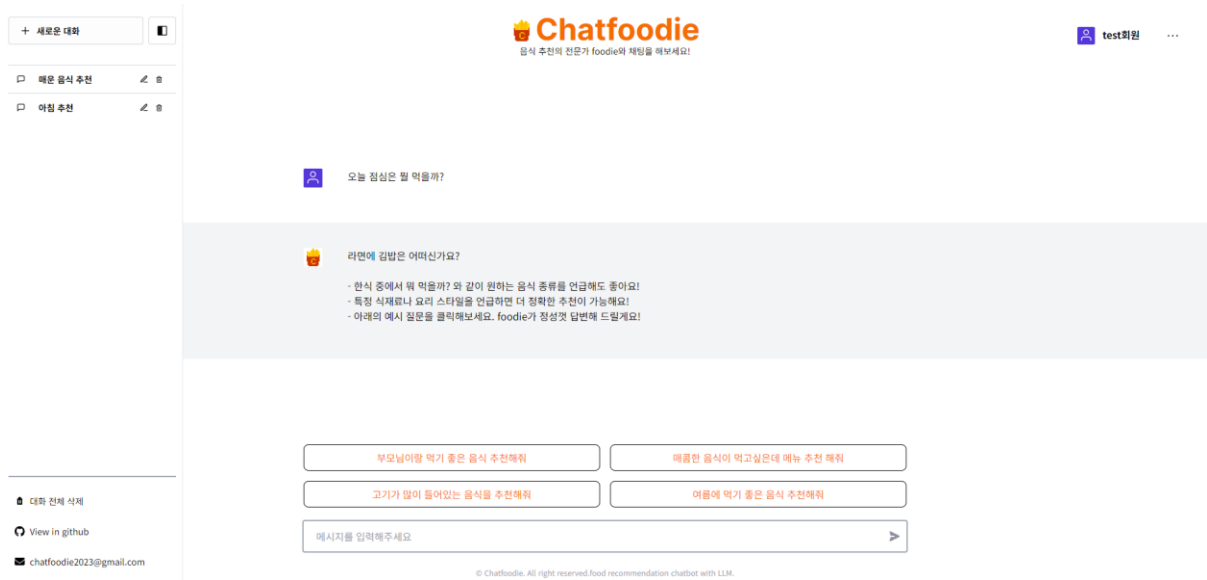
회원의 채팅의 경우 별도의 채팅 내역 대신 채팅방 번호를 받아, 서버의 db 에서 채팅내역을 조회해 답변을 생성한다. 또한, 사용자의 메시지를 전처리 과정을 통해 저장된 선호도 정보를 메시지에 추가하여 챗봇 api 를 호출하게 된다. 이 과정을 통해 사용자는 개인정보로 설정해놓은 자신의 선호도에 따라 맞춤형 대답을 얻을 수 있다.

```
{
  "input": "피자 때문에 배가 아픈데 속이 편한 음식을 알려줘",
  "chatroomId": 1,
  "regenerate": false
}
```

