캡스톤 프로젝트 최종 보고서

중앙대학교

자동 음성 녹음을 통한
사용자 음성 합성 시스템

팀 명: 초롱초롱 보들이
20176758 박진영
20172634 이민희
20162874 이준협

목차

1. 프로젝트 개요 및 동기

음성 합성이란 한 사람의 말소리를 일정한 음성 단위로 분할한 다음, 부호를 붙여 합성기에 입력하고 지시에 따라 필요한 음성 단위만을 다시 합쳐 말소리를 인위적으로 만들어내는 기술이다. 텍스트를 음성으로 변환한다는 의미로 텍스트 음성 변환(Text-to-Speech, TTS) 시스템이라고 부르기도 한다. 음성 합성 기술은 문자를 읽기 어려운 장애인을 위한 스크린 리더나 아이, 외국인 등의 언어 학습을 위해 쓰이기도 하며, 최근 대중적으로 쓰이고 있는 AI 비서, 음성 예약 서비스, 오디오 콘텐츠, 동영상 더빙 등 매우 다양한 분야에서 활용되고 있다.

최근 대두되고 있는 기술이지만 일반인이 편리하게 접하기에는 아직 무리가 있어 보인다. 일부 기업에서 사전에 합성된 AI 음성을 제공하거나, 일반인들이 사용하기에 부담스러운 가격으로 제공하고 있다. 또한 음성 합성에 이용될 한 사람의 목소리를 모으기 위해선 특정 문장들을 최소 100 문장에서 많게는 열 시간 이상의 녹음을 요구하기도 한다.

다양한 분야에 유용하게 쓰일 수 있는 AI 음성을 별도의 노력없이 누구나 쉽게 만들 수 있다면, AI 비서를 여자친구, 남자친구의 목소리로, 아이는 오디오 북을 부모님의 목소리로 들을 수 있지 않을까. 또, 자신의 목소리를 좋은 곳에 쓰일 수 있게 기부할 수도 AI 음성이 필요한 곳에 판매해 수익을 창출할 수도 있지 않을까.

따라서 우리의 프로젝트는 특정 성우가 시간을 투자해 녹음한 것으로 음성 합성을 하는 기존의 플랫폼에서 벗어나, 평범한 사람도 매일 사용하는 핸드폰을 이용해 별도의 녹음 없이 통화 중의 목소리로 음성 합성을 할 수 있는 앱을 만들고자 한다.

2. 프로젝트 목표

본 프로젝트의 목표는 다음과 같은 기능을 제공하는 핸드폰 어플리케이션을 만드는 것이다.

1. 앱에서 통화 중 녹음을 활성화하면 사용자가 통화를 할 때마다 사용자의 목소리를 녹음한다.
2. 녹음한 음성데이터를 사용해 음성 합성 모델을 학습시킨다.
3. 학습이 완료되면 음성 합성 기능이 제공되어 사용자는 원하는 텍스트를 자신의 AI 음성으로 들을 수 있다.
4. 완성된 AI 음성은 소셜을 통해 공유할 수 있으며 모델 자체 공유 또한 가능하다.

이 어플리케이션은 사용자 측면에선 개인화 음성 합성을 촉구하고, 회사 측면에서는 다양한 음성 합성에 의한 기술 개발을 촉진할 것이다.

위와 같은 앱을 사용자에게 제공하여 사용자는 자신의 목소리로 만들어진 AI 음성을 듣고, 사용할 수 있는 새로운 경험을 할 수 있다. 또한 기업에선 추가적인 프로젝트 없이, 우리의 앱을 통해 생성된 AI 음성 중 원하는 것을 구매하거나 제공받을 수 있다. 더 나아가 앱으로 생성한 AI 음성을 AI 비서의 음성으로 설정하거나 카카오톡 등의 메신저 앱과 연동시켜 메신저 앱 상에서 TTS 서비스를 이용할 수 있게 하는 발전 가능성도 고려할 수 있을 것으로 보인다.

3. 개발 배경과 구현 내용

3.1. 개발 배경

- **– 별다른 수고 없이, 다양한 사람을 대상으로, 음성 합성을 해주는 기술**

상용화된 음성 합성 기술은 큰 노동력을 요구하는데 이는 일반인의 사용성과 접근성을 저하시킨다. 위와 달리 별다른 수고 없이 일상생활 속에서 자동으로 사용자의 음성 데이터를 수집한다면 쉽게 음성 합성을 진행할 수 있지 않을까라는 생각을 하게 되었다.

사용자 음성 데이터를 수집하기엔 대부분의 사람들이 많은 시간 사용하고 소지하는 스마트폰이 적합하다 판단하였다. 스마트폰을 사용하는 여러 상황들 중, 전화하는 상황이 비교적 조용하고, 사용자와의 거리가 가까워 좋은 음질의 음성 데이터를 얻을 수 있을 것이라 판단하여, 전화하는 상황을 인지하여 자동으로 녹음을 해주는 안드로이드 어플로 개발을 진행하게 되었다.

또한 이러한 사용자의 별다른 수고 없이 자동으로 음성을 수집하는 앱은 음성 합성 제작 말고도 다양한 이유의 데이터 수집에 사용 가능 할 것으로 보인다.

3.2. 개발 환경

- 본 Server: Django, Django-rest-framework, Celery, Sox 등 사용. 자세한 개발 환경은 Github 의 requirement.txt 파일에 명시한다.
- Notification 알림을 위한 Server: Firebase
- Database: MariaDB
- Application: Android Studio

3.3. 구현 사항

- **로그인**

  사용자의 정보, 음성 데이터를 원활하게 획득하고, 서버단에서 관리하기 위해 로그인 기능을 사용한다.

- **통화 오디오 녹음**

  사용자가 스마트폰으로 전화하는 상황을 인식해, 사용자의 목소리를 녹음하여 데이터셋을 축적하는 기능이다. 사용자가 이 기능을 활성화했을 때만 동작하여 원치 않는 녹음을 피할 수 있다.

  통화 내용 전체를 녹음하는 것이 아닌 사용자의 목소리만을 녹음하는 이유는, 사람 인식의 어려움을 피하고, 좋은 음질의 음성 파일을 획득하기 위해서다.

