**< Personalized Music Recommendation Chatbot >**



**Subject : 종합설계프로젝트 2조**

**Class number : ICE3037\_48**

**Name : 2016313014 강동준**

**2016312917 김지연**

**2016311146 배지현**

**Date : 2022. 05. 30.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Capstone Design 과제 결과보고서 | | | |
| 과 목 명 | 종합설계프로젝트\_ICE3037\_48 | | | |
| 과 제 명 | Personalized Music Recommendation Chatbot | | | |
| 팀 명 | Team 2 | | | |
| 지도교수  (과제책임자) | 학과(부) | 소프트웨어융합대학 | 성명 | 이진규 |
| 팀장  (대표학생) | 학과(부)/학년 | 컴퓨터공학과/4 | 성명 | 김지연 |
| E-mail | 0715jiyounkim@gmail.com | 휴대전화 | 010-5042-6675 |
| 팀원 | 학과(부)/학년 | 컴퓨터공학과/4 | 성명 | 강동준 |
| 학과(부)/학년 | 컴퓨터공학과/4 | 성명 | 배지현 |
| 과제기간 | 2022 년 3월 3일 – 2022 년 5월 30일 (3개월) | | | |
| 위와 같이, 성균관대학교 산학협력선도대학(LINC+)육성사업단의 Capstone Design 지원사업 과제결과보고서를 제출합니다.    2022년 5월 30일 (월)  팀 장: 김지연(인)  지도교수: 이진규(인) | | | | |

|  |  |
| --- | --- |
| Capstone Design 과제 결과보고서 요약 | |
| **과 제 명** | Personalized Music Recommendation Chatbot |
| **팀 구성원 및 역할** | 강동준: 챗봇 시스템, classifier, 음악 추천 logic, 카카오 음악 데이터셋 전처리  김지연: 챗봇 시스템, 음악 추천 logic, 감성대화말뭉치 데이터 전처리, Chatbot Data v1.0 전처리  배지현: 챗봇 시스템 통신 및 스레드 관리, 서버, 프론트 엔드 구현 |
| **개발동기**  **목적 및 필요성** | 사용자 맞춤 추천 알고리즘이 많은 분야에서 사용이 되고 있다. 음악 추천도 그 중 하나이지만 많은 상용화 알고리즘은 사용자 감정을 배제한 채 기존 선호도 및 인기 차트로 음악을 추천하고 있다. 하지만 많은 연구를 통해 음악은 개인의 감정에 많은 영향을 미친다는 것을 알 수 있다.  사용자의 감정과 음악추천을 함께 고려한 선행 연구도 존재하지만, 해당 연구에서는 현재의 감정 상태와 선택하여 들은 음원의 무드를 수기로 작성하는 전문가 집단이 필요하며, 사용자 감정 및 음원 무드가 수기로 진행되기 때문에 적은 양의 데이터셋을 활용할 수밖에 없다. 또한, 사용자의 현재 감정을 직접 수치로 입력해야 한다는 번거로움이 있다. 혹은 음악 특징 추출에만 집중하여 사용자의 실시간 감정을 빠르게 업데이트 하여 현재 감정에 맞는 음악을 추천해주지 못한다는 문제도 있었다.  따라서 우리는 사용자에 감정에 알맞은 음악을 추천하는 챗봇을 통해 사용자가 직접 자신의 감정을 선택하거나 플레이리스트를 고르지 않아도 사용자와의 대화와 사용자 선호도를 바탕으로 음악을 추천하고자 한다. 또한 사용자가 챗봇과 계속해서 대화를 함에 따라 변화하는 감정까지 실시간으로 고려하여 고정된 플레이 리스트를 재생하는 것이 아닌 감정을 통해 새롭게 재구성된 추천 시스템을 제공하고자 한다. 그리고 현대사회에서 친근하게 자신의 이야기를 진솔하게 털어놓고 대화를 나눌 수 있는 챗봇을 필요로 하는 많은 사람들에게 서비스를 제공하는 것을 목표로 한다. |
| **과제 해결 방안 및 과정** | 수행 절차는 채팅, 감정분석, 음악 추천, 서비스 구현 단계가 있다.  채팅 모델은 자연어 처리 분야에서 압도적인 성능을 가진 Transformer 모델 중 한 번에 토큰을 하나씩 생성하는 GPT모델을 사용하였다. 이 종류의 모델은 사전학습(pre-train)과 미세 조정(fine-tuning)이라고 불리는 단계를 수행하는데, 본 과제에서는 한국어로 사전학습 된 SKT-KoGPT와 AIHub의 감성대화말뭉치를 사용하여 fine-tuning하여 사용자와 감성 대화를 나눌 수 있는 챗봇을 구현하였다.  음악 추천을 위한 감성분석 단계에서는 감정 소분류를 제대로 구별하기 힘들고 분류할 항목이 너무 많아 대분류를 분류하는 것으로 결정하였다. 또한 GPT의 output word vector를 사용하기에는 전체 포워딩 시간이 길어 context벡터는 GRU로 구하고 이 벡터를 Fully Connected Layer 통하여 감정을 분류한다.  즉 챗봇 시스템은 KoGPT-2 fine-tuning, 감정 분석 classifier는 GRU와 Fully Connected Layer를 통해 구현되었다.  음악 추천은 사용자로부터 입력된 대화 내역 텍스트를 통해 감정 분석을 진행하고 분류된 사용자 감정에 대한 설문조사에서 작성된 감정 별 장르 추출한다. 이후 이 장르로 음악을 1차 필터링하고 2차로 사용자 대화 내역 텍스트, 필터링 된 음악 태그의 유사도를 측정하여, 가장 높은 유사도 10개의 곡들을 추출한다.  서비스 구현 gateway server, front-end server, chatbot system, DB로 나눠 구성하였다. gateway server, front-end server, DB는 같은 하드웨어 즉, 같은 네트워크 안에 있지만 chatbot system은 AI 모델이기 때문에 GPU를 사용할 수 있는 외부 네트워크로 분리하였다. 따라서 gateway server가 chatbot system과 front-end server를 연결해 주는 역할을 하며, DB는 단순히 사용자 정보를 저장하는 역할과 더불어 공유 repository로 사용하여 chatbot system과 front-end server의 추가적 통신 수단을 제공한다. |
| **★대표적 결과물** | 텍스트이(가) 표시된 사진  자동 생성된 설명  채팅 응답 결과    음악 추천 결과      최종 구현 UI 결과물 |

**Table of Contents**

**1. 과제 개요 ---------------------------------------------- 1**

**2. 추진 배경 --------------------------------------- 1**

**3. 해결 방안 및 과정 ---------------------------------------- 2**

**3-1. 챗봇 ---------------------------------------------- 2**

**3-2. 음악 추천 -------------------------------------4**

**3-3. 감성 분류 --------------------------------------- 4**

**3-4. 서비스 시스템 ------------------------------------- 5**

**4. 해결 방안 세부 내용 ---------------------------------- 9**

**4-1. 챗봇 ---------------------------------------------- 9**

**4-2. 음악 추천 -------------------------------------11**

**4-3. 감성 분류 --------------------------------------- 11**

**4-4. 서비스 시스템 ------------------------------------- 12**

**5. 결과 ------------------------------------------------- 20**

**5-1. 챗봇 시스템 --------------------------------------- 20**

**5-2. 감정 분석 ----------------------------------------- 23**

**5-3. 음악 추천 -------------------------------------------- 24**

**5-3. 시나리오 -------------------------------------------- 24**

**6. 기대효과 ---------------------------------------------- 29**

**6-1. 챗봇 ---------------------------------------------- 29**

**6-2. 음악 스트리밍 서비스 ------------------------------ 29**

**6-3. 사용자 채팅 기반 음악 추천 알고리즘------------------- 29**

**6-4. 웹서비스--------------------------------------------- 29**

**7. 한계점 및 개선사항 --------------------------------------------- 29**

**8. 수행 일정 및 역할분담 ---------------------------------------- 30**

**9. 참고 문헌------------------------------------------- 31**

**표 목차**

**[표 1] 코드와 메시지 타입----------------------------------------- 13**

**[표 2] BLEU, ROUGE score-------------------------------------23**

**[표 3] 수행 일정---------------------------------------------30**

**그림 목차**

**[그림 1] 언어 모델의 종류-------------------------------------2**

**[그림 2] Transformer 구조---------------------------------3**

**[그림 3]** **순환 신경망의 구조-----------------------------------4**

**[그림 4] 제안서의 서비스 시스템 구조------------------------6**

**[그림 5] 변경된 서비스 시스템 구조-------------------------8**

**[그림 6]** **전처리된 챗봇 데이터셋-----------------------------9**

**[그림 7] Classifier 구조 예시-------------------------------11**

**[그림 8] 음악 추천 알고리즘---------------------------------12**

**[그림 9] 전체 시스템 구조---------------------------------12**

**[그림 10] Raspberry Pi 라즈베리 파이 4 모델 B------------------------13**

**[그림 11] 패킷 구조(단위: Byte)---------------------------------------13**

**[그림 12]** **DB schema---------------------------------------------------14**

**[그림 13]** **Front-end Server 로그 관리 구조------------------------------15**

**[그림 14] Front-end Server 대화 구조-----------------------------------15**

**[그림 15] Gateway Server 전체 구조--------------------------------------16**

**[그림 16]** **Gateway 역할을 하는 app\_gw 프로그램 구조-------------------16**

**[그림 17] Chatbot 시스템 구조-------------------------------------------17**

**[그림 18] 로그인 화면-------------------------------------------17**

**[그림 19] 설문조사 화면-----------------------------------------18**

**[그림 20] 대화 화면--------------------------------------------19**

**[그림 21] 대화 음악 추천 화면-----------------------------------19**

**[그림 22] 챗봇 대화 예시-----------------------------------------22**

**[그림 23]** **감정 분석 예시 -----------------------------------------24**

**[그림 24] gateway 패킷 처리 과정 -------------------------------24**

**[그림 25] 시나리오 여름 여행 --------------------------------------25**

**[그림 26]** **시나리오 취업 성공 ---------------------------------------26**

**[그림 27] 시나리오 이별 ---------------------------------------------27**

**[그림 28] 시나리오 EDM 축제 ----------------------------------------28**

그래프 및 도표 목차

**[그래프 1] Top-k ------------------------------------------10**

**[그래프 2] train\_loss ---------------------------------------20**

**[그래프 3]** **train\_ppl ----------------------------------------20**

**[그래프 4] valid\_loss ----------------------------------------21**

**[그래프 5] valid\_ppl -----------------------------------------21**

1. 과제 개요

요즘 사용자 추천 알고리즘이 많은 분야에서 사용이 되고 있다. 음악 추천도 그 중 하나이지만 많은 사람들은 개인이 선호하는 가수 또는 순위 차트에 있는 곡들을 단순히 재생하는 형태로 많이 사용하고 있다. 음악은 개인의 감정에 많은 영향을 미친다는 것을 뒷받침해주는 많은 연구가 활발히 진행됨을 통해 확인할 수 있었는데, 따라서 우리는 사용자에 감정에 알맞은 음악을 추천하는 챗봇을 통해 사용자가 직접 자신의 감정을 선택하거나 플레이리스트를 고르지 않아도 사용자와의 대화 등 여러 feature를 바탕으로 음악을 추천하고자 한다. 또한 사용자가 챗봇과 계속해서 대화를 함에 따라 변화하는 감정까지 실시간으로 고려하여 고정된 플레이 리스트를 재생하는 것이 아닌 감정을 통해 새롭게 재구성된 추천 시스템을 제공하고자 하며, 현대사회에서 친근하게 자신의 이야기를 진솔하게 털어놓고 대화를 나눌 수 있는 챗봇을 필요로 하는 많은 사람들에게 서비스를 제공하는 것을 목표로 한다.

2. 과제 추진배경(개발 동기) 및 목적, 필요성

기존 맞춤형 음악 추천 연구로는 사용자 감성과 음원 무드기반 음악 추천 시스템[1]과 음악 추천을 위한 가사정보 및 음악신호 기반 특성 탐색 연구[2]가 있다.

사용자 감성과 음원 무드기반 음악 추천 시스템은 맞춤형 음악 추천 서비스를 제공하지만, 현재의 감정 상태와 선택하여 들은 음원의 무드를 수기로 작성하는 전문가 집단이 필요하며, 사용자 감정 및 음원 무드가 수기로 진행되기 때문에 적은 양의 데이터셋을 활용할 수밖에 없다. 또한, 사용자의 현재 감정을 직접 수치로 입력해야 한다.

우리가 구현하는 모델은 사용자가 현재 감정 상태를 수기로 입력할 필요 없이, 챗봇의 대화를 통해서 사용자 감정을 탐지하며 이를 바탕으로 음악 추천을 진행하기 때문에 기존 연구보다 사용자가 더욱 편하고 자연스럽게 사용 가능하다는 장점이 있다. 또한, 기존 연구에서는 18명의 전문가를 통해 적은 양의 수기로 생성된 데이터셋을 사용하지만, 우리 모델은 대량의 데이터셋을 인공지능 모델에 넣어 주기 때문에 더욱 정확하다는 차별성이 있다.