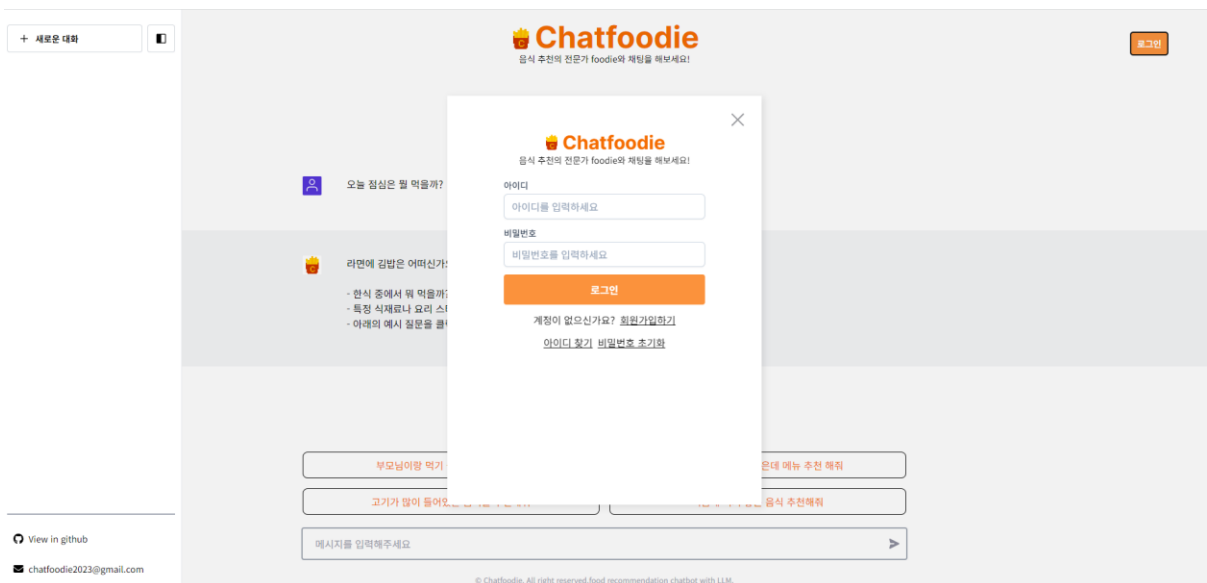
회원 request message 예시

3.5. 프론트 페이지 개발

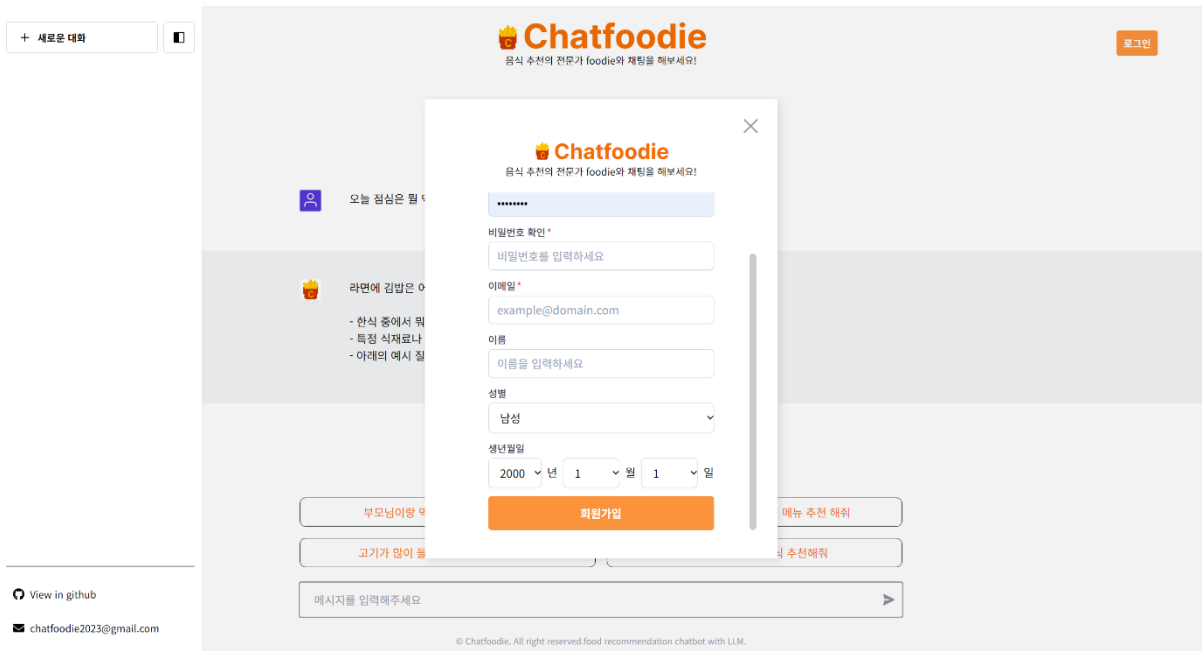
Next.js와 TypeScript를 사용하여 채팅 웹 UI를 제작하였다. 필요한 기능(화면)으로 로그인, 회원가입, 이메일 인증, 메인 UI, 선호도조사, 채팅 방이 있다.