  하지만 조용한 상황이 아닌 여러 상황에서 전화를 할 수 있으므로 깨끗하지 않은 음성이 녹음된다면 잡음 제거, 파일 분할 등 전처리를 진행하고, 그래도 좋지 않은 데이터라면 필터링 과정을 거친다. 또한, 녹음 데이터에 해당하는 텍스트 데이터를 생성하기 위해 Speech-to-Text(STT)를 사용한다.

- **추가 녹음**

  평소 잘 말하지 않는 음절에 의해 모델 생성이 늦춰지거나, 정확도가 낮다면 추가 녹음 창에서 사용자가 원하는 음절을 직접 선택해 녹음할 수 있다. 또한 모델 생성 후 사용자의 목소리와 비슷하지 않다 거나 어색하게 들리는 등 마음에 들지 않는 음절 부분이 있다면, 음절을 선택하고 이에 대한 추가 녹음을 진행할 수 있다.

- **TTS 모델 학습**

  수집한 사용자의 데이터로 TTS 모델로 학습시켜 사용자 개인의 목소리로 TTS 테스트를 할 수 있도록 한다.

- **음성 합성 진행 현황**

  현재 어느 정도 모델 생성이 진행되었는지, 모델을 만들기에 부족한 음절이 있는지를 보여준다. 이 기능으로 사용자는 쉽게 진행 현황을 보고, 이해할 수 있을 것이다.

- **결과 확인 / 테스트**

  일정량의 데이터가 수집이 되어 모델 생성 후, 원하는 문장을 사용자의 합성 음성으로 들을 수 있는 TTS 기능을 제공한다.

- **음성 파일 다운**

  원하는 문장을 입력하고 그에 대한 TTS 음성 파일을 다운받을 수 있다.

- **모델 공유**

  주변 사람에게 사용자의 음성 합성을 자랑하거나, 사용 가능하도록 공유할 수 있다. 추후 다른 어플과 연동을 통해 다양한 방면으로 접목이 가능하다.

4. 상세 개발 내용

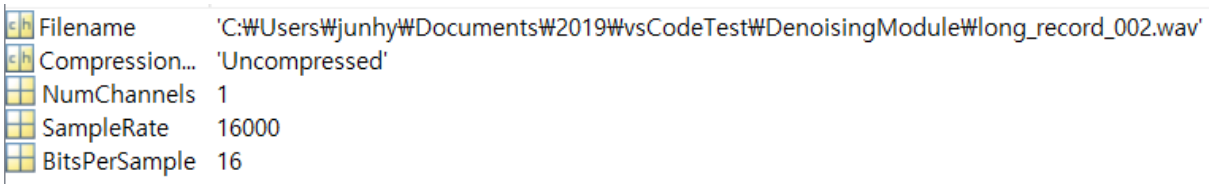
4.1. 전처리 및 STT

– 데이터 수집

소음제거 전처리 모델을 학습시키기 위해서는 상당히 많은 양의 데이터가 필요했다. 팀원들이 직접 녹음해서 수집하기에 무리가 있어 온라인에 공개된 데이터셋들을 수집했다. 한국어 음성녹음 데이터는 공공 인공지능 오픈 API·DATA 서비스 포털(http://aiopen.etri.re.kr/)에서 API key 를 받아 수집했다. 음성 데이터는 텍스트쌍과 매칭이 되어있었으며 50 명의 발화자가 100 문장씩 녹음한 데이터이다. 추가 데이터는 AI HUB(http://www.aihub.or.kr/)에서 수집하였다. 전처리 모델을 학습시키기 위해서는 원본의 조용한 데이터와 해당 데이터에 소음이 합성된 형태의 합성 데이터가 필요했는데 이 때 합성할 소음 데이터 또한 종류의 다양성을 추구하기위해 직접 녹음하지 않고 공공장소 소음데이터로 어느정도 인지도가 있는 UrbanSound8K 데이터를 수집하였다(https://urbansounddataset.weebly.com/). 또한, 다양한 상황의 소음 데이터 중에 에어컨, 자동차 경적, 아이들 소리, 길거리 음악 등 학습에 의미 있을 것 같은 데이터만 선택해 사용하였다.

– 데이터 전처리

기본적으로 여러 사이트에서 데이터를 수집했고 모바일 디바이스를 통해서도 데이터를 수집했기 때문에 파일의 형식들이 전부 뒤죽박죽이었다. 음성파일들을 통일시키기 위해서는 Sampling rate, Bit per Sample, Channel, Data type, Extension 들을 맞춰 주어야한다. 온라인 사이트로는 이러한 것들을 전부 상세하게 맞추어 주기 쉽지 않아서 SoX(Sound Exchange)라는 프로그램을 통해 음성 포맷을 **[그림 1]**과 같이 통일해주었다. Float 과 Integer 을 오가는 과정에서 약간의 데이터 손실이 일어났지만 크게 영향을 미치지는 않았다.

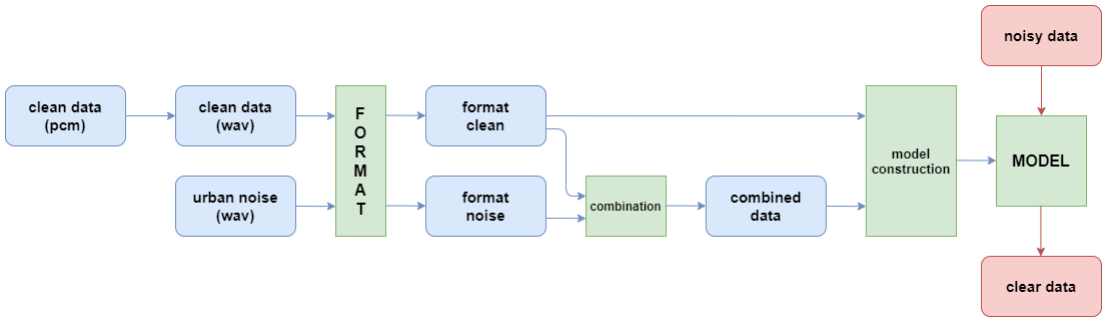


[그림 1]