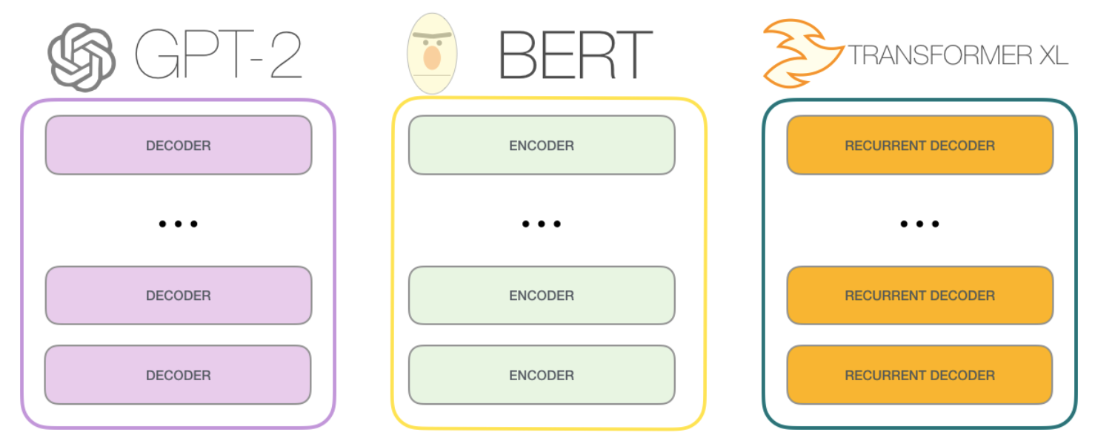
또한, 음악 추천을 위한 가사정보 및 음악신호 기반 특성 탐색 연구는 사용자에 맞는 음악을 추천한다는 점에서 우리가 구현하려는 모델과 비슷하지만, 사용자의 실시간 감정을 빠르게 업데이트 하여 현재 감정에 맞는 음악을 추천해주지 못한다는 문제가 있다. 우리 모델은 해당 문제를 챗봇과의 대화를 통한 정확한 현재 감정 파악을 통해서 진행한다.

우리 모델은 챗봇을 통해 실시간으로 사용자가 감정적 대화를 할 수 있는 서비스를 제공하며, 사용자의 현재 감정과 채팅 내용을 바탕으로 맞춤형 음악 추천을 진행하는 서비스를 제공한다.

3. 과제 해결 방안 및 과정

3-1. 챗봇 시스템

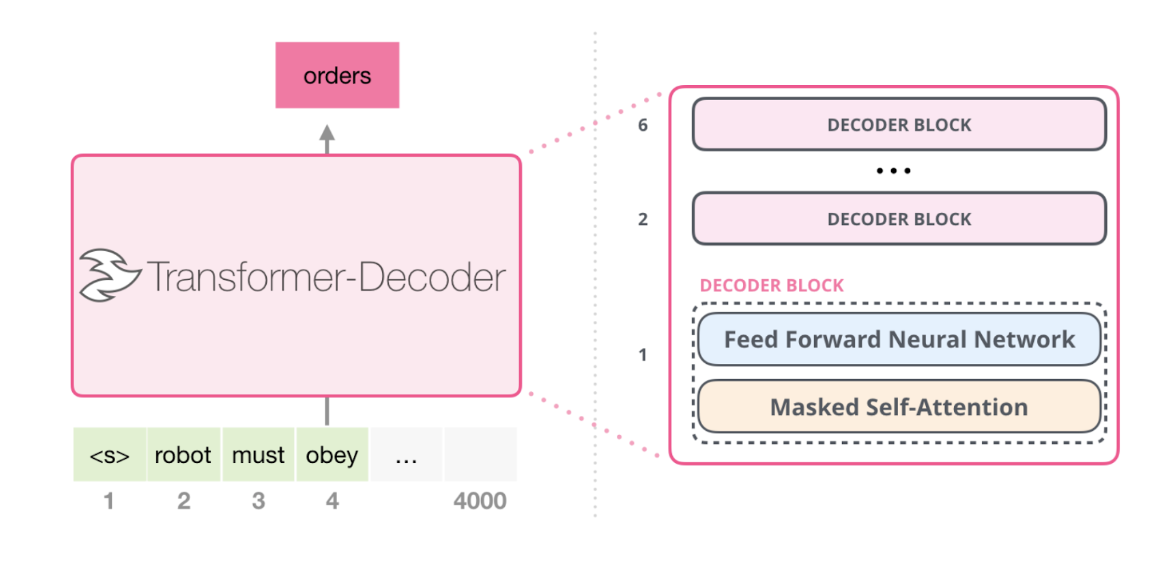
챗봇 시스템에는 GPT-2 와 같은 생성 모델이 주로 사용되고 있다. GPT-2는 영어 데이터 기반의 모델로, 40 GB의 WebText 데이터셋으로 학습되었고, start 토큰을 시작으로 문장 생성의 다음에 올 단어를 하나씩 생성하는 모델이다. BERT 모델에서는 Transformer 인코더를 사용하는 반면, KoGPT-2 와 같은 GPT 기반 모델은 Transformer 디코더를 사용해서 토큰을 한 번에 하나씩 (autoregressive) 생성한다. 생성된 토큰은 모델의 input으로 들어가서 다음 토큰을 생성하는데 사용된다. 여기서 Transformer 디코더의 stack 개수가 GPT-2의 모델 크기를 결정하며, 모델은 GPT-2 small, medium, large, extra large으로 구성된다.



<그림 1: 언어 모델의 종류>

GPT-2의 모델 구조는 아래 그림과 같으며, Transformer 디코더 여러 개로 구성된 것을 볼 수 있다. 각 디코더 블록은 masked-self-attention 과 feed forward neural network으로 구성되어 있다. Masked-self attention 은 모델이 현재까지의 input token 만을 참조할 수 있도록 하며, 현재 input의 오른쪽 token 들은 참조하지 못하도록 한다.

GPT-2 모델을 사용한 최근 챗봇 시스템으로는 구글 람다 (LaMDA, Language Model for Dialogue Applications) 가 있다. 람다의 높은 성능은 다양한 성능 측정 방식의 복합에 있으며, 다음 7가지 방식을 사용한다. 안정성 (Safety)는 답변이 사용자에게 해를 끼치거나 차별적인 발언을 수행하는지에 대한 평가이다. 합리성 (Sensible) 은 답변이 대화 내용과 모순되지 않고 맞는 흐름을 생성하는지에 대한 것이다. 구체성 (Specificity)는 답변이 일반적이지 않고 구체적인지에 대한 부분이다. 재미 (Interestingness)는 답변이 사용자에게 재미를 주는지에 대한 부분이다. Fact-based는 답변이 얼마나 사실 기반 된 내용인지에 대한 것으로, 정보성과 인용 정확성을 포함한다. 도움(Helpfulness)는 답변이 사용자가 원하는 정보를 얼마나 정확하게 전달해주는지에 대한 부분이다. 역할 일관성(Role Consistency)은 답변들이 얼마나 역할에 일관적인지에 대한 파악이다.



<그림 2: Transformer 구조>

구글 리서치에서는 람다 모델을 정확하게 학습시키기 위해 클라우드 워커들을 람다와 대화하도록 하여, 대화의 점수를 매기도록 하였다. 안정성 부분의 편향성을 줄이기 위해 다양한 인종, 성별, 나이의 사람들을 사용하여 람다와의 대화를 평가하도록 했다. 람다의 구조는 GPT 와 유사한 구조의 트랜스포머 기반 모델이다. Generator 모델로 답변을 생성한 후, 답변이 Classifier 모델을 통과하도록 하여, Safety, Sensible, Specific의 성능 측정 항목별 점수를 생성하도록 한다. 여기서 낮은 안정성 점수를 받은 답변을 제거하며, Sensible, Specific, Interesting 점수가 높은 답변을 하도록 한다.

위의 GPT-2는 대량의 영어 데이터셋으로 학습된 모델이기 때문에 한국어 기반의 챗봇에는 높은 성능을 보이지 못한다. 이러한 문제를 해결하기 위해 SK 텔레콤에서 한국어 데이터셋으로 학습된 KoGPT-2 모델을 발표하였다. KoGPT-2는 최근 한국어 기반 챗봇 시스템 개발에 많이 사용되고 있으며, 한국어 데이터 40 GB 이상의 텍스트로 학습된 Transformer Decoder 언어 모델이다. KoGPT-2는 한국어 위키 백과, 모두의 말뭉치 v1.0, 청와대 국민 청원 등의 데이터를 사용하여 학습되었다.

따라서 우리는 현재 한국어에 대해 우수한 생성 성능을 보이는 KoGPT-2 모델을 사용하여 챗봇으로 사용하고자 하였으며, 아래와 같이 학습하였다.

1. 한국어 챗봇 데이터셋을 사람이 입력하는 q와 챗봇의 대답인 a로 나누어 tokenizing 하도록 전처리한다.
2. KoGPT-2 모델을 사용하여 fine-tuning 시킨다.

이때 사용되는 챗봇 데이터셋의 q와 a의 경우 사람과 챗봇의 대화를 1-turn으로 끊어서 학습을 진행하였다

3-2. 감정 분석 classifier

감정 분석 classifier의 경우, 사용자와 챗봇이 대화한 전체 context 중 사용자의 입력만을 고려하여 다음과 같은 [기쁨, 슬픔, 불안, 분노, 상처, 당황] 6가지 감정 class 중 하나를 구분하는 것을 목표로 한다.

Classifier는 기본적으로 Gated Recurrent Units (GRU)를 사용하여 구현한다. GRU란 RNN, LSTM과 같은 순환 신경망 중 하나이며, LSTM과는 달리 출력 게이트가 없는 간단한 구조를 가진다. 따라서 전체 context를 GRU에 인풋으로 넣고, 마지막 cell의 hidden state를 Fully-Connected Layer에 연결하여 최종적으로 softmax 함수를 통해 감정을 예측하는 방식이다.

텍스트, 시계, 표지판이(가) 표시된 사진

자동 생성된 설명

<그림 3: 순환 신경망의 구조>

3-3. 사용자 채팅 기반 음악 추천 알고리즘

사용자가 챗봇과 채팅을 진행하다가 좌측의 ‘Music’ 버튼을 클릭하면, 버튼 클릭 이벤트가 발생함에 따라 사용자가 챗봇에게 전송한 채팅 최근 5개가 MusicRecommendLogic 모델에게 전달된다. 또한, 사용자가 채팅 시작 직전에 작성한 폼의 내용에 따라 [기쁨, 슬픔, 불안, 분노, 상처, 당황] 순서로 감정에 따라 어떤 장르의 음악을 듣는지도 함께 MusicRecommendLogic 모델에게 전달된다. MusicRecommendLogic 모델은 1) 감정 분석 Classifier 모델과 2) 감정 기반 음악 추천 알고리즘으로 구성되어 있다.

먼저 사용자의 최근 5개 채팅 텍스트를 받아서 감정 분석 Classifier 모델을 통해 사용자의 현재 감정 분석을 진행한다. 여기서 감정 분석 Classifier 모델은 앞서 3-2을 통해 감성대화말뭉치 데이터셋으로 학습된 state dictionary를 로드 하여 초기화된 모델을 사용하여 사용자 감정 classification을 진행한다. 채팅으로 classification 될 수 있는 감정은 [기쁨, 슬픔, 불안, 분노, 상처, 당황] 로, 총 6가지 중 하나이다.

사용자 감정이 분류되면, 사용자가 폼에 작성한 내용대로 해당 감정에 따라 자주 듣는 장르 정보를 추출한다. 장르에 해당하는 곡들을 음악 csv 파일 5만개 row에서 필터링한다. 사용자가 입력한 채팅 내용과 유사한 음악이 추천될 수 있도록 채팅 내용과 음악 태그의 유사도를 측정한다. 구체적으로, 사용자가 입력한 채팅이 tokenizer를 통과해서 토큰 단위로 끊기도록 하고, 장르로 필터링 된 음악의 각 row의 태그 또한 tokenizer통과해서 토큰 단위로 끊어준다. 이때, 사용자가 입력한 텍스트의 토큰들과 가장 유사도가 높은 음악 태그 토큰들을 순서대로 나열하여 최대 10개의 음악을 순서 매긴다.

