



386 RANGERS

AI 기반 악성 메시지 필터링이 적용된 아티스트-팬덤 채팅 서비스



* 386 RANGERS *

이예지 강혜정 권성재 윤철식 이준희

PLAYDATA - DE 19

23.06.16.



386 RANGERS

; 악성 메시지로 고통받는 연예인을 수호하는 레인저스

강혜정 

데이터분석

*** 데이터 전처리

*** 모델 학습

윤철식 

프론트엔드

*** 웹 화면 설계

*** 웹 서비스 구현



이준희 

백엔드

*** DB 설계 및 구현

*** 서버 연결

권성재 

데이터분석

*** 모델 학습

*** 모델 평가

※리더※
이예지 

데이터분석

*** 데이터 수집

*** 모델 학습

CONTENTS



1. 프로젝트 개요

- 1) 프로젝트 배경
- 2) 프로젝트 목적
- 3) 기대효과
- 4) 프로젝트 일정
- 5) 사용한 기술

2. 데이터 수집 및 분석

- 1) 데이터 수집
- 2) 데이터 전처리
- 3) 딥러닝 모델 학습
- 4) 딥러닝 모델 평가

3. 서비스 설계 및 구현

- 1) 서비스 설계
- 2) 설계 구현
- 3) 테스트

4. 서비스 시연 및 질의응답

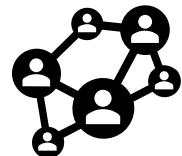


1. 프로젝트 개요

* 프로젝트 배경 — 프로젝트 목적 — 기대효과 — 프로젝트 일정 — 사용한 기술

1. 프로젝트 개요

기존 서비스의 문제점 ➡️



사회적 문제

연예인 대상 악성 게시물과 메시지
악플러 제재에 대한 사회적 요구 증가

[OK!제보] 팬커뮤니티 '위버스'에 끊이지 않는 악성 게시물
다음, 뉴스댓글 대신 실시간채팅…
네이버는 악플러 노출(종합)

2022-06-02 07:13 2023-06-08 10:27

이용자들 "악성 게시물에 청소년·아티스트 무방비 노출"
위버스 "기술적 시스템·인력 모두 동원해 적극 대응중"

다음, '타임톡' 서비스 시작…댓글 24시간 제한하고 기록도 안 남아
네이버, 댓글 제한 이용자 프로필서 닉네임과 아이디 일부 공개



기술적 문제

금칙어 기반 필터링:
대화의 맥락을 고려하지 않음

김요한 선수들 세레모니보면 Google 번역 PM 08:06

김요한 흥부가 응장해진다 이말이에요 Google 번역 PM 08:06

김요한 금칙어라네 참나 Google 번역 PM 08:07



1. 프로젝트 개요



사회적 문제

- ⌚ 악성 게시물 및 댓글 관리의 경우, 금칙어 기반 기계적 필터링과 관리자의 수동 제재로 이루어짐
- ⌚ 줄어들지 않는 악의적인 글로 인해 정신적 고통을 호소하는 아티스트 및 팬들의 불만이 고조됨

[엔터] '악플 관리 못하면서 유료화 도입?' 위버스, 올해 상반기 유료 구독 서비스 도입 [공식]

그런데 라이브 방송 중 도 넘는 악플이 지속적으로 달렸고, 악플 중에서는 부모님을 향한 욕설까지 포함되어 있던 것으로 알려졌다. 이채영은 무시하고 계속해서 소통을 이어가고자 했지만 도를 넘는 글들이 잇따르면서 이내 눈물을 흘렸고, 방송을 종료했다.

[OK!제보] 팬커뮤니티 '위버스'에 끊이지 않는 악성 게시물

2022-06-02 07:13

요약

가 🔍 1

**다음, 뉴스댓글 대신 실시간채팅…
네이버는 악플러 노출(종합)**

2023-06-08 10:27

요약

가 🔍

이용자들 "악성 게시물에 청소년·아티스트 무방비 노출"
위버스 "기술적 시스템·인력 모두 동원해 적극 대응중"

다음, '타임톡' 서비스 시작…댓글 24시간 제한하고 기록도 안 남아
네이버, 댓글 제한 이용자 프로필서 닉네임과 아이디 일부 공개

배경

목적

기대효과

일정

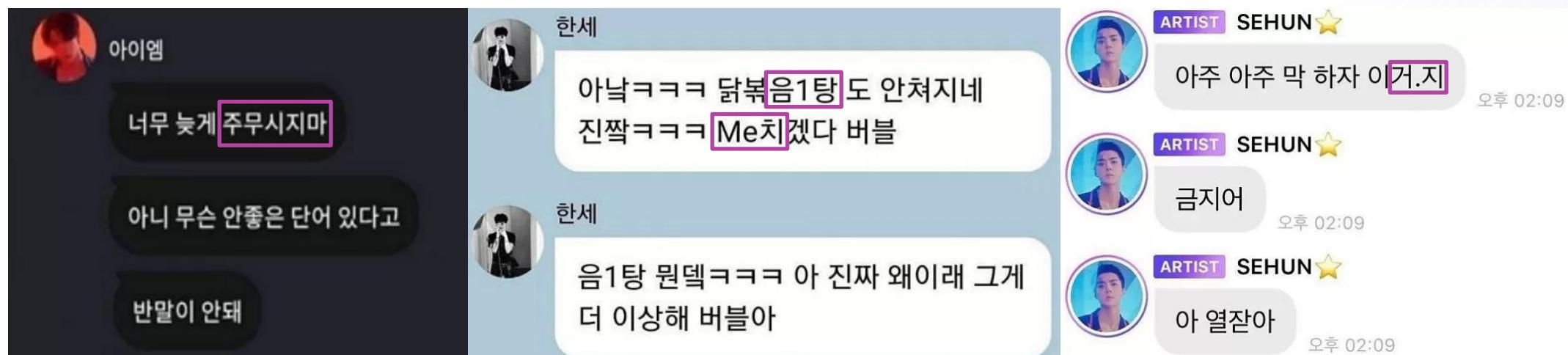
기술

1. 프로젝트 개요



기술적 문제

- 기존 서비스에서 부적절한 표현이라고 판단되면 대화의 맥락과 상관없이 메시지 전송 불가
- 하지만, 금칙어를 변형할 경우 필터링 없이 전송 가능
- 해당 어플리케이션을 사용하는 아티스트 및 사용자들의 불편감 토로



배경



목적

기대효과

일정

기술

1. 프로젝트 개요

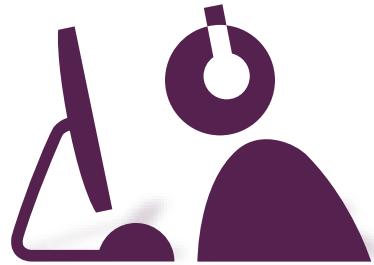


❖ AI 기반 악성 메시지 필터링이 적용된 아티스트-팬덤 채팅 서비스 ❖

- ⌚ 아티스트와의 소통에서 비윤리적 문장을 자동으로 필터링하여 건강한 채팅 문화를 형성
- ⌚ 웹 기반의 채팅 서비스로 접근성 향상
- ⌚ 아티스트 피드, 오픈 메시지 등 아티스트와 소통할 수 있는 서비스 제공



1. 프로젝트 개요



사용자 관점

쾌적한 커뮤니티 환경 제공

비윤리적 언어를 자동으로
필터링함으로써 유저들의
긍정적 경험 증대



관리자 관점

관리자의 시간과 비용 감소

수동 제재 방식에서 자동
필터링으로 변화함으로써
인적 비용 감소
기계적 필터링에서 AI 기반
필터링으로 서비스 개선,
유저 이탈률 감소



공익적 효과

아티스트 온라인 공격 보호

무분별하게 전달되는 악성
메시지로부터 자유로운
온라인 공간을 제공
사회적 문제 해결에 도움



비즈니스 활용

다른 서비스에 적용 기대

학습된 모델을 이용하여
채팅서비스 뿐만 아니라
커뮤니티 게시물, SNS 등
여러 분야에 적용 가능



배경



목적



기대효과



일정



기술

1. 프로젝트 개요

5/11 (목)

5/15 (월)

5/18 (목)

5/22 (월)

5/25 (목)

5/29 (월)

6/5 (월)

6/8 (목)

6/12 (월)

1차 목표: 채팅 서비스 구현	
공통	협업 툴 참여하기: Notion, Github, Figma, Google Drive
▶ 서비스 설계	
프론트	화면 설계도 작성
백엔드	DB 설계도 작성
백엔드	시스템 구성도 작성
▶ 서비스 구현 및 테스트	
프론트	메인 페이지 및 기능 구현
프론트	로그인 페이지 및 기능 구현
프론트	회원가입 페이지 및 기능 구현
백엔드	유저 DB 구성
백엔드	채팅 기능 구현
프&백	웹페이지 서버 연결
▶ 비윤리 문장 탐지 모델 만들기	
데이터	핵심 데이터 수집
데이터	데이터 전처리
데이터	NLP 모델 탐색 및 데이터 학습
데이터	NLP 모델 비교 & 모델 선정