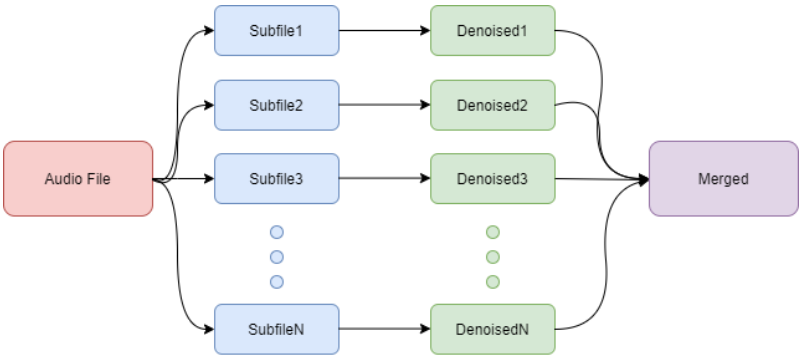
사용한 소음 제거 학습 모델이 32 bit per sample 형식을 고집해서 다른 형식들은 전처리 과정동안 변하지 않았지만 bit per sample 은 16->32->16 순으로 변화하였다. 학습을 진행시키기 위해서는 기존에 수집한 한국 음성 데이터에 어느정도 소음이 낀 데이터가 필요했는데 이를 위해 수집한 데이터들의 추가가공을 진행했다. UrbanSound8k 에서 뽑아낸 소음 데이터와 한국 음성 데이터 합성 시, 한국음성의 max value 가 1 이 되도록 정규화를 진행해주고 소음의 max value 가 0.5 가 되도록 정규화를 해서 합성했다. 이 때 소음의 max value 값을 변경시키면 다양한 신호 대 잡음비 SNR(Sound to Noise Ratio)가 완성되는데 학습시간이 너무 오래 걸려 max value 는 0.5 로만 진행되었다. 또한, 수집한 한국어 음성 데이터는 파일별로 그 시간이 서로 달랐는데 이를 4 초로 통일해주었고 합성되는 소음 데이터도 4 초이상이면 4 초까지 잘라서 합성해주었고 4 초 미만이었을 경우 다시 반복시켜 언제나 4 초가 되도록 통일시켜주었다. 따라서 소음제거 모델 합성 시 사용된 학습 데이터는 전부 4 초다.

– 음성 전처리

추가 녹음 혹은 전화 녹음 데이터 둘 다 스마트폰을 통해 수집되기 때문에 전문적인 녹음 장비에 비해 음질도 뛰어나지 않고 다양한 환경의 주변 소음이 끼게 된다. 이러한 문제점들은 추후에 적용되는 Speech-to-Text 모델과 Text-to-Speech 모델 모두에게 큰 성능 저하를 불러온다. 따라서 소음을 제거해주고 음질을 높이는 방향의 음성 전처리 과정을 진행했다. 원본데이터로부터 소음과 깨끗한 소리를 분리해주는 학습 모델은 데이터 전처리 단계에서 가공한 데이터들로 학습을 진행한다. 모델 학습을 새로 데이터가 들어올 때 하는 방식이 아닌 학습 데이터들로 미리 Pretrained Model 을 만들어서 새로운 데이터를 모델에 적용하는 방식으로 진행된다. 학습데이터가 전부 4 초로 통일되었기 때문에 테스트 데이터도 소음제거 모델에 넣어 주기 전에 4 초 단위로 전부 나누어 주고 해당 파일들을 전처리 하고나서 합쳐주는 방식으로 진행된다. (**[그림 2,3]**참조)



[그림 2]



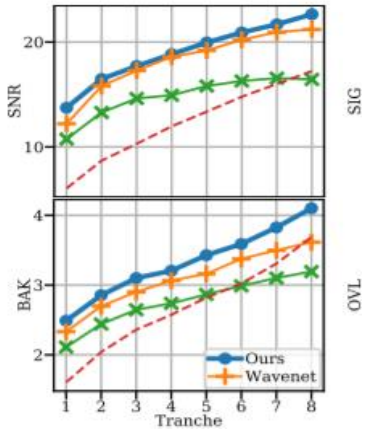
[그림 3]

– 전처리 모델

Wavenet 을 사용한 Speech Denoising Model(https://github.com/drethage/speech-denoising-wavenet)을 초기 모델로 사용했는데 개인이 만든 프로젝트여서 최적화 및 성능에서 아쉬운 성능을 보여주었다. 또한 모델학습을 위해 만든 인공 합성 데이터로 학습 시 SNR 이 0 이 되면서 완전히 뭉개지는 현상이 종종 발생했다. 원인을 찾지 못해 새로운 모델로의 변경을 결정했다.

새로운 모델은 소음제거시 많이 사용되는 Wavenet 이 아닌 독자적으로 개발한 Deep Feature Loss 를 사용해서 음성 전처리를 진행한다. [그림#]을 보면 새로운 모델이 기존의 모델(Wavenet)보다 소음 제거 기술이 더 뛰어난 것을 확인할 수 있다. 예시 소음 제거 파일들을 들어보면 약 2.5dB 의 합성음까지는 문제없이 제거해냈다. 또한 End-to-End 방식으로 구현되어 있어 모델에서 요구하는 학습파일의 형식과 전처리하고 싶은 음성 파일들의 포맷(16000Fs, 32BpS, Mono, float)들만 잘 맞춰주면 어렵지 않게 모델을 사용할 수 있다. 해당 모델로 바꾼 뒤로 더욱 더 말끔한 소리를 분리해냈으며, 전 모델에서 발생하던 소리가 뭉개지는 현상도 더 이상

발생하지 않았다.

