TEAM PROJECT

AURA

AI 보이스피싱 탐지 어플

발표일: 2025. 09. 25.

TEAM AURA

00

TEAM AURA

- 김시준 : 프로젝트 총괄, whisper모델, 백엔드
- 김채연 : 피그마 문서 작업 및 발표 자료 제작
- 이민서 : UniSpeech-SAT 모델, 앱 배포
- 이민우 : LLM모델(대화분석+챗봇), 프론트
- 한상원 : 발표 + 라이센스 정리

CONTENTS

01

어플 개발 배경

01

어플 개발 배경

프로젝트 개요

보이스피싱 증가

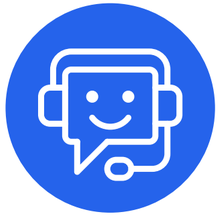
보이스피싱 증가 추세 대응형 시스템
현황 분석 기반 능동적 대응

기존 방식의 한계

기존 방식의 한계 극복 솔루션 제시
구조적 문제점 해결 중심 설계

선제적 예방

선제적 예방조치 통합 플랫폼
사전 차단 중심의 방어체계 구축



AI 기능 도입

AI 기능 도입 스마트 방어 시스템
인공지능 기반 고도화된 탐지 능력

어플 개발 배경

AURA의 차별점

보이스 피싱 탐지

데이터 소스

- 실시간 통화
- 녹음된 통화 음성 파일

기술 구성 요소

- 음성 변조 탐지 : Whisper 기반 학습 모델
- 사기 통향 탐지 : LLM(Qwen 모델) 기반 탐지

정확도 향상

모델 성능 개선

- 일반음성 + 변조음성 데이터셋 구축
- Whisper 모델 추가 파인튜닝 수행

전처리 기법 적용

- 잡음 제거
- 음성 향상
- 음성 품질 최적화

확장성

서비스 확장 방향

- 일정 등록 시스템 연동
- 챗봇 상담 서비스 통합
- 통화 요약 기능

추가 기능 개발

- 실시간 위험도 알림
- 사기 패턴 학습 및 업데이트
- 다국어 지원

어플 개발 배경

01

Whisper 모델 소개

장점

- 다국어 지원
- 잡음 환경에서도 우수한 성능
- STT 품질 우수 → 텍스트 기반 사기 패턴 분석 최적

단점

- 스푸핑 전용 탐지 성능은 SSL 대비 약함
- 추가 헤드/파인튜닝 필요
- 모델 사이즈 큼 → 실시간성·배포 부담

추천 활용

- Whisper: STT 전담

  (음성→ 텍스트 변환)
- 스푸핑 탐지: SSL 모델(Wav2Vec2, DistilHuBERT 등) 활용
- 선택적: Whisper Encoder 임베딩 + 이진 분류기 결합