배경

목적

기대효과

일정

기술

1. 프로젝트 개요

5/11 (목)

5/15 (월)

5/18 (목)

5/22 (월)

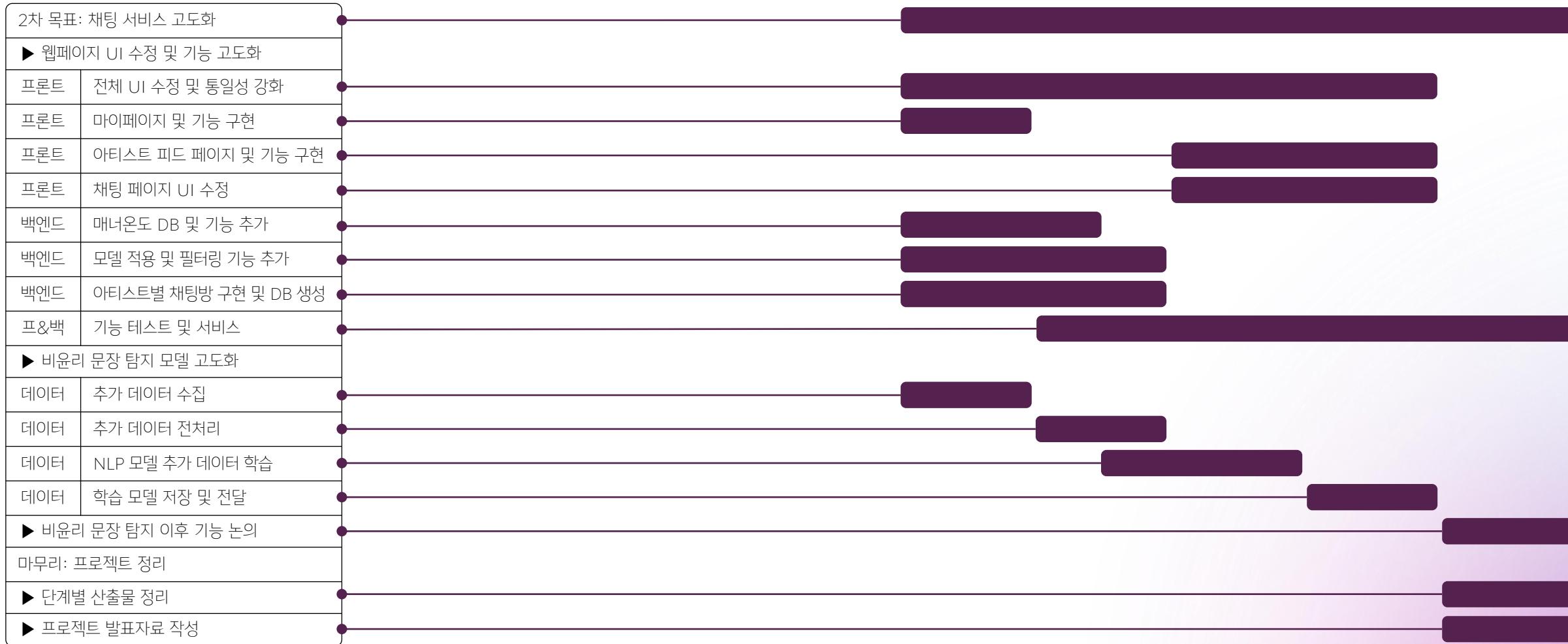
5/25 (목)

5/29 (월)

6/5 (월)

6/8 (목)

6/12 (월)



배경

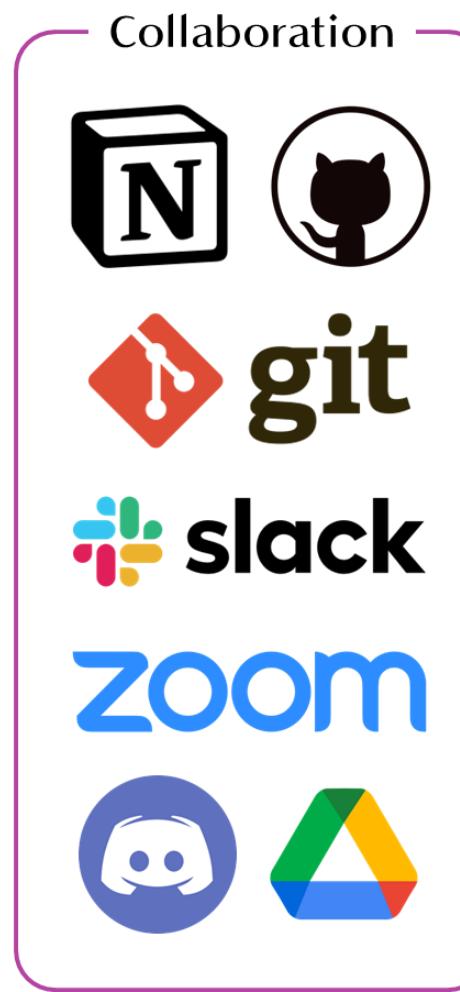
목적

기대효과

일정

기술

1. 프로젝트 개요



배경

목적

기대효과

일정

기술

2. 데이터 수집 및 분석

* 데이터 수집 — 데이터 전처리 — 딥러닝 모델 학습 — 딥러닝 모델 평가

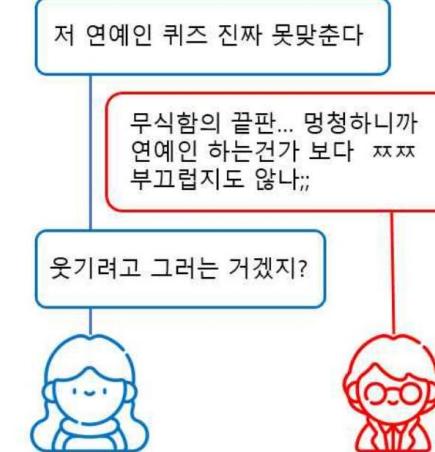
2. 데이터 수집 및 분석

❖ 텍스트 윤리검증 데이터

<https://aihub.or.kr/aihubdata/data/view.do?dataSetSn=558>

The screenshot shows the AI Hub website interface. At the top, there are navigation links: AI 데이터찾기, AI 개발지원, 참여하기, 정보공유, 고객지원, AI 허브소개, and a login link. Below the header is a search bar with three tags: #인공지능 윤리, #윤리검증, and #대화형 에이전트. The main content area features a large image of a smiling magnifying glass icon. The title '텍스트 윤리검증 데이터' is displayed prominently. Below the title, there are buttons for '다운로드' (Download) and '샘플 데이터' (Sample Data). Technical details are listed: 간신년월: 2023-05, 구축년도: 2021, 조회수: 6,942, 다운로드: 505, 용량: 45.52 MB. A '관심데이터 등록' button with a count of 42 is also present.