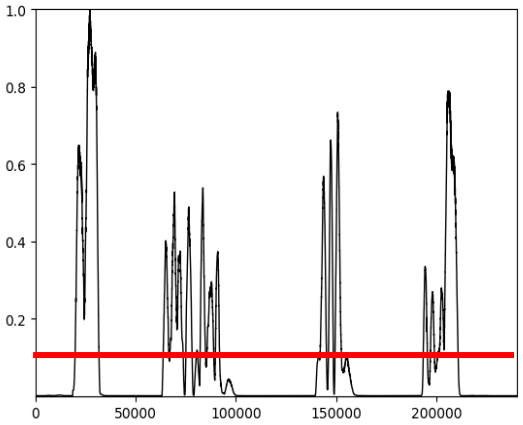


[그림 4]

두 모델 다 고질적으로 가지고 있는 문제점이 있었는데 파일 당 전처리 시간이 너무 오래 걸린다는 점이다. Pretrained Model 을 사용하는데 이때 GPU 를 할당해 전처리를 진행해도 파일당 약 15 초~30 초 정도 소요된다.

– VAD(Voice Activity Detection)

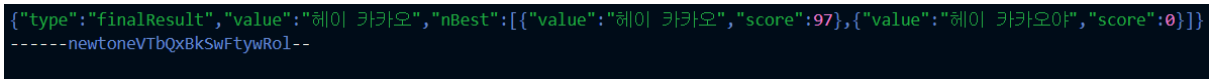
음성신호의 공백 부분을 찾아내 제거하고 필요한 음성 신호만 들어있도록 신호를 분리하는 기술이다. 정밀하고 수준 높은 VAD 를 구현하기 위해서는 머신 러닝이나 다른 기술들이 필요하지만 우리 프로젝트가 요구하는 공백 분리의 수준은 그렇게 높지 않다. 추가 녹음은 문장단위로 녹음하기 때문에 공백단위로 분리할 필요가 없고 공백 분리가 필요한 전화 녹음마저 통화 특성상 주고받기 때문에 문장과 문장 사이에 상당히 긴 공백이 존재하기 때문이다. 다음과 같은 이유로 흔히 접할 수 있는 간단한 수준의 공백 분리 기술을 적용하였다. 0.1 초 단위로 슬라이딩 윈도우 기법을 사용해서 시간영역의 에너지를 구해준다. 그 결과 energy domain 에서 음성 신호를 바라볼 수 있게 되는데 max normalization 를 적용해주고 0.1 보다 커질 때 음성 신호가 시작된다고 판단했다. (**[그림 5]**참조)



[그림 5]

- Speech-to-Text

STT 모델은 카카오톡에서 무료로 제공하는 Kakao Speech API 를 사용했다. API 사용 요청시 제공되는 rest_api_key 를 가지고 지정된 url(https://kakaoi-newtone-openapi.kakao.com/v1/recognize)에 POST 로 request 를 보내면 지정된 음성파일에 대한 Speech-to-Text 결과가 함수 반환형으로 돌아옵니다. 이 때 돌아오는 정보의 형식이 json 인데 여기서 final result 값만 읽어서 텍스트 파일 형태로 저장된다. (**[그림 5]**참조)



[그림 5]

4.2. TTS 모델

– TTS 모델 선정

딥러닝을 통한 음성 합성엔 Tacotron, Tacotron2, Deep Voice1-3 등 다양한 모델이 있으며, 모델들의 장단점을 분석하고 선택해 직접 구현하기까지 오랜 시간이 걸릴 것이라 판단하여 모델을 구현해 놓은 오픈소스를 사용하기로 결정하였다. 한국어 사용이 가능한 오픈소스 모델은 두 가지로, 기존 Tacotron 모델에 Deepvoice2 의 Multi-Speaker 를 결합한 [Multi-Speaker Tacotron in TensorFlow, https://github.com/carpedm20/multi-Speaker-tacotron-tensorflow]와 Tacotron2 모델과 Wavenet Vocoder 를 결합한 [Multi-Speaker Tocotron2 + Wavenet Vocoder + Korean TTS, https://github.com/hccho2/Tacotron2-Wavenet-Korean-TTS]이다. 비교적 최신 버전인 두 번째 오픈소스를 이용하였으며, Wavenet Vocoder 를 학습시켜 사용하기엔 시간 상 어려움이 있어 Griffin-Lim Synthesizer 를 사용하였다.

– TTS 모델 성능 최적화를 위한 실험

1) 실험의 목적

다양한 테스트셋을 가지고 여러 방법으로 모델 학습을 실험하였다. 실험의 목적은 본 프로젝트에서 통화, 추가녹음으로 수집한 데이터를 최대한 효율적으로 학습시킬 수 있게 아래와 같은 사항들을 결정하기 위해서이다.

- *TTS 모델 학습을 어느 정도의 데이터가 쌓였을 때 시작할 것인가?*
- *Single-Speaker와 Multi-Speaker 모델 중에 어떤 것을 사용할 것인가?*
- *Pre-trained Model을 사용할 것인가?*
- *Learning_rate의 최적값은 무엇인가?*
- *Max iteration을 얼마로 설정할 것인가?*

2) 실험 트레이닝 데이터셋

① 문재인 대통령, 손석희 아나운서: 각 10분, 5분 데이터 (오픈 소스에 포함된 데이터)

② Korean Single speaker Speech Dataset: 여성 한 명, 약 7시간

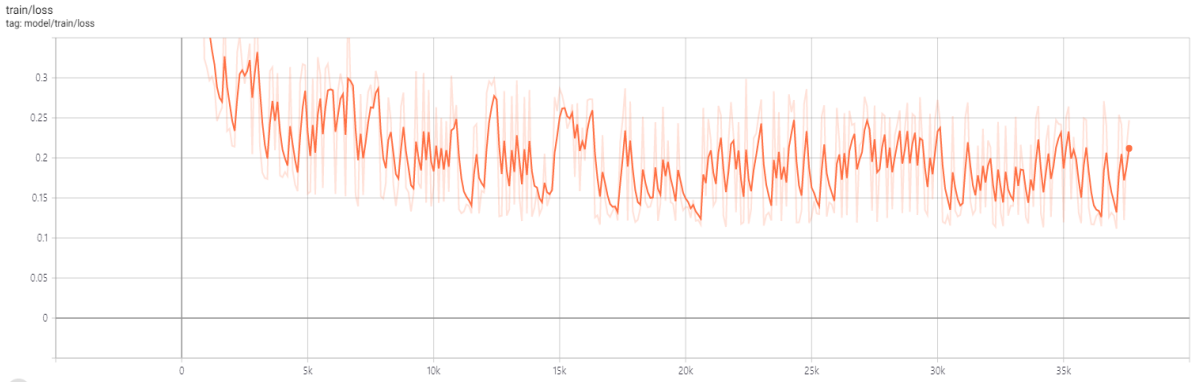
https://www.kaggle.com/bryanpark/korean-single-speaker-speech-dataset#transcript.v.1.4.txt