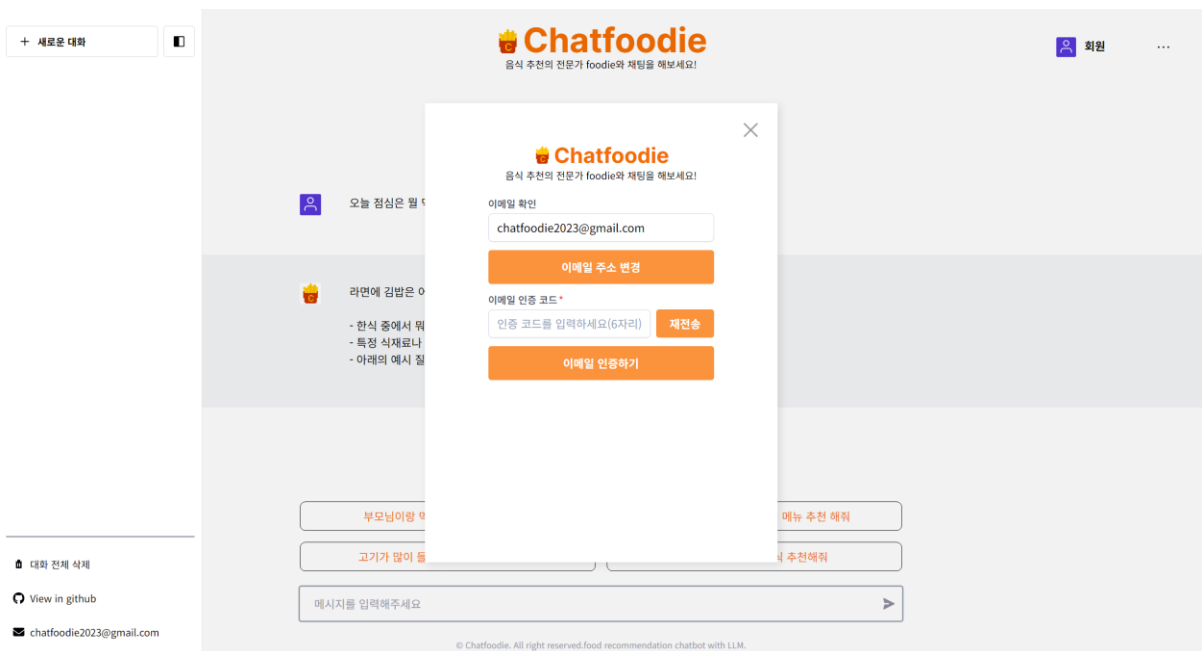
<그림 1 2 > 메인 화면



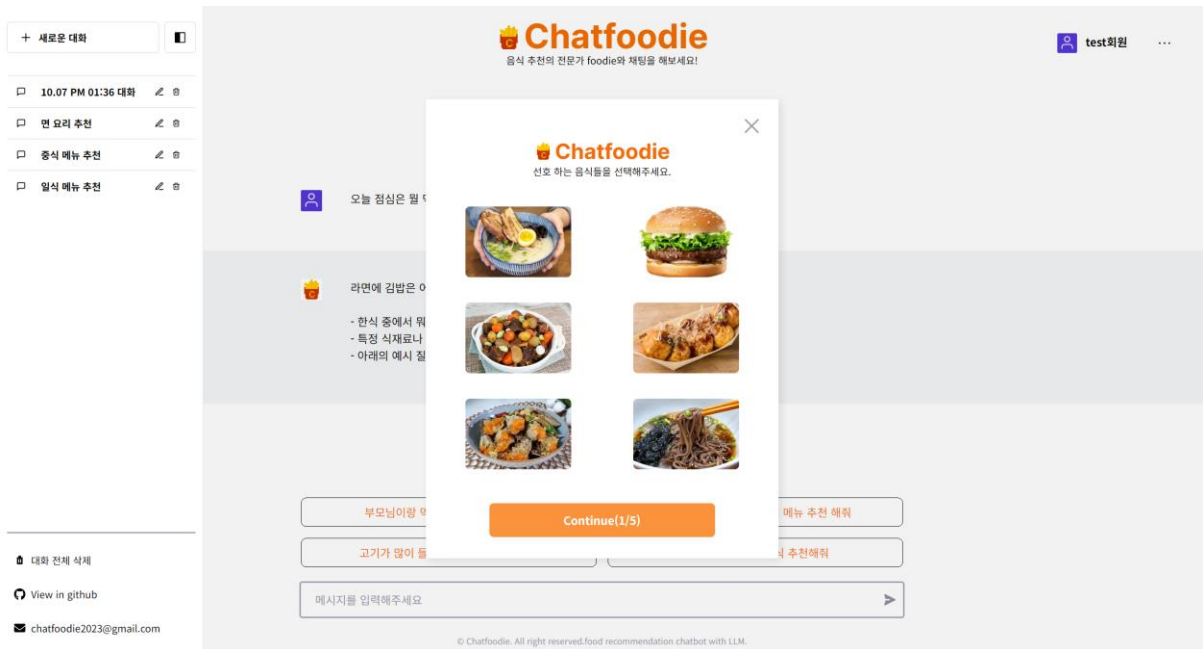
<그림 1 3 > 로그인 화면



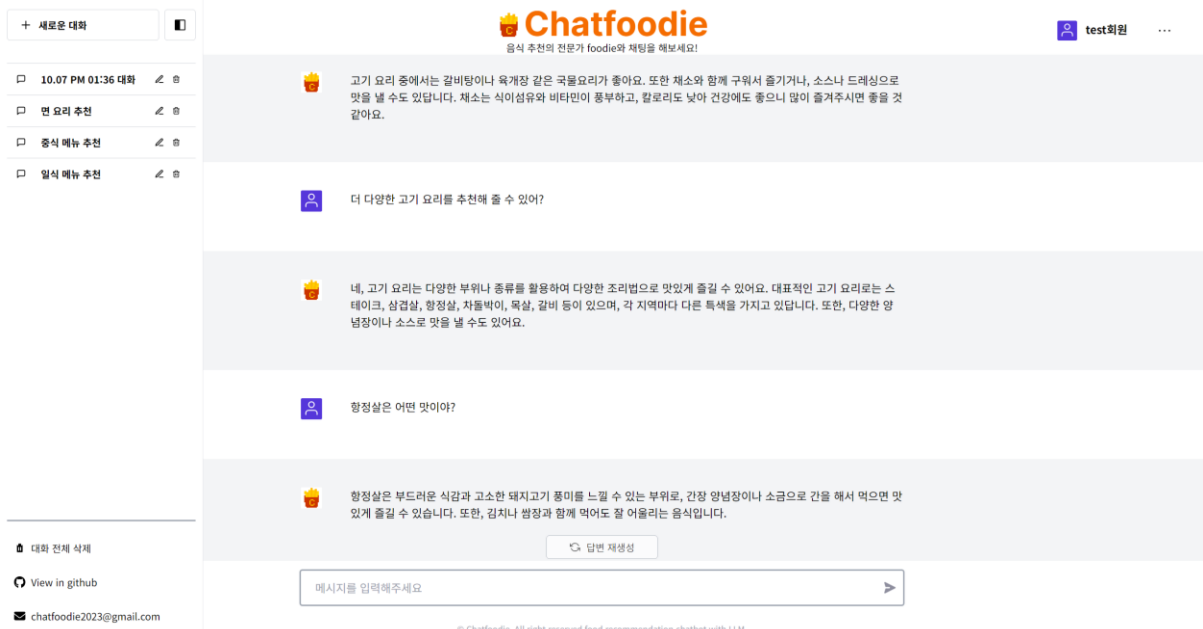
<그림 14> 회원가입 화면



<그림 15> 이메일 인증 화면



<그림 16> 선호도 조사 화면



<그림 17> 채팅방 화면

메인 화면

메인 화면에는 로고와 대화 창 및 입력창이 표시되고 입력 창 위의 답변 재생성 버튼을 통해 답변을 재생성 할 수 있다. 메인 화면 좌측에는 대화 생성 및 대화 기록 및 기록 삭제기능 등이 위치해 있고, 우측의 버튼을 통해 로그인 및 회원가입이 가능하다.

로그인 화면

메인 UI 우측 상단의 로그인 화면을 누르면 화면 위에 로그인 창이 뜨고 아이디와 비밀번호를 입력하고 로그인 버튼을 누르면 로그인이 가능하다

회원가입 화면

로그인 화면에서 회원가입하기 버튼을 누르면 화면 위에 회원가입 창이 뜨고 정보를 넣고 회원가입 버튼을 누르면 이메일 인증 창으로 넘어간다.