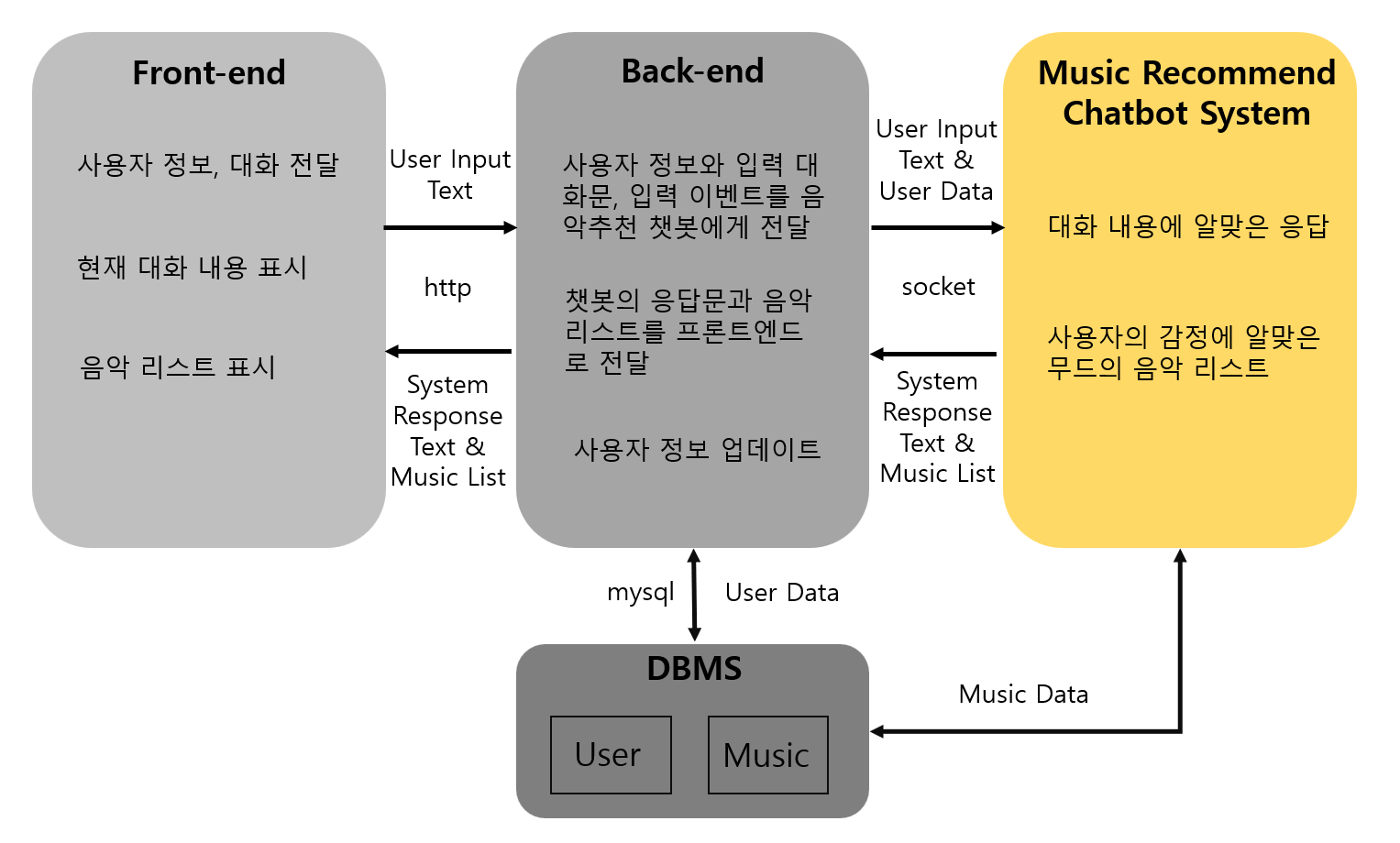
결과적으로, 사용자가 특정 기분에 따라 자주 듣는 장르의 음악을 기반으로, 사용자 채팅과 유사도가 높은 음악을 추출하여 맞춤형 음악 추천을 가능하도록 한다.

3-4. 서비스 제공

제안서에는 위와 같이 서비스 시스템을 구성할 것이라 계획했지만 하드웨어 성능, 구성의 물리적 위치, 통신 등의 원인으로 문제들이 발생하였다. 아래는 프로젝트 전반적인 구현 이슈 및 구현 중 발생한 문제들과 해결해 나간 방법을 기술한다.

a) 서버 구축

이 프로젝트 진행하는 모든 팀원이 웹 서비스 구축 경험이 없었고 관련 서비스 이용자도 없어서 AWS와 같이 클라우드 자원을 신청, 사용하기에는 시간이나 비용이 부담되었다. 따라서 무료 DDNS(동적 도메인 이름 시스템) 서비스를 이용하여 상시 동작할 수 있는 개인 컴퓨터, Raspberry Pi에 서버를 구축하였다. 그래서 해당 기기가 있는 네트워크의 무선 라우터(공유센터)에 포트 포워딩을 하여 해당 서비스를 배포하였다. 그 결과 해당 컴퓨터의 성능의 한계로 React, Flask, Django와 같은 무거운 프레임워크 및 라이브러리를 사용할 수 없었고 Apache, PHP와 같은 가벼운 프레임워크를 사용했고 그 결과 나머지 소켓 통신도 C언어로 낮은 레벨부터 구현했다.



<그림 4: 제안서의 서비스 시스템 구조>

b) Music Recommend Chatbot System Forwarding 시간

첫번째로 예상했던 문제이지만, 음악 추천 챗봇 시스템이 GPT 등 연산과정이 많은 AI 모델을 사용하기 때문에 서버가 동작하고 있는 Raspberry Pi 하드웨어에서 구동하기 어려웠다. 예를 들어, 기본적으로 사전 학습된 SKT-KoGPT 모델이 max\_len = 8로 forwarding 하는 시간만 30초가 넘게 걸려 서버 안 AI 모델의 정상 작동을 기대하기 어려웠다. 따라서 서버와 챗봇 모델을 분리할 필요성이 생겼고, Raspberry Pi에 동작하는 서버는 client와 모델을 연결해 주는 gateway 역할이 추가적으로 필요했다. 따라서 웹 서버 안의 gateway는 1-32개의 챗봇 채널을 유지하며 client로부터 생성된 응답 요청을 챗봇 채널과 연결해 주는 역할을 한다.

c) 추천 음악 데이터와 음악 DB

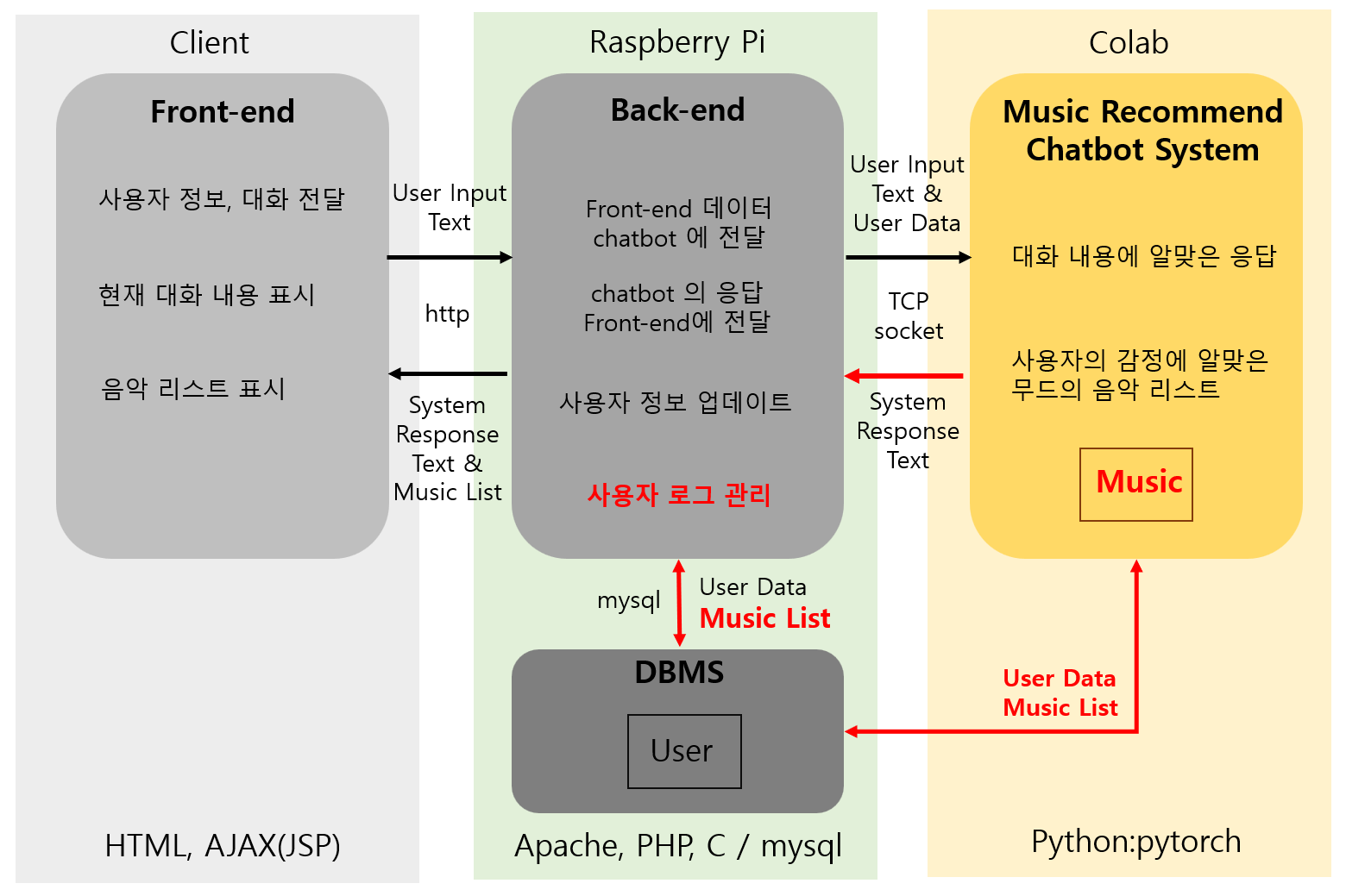
위 3.4.1의 이유로 웹 서버와 채팅 모델이 물리적으로 분리되면서 음악 데이터를 게이트 서버 내부에 유지하는 것이 효율성이 떨어지게 되었다. 따라서 채팅 모델 시스템이 음악 DB를 포함하게 되었고 채팅 모델과 음악 추천 모델이 하나의 시스템이 되어 게이트 서버와 같은 채널로 통신하게 되었다. 이는 게이트 서버의 구현이 할당된 챗봇 채널 하나의 응답이 timeout시간 내에 도착할 때까지 채널을 차지하고 있으므로 음악 추천 모델과 채팅 모델을 한 번에 하나만 쓸 수 있는 문제가 발생했다. 심지어 추천 시간이 평균적으로 3초를 차지하여 이는 비효율적인 자원 분배였다. 따라서 음악 추천 모델과 채팅 모델을 각기 다른 스레드로 분리하였고, 중간에 blocking que를 두어 두 스레드간 데이터 통신을 처리하였다. 또한 추천 모델에서 생성한 음악 추천 리스트 데이터는 패킷으로 보내지 않고, 서버의 DBMS에 직접 접근하여 데이터를 업데이트하여 웹 서버에게 정보를 전달하는 방식으로 구조를 변경하였다. 즉 소켓 대신 Maria DB를 공유 repository로 사용하여 서버와 채팅 시스템이 음악 데이터를 통신하게 되었다.

d) 사용자 로그

중간에 받은 피드백에서 사용자 감정 분석 시점과 관련한 피드백이 있었다. 사용자의 감정이 일련의 연속된 대화속에서 최근 시점을 사용하는 것을 정책으로 정하였다. 따라서 사용자 로그 데이터가 일정 기간동안 저장되는 전체 대화 로그와 최근 대화 로그로 나누었고, 사용자마다 두 파일로 관리하게 되었다. 최근 대화는 대화 창을 나갈 때 마다 지워지고 전체 대화 로그는 하루마다 지워지게 된다. 물론 최근 대화 데이터도 하루가 지나면 로그 관리 데몬 프로그램이 지우게 된다. 또한 Delete Log 버튼을 통해 사용자가 직접 자신의 대화 로그를 모두 지울 수 있다.

e) 프로세스 비동기화

테스트하는 과정에서 챗봇과 연결이 불안정 할 때 서버와 클라이언트 브라우저에서 과부하가 걸리는 것을 확인했다. 이는 AJAX 사용 미숙에 의한 것이었다. AJAX을 통해 다른 PHP 프로그램으로 데이터를 전달하는데, AJAX이 포함된 함수는 비동기 처리방식으로 동작하지만, 해당 PHP 프로그램의 동작이 끝나 응답이 올 때까지 여러 스레드를 생성하여 응답을 요청하는 것을 발견했다. 해당 문제의 부분은 챗봇으로 요청을 보내고 응답을 기다리는 대화 메시지 요청과 음악 리스트 요청 두 부분이었다. 음악 리스트는 3.4.3에서 설명한 방식으로 해결했지만, 대화 메시지 요청 부분은 gateway 서버 로직부터 변경해야 했으므로 고치지 못하였다.



<그림 5: 변경된 서비스 시스템 구조>

따라서 앞에 언급한 문제점들을 해결하는 과정에서 시스템 구조는 위와 같이 바뀌었다. 각 하위 시스템의 자세한 구조는 다음 4.4에서 설명할 예정이다.

4. 과제 세부내용

4-1. 챗봇 시스템

챗봇 시스템에 사용된 데이터셋은 AIHub에 공개되어 있는 감성대화말뭉치를 사용하였으며, 전처리된 형태는 아래와 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

<그림 6: 전처리된 챗봇 데이터셋>

주요 특징으로는 사용자와 챗봇의 1-turn 기반의 데이터셋을 구성하였고, 중복 데이터 및 결측값을 제거하였다. 그리고 추후 사용될 가능성이 있는 대분류의 emotion\_1과 소분류 emotion\_2를 제외하고 나머지 column은 drop하였다.

현재 baseline 챗봇 모델은 q column의 데이터를 넣어 a column의 데이터를 generation 할 수 있도록 진행하며, 이에 맞추어 사용자와 챗봇의 답변 토큰으로 각각 "<usr>", “<sys>"을 추가하여 모델이 하여금 이를 구분하여 학습할 수 있도록 하였다. 자세한 구현 세부사항 및 hyperparameter는 다음과 같다.