③ 잡음처리 및 음성검출을 위한 스마트폰 환경 연속어 음성 데이터: 50명, 각 100문장 발화

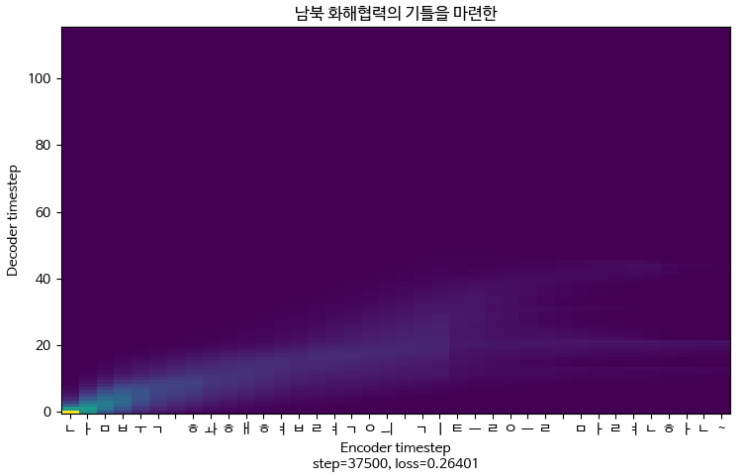
http://aiopen.etri.re.kr/aidata_download.php

3) 실험 결과

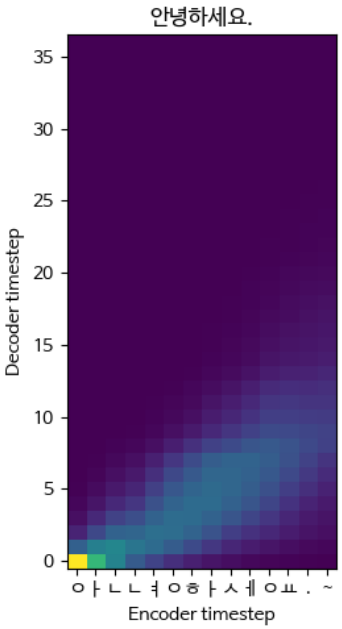
실험 1. ①번 데이터셋을 이용하여 Multi-Speaker Model 학습을 37.5K step 진행하였다. 학습 결과 Loss 가 잘 감소하지 않고, 트레이닝셋에 포함된 문장과 포함되지 않았지만 매우 간단한 문장 두 경우 모두 음성을 잘 합성하지 못하였다.