가정된 대화 상황

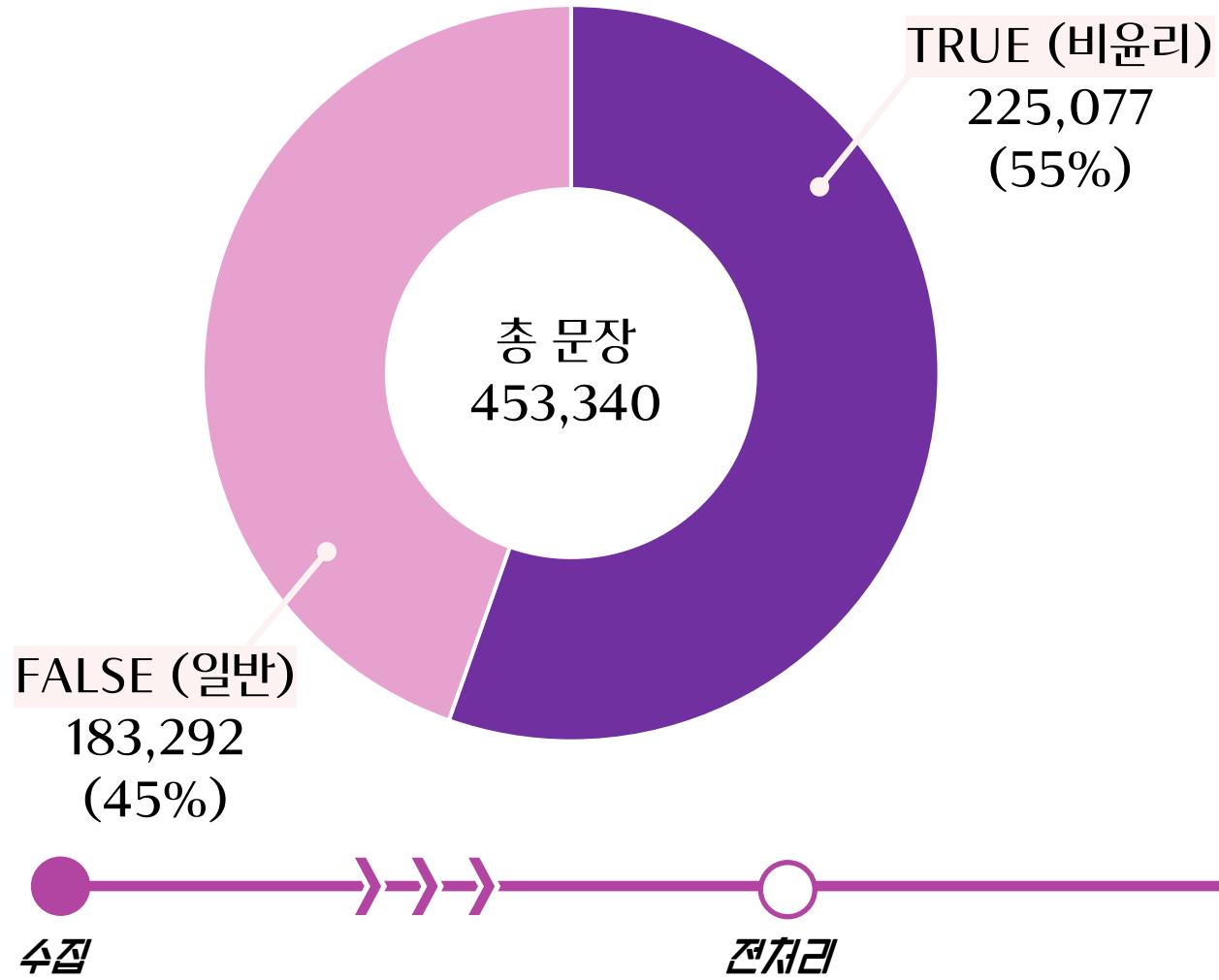


[그림] 대화세트 예시

텍스트

화자 1	저 연예인 퀴즈 진짜 못 맞춘다 @비윤리: False
화자 2	"무식함의 끝판... 멍청하니까 연예인 하는건가 보다 짜ㅉ 부끄럽지도 않나;;" @비윤리: True @비윤리 유형: 비난 @비윤리 강도: {1점, 1점, 1점, 1점, 1점} (3점 만점으로 5명이 투표) @문형: V1(무식하다)ㅁ의 N1(끝판) V2(멍청하 다)니까 N2(연예인) V3(하다)는N3(거)ㄴ가 V4(보다)다 V5(부끄럽다)지도 V6(않다)나
화자 1	"웃기려고 그러는 거겠지?" @비윤리: False

◆ 텍스트 윤리검증 데이터

<https://aihub.or.kr/aihubdata/data/view.do?dataSetSn=558>

원시데이터 특성

- 453,340 문장 (132,807 대화세트)
- 참고 데이터 활용 – 비윤리적 문장 직접 생성
- 1개 이상의 비윤리 문장을 포함하는,
발화자 2명 이상의 대화세트로 구성
- 직접적 비윤리 표현은 금칙어 지정

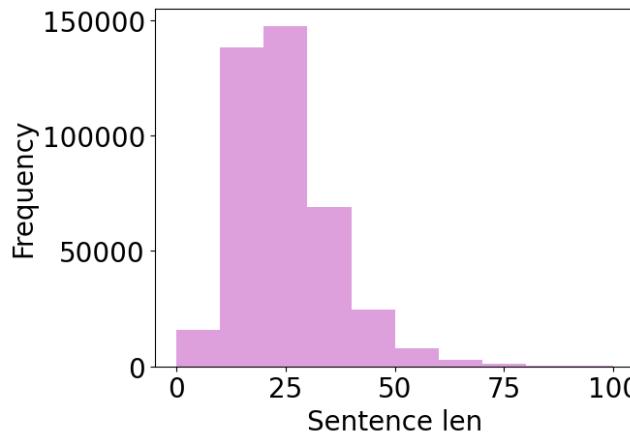
데이터 정제

- 부정적 맥락 속 인물/기업/상품 등 비식별화

라벨링

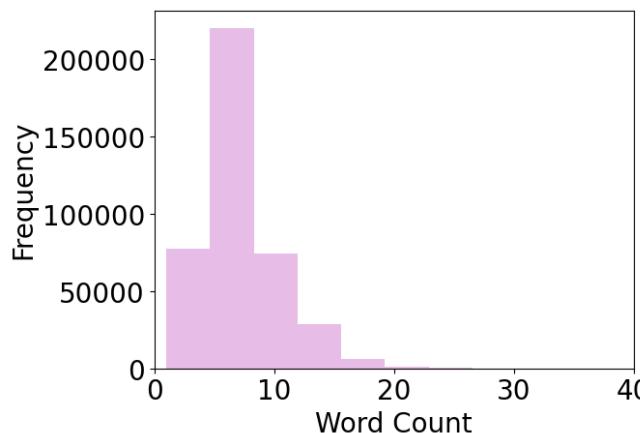
- 5명의 크라우드 워커가 문장에 대해 비윤리 여부, 비윤리 유형, 비윤리 강도 라벨링

2. 데이터 수집 및 분석



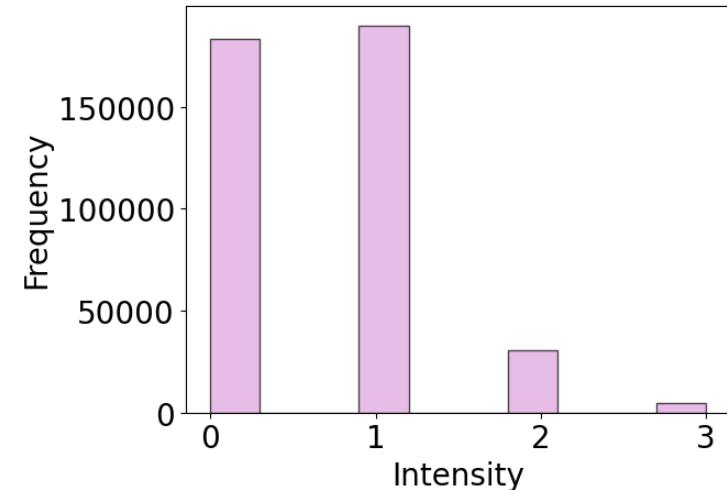
문장 길이 분포도

- * Min Length: 1
- * Max Length: 297
- * 대부분 10~50 이내 문장
- * 인코딩 과정에서 패딩 처리



문장 단어 수 분포도

- * nltk 라이브러리의 word_tokenize
- * 각 문장의 단어 수 분포
- * 대부분 10개 이내의 단어로 이루어짐



비윤리 문장 강도 분포도

- * 0(False)은 비윤리적이지 않은 문장
- * 1~3(True)은 비윤리적인 문장
- * 3으로 갈수록 비윤리 강도가 큰 문장

수집

전처리

모델 학습

모델 평가

* JSON 파일 구조

1) id	대화세트 id																																		
2) sentences	<table border="1"> <tr> <td>2-1) id</td><td>문장 id</td></tr> <tr> <td>2-2) speaker</td><td>대화세트 내 발화자 구분</td></tr> <tr> <td>2-3) origin_text</td><td></td></tr> <tr> <td>2-4) text</td><td>비식별화된 텍스트</td></tr> <tr> <td>2-5) types</td><td>비윤리 문장 유형(8가지)</td></tr> <tr> <td>2-6) is_immoral</td><td>비윤리 문장 여부(2가지)</td></tr> <tr> <td>2-7) intensity</td><td>비윤리 강도 평균 점수</td></tr> <tr> <td>2-8) intensity_sum</td><td>비윤리 강도 총합 점수</td></tr> <tr> <td>2-9) votes</td><td> <table border="1"> <tr> <td>intensity</td><td></td></tr> <tr> <td>voter</td><td>gender</td></tr> <tr> <td></td><td>age</td></tr> </table> </td></tr> <tr> <td>2-10) frame_id</td><td></td></tr> <tr> <td>2-11) mapped_slots</td><td> <table border="1"> <tr> <td>slot</td><td></td></tr> <tr> <td></td><td>token</td></tr> <tr> <td></td><td>lu_id</td></tr> </table> </td></tr> </table>	2-1) id	문장 id	2-2) speaker	대화세트 내 발화자 구분	2-3) origin_text		2-4) text	비식별화된 텍스트	2-5) types	비윤리 문장 유형(8가지)	2-6) is_immoral	비윤리 문장 여부(2가지)	2-7) intensity	비윤리 강도 평균 점수	2-8) intensity_sum	비윤리 강도 총합 점수	2-9) votes	<table border="1"> <tr> <td>intensity</td><td></td></tr> <tr> <td>voter</td><td>gender</td></tr> <tr> <td></td><td>age</td></tr> </table>	intensity		voter	gender		age	2-10) frame_id		2-11) mapped_slots	<table border="1"> <tr> <td>slot</td><td></td></tr> <tr> <td></td><td>token</td></tr> <tr> <td></td><td>lu_id</td></tr> </table>	slot			token		lu_id
2-1) id	문장 id																																		
2-2) speaker	대화세트 내 발화자 구분																																		
2-3) origin_text																																			
2-4) text	비식별화된 텍스트																																		
2-5) types	비윤리 문장 유형(8가지)																																		
2-6) is_immoral	비윤리 문장 여부(2가지)																																		
2-7) intensity	비윤리 강도 평균 점수																																		
2-8) intensity_sum	비윤리 강도 총합 점수																																		
2-9) votes	<table border="1"> <tr> <td>intensity</td><td></td></tr> <tr> <td>voter</td><td>gender</td></tr> <tr> <td></td><td>age</td></tr> </table>	intensity		voter	gender		age																												
intensity																																			
voter	gender																																		
	age																																		
2-10) frame_id																																			
2-11) mapped_slots	<table border="1"> <tr> <td>slot</td><td></td></tr> <tr> <td></td><td>token</td></tr> <tr> <td></td><td>lu_id</td></tr> </table>	slot			token		lu_id																												
slot																																			
	token																																		
	lu_id																																		