Model = skt/kogpt2-base-v2

Learning Rate = 3e-5

Loss function = CrossEntropyLoss

Optimizer = AdamW

Batch\_size = 16

Best epoch = 4

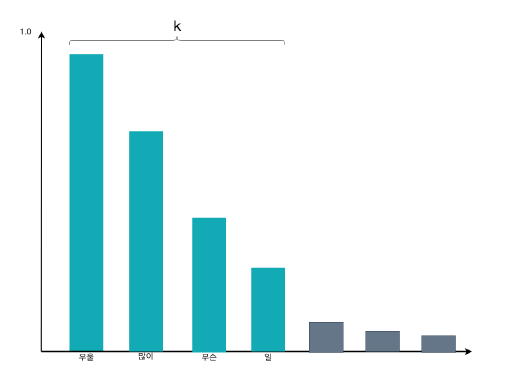
KoGPT-2에서의 generation method로는 greedy와 top-k 등이 존재하며 각각의 특징은 greedy, top-p가 있다.

a) greedy



특정 타임스텝 t에서 가장 높은 확률을 가지는 토큰을 다음 토큰으로 선택하는 전략으로 매 스텝의 관점에선 최선의 확률을 선택하지만 전체적인 문장의 관점에선 최선이 아닐 수도 있다는 문제가 있으며, 이러한 문제의 원인으로 동어 반복 문제가 생길 수 있다.

b) top-p



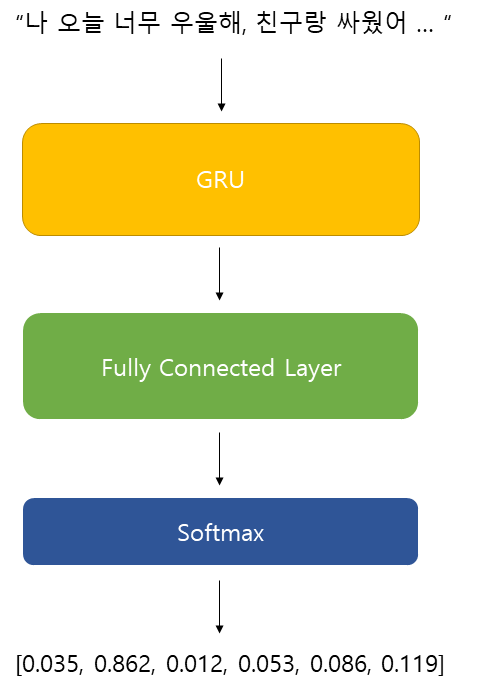
<그래프 1: Top-k>

일반 sampling 과정에서 아주 낮은 확률의 단어가 나올 수 있으므로, generation 과정에서 누적 확률이 일정 임계치를 넘지 않는 p개 만큼의 확률이 높은 순서의 단어들만으로 생성 제한을 두는 방식이다. context와 관련 없는 단어를 제외할 수 있다는 장점이 있다.

본 챗봇 시스템에서는 둘 중 우수한 fluency를 가진다고 알려진 top-k generation method를 선택하여 generation에 사용하였다.

4-2. 감정 분석 classifier

감정 분석 Classifier의 경우, 사용자의 전체 입력을 인풋으로 하여 GRU의 last hidden state가 다시 Fully Connected Layer, Softmax 함수를 거쳐 최종 6가지 class의 감정에 해당하는 벡터가 아웃풋으로 출력이 되는 구조이다. 이를 그림으로 표현하면 아래와 같다.



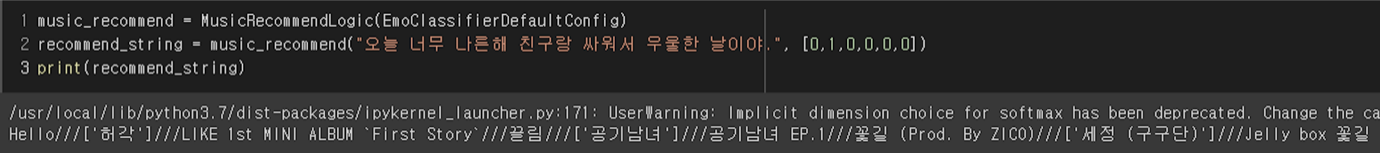
<그림 7: Classifier 구조 예시>

이때 input의 처음과 끝에 Beginning Of Sentence(BOS)와 End Of Sentence(EOS) 토큰을 추가하여 인풋을 받을 수 있도록 처리하였다.

4-3. 사용자 채팅 기반 음악 추천 알고리즘

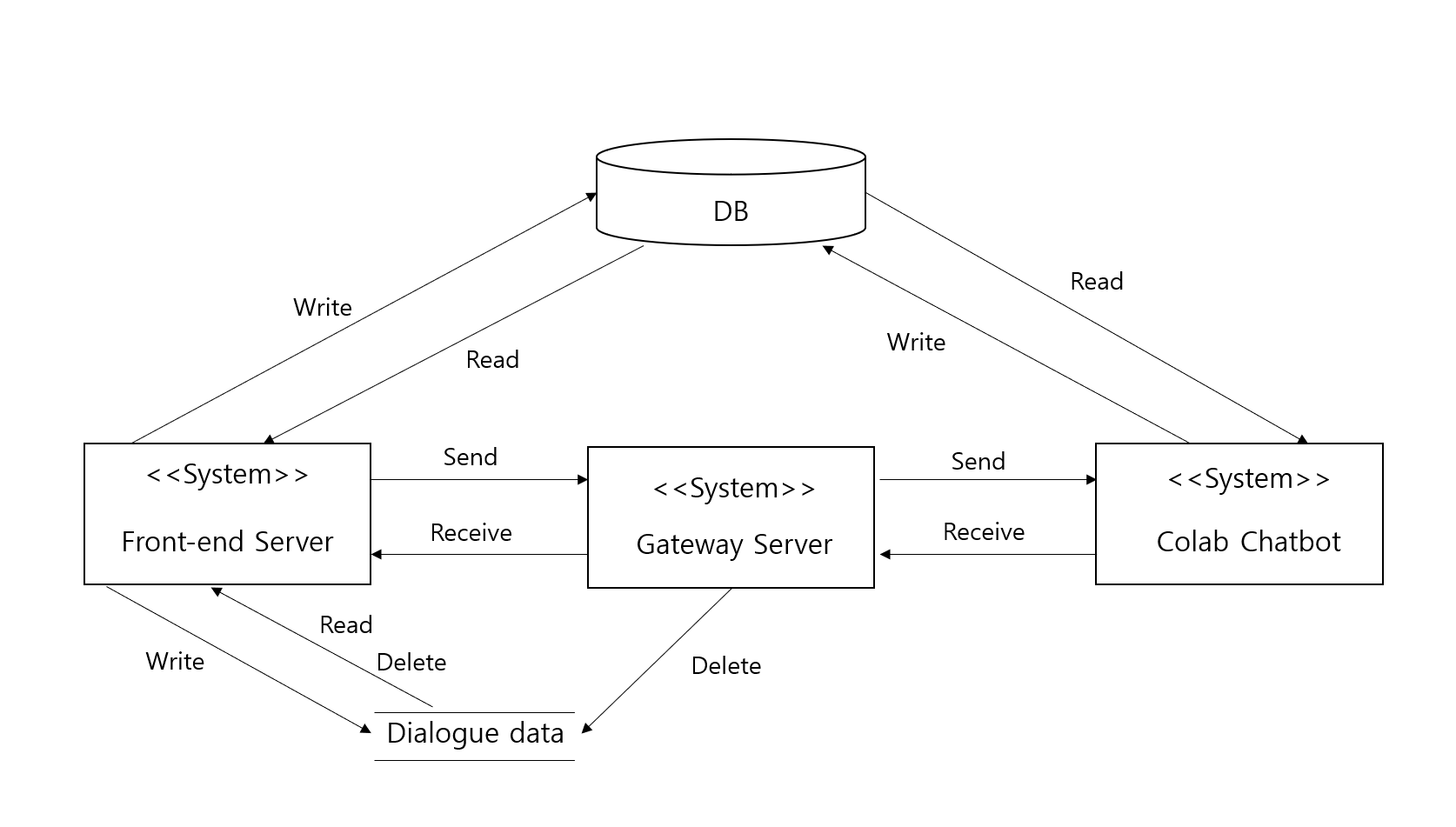
아래 그림처럼 MusicRecommendLogic 모델에 1) 사용자 채팅 내용과 2) [기쁨, 슬픔, 불안, 분노, 상처, 당황] 감정에 따라 자주 듣는 장르가 0-8 로 입력된다.

사용자 채팅 기반 감정 classification 이 진행되고 감정에 따라 자주 듣는 장르의 곡들로 음악 csv 가 필터링 된 후, 알고리즘을 통과하면, 최종적으로 음악 제목 명, 아티스트명, 앨범명이 반환된다.



<그림 8: 음악 추천 알고리즘>

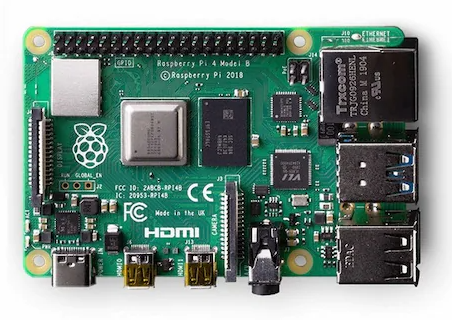
4-4. 서비스 구현



<그림 9: 전체 시스템 구조>

전체 시스템 구조를 보면 위와 같이 구성되어 있다. 사용자의 대화 입력을 받은 Front-end Server는 Gateway Server를 통해 Colab Chatbot으로 패킷으로 메시지를 전달하고 같은 경로로 응답 메시지를 받게 된다. 이러한 일련의 대화 내역의 파일로 저장되고 사용자가 Delete 버튼을 누르거나, 만 하루가 지나면 삭제된다. 음악 추천의 경우 위와 같은 경로로 메시지를 전달하고 성공 응답을 받으면 Front-end Server는 1초에 한 번씩 DB를 읽는다. Colab Chatbot은 패킷으로 받은 데이터와 DB로부터 읽은 사용자 정보를 통하여 생성한 음악 리스트를 DB에 입력하면 Front-end Server는 사용자에게 그 정보를 보여주게 된다. 아래에는 서버 구축, 패킷, DB, Front-end 서버, Gateway Sever 서버 구조에 관한 자세한 설명이 기술되어 있다.

a) 서버 구축

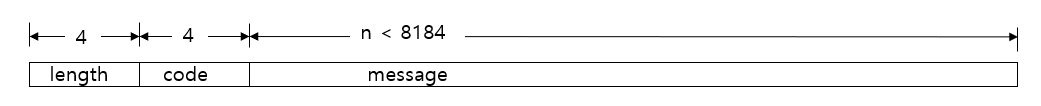


<그림 10: Raspberry Pi 라즈베리 파이 4 모델 B>

서버 하드웨어는 Raspberry Pi 4 모델 B를 사용하였다. RAM은 8G, 디스크는 256G, CPU는 ARMv7 4개로 구성되어 있다. OS는 Linux raspberrypi 5.10.17를 사용한다. Apache/2.4.38 (Raspbian), PHP 7.3.31, JSP 3.5.1 jquery.min.js 버전을 사용했다. DDNS는 Duck DNS[[1]](#footnote-1)의 무료 서비스를 이용하였고, d외부 포트는 18080을 열었다. 따라서 아래 주소로 이동하면 서비스를 이용할 수 있다. 다만 코랩의 런타임 시간 한계 때문에 연결해 놓지 않아 채팅은 불가능하다.