02

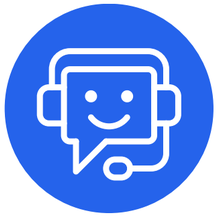
핵심 기능 및 코드

핵심 기능 및 코드

핵심 기능

음성 변조 탐지

Whisper 기반 모델을 활용하여 실시간 통화나 녹음된 음성 파일에서 인위적 음성 변조 및 딥페이크 음성을 탐지

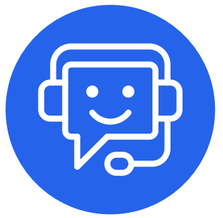


보이스피싱 탐지AI

Qwen LLM을 활용하여 통화 내용을 실시간 분석하고 사기 패턴, 키워드, 화법을 종합적으로 판단하는 지능형 탐지 시스템

어플 연동

어플 연동을 통해 사용자의 편의성 확보 및 보안 강화

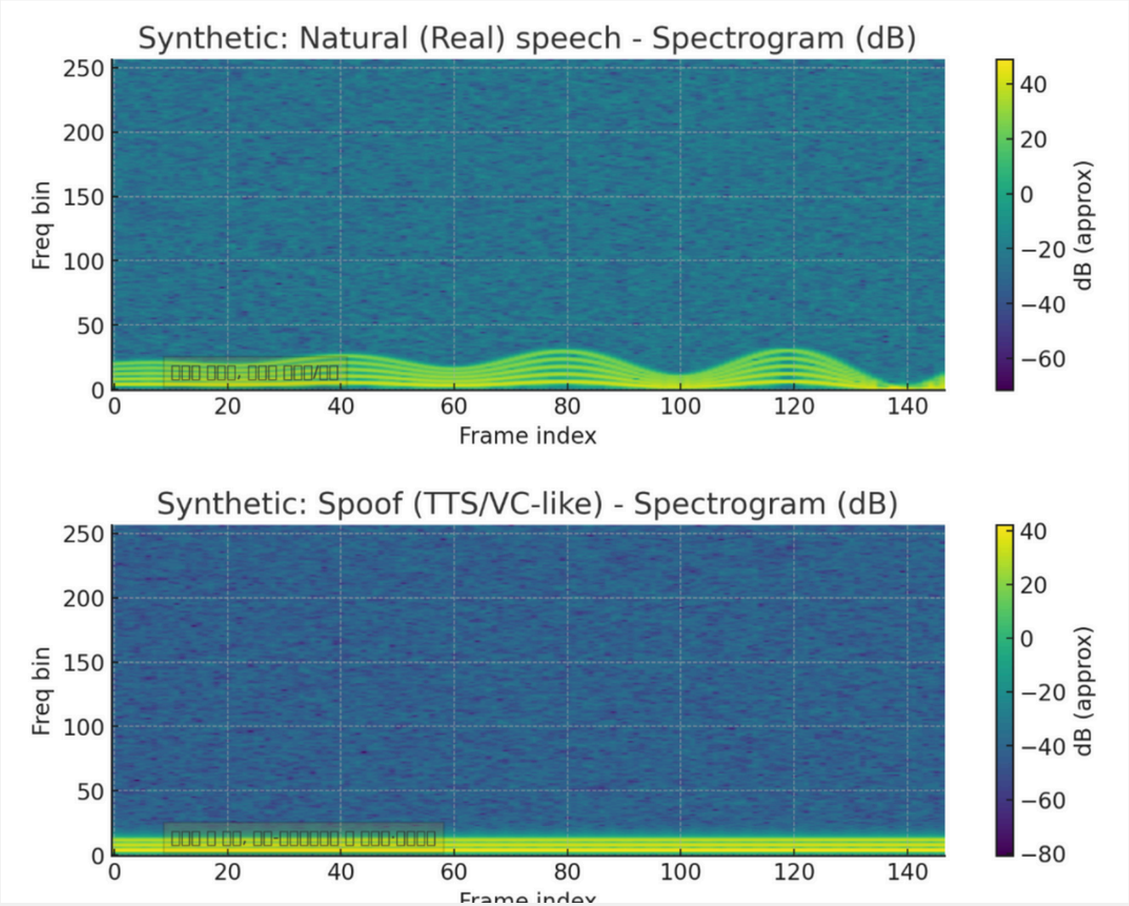


챗봇

사기 탐지 결과에 대한 상담 서비스 제공 및 피해 예방을 위한 가이드라인 안내, 신고 절차 지원

핵심 기능 및 코드

핵심 기능



원본음성과 변조음성의 차이

[주요 차이점]

구분	실제 음성	변조(TTS/VC 등) 음성
고조파 구조	f0와 고조파가 자연스럽게 흔들림 (미세 변동)	고조파가 규칙적·일정하거나 약화됨
스펙트럼 잡음/엔트로피	호흡·마찰음 등으로 고주파 비조화 성분 풍부	고주파 성분 부족, 스펙트럼 평활화
미세 발성 변화 (micro-prosody)	RMS·주파수·위상이 짧은 시간에도 불규칙하게 변동	변동이 부족하거나 지나치게 규칙적
자음-모음 전환	자연스럽고 불규칙	매끄럽거나 반대로 인위적으로 날카로움
주기성·위상	녹음/재생 시 위상 왜곡·리버브 등 흔적	모델 위상 재구성으로 특정 대역에서 비정상적 패턴
보컬 노이즈	호흡, 입술 소리 등 다양	부족하거나 부자연스러움
코덱/전달 아티팩트	녹음·재생 시 발생 (대역감쇠, 잡음 패턴)	vocoder·코덱 특유의 스무딩, 평활화