전처리

모델 학습

모델 평가

수집



2. 데이터 수집 및 분석

◆ 모델 학습 데이터 전처리

```

rows = []

# 데이터 추출
for item in data:
    sentences = item["sentences"]
    for sentence in sentences:
        row = [
            sentence["id"].split('-')[1],
            sentence["is_immoral"],
            sentence["intensity"],
            sentence["speaker"],
            sentence["types"],
            sentence["text"]
        ]
        rows.append(row)

# 데이터프레임 생성
df = pd.DataFrame(rows, columns=["order", "Is_Immoral", "Intensity", "Speaker", "Types", "Text"])

# CSV 파일로 저장
csv_file = "train_1.csv"
df.to_csv(csv_file, index=False)

```

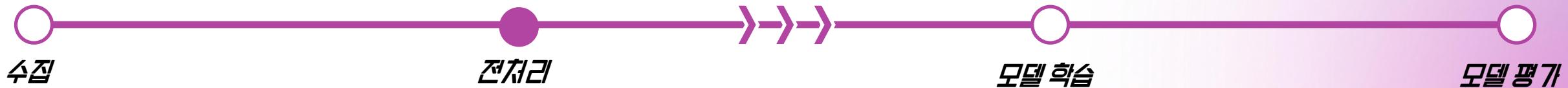
order	Is_Immoral	Intensity	Speaker	Types	Text
1	TRUE	2	1	['CENSURE']	부랴부랴 왔는데 아무도 안왔네. 시간개념들이 없네
2	TRUE	2	2	['CENSURE']	맞아. 사람들이 진짜 개념이없다
3	TRUE	2	1	['CENSURE']	저렇게는 살지 말아야지
4	TRUE	2	2	['CENSURE']	그러게 게으르고 멍청한 사람들은 맞아야해
1	TRUE	1	1	['CENSURE']	인방 보는 남자는 거르는게 맞다
2	TRUE	2	2	['HATE', 'CENSURE', 'SEXUAL']	특히 벗방보는 애들은 진짜 거세 시켜야함
3	TRUE	2	3	['CENSURE', 'SEXUAL']	볼 게 없어서 벗방을 보느냐 ㅋㅋㅋ
4	TRUE	1	4	['CENSURE']	사회성이 얼마나 떨어지면 그려느냐 ㅋㅋㅋ

(참고) 대화셋 2세트 예시

전처리 후 훈련 데이터셋

- 408,369 문장
- 비윤리 여부(Is_Immoral)

TRUE 225,077개, FALSE 183,292개

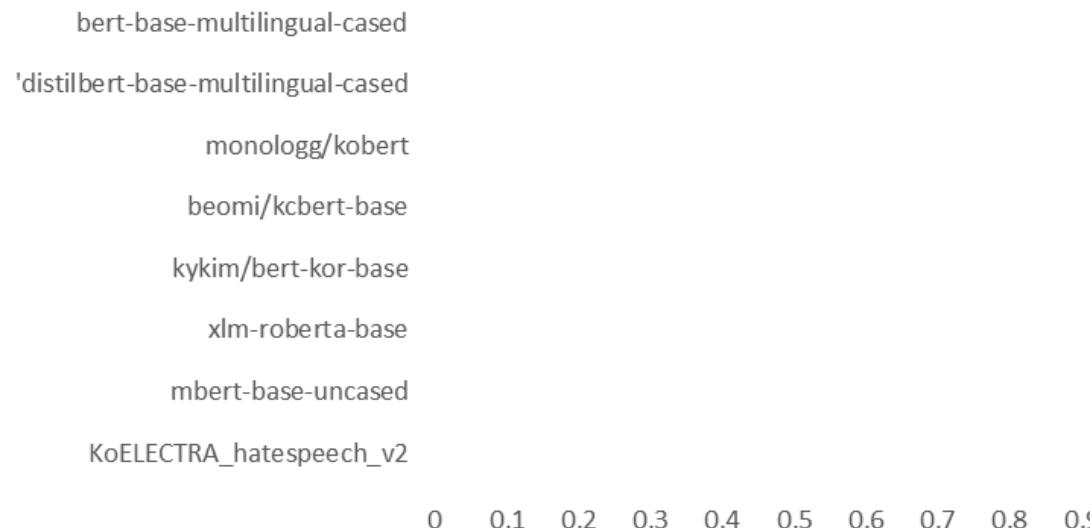


2. 데이터 수집 및 분석

◆ 학습 모델 정확도 비교



<https://huggingface.co/models>



학습환경

CPU : Intel Core i7-12700
 GPU : GeForce RTX 3060TI(Nvidia) CUDA ver:11.2
 RAM : 32GB
 OS: Window10

수집

전처리

모델 학습

모델 평가

<한국어 자연어 딥러닝 모델>

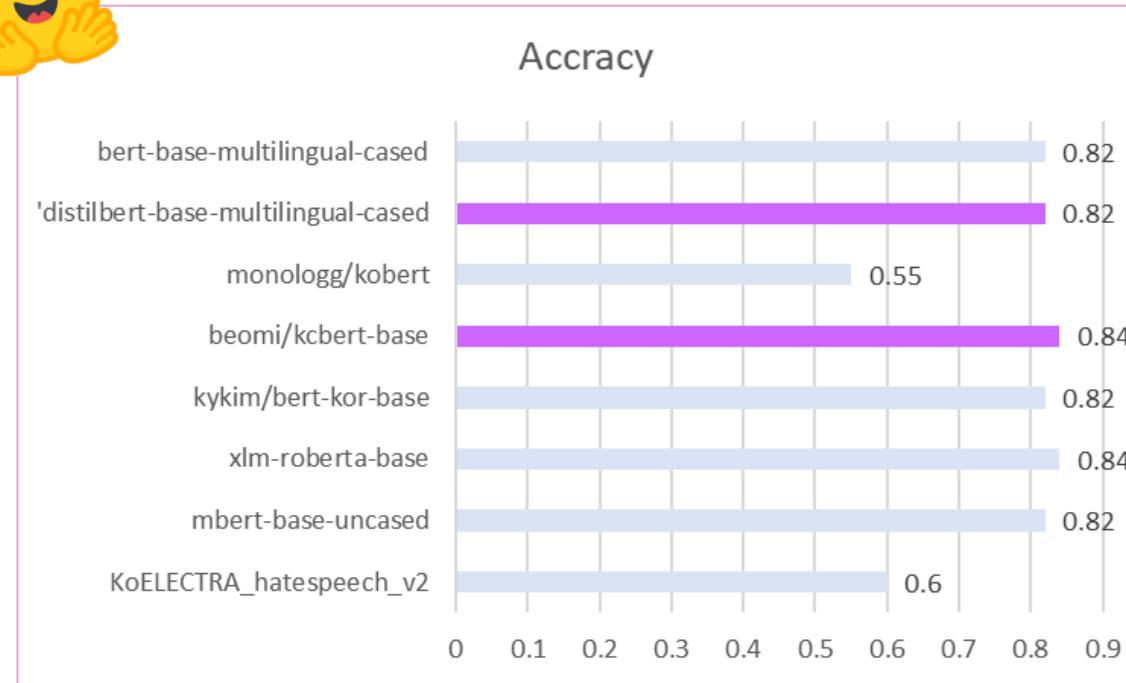
1. BERT – NLP 관련 대표적 모델.
마스크드 언어 모델(MLM), 다음 문장 예측(NSP) 방법으로 사전 훈련됨.
2. DistilBERT – BERT 모델의 경량화 버전. 다국어 지원.
3. KoBERT – BERT 모델의 한국어 학습 모델. 한국어 위키, 기사 등 학습.
4. KcBERT – 구어체, 신조어가 포함된 뉴스 댓글 학습 모델.
5. BERT-kor – 한국어 주제별 텍스트 학습 BERT 모델.
6. xlm-RoBERTa – 튜닝된 BERT의 다국어 지원 모델.
7. mBERT – 언어 간 특성을 학습한 모델. 기계 번역에 주로 활용.
8. KoELECTRA-hatespeech – 한국어 혐오 표현을 학습한 모델.