<http://lbsg98.duckdns.org:18080/chat/index.php>

b) 패킷 구조



<그림 11: 패킷 구조(단위: Byte)>

위 그림은 패킷의 전반적인 구조이다. TCP socket을 사용하기 때문에 packetizing을 고려하여 length와 code로 이루어진 헤드와 메시지를 두 번에 나눠 읽는 two-read 방식을 사용하였다. length와 code는 각각 4byte 스트링 값이고 패킷의 message 데이터는 모두 utf-8 인코딩 된 바이트 문자열이며 length는 message 데이터의 길이이다. 아래는 해당 코드 별 메시지 타입을 나열한 표이다.

|  |  |
| --- | --- |
| code | Message type |
| 0000 | Link check packet between gateway server and chatbot. |
| 2001 | General message for dialogue data between gateway, front-end server and chatbot. |
| 2002 | Music recommendation message between front-end and chatbot. |

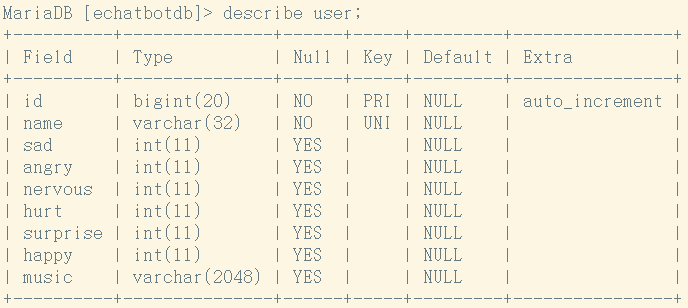
<표 1: 코드와 메시지 타입>

0000 코드는 챗봇과 gateway 서버 사이의 연결 상태를 확인하는 패킷이다. 70초 마다 패킷을 주고받으며 그 시간 내에 보낸 패킷이 도착하지 않는 경우 해당 채널을 삭제한다. 이는 테스트하는 과정에서 네트워크의 문제로 서버와 챗봇 모두 연결이 establish 된 상태이지만 중간 연결이 끊겨 해당 connection이 일치하지 않아 통신을 하지 못하는 경우가 발생했기 때문이다. 0000, 2001번 코드는 모두 메시지가 일련의 문자열이고, 2002번 코드의 경우 아래와 같이 message 부분이 구성되어 있다.

user\_name<///userName///>user\_dialogue\_data

c) DB 구조

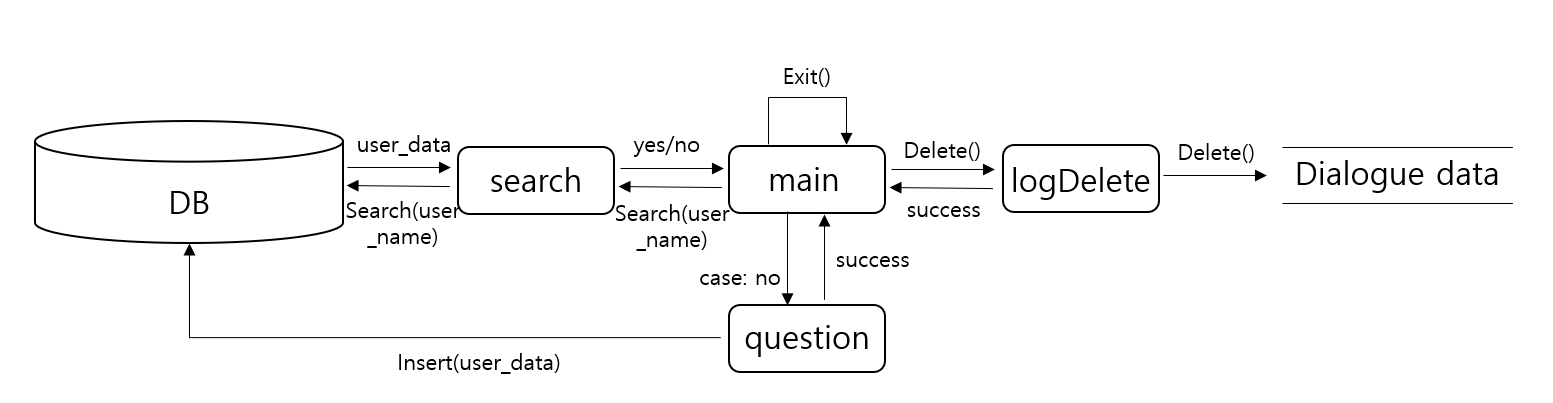
데이터 구조는 아래와 같이 user 테이블 하나가 존재한다. 구조는 사용자 입력 name, 선호도 조사 값인 sad, angry, nervous, hurt, surprise, happy한 감정에 듣는 장르 값, 자동 생성되는 key값인 id 그리고 음악 추천 챗봇과 데이터를 주고받는 music 열로 총 9개의 열이 존재한다. 사용자 로그인과 관련하여 인증과정이 없으며 하드웨어 한계와 간단한 데이터 구조의 이유로 디스크 DB와 메모리 DB를 나누지 않고 모두 디스크 DB 즉 InnoDB만 사용했다.



<그림 12: DB schema>

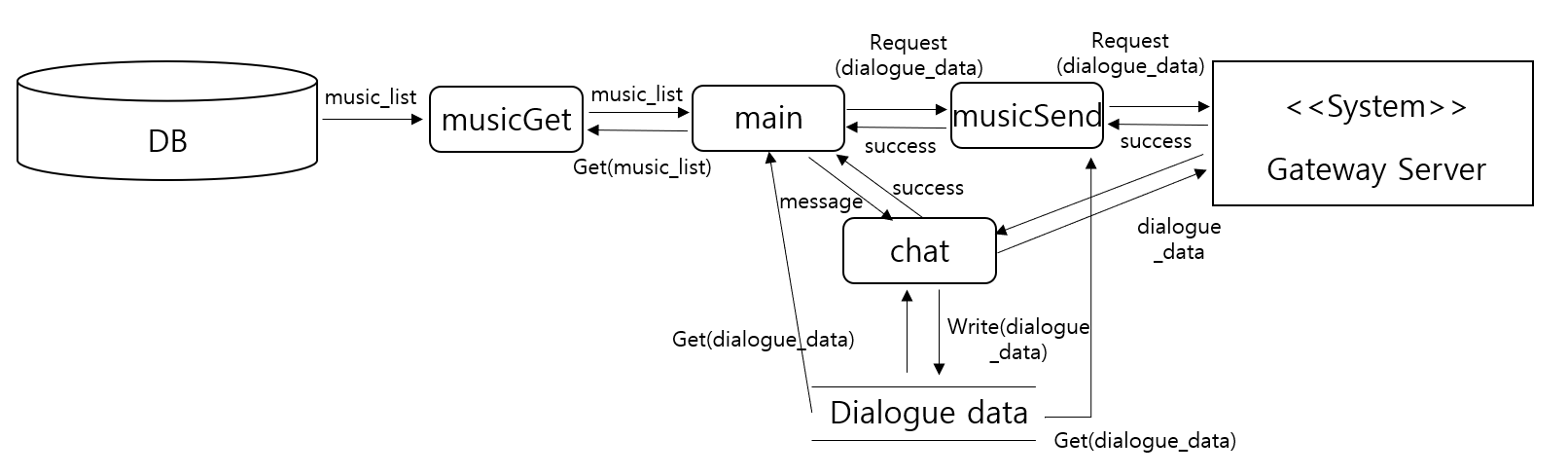
d) Front-end Server 구조

프론트 엔드 서버의 구조는 아래와 같이 로그와 채팅과 관리하는 구조로 나눌 수 있다.



<그림 13: Front-end Server 로그 관리 구조>

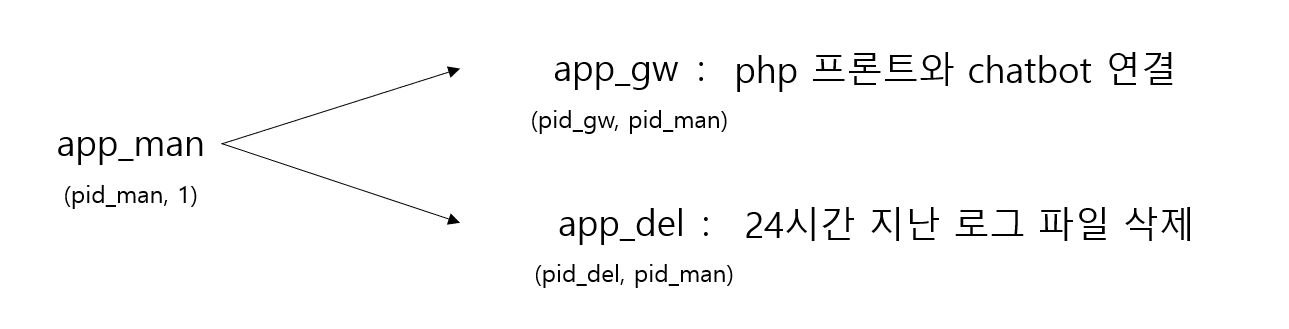
사용자가 main화면에 진입하면 사용자 이름을 입력하여 세션을 시작한다. 이때 search 모듈은 사용자의 정보가 DB에 있는지 확인하고 결과 값을 반환한다. 사용자 정보가 존재하는 경우 바로 채팅 화면으로 전환된다. 사용자 정보가 DB에 존재하지 않을 경우 question 페이지로 이동하여 sad, angry, nervous, hurt, surprise, happy 감정 별 듣는 발라드, 댄스, 랩/힙합, 락/매탈, R&B, 인디, 팝, 클래식, 재즈 장르를 조사한다. 이후 이를 DB에 사용자 이름과 같이 저장하고 성공 메시지를 main 페이지에 전달하고 채팅 화면으로 전환된다. 상단에는 exit 버튼과 deleteLog 버튼이 있다. Exit 버튼을 클릭하면 세션을 삭제하고 로그인 화면으로 전환된다. deleteLog 버튼을 클릭하면 ~/log 디렉토리 안에 생성된 해당 사용자의 로그 파일을 삭제한다.



<그림 14: Front-end Server 대화 구조>

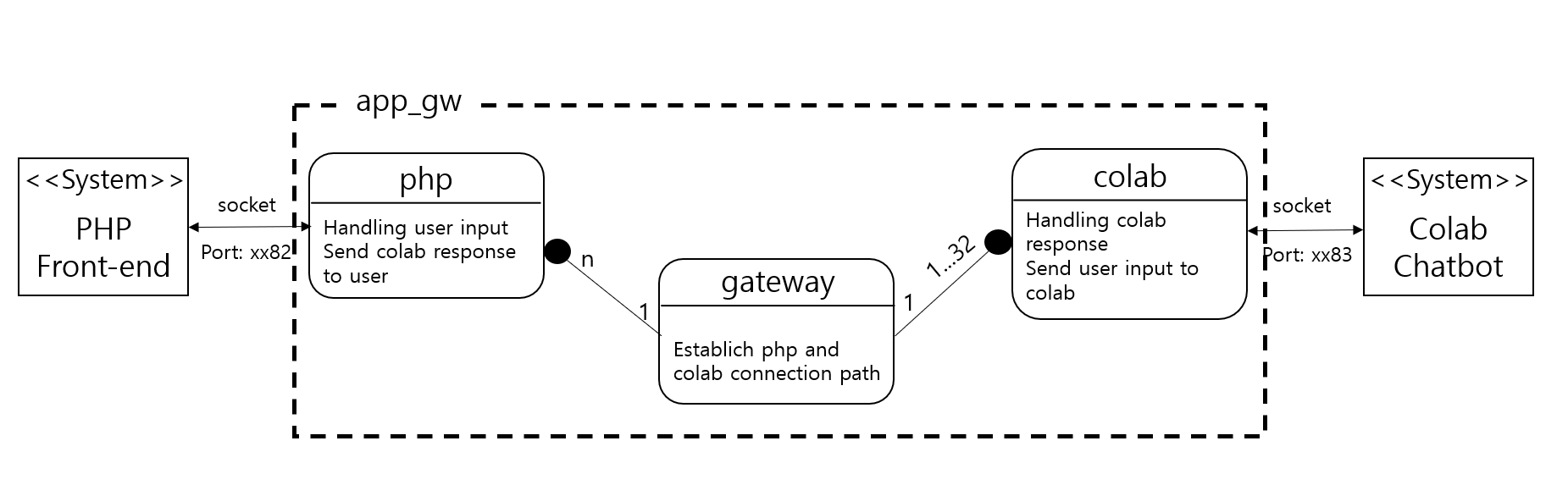
사용자가 채팅 화면에 진입하여 메시지를 입력하면 chat 모듈이 gateway 서버에게 이전 대화 내역과 합쳐서 전달한다. 그리고 gateway로부터 받은 응답 데이터를 dialogue data 파일에 기입하고 성공 메시지를 반환한다. Main 모듈은 주기적으로 해당 파일을 읽으면서 대화 내역을 사용자에게 업로드 한다. 이 부분에서 응답을 기다리는 부분이 비동기화가 부족하여 챗봇과 연결이 불안정 할 때 과부하를 발생시킨다. 좌측 상단 music 버튼을 클릭하면, musicSend 모듈이 gateway 서버에게 사용자 이름과 대화 내역을 하나의 패킷으로 전달한다. 그리고 성공 메시지를 받으면 main 모듈에 전달한다. 이후 musicGet 모듈이 1초에 한 번씩, 데이터를 획득할 때까지 최대 timeout 90초 동안 DB의 해당 사용자 행의 music 열을 읽으며 추천 음악 리스트 데이터를 읽어 사용자에게 표시한다. 기다리는 동안 “waiting.”의 “.” 문자열이 길어지는 것으로 로딩을 표시하고 timeout 시간 내에 데이터가 도착하지 않는 경우 "Sorry, connection time out." 문구를 표시한다.

e) Gateway Server 구조



<그림 15: Gateway Server 전체 구조>

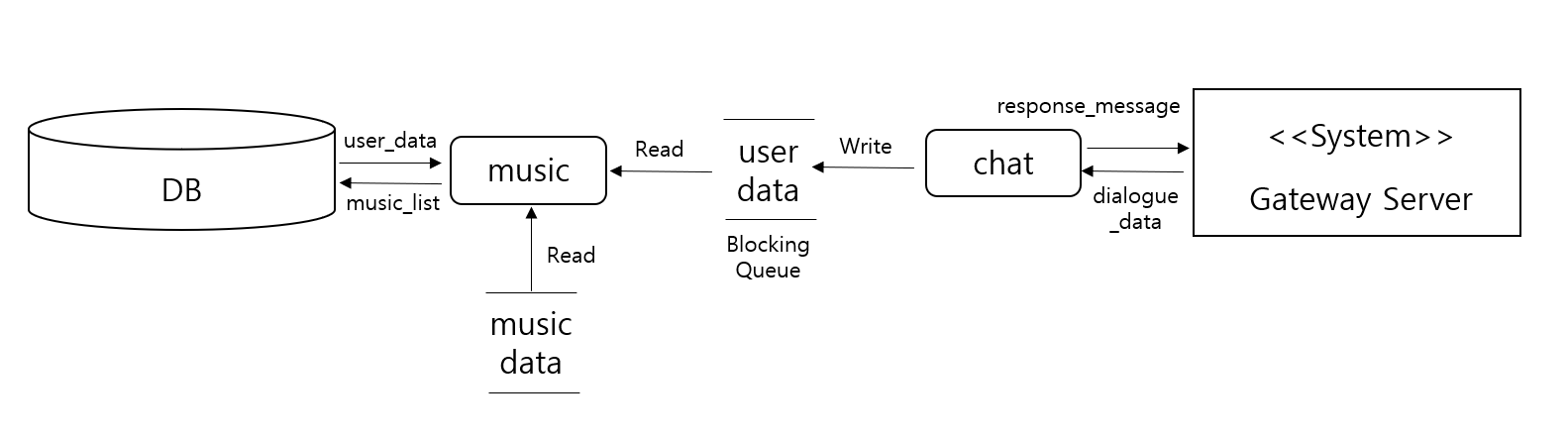
Gateway Server의 구조는 다음과 같다. app\_man 프로그램은 app\_gw와 app\_del 프로그램의 부모가 되어 모종의 이유로 종료되면 다시 시작시키는 데몬 프로그램이다. app\_del은 30분에 한 번씩 ~/log 디렉토리에 있는 파일을 검사하여 만 하루가 지난 파일을 삭제하는 프로그램이다. app\_gw는 프론트 엔드 서버와 챗봇 서버를 연결해 주는 gateway역할을 하는 프로그램이다.