02

핵심 코드

Whisper 모델 코드

```python
class WhisperSpoofClassifier(torch.nn.Module):
    def __init__(self, whisper_model, hidden_dim=256, dropout=0.2):
        super().__init__()
        self.encoder = whisper_model.encoder
        self.dim = whisper_model.config.d_model

        self.classifier = torch.nn.Sequential(
            torch.nn.Linear(self.dim, hidden_dim),
            torch.nn.ReLU(),
            torch.nn.Dropout(dropout),
            torch.nn.Linear(hidden_dim, 1)
        )

    def forward(self, input_features):
        x = input_features["input_features"]
        enc = self.encoder(x).last_hidden_state
        pooled = enc.mean(dim=1)
        return self.classifier(pooled).squeeze(-1)
```

Whisper Encoder + 분류기 헤드

데이터셋: [KSS, ASVspoof 2019, LibriSpeech, VCTK, WaveFake]

[주요 동작 방식]

Whisper Encoder 활용

- 입력된 음성을 whisper_model.encoder를 통해 고차원 벡터 시퀀스(특징 벡터들)로 변환 후 추출

1. Pooling (평균화)
   - 시퀀스 길이가 가변적이므로 전체 타임스텝의 평균값을 사용
   - → 고정 길이 벡터로 변환
2. Classifier (MLP)
   - 2층 신경망(Linear → ReLU → Dropout → Linear)
   - 입력: Whisper에서 뽑은 대표 특징 벡터
   - 출력: 스칼라 값 (양수 = 진짜, 음수 = 스푸핑)
3. 최종 출력
   - 예측 결과 (logit 값)를 반환 → 시그모이드(σ) 함수 적용하면 **확률(0~1)**로 변환 가능

핵심 기능 및 코드

핵심 코드

-화자 임베딩 코드-

```python
def match_speaker(embedding_vec: np.ndarray, threshold: float = 0.4):
    global _next_speaker_id
    if embedding_vec is None:
        return None
    if embedding_vec.ndim != 1:
        embedding_vec = embedding_vec.reshape(-1)
    if not speaker_embeddings:
        spk = f"SPEAKER_{_next_speaker_id}"
        speaker_embeddings[spk] = [embedding_vec]
        _next_speaker_id += 1
        return spk
    best_spk = None
    best_sim = -1.0
    for spk, vecs in speaker_embeddings.items():
        avg = np.mean(vecs, axis=0)
        sim = cosine_similarity(embedding_vec.reshape(1, -1), avg.reshape(1, -1))[0,
0]
        if sim > best_sim:
            best_sim = sim
            best_spk = spk
    if best_sim >= threshold:
        speaker_embeddings[best_spk].append(embedding_vec)
        return best_spk
    else:
        spk = f"SPEAKER_{_next_speaker_id}"
        speaker_embeddings[spk] = [embedding_vec]
        _next_speaker_id += 1
        return spk
```

발언한 화자의 동일성 구분

[주요 동작 방식]

1. 입력확인
   - embedding_vec이 비어있거나 차원이 안 맞으면 안전하게 변환
2. 첫 번째 화자 등록
   - 아직 저장된 화자가 없다면 → SPEAKER_0으로 새로 등록
3. 기존 화자들과 비교
   - 각 화자마다 지금까지 저장된 임베딩들의 평균 벡터를 구함
   - 새로운 벡터와 평균 벡터 간 코사인 유사도(cosine similarity) 계산
   - 가장 유사도가 높은 화자(best_spk)를 선택
4. 임계값(threshold) 비교
   - 유사도가 기준값 이상이면 → 같은 화자로 간주하고 벡터를 추가 저장
   - 기준값보다 낮으면 → 새로운 화자로 등록 (SPEAKER_1, SPEAKER_2, …)