실험 1-Train loss

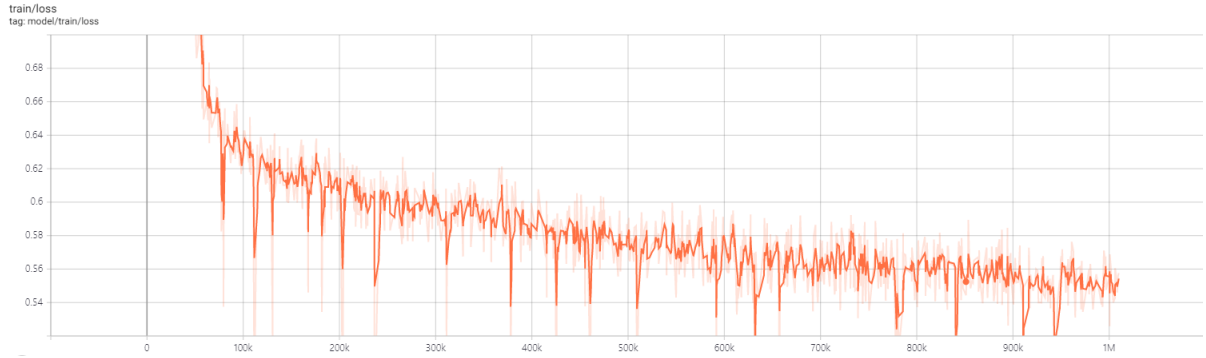


실험 1-트레이닝셋 포함 문장 테스트

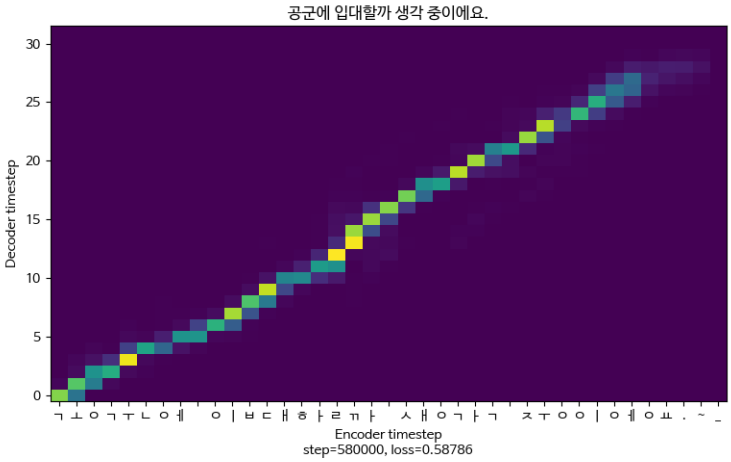


실험 1-새로운 문장 테스트

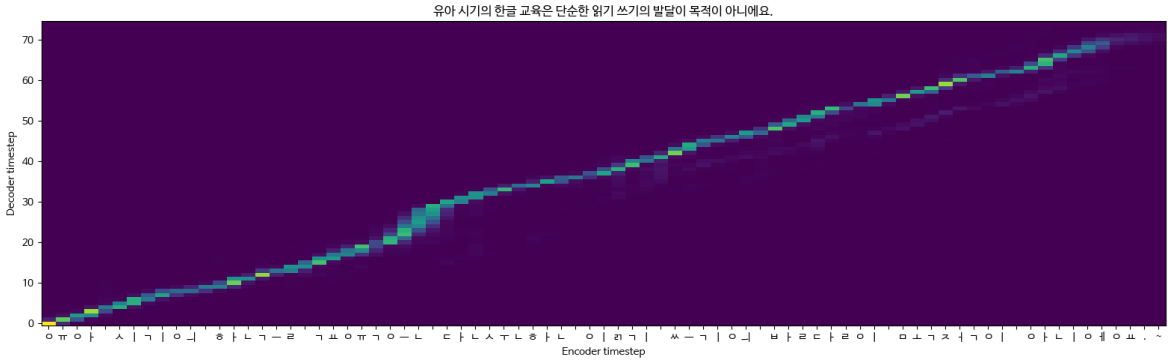
실험 2. ②번 데이터셋을 Single-Speaker Model 로 학습시켰으며, 100K step 진행하였다. 오랜 시간 학습이 진행되었음에도 꾸준히 Loss 가 감소하였으며, 트레이닝셋에 포함된 문장과 포함되지 않은 길고 복잡한 문장 두 경우 모두 매끄러운 음성을 합성해냈다.



실험 2-Train loss

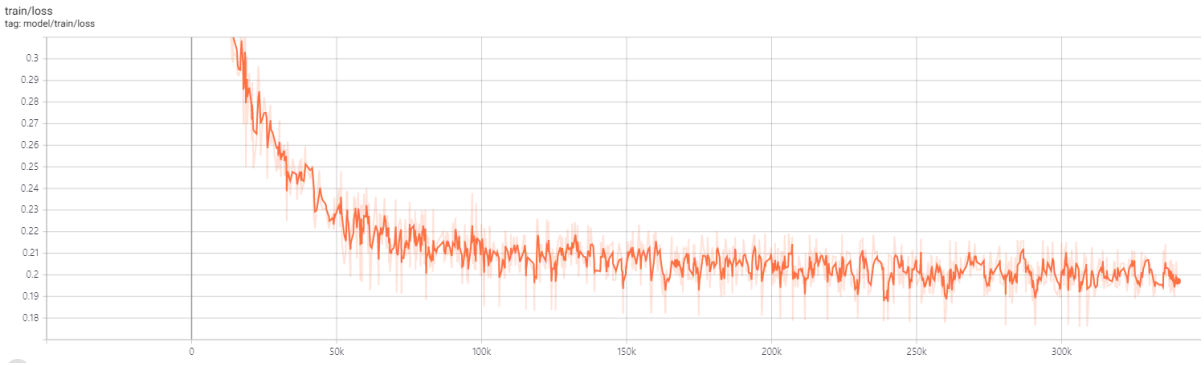


실험 2-트레이닝셋 포함 문장 테스트

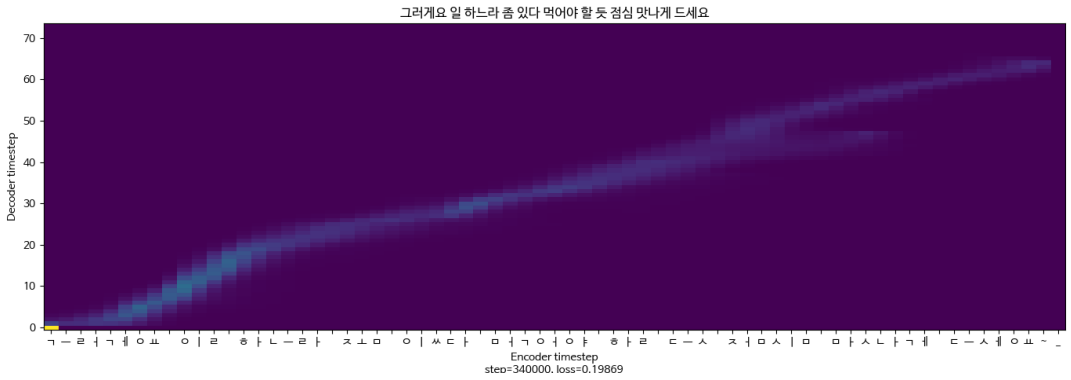


실험 2-새로운 문장 테스트

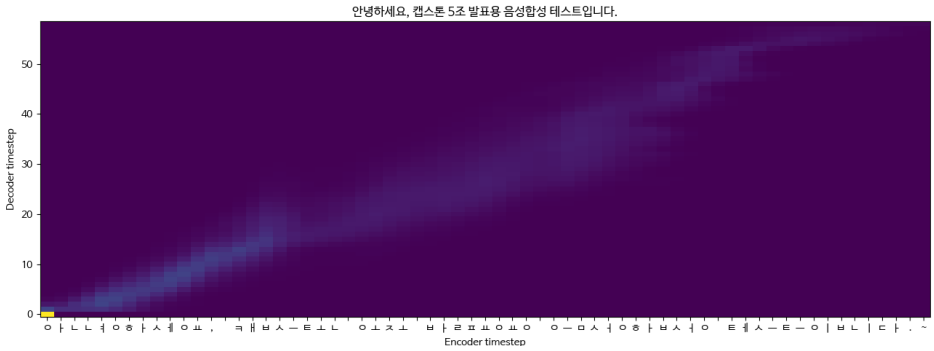
실험 3. ③번 데이터셋의 1-4 번 발화자의 데이터로 Multi-Speaker Model 학습을 340K step 진행했다. 학습 결과, 데이터양의 부족으로 트레이닝셋에 포함된 문장과 새로운 문장 두 경우 모두 음성 합성이 전혀 되지 않았다.