<그림 16: Gateway 역할을 하는 app\_gw 프로그램 구조>

app\_gw 자세한 구조는 위와 같다. php와 colab, gateway 세개의 모듈로 구성되어 있고 php는 프론트 엔드와, colab은 챗봇과 연결을 담당하며 이 둘의 통신은 gateway 모듈이 담당한다. 프론트 엔드 서버로부터 사용자의 입력 데이터가 들어오면 gateway 스레드가 php 스레드를 생성하고 생성된 php 스레드는 timeout 60초 동안 colab 채널 스레드 할당을 기다린다. Timeout 시간동안 할당 받지 못하면 gateway는 “colab 채널 할당 오류입니다.” 메시지를 반환한다. 채널을 할당 받으면 php 스레드는 해당 데이터를 colab 채널에 보낸다. colab 채널은 연결되어 있는 외부 챗봇 시스템에게 데이터를 전달하고 응답을 받는다. 받은 응답은 다시 php 스레드에게 전달되고 php 스레드는 프론트 엔드 서버에 전달하고 스레드를 종료한다. Colab 채널은 다음 응답 요청을 기다리며 link check 패킷을 70초 마다 한 번씩 보내 연결을 유지한다. 이때 colab 채널은 최소 1개에서 최대 32개까지 연결될 수 있으며, 테스트 결과 챗봇 동시 연결 처리는 챗봇의 성능의 한계로 클라이언트와 최대 1:4 정도의 부하를 감당할 수 있었다.

f) Chatbot 구조



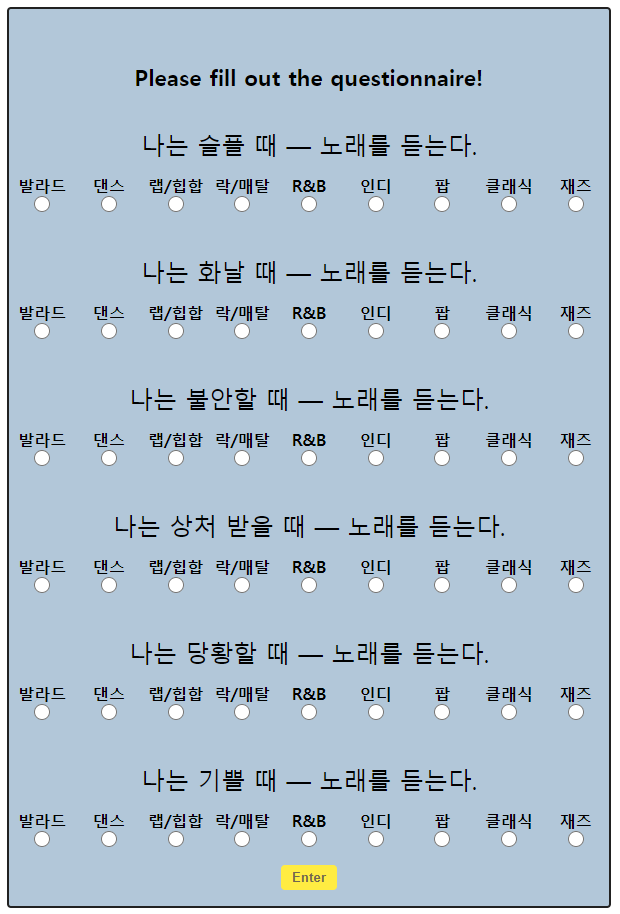
<그림 17: Chatbot 시스템 구조>

챗봇은 gateway 및 front-end 서버와 물리적으로 분리된 하드웨어에 존재하고 전체 챗봇의 구조는 음악 추천을 수행하는 music 모듈과, 챗봇의 응답을 생성하고 gateway 서버와 통신하는 chat 모듈로 구성되어 있다. 앞서 설명한 이유로 두 모듈은 각각의 스레드로 동작하고 Blocking Queue를 통하여 두 스레드 사이의 데이터를 주고받는다. chat 모듈은0000 코드의 link check 패킷을 받으면 즉시 gateway 서버로 패킷을 전송한다. 2001코드의 일반 대화 메시지를 받으면 챗봇의 GPT모델이 대답을 생성하여 gateway 서버에게 전달한다. 2002코드의 음악 추천 메시지를 받으면 해당 사용자 이름과 선호도, 대화 내역을 dictionary 형태로 Blocking Queue에 넣는다. music 모듈은 1초에 한 번씩 Queue를 읽으면서 데이터를 확인한다. 들어온 데이터가 있으면, 사용자 선호도와 대화 데이터로 추천 음악 리스트를 생성하고 해당 사용자 DB 행의 music 열에 생성한 데이터를 입력한다.

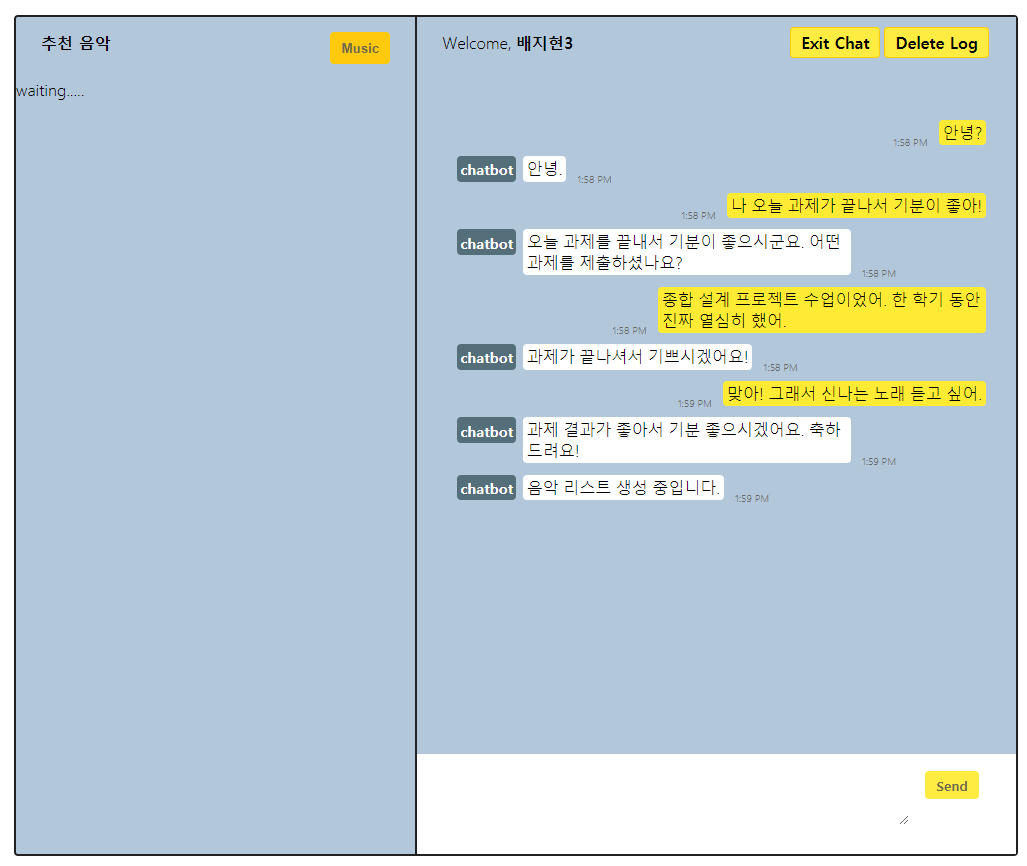
g) UI



<그림 18: 로그인 화면>



<그림 19: 설문조사 화면>



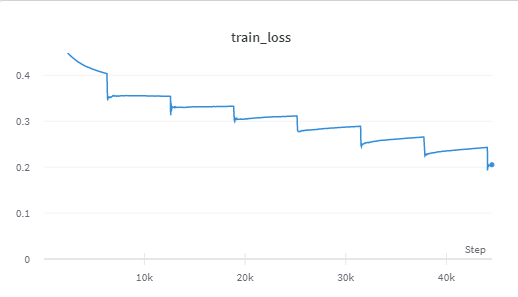
<그림 20: 대화 화면>



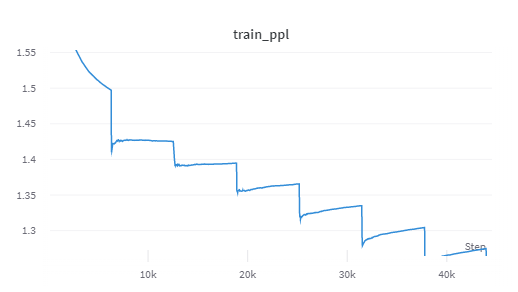
<그림 21: 대화 음악 추천 화면>

5. 결과

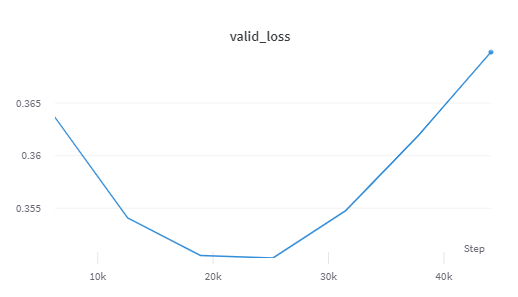
5-1. 챗봇 시스템



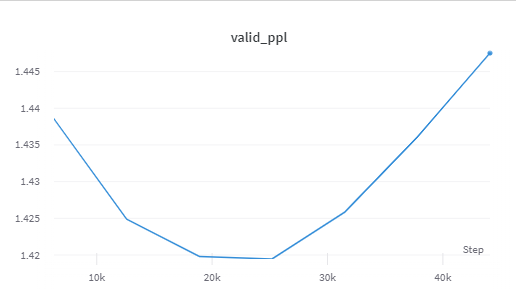
<그래프 2: train\_loss>



<그래프 3: train\_ppl[[2]](#footnote-2)>



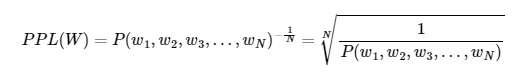
<그래프 4: valid\_loss>



<그래프 5: valid\_ppl>

위와 같은 방법으로 Train 시켰을 때의 train validation 과정 loss, ppl은 다음과 같이 나타났으며 학습을 진행함에 따라 정상적으로 train loss와 ppl이 줄어드는 것을 확인할 수 있었다.

PPL (Perplexity Score)는 언어 모델을 평가하기 위한 평가지표로, 문장의 길이로 정규화된 문장 확률의 역수를 나타낸다. 따라서 언어 모델의 ppl 수치가 낮을수록 좋은 성능을 의미한다.



학습을 진행하며 가장 validation ppl이 낮은 epoch 4 state를 본 챗봇 시스템에 적용하였다.

아래는 학습이 완료된 챗봇의 대화 예시이며, fine-tuning을 통해 적절한 응답을 보내는 것을 확인할 수 있었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

<그림 22: 챗봇 대화 예시>

또한 챗봇의 성능을 평가하기 위하여 ROUGE, BLEU와 같은 metric을 사용하여 이를 측정하였다. 각각의 metric에 대한 설명은 다음과 같다.

a) BLEU (Bilingual Evaluation Understudy)

BLEU는 기계 번역 결과와 사람이 직접 번역한 결과가 얼마나 유사한지 비교하여 번역에 대한 성능을 측정하는 방법이다. 측정 기준은 n-gram에 기반하며 BLEU는 PPL과는 달리 높을 수록 성능이 더 좋음을 의미한다. 단어 개수 카운트 측정, 중복 제거 보정, n-gram 확장 및 짧은 문장 길이에 대한 패널티와 같은 여러 측면을 고려하여 종합적인 score를 나타내게 된다. 우리가 하고자 하는 챗봇 시스템의 경우, reference와 모델이 생성한 response 두 문장을 비교하는 식으로 사용하였다.

b) ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

텍스트 요약 모델의 성능 평가 지표이다. ROUGE는 텍스트 자동 요약, 기계 번역 등 자연어 생성 모델의 성능을 평가하기 위한 지표이며, 모델이 생성한 요약본 혹은 번역본을 사람이 미리 만들어 놓은 참조본과 대조해 성능 점수를 계산한다. 이것 또한 BLEU와 같이 점수가 높을수록 성능이 높음을 의미한다.