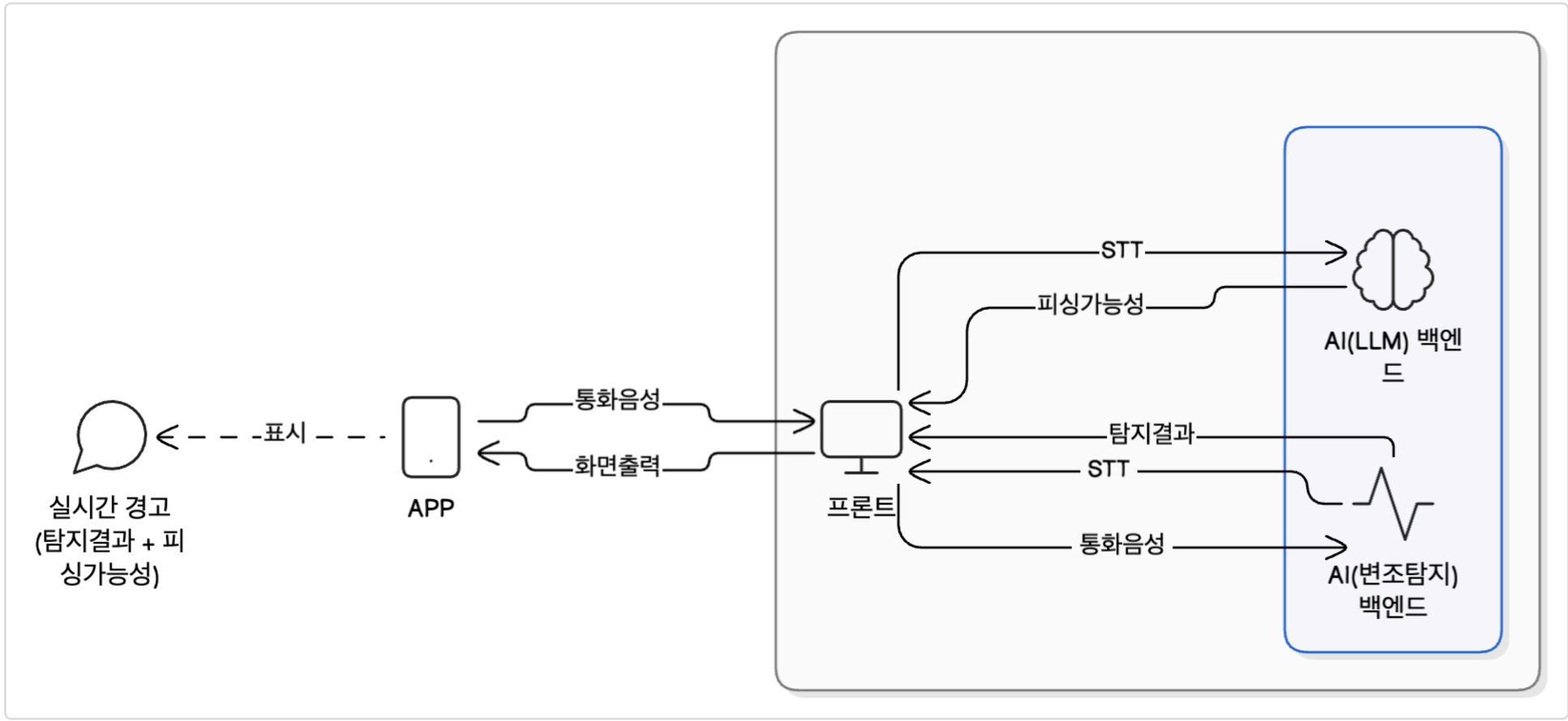
03

아키텍쳐

01

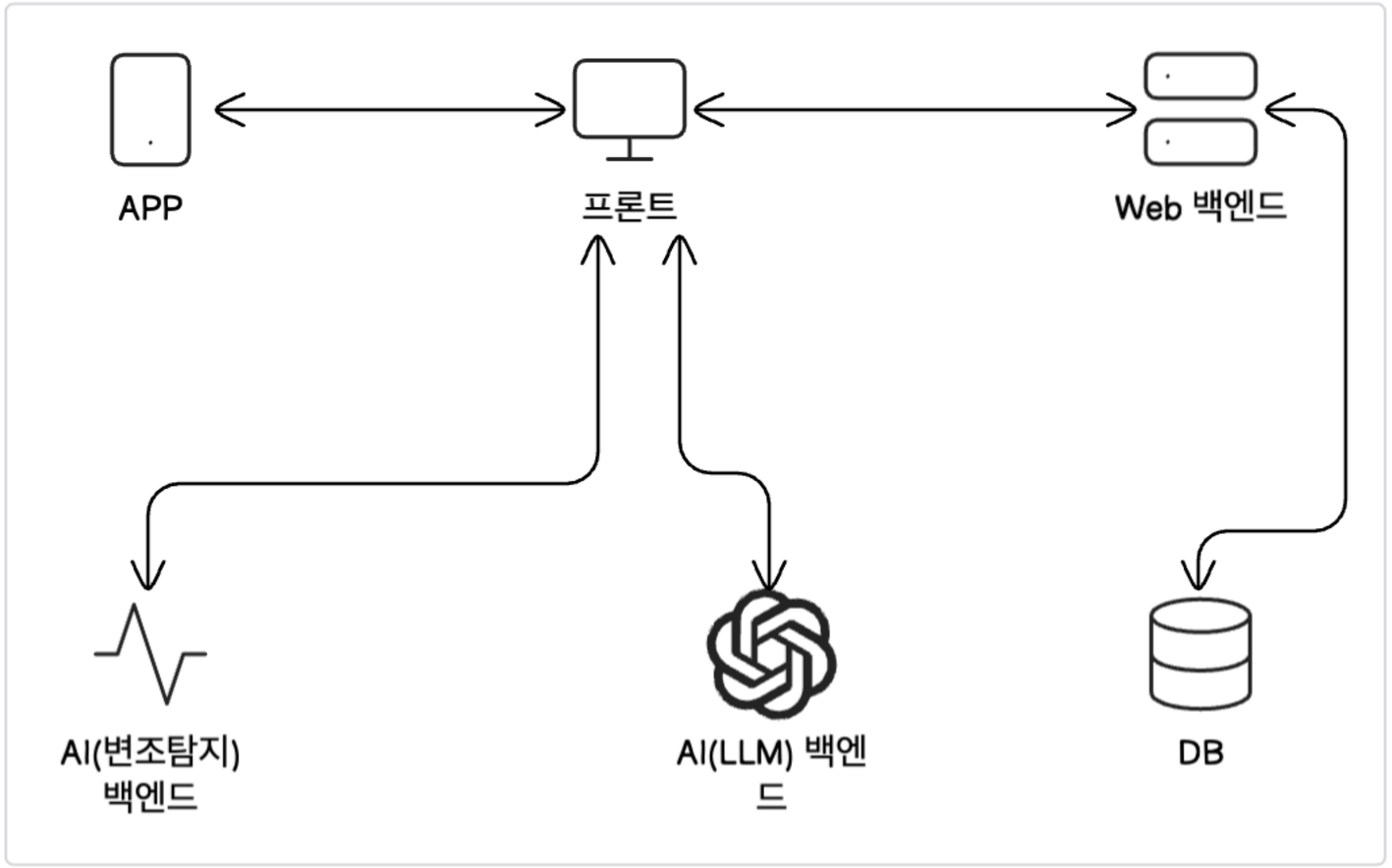
아키텍쳐

모델 처리 구조



02

사용자 데이터 전달 아키텍쳐



04

개발 툴

04

개발 툴

IDE 및 협업도구

IntelliJ

Visual Studio Code

GitHub

android studio

라이브러리

React

Transformers

PyTorch

Speech Brain

Lombok

JWT

Android X

프레임워크

NEXT.JS

Tailwind CSS

Spring Boot 3

FastAPI

Native

WebView

데이터베이스

MariaDB

05

시연

06

트러블 슈팅 및 개선점

05

트러블슈팅 (1)

STEP. 1

문제 발생

- 상대방 음성까지 포함된 실시간 통화 음성을 가져올 수 없음

STEP. 2

문제 원인 분석