이메일 인증 화면

회원가입 창에서 입력한 이메일 주소에서 인증코드를 확인하고 입력한 뒤 이메일 인증하기 버튼을 누르면 회원가입이 완료된다.

3.6. 사용자의 선호 음식에 따른 대답 테스트

사용자의 입력 텍스트는 서버에서 전처리된 후 선호도 정보와 함께 챗봇에 전달된다. 서버에는 사용자가 선호하는 음식 목록이 저장되어 있으며, 이 목록이 사용자 입력 앞에 추가되어 챗봇에게 전송된다. 이를 통해 챗봇은 각 사용자의 음식 선호도에 따라 맞춤형 답변을 생성하게 된다. 아래는 "음식 추천해줘"라는 질문에 대한 챗봇의 사용자의 선호 음식 별 대답이다.

선호 음식	챗봇의 응답
햄버거, 스테이크, 치킨, 찜닭, 닭꼬치, 탕수육, 카츠산도	닭꼬치와 탕수육을 추천드려요! 고기의 담백함과 소스의 감칠맛이 조화로운 조합이에요. 중국집이나 분식집에서 많이 즐겨먹는 음식 중 하나입니다. 튀김의 바삭함과 소스의 풍부한 맛이 어우러져 맛있게 즐길 수 있어요.
핫도그, 토스트, 호떡, 삼각김밥, 라면	소시지와 토스트, 베이컨과 계란을 넣은 와플, 치즈와 감자튀김, 햄과 피망을 넣은 샌드위치, 새우와 야채를 넣은 볶음면 등이 좋은 선택일 것입니다.
짬뽕, 김치찌개, 부대찌개, 콩나물국밥	콩나물국밥은 시원하고 담백한 맛이 일품이며, 김치찌개나 부대찌개도 매콤한 맛으로 많은 사람들이 좋아하는 메뉴입니다. 또한, 비빔밥이나 떡볶이 같은 한국 전통 음식도 추천할 만합니다. 다양한 재료와 양념을 조합하여 자신만의 독특한 음식을 만들어보세요!
라면, 텐동, 연어덮밥, 마제소바, 카츠동, 후토마키	연어덮밥, 텐동, 에비동, 라멘, 돈부리, 규동 등은 다양한 맛과 스타일로 인기있는 음식입니다. 일본 요리 중에서도 맛있게 즐길 수 있으며, 특히 연어덮밥은 신선한 연어와 부드러운 밥알, 풍부한 소스가 어우러져 매우 맛있게 느껴집니다.
쌀국수, 짬뽕, 라면, 우동, 밀면	쌀국수, 짬뽕, 라멘, 우동, 메밀국수 등 면요리는 다양한 맛과 풍미를 느낄 수 있는 음식입니다. 또한, 순살치킨이나 양념갈비 등의 고기 요리도 야식으로 즐겨먹을 수 있는 건강한 음식입니다.
돈카츠, 텐동, 카츠동, 카츠산도	돈카츠는 부드럽고 바삭한 식감과 함께 고소한 고기 소스와 함께 즐기는 일본 요리입니다. 텐동은 바삭하고 뜨거운 튀김요리로, 다양한 재료와 양념을 추가하여 맛과 풍미를 더할 수 있습니다. 카츠동은 돼지고기 살과 간단한 야채를 함께 볶아 간장 소스로 간을 맞춘 일본식 덮밥 요리입니다. 카츠동은 돈까스 덮밥 중에서도 가장 유명한 음식 중 하나입니다.

<표 7> 질문 “음식 추천해줘”에 대한 선호 음식 별 응답

4. 결론 및 향후 발전 방향

본 과제에서는 LLM을 기반으로 한 AI 챗봇 서비스를 구현하였다. 모델 학습에서 Self-instruct와 LoRA와 같은 최신 기법을 접목해 성능을 최적화하는 동시에 학습 시간과 자원 소모를 줄일 수 있었다. 최종적으로 학습된 모델을 챗봇 서비스로 제공하기 위해 서버와 web 화면을 만들어 많은 사용자들이 쉽게 사용할 수 있도록 배포하였다. 배포된 페이지는 <https://chatfoodie.net> 에서 확인 가능하다.