2. 데이터 수집 및 분석

◆ 학습 모델 정확도 비교



<https://huggingface.co/models>



학습환경

CPU : Intel Core i7-12700
GPU : GeForce RTX 3060TI(Nvidia) CUDA ver:11.2
RAM : 32GB
OS: Window10

<모델별 특성과 비교 기준>

- 정확도가 낮은 모델 제외 – KoBERT, KoELECTRA
- 변형된 욕설 탐지에 강점을 보인 모델 – mBERT
3. 이름(예: 철식아)에 대한 오탐지가 없는 모델
 - DistilBERT-multilingual, KcBERT
- 문장이 길어질수록 비윤리 문장으로 오탐지하는 경우가 있음

수집

전처리

모델 학습

모델 평가

2. 데이터 수집 및 분석

◆ 학습 모델 정확도 비교

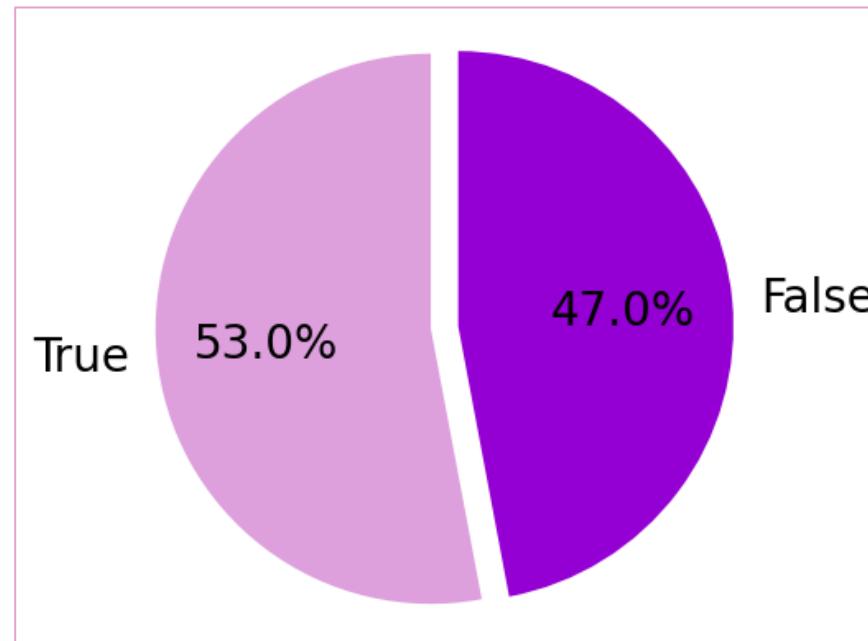
모델 종류	Distilbert-base-multilingual-case	beomi/kcbert-base
모델 특징	<ul style="list-style-type: none"> → BERT 모델을 경량화한 버전 → 다국어 이진 분류 작업에 적합 → 기본 BERT보다 모델 크기가 작고 추론 속도가 빠름 → 다국어 문장 분류, 감성 분석, 스팸 분류 등에 활용 	<ul style="list-style-type: none"> → 한국어 문장 분류를 위한 사전 학습된 BERT 모델 → 한국어 테스트의 이진 분류 작업에 특화 → 감성 분석, 스팸 분류 등에 활용
모델 Test 결과	<ul style="list-style-type: none"> → 80%이상의 정확도로, 비교적 안정적인 속도와 일관성을 보임 → 이름과 고유명사는 비윤리적이지 않은 문장으로 판단 (오탐지율 낮음) → 변형된 욕설과 신조어가 들어가 있는 문장에 대한 오탐지 가능성 존재 	



2. 데이터 수집 및 분석

◆ 데이터 추가 수집 및 학습

- 스마일게이트 한국어 욕설 데이터
 - <https://github.com/2runo/Curse-detection-data/blob/master/dataset.txt>
- 온라인 뉴스 기사 혐오 댓글 데이터 (korean-hate-speech)
 - <https://github.com/kocohub/korean-hate-speech/blob/master/labeled/train.tsv>



채팅 필터링 성능을 향상시키기 위해
직접적인 욕설과 변형된 욕설이 포함된 문장,
비윤리 강도가 높은 문장 추가 수집

문장(sentences): 13,721개
비윤리 문장 TRUE (6,454개)
일반 문장 FALSE (7,267개)

Is_Immoral	Text
TRUE	(현재 호텔주인 심정) 아18 난 마른하늘에 날벼락맞고 호텔망하게생겼는데 누군 계속 추모받네....
FALSE한국적인 미인의 대표적인 분...너무나 곱고아름다운모습...그모습뒤의 슬픔을 미처 알지못했네요ㅠ

수집

전처리

모델 학습

모델 평가

2. 데이터 수집 및 분석

◆ 최종 모델 선정

KcBERT: Korean Comments BERT

- ☰ 구어체와 신조어, 특수문자, 이모지 등 한국어 댓글 데이터
약 1억 1천 만 개를 활용하여 사전 훈련된 BERT 모델
- ☰ 한국어 자연어 처리 작업에 특화

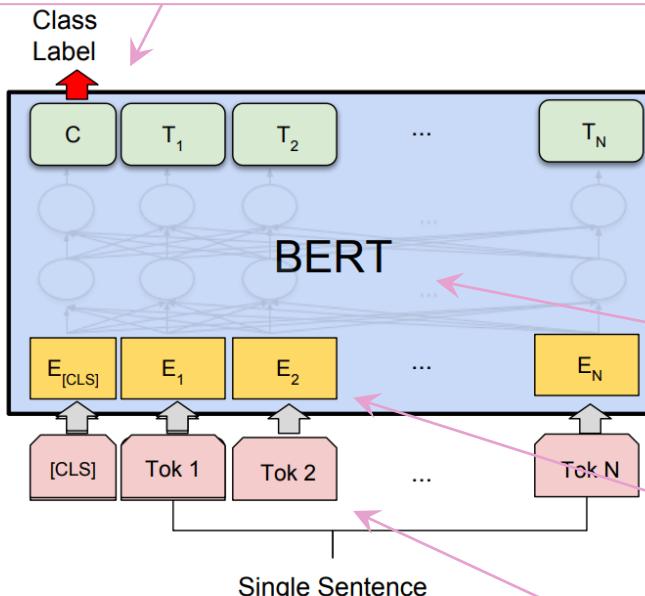
3. BERT 모델 적용: 인코더가 여러 층을 거치며 self-attention과 MLP를 계산하고 각 토큰의 문맥 표현을 생성

2. 임베딩된 토큰은 BERT 모델에 인코더에 입력

1. 문장을 토큰 단위로 분리, 각 토큰에 대한 임베딩 생성

4. 분류를 위한 레이어 추가

model: BERTForSequenceClassification()



(b) Single Sentence Classification Tasks:
SST-2, CoLA

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

2. 데이터 수집 및 분석

◆ 모델 평가 및 테스트 – ‘Beomi/kcbert-base’

```

import torch
import torch.nn as nn
from transformers import AutoTokenizer, AutoModelForSequenceClassification
# 학습된 모델을 CPU에 로드합니다.
model_path = 'Add_beomi_model.pt'
model = AutoModelForSequenceClassification.from_pretrained('beomi/kcbert-base')
model.load_state_dict(torch.load(model_path, map_location=torch.device('cpu')))
model.eval()
# 토크나이저를 로드합니다.
tokenizer = AutoTokenizer.from_pretrained('beomi/kcbert-base')

```

추가 학습 완료한 모델 로드

토크나이저 로드

```

def classify_sentence(sentence):
    inputs = tokenizer.encode_plus(
        sentence,
        add_special_tokens=True,
        max_length=128,
        padding='max_length',
        truncation=True,
        return_tensors='pt'
    )
    input_ids = inputs['input_ids']
    attention_mask = inputs['attention_mask']

```

분류 모델을 적용한 분류 함수

1. 입력받은 문장 전처리

```

with torch.no_grad():
    outputs = model(input_ids, attention_mask=attention_mask)
    logits = outputs.logits

```

```

predicted_class = torch.argmax(logits, dim=1) # 가장 높은 로짓 값을 가진 클래스 선택
print(predicted_class, logits, outputs)
if predicted_class.item() == 1:
    return "비윤리적인 문장인가요? True"
else:
    return "비윤리적인 문장입니다? False"

```

1) 모델 TEST 진행

모델 test set : 81,673개 문장

Test Loss : 0.3898

Test Accuracy : 0.8401

2) 직접 평가 진행

```

# 성능 Test
sentence1 = "안녕하세요~~ 저는 파이썬을 공부하는 이름입니다."
sentence2 = "넌 진짜 사람이 왜그러냐.. 나같으면 너처럼 안할듯ㅋㅋ"

```

```

class1 = classify_sentence(sentence1)
class2 = classify_sentence(sentence2)

```

```

print(class1)
print(class2)