실험 3-Train loss



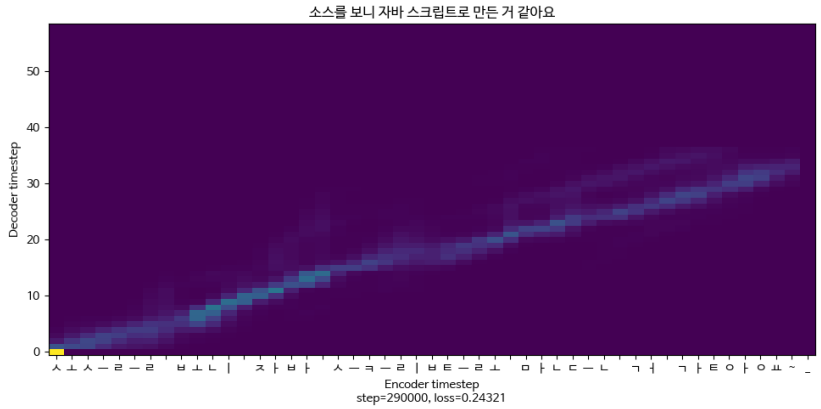
실험 3-트레이닝셋 포함 문장 테스트



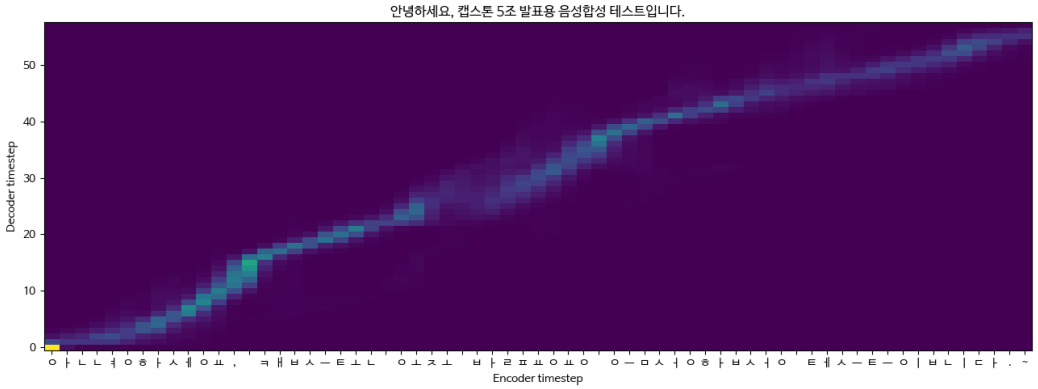
실험 3-새로운 문장 테스트

실험 4. ③번 데이터셋의 1-10 번 발화자 데이터를 Multi-Speaker Model 로 290K step 학습시켰다. 학습 결과, 실험 3 에 비해 어느 정도 음성 합성이 되긴 했지만, 발음이 부정확하고 기계음 같은 음성이 생성되었다.

실험 4-Train loss

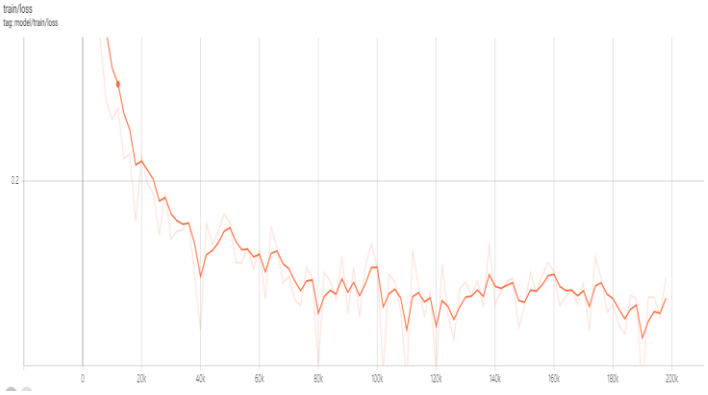


실험 4-트레이닝셋 포함 문장 테스트

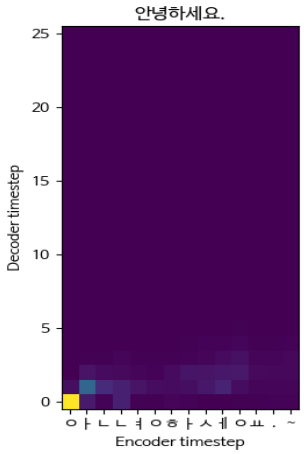


실험 4-새로운 문장 테스트

실험 5. 실험 2 를 Pretrained-model 로 ③번 데이터셋의 1 번 발화자 데이터를 학습시켰다. Single-Speaker Model 이며, 200K step 학습을 진행한 결과 데이터양의 부족으로 간단한 문장도 제대로 음성을 합성하지 못했다.