본 과제는 사용자의 세부적인 음식 취향과 요구사항을 바탕으로 맞춤형 음식 메뉴를 추천하는 것이 주 목적이었다. 통상적인 추천 시스템과는 달리, LLM 기반 챗봇으로 사용자와의 대화에서 보다 구체적이고 다양한 맞춤형 추천을 제공할 수 있다. 쉽게 접근할 수 있는 웹 환경과 간편한 웹 UI를 통해 사용자는 단순히 채팅만을 할 수 있게 하여 서비스에 대한 진입장벽이 낮추었다. 또한, 채팅방 생성 및 채팅 내역 기록, 챗봇의 이전 답변 재생성 등의 편의 기능들을 제공하여 사용하기에 불편함이 없도록 하였다.

현재의 추천 시스템은 사용자 별 선호 음식 위주로 구현되어 있다. 선호 음식 외에, 사용자의 다양한 생활 스타일, 건강 상태, 알레르기 정보 등을 추가적으로 반영하여 보다 맞춤형인 서비스를 제공할 수 있을 것이다. 챗봇 답변의 만족도를 좋아요 버튼과 같은 수단으로 사용자 피드백을 수집할 수 있다. 이를 반영하여 사용자가 만족한 질문-답변 세트로 모델을 지속적으로 재학습하여 업데이트함으로써 챗봇의 성능을 향상시킬 수 있을 것이다. 현재는 텍스트 기반의 답변 정보만을 제공하지만, 추천되는 음식에 해당하는 사진 등의 시각적인 정보를 함께 제공하여 사용자 경험을 향상시킬 수 있을 것이다.

마치며, 본 과제를 통해 개발된 AI 챗봇 서비스는 사용자 중심의 맞춤형 서비스 제공을 통해 기존의 추천 시스템의 한계를 극복하고, 새로운 추천 서비스의 가능성을 제시하였다. 향후 연구를 통해 더욱 발전된 서비스를 제공할 수 있기를 기대한다.

5. 구성원별 역할 및 개발 일정

이름	역할
안혜준	<ul style="list-style-type: none"> 서비스 시스템 디자인 데이터베이스 설계 개발 환경 및 서버 배포 환경 구축 챗봇 websocket API 구축 및 테스트 백엔드(인증 필터, 웹소켓 채팅 api) 개발 프론트엔드(웹소켓 채팅 UI) 개발
박성민	<ul style="list-style-type: none"> self-instruct를 위한 시드 데이터 수집 self-instruct 코드 작성 및 데이터 생성 모델 fine-tuning 백엔드(유저, 인증 관련 api) 개발 프론트엔드(회원가입, 이메일 인증, 유저 관련 UI) 개발
박진영	<ul style="list-style-type: none"> 서비스 UI디자인 self-instruct를 위한 시드 데이터 수집 데이터베이스 쿼리 작성 백엔드(채팅방, 음식/선호도 api) 개발 프론트엔드(채팅방 drawer, 선호도 조사 UI) 개발

<표 8> 구성원별 역할

6월		7월					8월					9월					
4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	
DB 설계																	
	UI디자인																
	서버환경 구축																
		seed data수집															
		언어 모델 환경 테스트															
		백엔드 환경 구축															
		Self-instruct 수행 Fine-tuning 수행															
							백엔드 개발										
									백엔드 api 배 포/버그 수정								
											프론트 개발						
											배포 테스트						
												실제 배포 및 운영					
												버그 유지보수					
												최종 발표 준 비 및 보고서					

<표 9> 개발 일정

6. 참고 문헌

- Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models." arXiv preprint arXiv:2106.09685 (2021).
- Wang, Yizhong, et al. "Self-instruct: Aligning language model with self generated instructions." arXiv preprint arXiv:2212.10560 (2022).
- "Polyglot Ko 5.8B." Hugging Face, EleutherAI, 2021, <https://huggingface.co/EleutherAI/polyglot-ko-5.8b>.
- "KoAlpaca Polyglot Ko 5.8B." Hugging Face, beomi, 2023, <https://huggingface.co/beomi/KoAlpaca-Polyglot-5.8B>
- "SNS 고도화 데이터". AI Hub. <https://aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&aihubDataSe=realm&dataSetSn=71343>

모든 코드는 [jagaldol/chat-foodie | GitHub](#) 저장소에 오픈소스로 공개되어 있습니다.