```

✓ 0.4s

```

비윤리적인 문장입니다? false
비윤리적인 문장인가요? True

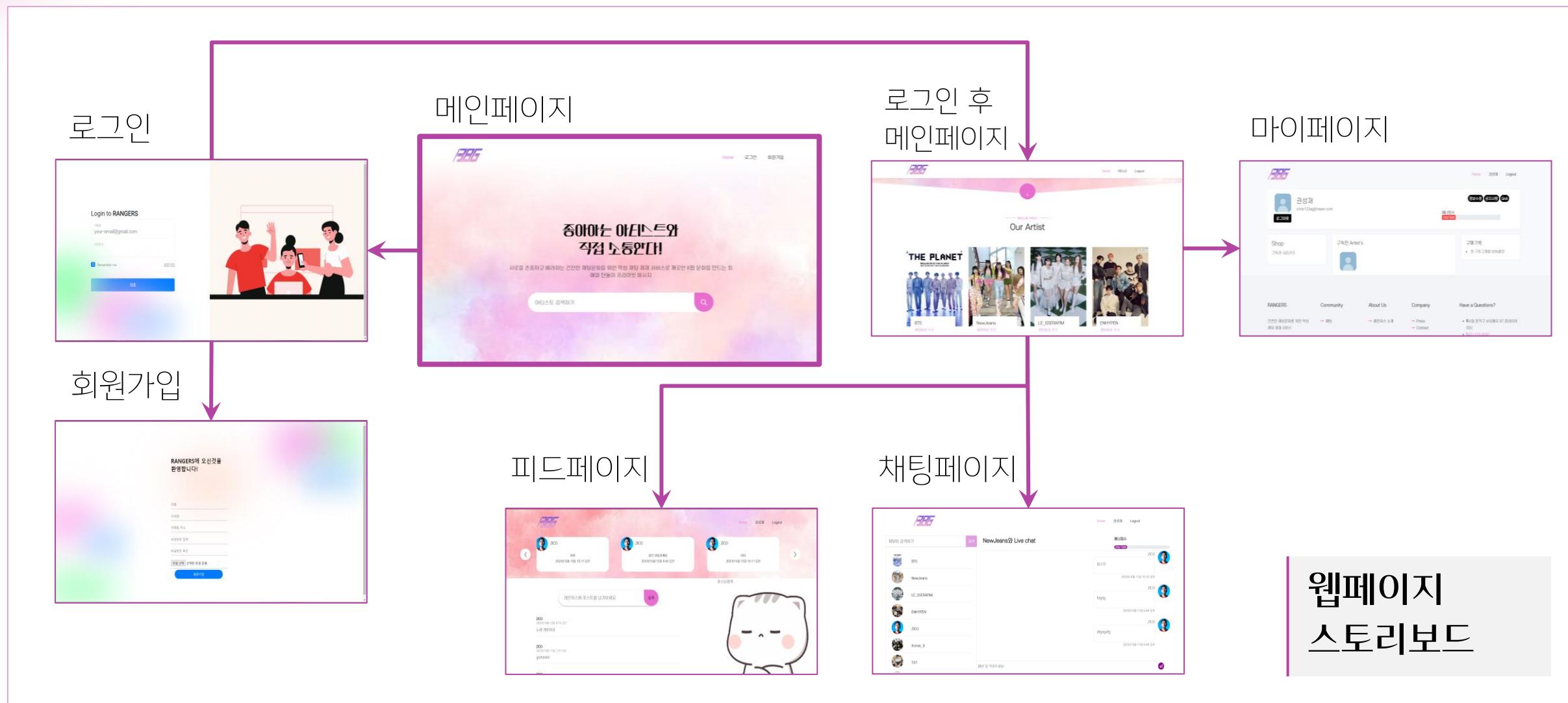
```