ROUGE-1은 시스템 요약본과 참조 요약본 간 겹치는 unigram의 수를 보는 지표이며, ROUGE-2는 시스템 요약본과 참조 요약본 간 겹치는 bigram의 수를 보는 지표이다. ROUGE-L은 LCS(Longest Common Subsequence problem) 기법을 이용해 최장 길이로 매칭되는 문자열을 측정하는 방식이다. LCS의 장점은 ROUGE-2와 같이 단어들의 연속적 매칭을 요구하지 않고, 어떻게든 문자열 내에서 발생하는 매칭을 측정하기 때문에 보다 유연한 성능 비교가 가능하다는 것이다. BLEU와 ROUGE를 테스트 데이터셋과 비교한 결과는 다음과 같다.

|  |  |
| --- | --- |
|  | Score |
| BLEU | 0.235 |
| ROUGE-1 | 0.235 |
| ROUGE-2 | 0.099 |
| ROUGE-L | 0.233 |

<표 2: BLEU, ROUGE score>

위와 같은 baseline의 성능을 측정해볼 수 있었으며, 우리의 baseline과 비교할 만한 챗봇은 시간상의 문제로 인하여 찾지 못하였다. 또한 BLEU와 ROUGE의 경우 통계적으로 얼마나 응답이 적절한지를 나타내는 metric으로, 의미를 고려하여 evaluation하기에는 아직까지는 어려운 부분이 존재한다. open-domain conversation의 경우 현재 활발히 연구가 진행되고 있는 분야로, 나중에 이를 고려한 metric을 통하여 챗봇을 좀 더 다각도로 평가할 수 있을 것으로 기대한다.

5-2. 감정 분석 classifier

GRU와 Fully Connected Layer로 구성된 classifier를 train 시킨 후 test data에 대하여 감정을 예측해본 결과는 다음과 같이 나타났다.

텍스트이(가) 표시된 사진

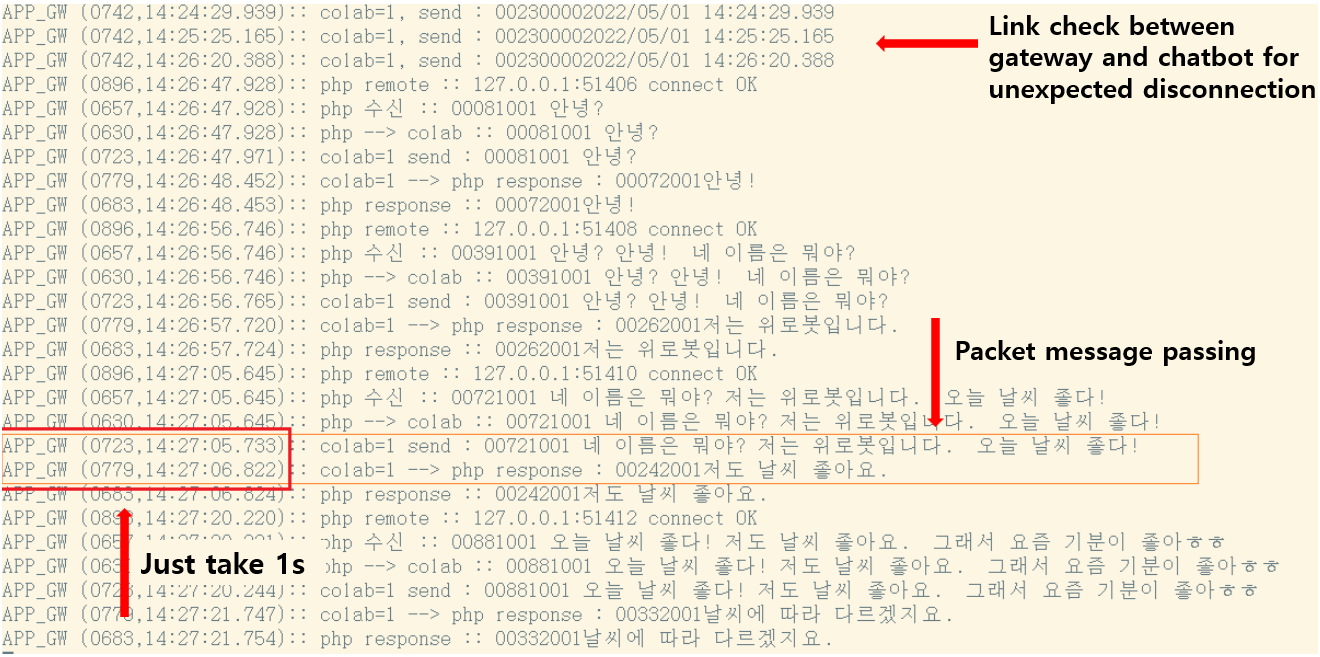
자동 생성된 설명

<그림 23: 감정 분석 예시>

이때 간단한 구조임에도 불구하고 classifier가 여러 감정을 예측하는 것을 확인할 수 있으며, 실제로 감정이 일치하는지를 나타내는 accuracy 또한 0.687로 측정할 수 있었다.

5-3. 서버

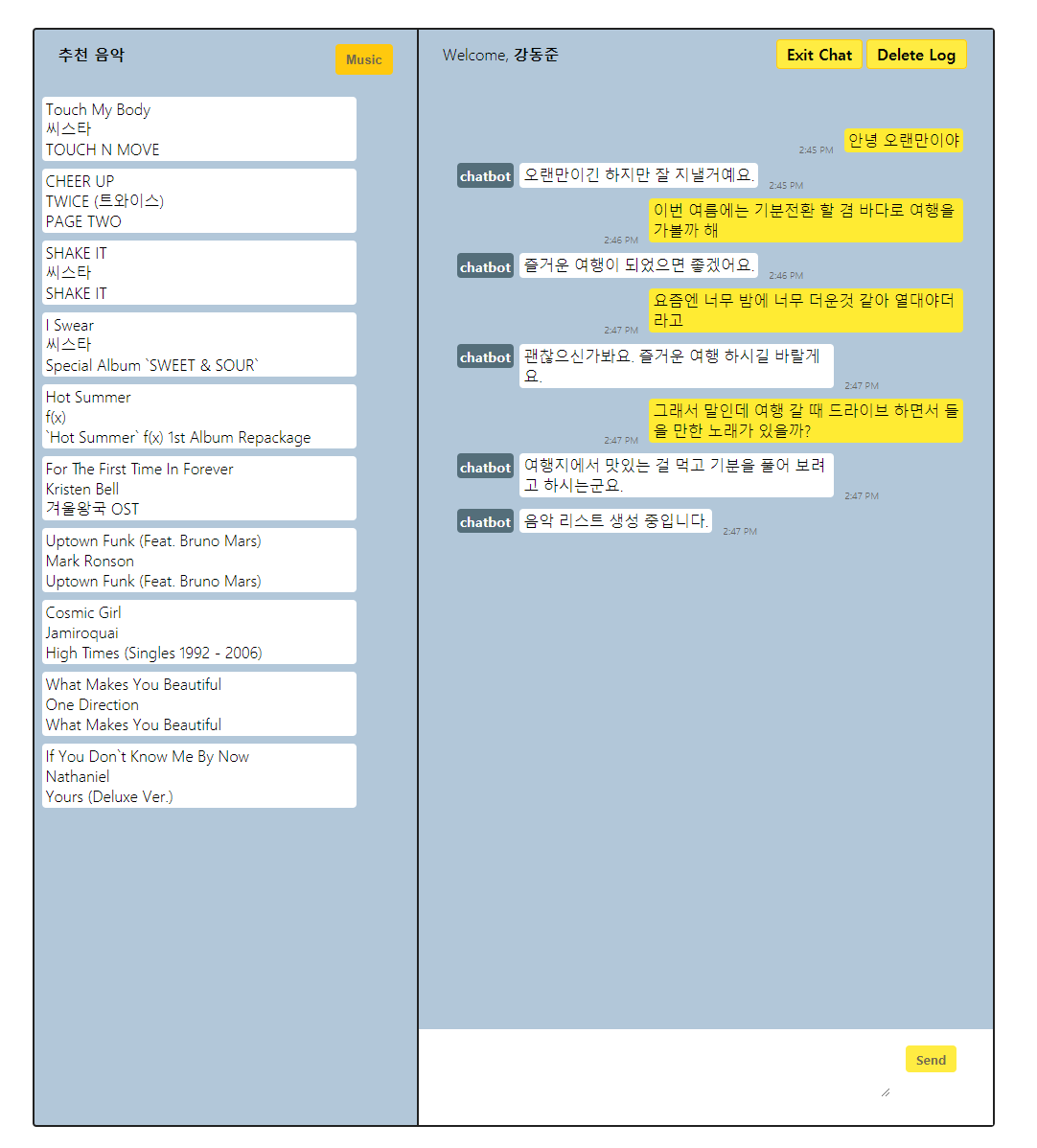
다음은 패킷 데이터가 gateway 서버에서 처리되는 과정이 표시된 화면이다. 우측 상단을 보면 link check 패킷을 확인할 수 있고 주황색 박스 안을 보면 대화 내역과 응답이 잘 전달되는 것을 볼 수 있다. 좌측 시간을 보면 응답을 받는데 걸리는 시간이 1초 정도로 안정적이고 주로 챗봇이 응답을 생성하는 것에 많은 시간을 소요하는 것을 확인할 수 있다.



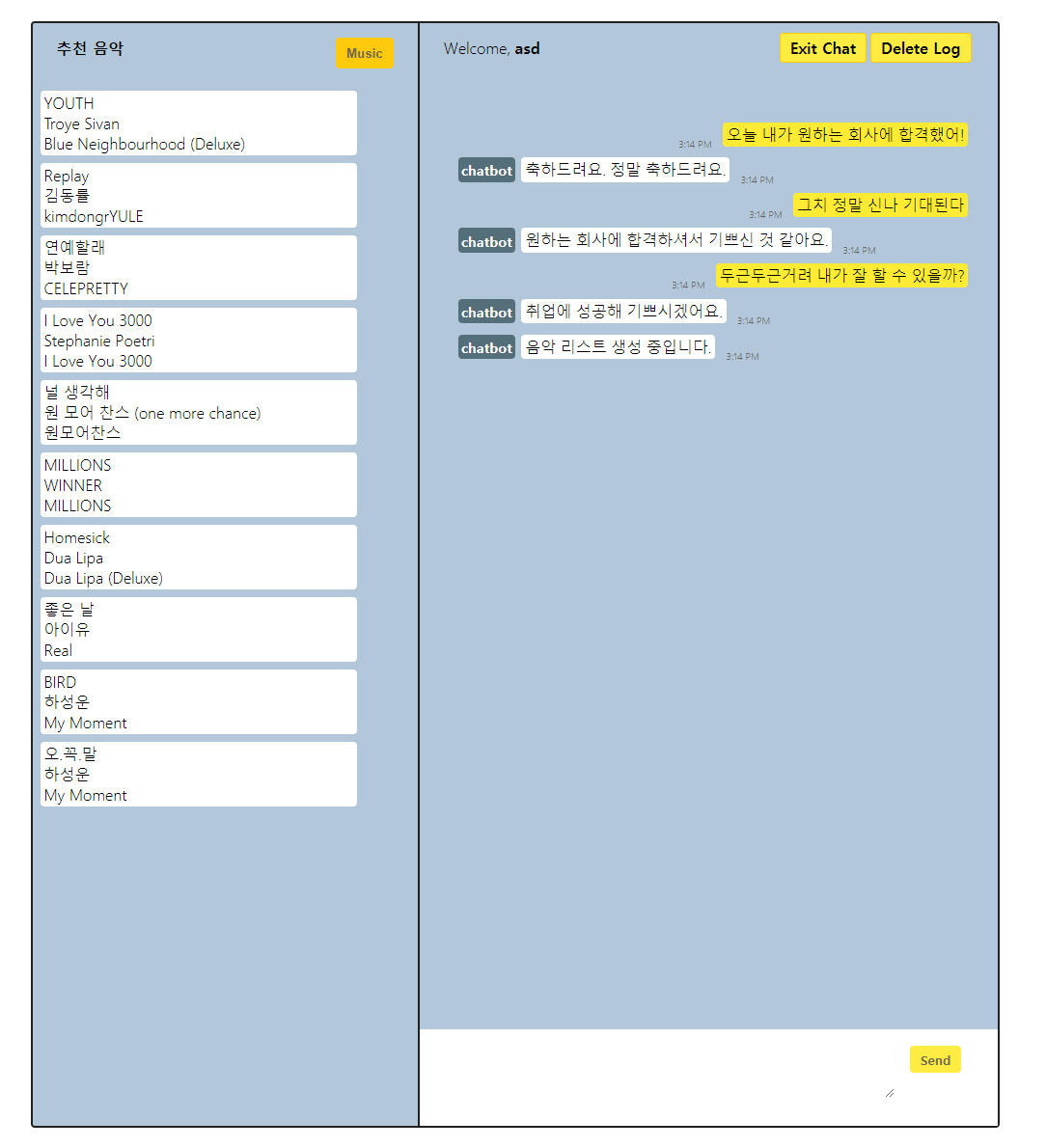
<그림 24: gateway 패킷 처리 과정>

5-4. 최종 시나리오

아래의 예시는 챗봇과의 대화와 음악 추천에 관한 9가지 시나리오를 바탕으로 분석한 결과이다.