- 안드로이드 9버전 이후로 실제 실시간 통화 음성에 접근하기 위해서는 통신사와 시스템(OS)권한이 필요함

STEP. 3

개선 방안

- 웹 소켓을 이용한 가상의 실시간 통화를 구현
- 발전 가능성으로 추후 실제 상용화 시 권한 허가를 받으면 실제 통화에도 적용 가능함

트러블슈팅 및 개선점

트러블슈팅 (2)

STEP. 1

문제 발생

- whisper기반 변조 탐지 모델을 실시간 통화에 연결시 통화 상태에 따라 음성이 깨질시 일부 변조 음성으로 판단함.

STEP. 2

문제 원인 분석

- 통화시 통화 품질의 문제로 음성이 깨짐현상이 발생할 수있음.
- 음성깨짐 현상 발생시 음성의 파형이 완벽하지 않아 주파수가 일그러져 일시적으로 변조로 판단.

STEP. 3

개선 방안

- 변조음성은 음성 깨짐 현상과 상관없이 주파수 특징 차이로 지속적으로 변조 음성으로 판단됨
- 큐를 이용하여 일정 구간에서 기준 횟수 이상 변조가 찍힐 경우에 실제 변조 음성으로 판단.

07

Q & A

TEAM PROJECT

THANK YOU
감사합니다.

AURA

AI 보이스피싱 탐지 어플