3. 서비스 설계 및 구현

* 서비스 설계 – 설계 구현 – 테스트

3. 서비스 설계 및 구현

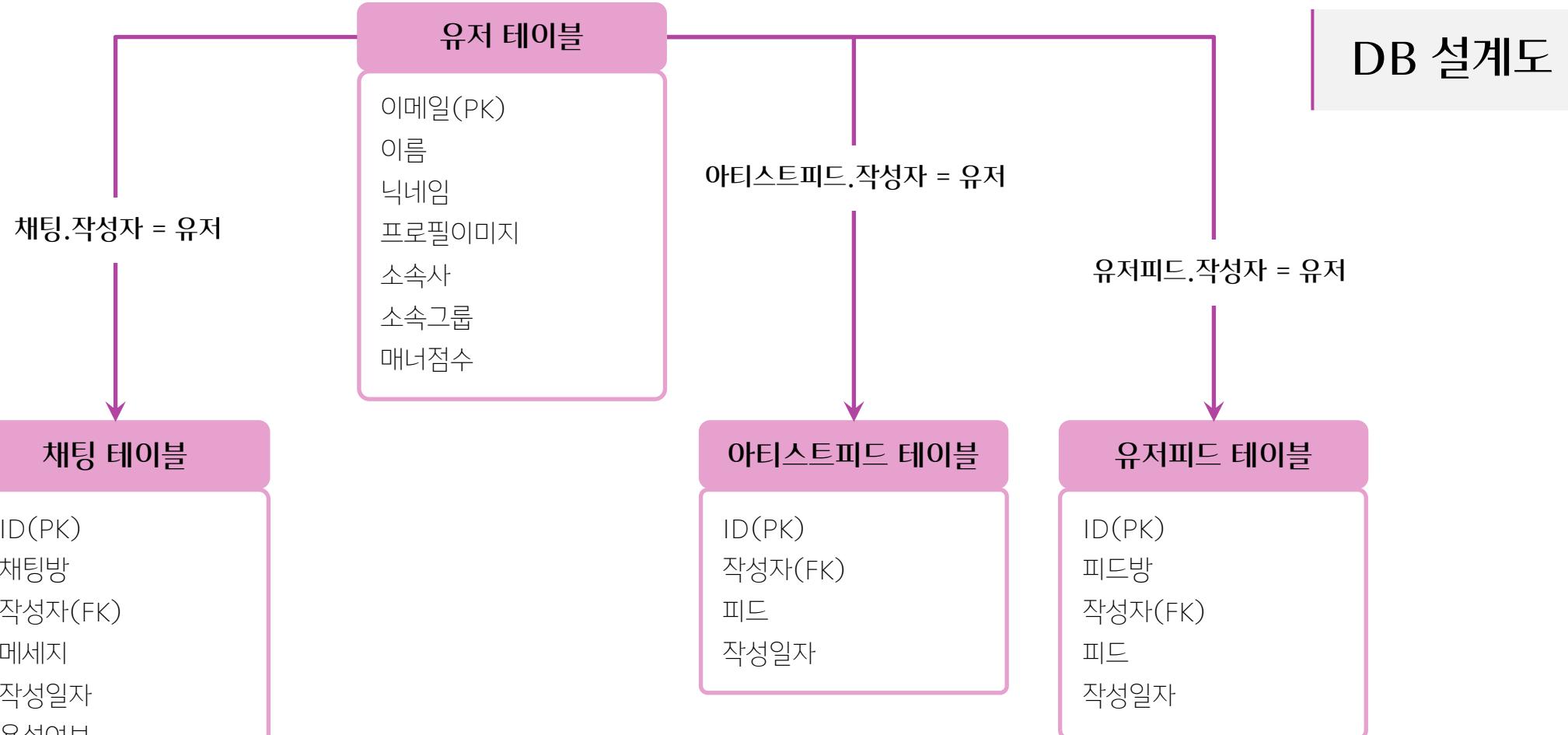


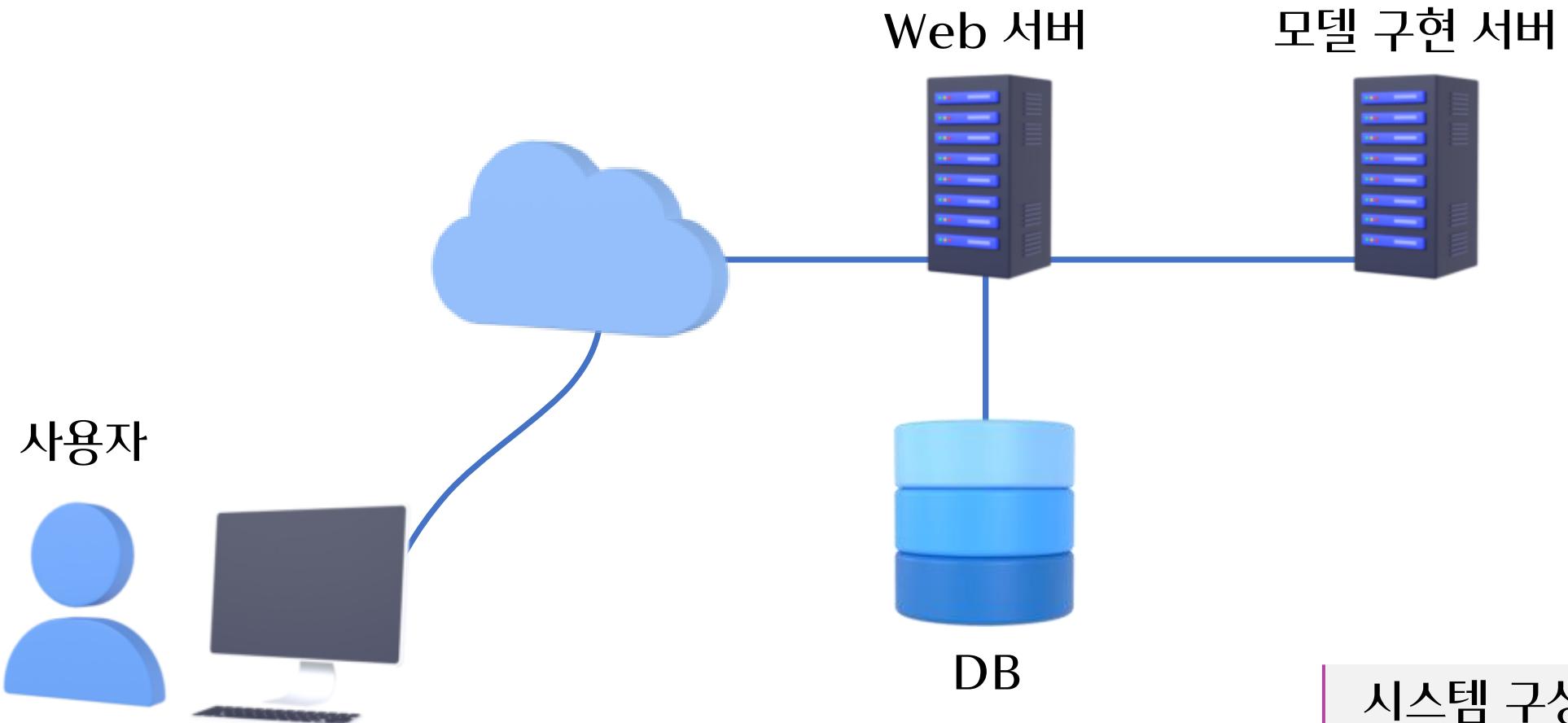
서비스 설계

설계 구현

테스트

웹페이지
스토리보드



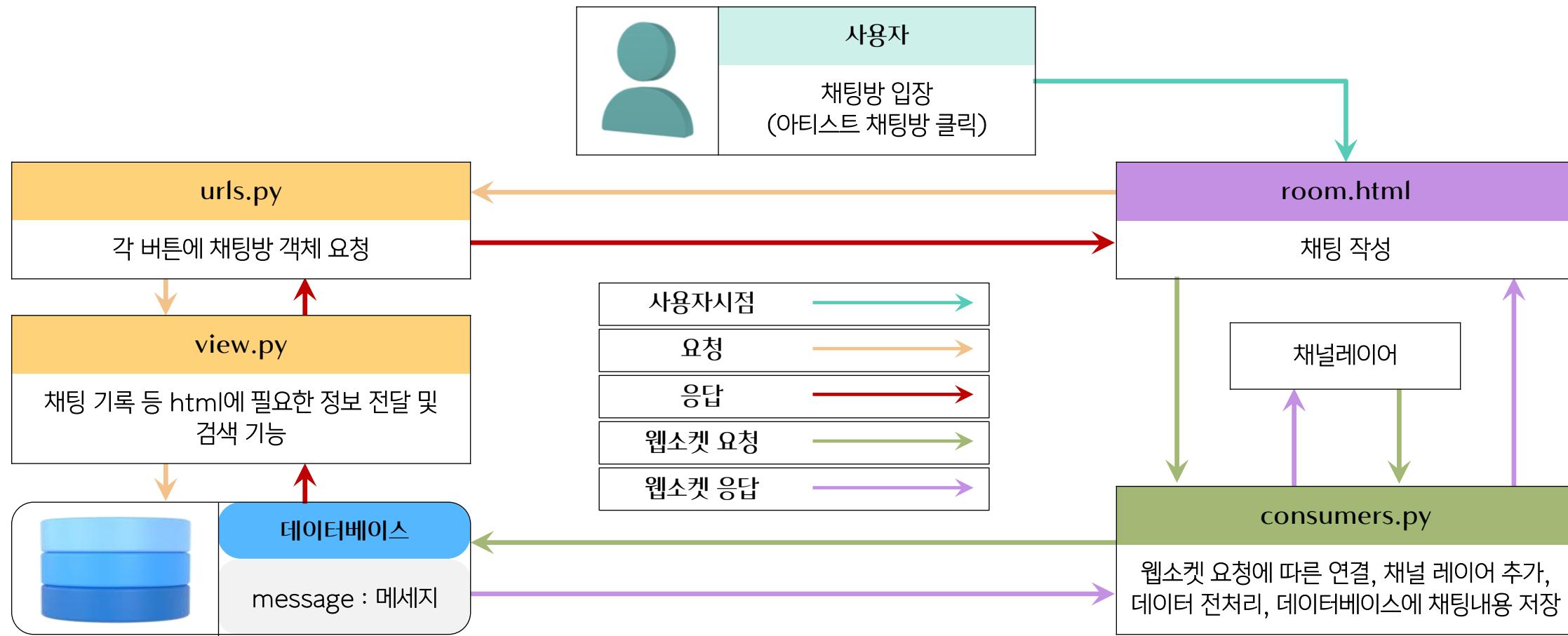


서비스 설계
→→→

설계 구현
→

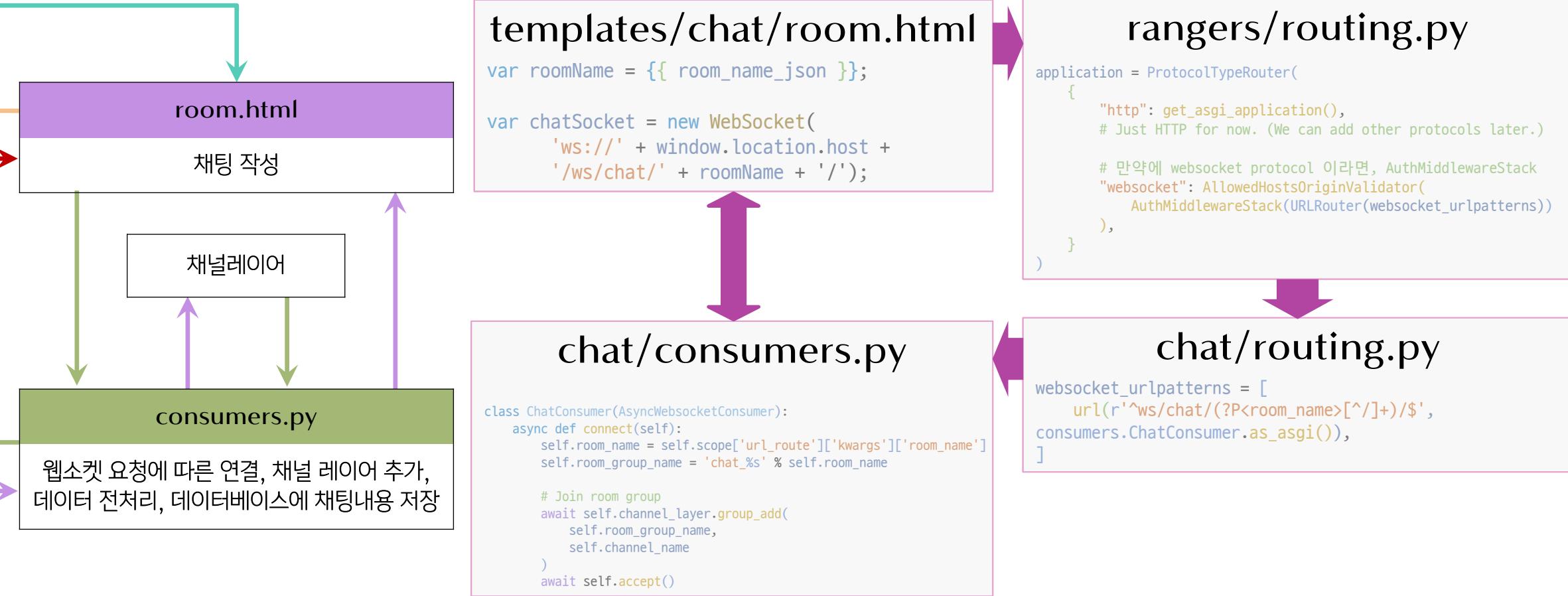
테스트

◆ 채팅 기능 구현: 웹 소켓을 활용한 비동기 처리 시스템



3. 서비스 설계 및 구현

◆ 채팅 기능 구현: 웹 소켓을 활용한 비동기 처리 시스템(연결)



◆ 채팅 기능 구현: 웹 소켓을 활용한 비동기 처리 시스템(메세지 전송)