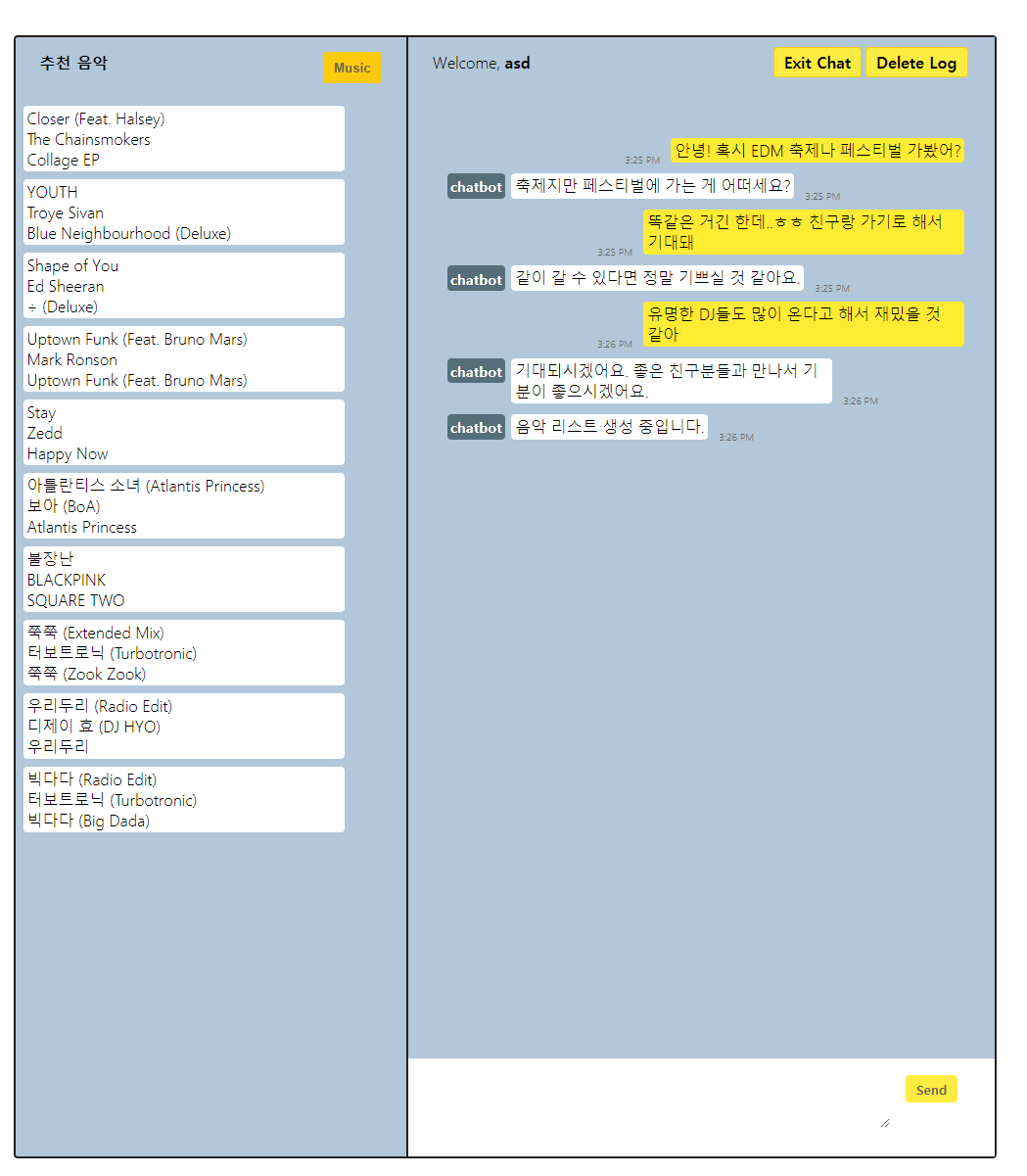
<그림 25: 시나리오 여름 여행>



<그림 26: 시나리오 취업 성공>



<그림 27: 시나리오 이별>



<그림 28: 시나리오 EDM 축제>

6. 기대 효과

우리 모델은 챗봇을 통해 사용자가 챗봇과 감정적인 대화를 진행하는 것뿐 아니라, 사용자 감정에 맞는 노래를 추천하여 노래를 통한 감정적 위로를 제공한다. 우리 서비스를 통해 얻을 수 있는 구체적 항목별 기대효과는 다음과 같다.

6-1. 챗봇

구글 람다와 같은 챗봇 서비스를 주기능으로, 사용자가 챗봇과 감정적 대화를 나눌 수 있도록 하며, 부기능으로 챗봇이 사용자 실시간 감정 맞춤형 음악을 추천해주고 위로를 돕는다.

6-2. 음악 스트리밍 서비스

음악 스트리밍 서비스에 챗봇 기능을 탑재하여 사용자 맞춤형 음악 추천을 주 기능으로 수행하며, 챗봇의 역할은 사용자의 실시간 감정을 자연스럽게 파악할 수 있도록 하는 것에 초점을 둔다.

6-3. 사용자 채팅 기반 음악 추천 알고리즘

사용자가 기분에 따라 자주 듣는 음악 장르를 기반으로 음악을 추천하여 정확한 추천이 가능하며, 사용자의 채팅 텍스트와 유사도가 높은 곡들을 필터링하기 때문에 맞춤형 추천이 가능하다.

6-4. 웹서비스

동시에 여러 사용자가 접속이 가능하도록 설정했으며, 웹서비스가 AI 모델과 통신하고 데이터를 전달하는 로직으로 구현하여, 사용자가 웹에서 편리하게 추천을 받을 수 있도록 하였다. 또한, frontend 상에서도 카카오톡과 유사한 UI를 구현하여, 사용자에게 친숙한 UI를 제공할 수 있었다.

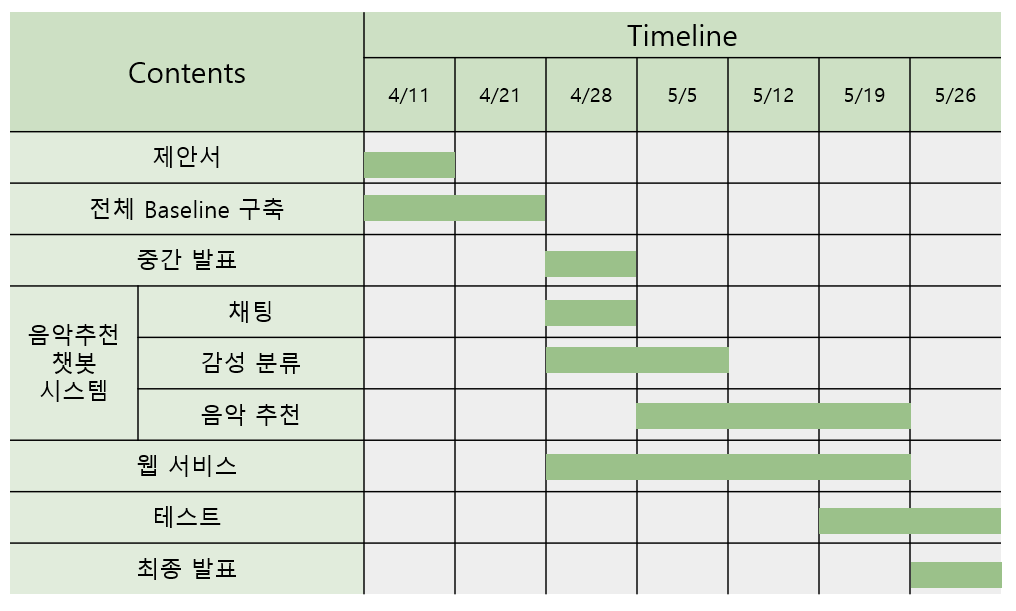
7. 한계점 및 개선사항

가장 먼저 챗봇의 성능에 대한 아쉬움이 크다. 우리가 처음 구현하려고 했던 챗봇은 현재 서비스되어지고 있는 이루다와 같이 사람과 원활하게 소통하는 모델이었지만, 테스트를 해보면서 아직까지 사용자의 대화나 문맥을 완벽하게 고려하여 제대로 된 답변을 내놓지 못하는 경우가 많았다. 이를 분석해보면 기본적으로 높은 퀄리티의 데이터셋을 사용하지 못한 점, 그리고 데이터 셋의 양이 절대적으로 부족한 점을 꼽을 수 있을 것 같다. 우리가 train에 사용한 dataset의 크기는 약 1만 건이었지만, 이루다의 경우 1억건 이상의 높은 퀄리티의 데이터를 통하여 학습된 것으로 미루어 볼 때 우리가 이루다와 같이 문맥을 잘 이해하고 적절한 응답을 내는 것처럼 좋은 성능을 얻기는 힘듦을 짐작할 수 있다.

Classifier의 경우, GRU를 통하여 학습을 진행하였지만, classification task에서 성능이 더 좋은 BERT와 같은 Language Model을 하드웨어 memory 문제로 쓰지 못했다. GPT모델과 함께 돌려야 하기에 어쩔 수 없는 선택이었지만, 이를 사용했더라면 감정 분석에서 accuracy가 훨씬 높게 나왔을 것으로 생각된다. 이와 더불어 사용한 훈련 데이터 셋의 한계로 감정 class가 6가지에 그쳤고, 긍정적인 감정은 기쁨 하나뿐이기 때문에 여러 가지 감정을 느끼는 사용자에게 맞춤 음악을 추천하기에 어려움이 있었다. 또한, 장르 기반의 음악 추천을 진행하기 때문에 다양한 장르를 섞어서 추천해주지 못하는 점이 있다. 따라서 이를 개선하기 위해서는 먼저 많은 양의 데이터와 더욱 더 세분화된 음악 추천 logic이 필요할 것으로 생각이 된다.

프론트와 백 관련에서는 socket이 받는 부분을 비동기 처리하는 것이 미완료되어 google colab과 통신이 불안정 할 때 서버 및 client 과부하가 발생한다는 단점이 있었다. 또한 하드웨어 성능이 부족하여 수용 가능한 사용자의 수가 적다. 또한 웹 페이지 상 사용자가 가장 처음 선택한 폼에서 기분에 따른 음악 선호도를 바꾸는 기능이 부족한 점, 사용자 가입 시 인증이 부족한 점 등 디테일한 부분에서 기능 구현이 부족하였다. 이렇듯 아쉬운 점이 존재하지만 그럼에도 불구하고 직접 챗봇을 구현하고 처음으로 음악 추천까지 해주는 모델을 끝까지 성공했으며, 기회가 된다면 future work을 통하여 더욱 더 발전시켜 나갈 계획이다.

8. 수행 일정 및 역할분담



<표 3: 수행 일정>

역할분담

강동준 – 챗봇 시스템, classifier, 음악 추천 logic, 카카오 음악 데이터셋 전처리

김지연 – 챗봇 시스템, 음악 추천 logic, 감성대화말뭉치 데이터 전처리, Chatbot Data v1.0 전처리

배지현 – 챗봇 시스템 통신 및 스레드 관리, 서버, 프론트 엔드 구현

9. 참고문헌

[1] 최현석 외 5, 사용자 감성과 음원 무드기반 음악 추천 시스템, 한국방송공학회, 2010, (<https://www.koreascience.or.kr/article/CFKO201033254124211.pdf> )

[2] 음악 추천을 위한 가사정보 및 음악신호 기반 특성 탐색 연구, 2019, (<https://s-space.snu.ac.kr/bitstream/10371/151420/1/000000154848.pdf>)

[3] Susan David, 『 Emotional Agility 』, Wall Street Journal-September 6, 2016

[4] 박광배, 「‘이루다’ 사건에 대한 개인정보위 결정의 의미와 시사점」, 『법률신문』, 2021년 05월 11일, https://m.lawtimes.co.kr/Content/LawFirm-NewsLetter?serial=170076

[5] Under review as a conference paper at ICLR 2017, OUTRAGEOUSLY LARGE NEURAL NETWORKS (https://arxiv.org/pdf/1701.06538.pdf)

[6] 한병준, 황인준, 음악 추천을 위한 감정 전이 모델 기반의 음악 분류 기법, 전기전자학회논문집, 2009

[7] 임영범 외 4, KE-T5 기반 한국어 대화 문장 감정 분류, 한글 및 한국어 정보처리대회 33, 2021

[8] Alec Radford 외 5, Language Models are Unsupervised Multitask Learners, OpenAI blog, 2019

1. <https://www.duckdns.org/> [↑](#footnote-ref-1)
2. PPL : Perplexity Score [↑](#footnote-ref-2)