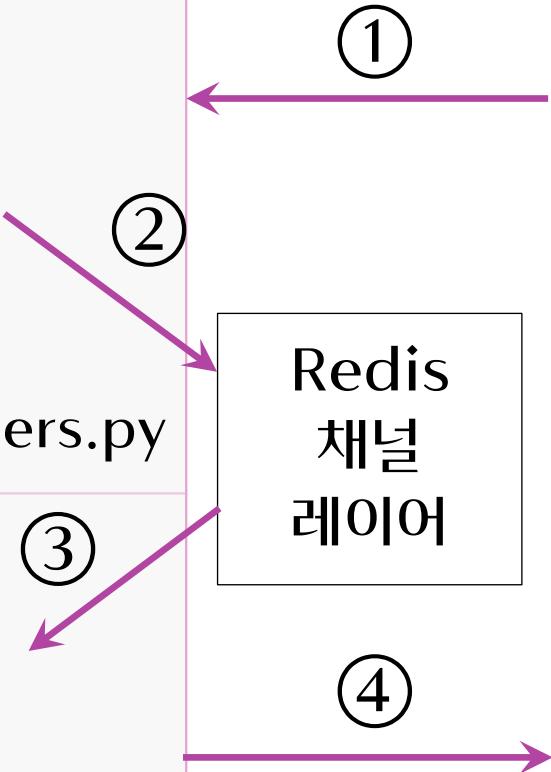
```
# Receive message from WebSocket
async def receive(self, text_data):
    text_data_json = json.loads(text_data)
    message = text_data_json['message']
    ...

# Send message to room group
await self.channel_layer.group_send(
    self.room_group_name,
    {
        'type': 'chat_message',
        'message': message,
        ...
    }
) # chat/consumers.py

# Receive message from room group
async def chat_message(self, event):
    message = event['message']

    ...

# Send message to WebSocket
await self.send(text_data=json.dumps({
    'message': message,
    ...
}))
```



templates/chat/room.html

```
document.querySelector('#chat-message-submit').onclick = function(e) {
    var messageInputDom = document.querySelector('#chat-message-input');
    var message = messageInputDom.value;
    ...
    chatSocket.send(JSON.stringify({
        'message': message,
        ...
    }));
    messageInputDom.value = '';
}

chatSocket.onmessage = function(e) {
    var data = JSON.parse(e.data);
    var message = data['message'];
    ...
    const element = document.getElementById('add_message');
    ...
    element.innerHTML += (
```

3. 서비스 설계 및 구현

◆ 딥러닝 모델 호출 – 비윤리 문장 판별

```

model_path = os.path.join(os.path.dirname(__file__), 'Add_beomi_model.pt')
model = AutoModelForSequenceClassification.from_pretrained('beomi/kcbert-base')
model.load_state_dict(torch.load(model_path, map_location=torch.device('cpu')))
model.eval()
tokenizer = AutoTokenizer.from_pretrained('beomi/kcbert-base')

# 새로운 문장을 분류하는 함수를 정의합니다.
def classify_sentence(sentence):
    inputs = tokenizer.encode_plus(
        sentence,
        add_special_tokens=True,
        max_length=128,
        padding='max_length',
        truncation=True,
        return_tensors='pt'
    )
    input_ids = inputs['input_ids']
    attention_mask = inputs['attention_mask']
    # 추론을 수행합니다.
    with torch.no_grad():
        outputs = model(input_ids, attention_mask=attention_mask)
        logits = outputs.logits
    # 소프트맥스 활성화 함수를 적용하고 예측된 클래스를 가져옵니다.
    probabilities = nn.functional.softmax(logits, dim=1)
    predicted_class = torch.argmax(probabilities, dim=1)
    return predicted_class.item()

```

chat/filtering.py

```

user = User.objects.get(nickname=nickname)

lastdate = lastdate.split(':')
lastdate = int(lastdate[0])*24*60+int(lastdate[1])*60+int(lastdate[2])

nowtime = time.split(' ')[1]
nowtime = nowtime.split(':')
nowtime = int(nowtime[0])*24*60+int(nowtime[1])*60+int(nowtime[2])
vilification = bool(classify_sentence(message))

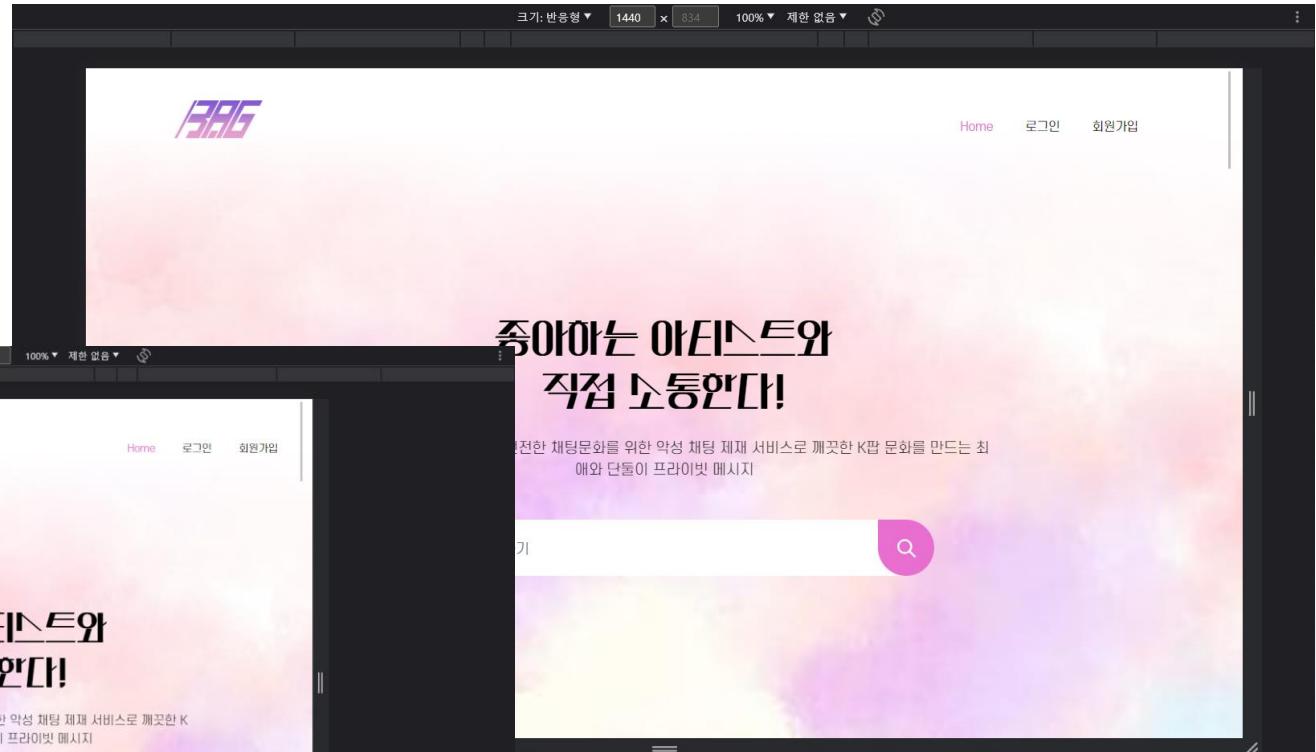
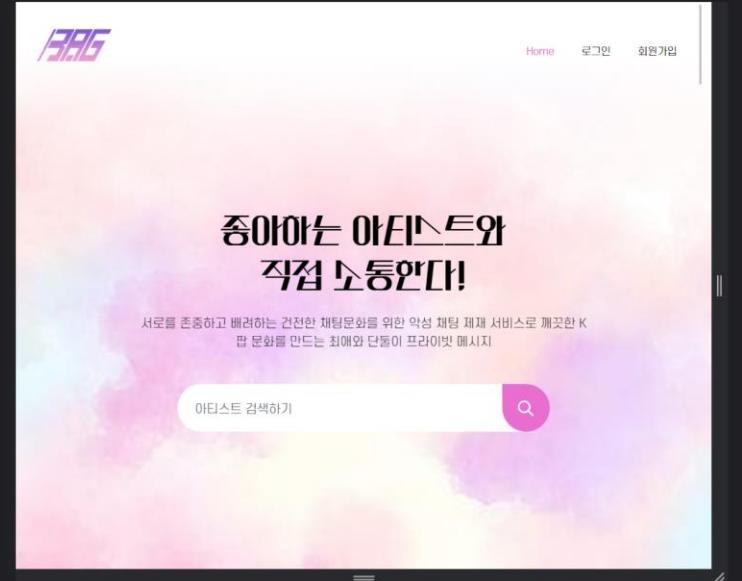
if message != '':
    if not (user.id==lastnickname and message==lastchatmessage and abs(lastdate-nowtime)<=2):
        if vilification: # 비윤리적 문장이면
            Chat.objects.create(roomname=urllist[-2],author_id=user.id,
                                message=message,create_date=time,vilification=vilification)
            user.counting-=30
            if user.counting<0:
                user.counting = 0
            user.save()
        else: # 비윤리적 문장이 아니면
            Chat.objects.create(roomname=urllist[-2],author_id=user.id,
                                message=message,create_date=time,vilification=vilification)
            if user.counting<1000:
                user.counting+=1
            user.save()

```

chat/consumers.py



3. 서비스 설계 및 구현



서비스 설계

설계 구현

레스터



4. 서비스 시연

[Home](#)[로그인](#)[회원가입](#)

좋아하는 아티스트와 직접 노동한다!

서로를 존중하고 배려하는 건전한 채팅문화를 위한 악성 채팅 제재 서비스로 깨끗한 K팝 문화를 만드는 최
애와 단둘이 프라이빗 메시지

[아티스트 검색하기](#)



Q E A

BIG

THANK YOU

Special thanks to 이원영 멘토님

* 386 RANGERS *

이예지 강혜정 권성재 윤철식 이준희

PLAYDATA - DE 19

23.06.16.