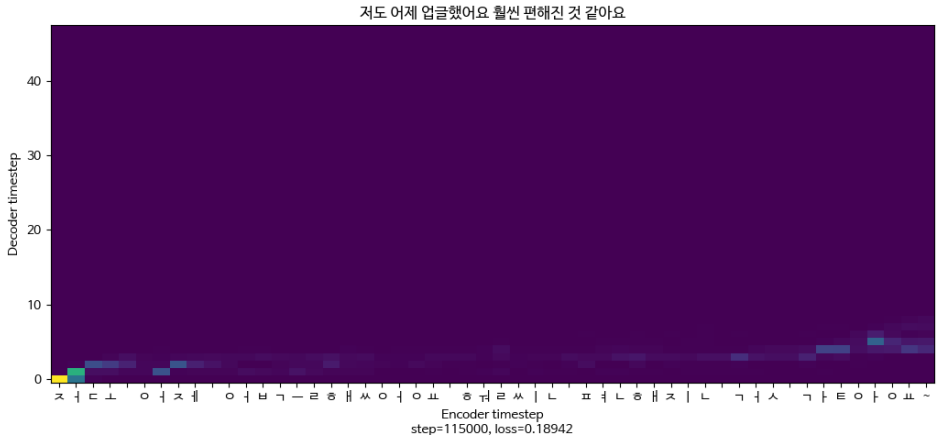
실험 5-Train loss



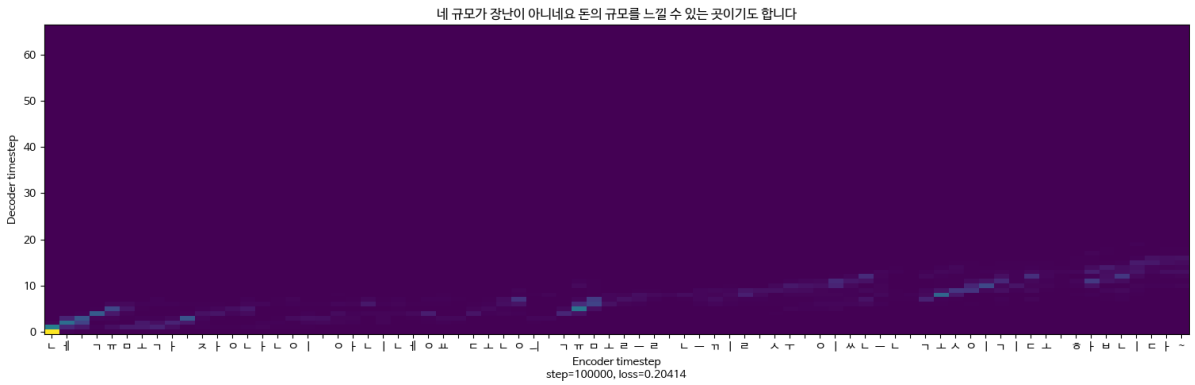
실험 5-새로운 문장 테스트

실험 6. 실험 5 에서 Learning rate 와 Learning rate decay 값을 다양하게 바꿔가며 실험하였다.

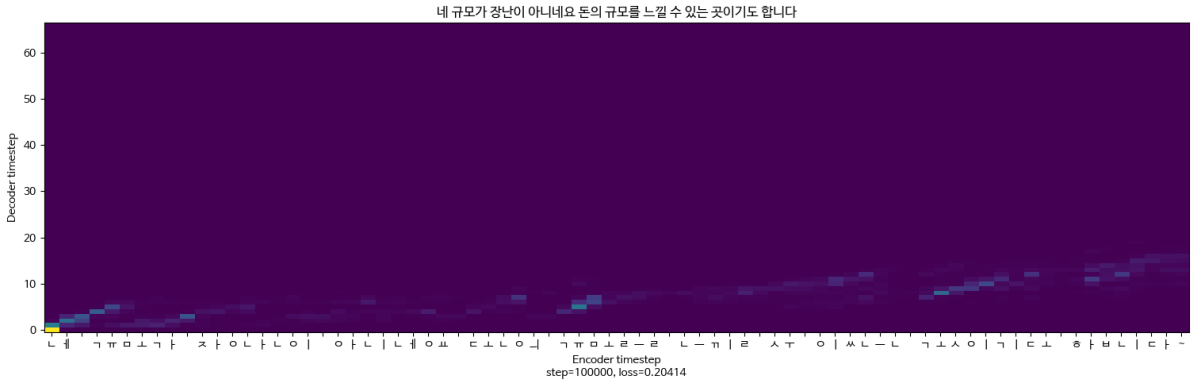
3 번에 걸친 실험 모두 트레이닝셋에 포함된 문장조차 제대로 음성을 합성하지 못했다. 따라서 이후 실험들은 Pre-trained Model 을 사용한 Single Speaker 학습이 아닌, Multi-Speaker Model 로 실험을 진행하였다.



실험 6-Learning rate=1e-3, decay=False

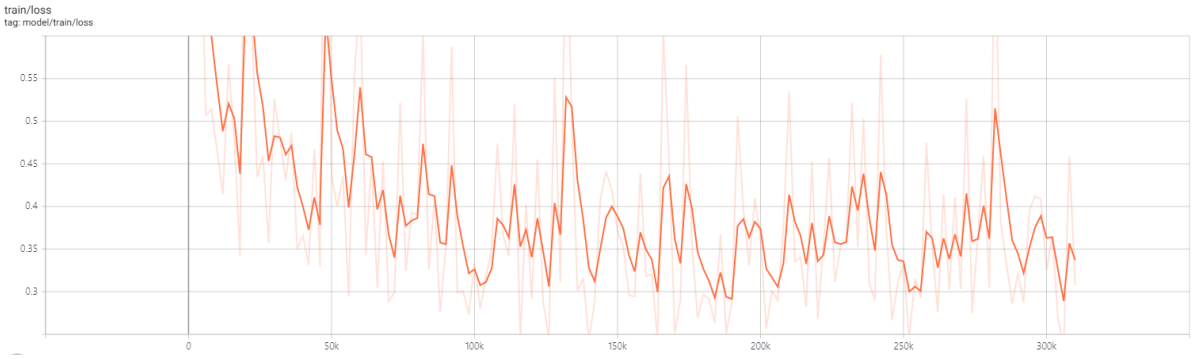


실험 6-Learning rate=1e-3, decay=True, last learning rate=1e-4

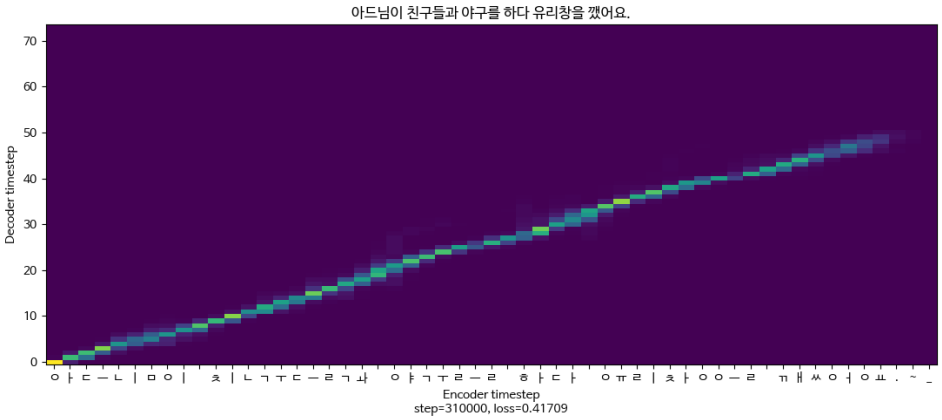


실험 6-Learning rate=1e-4, decay=True, last learning rate=1e-5

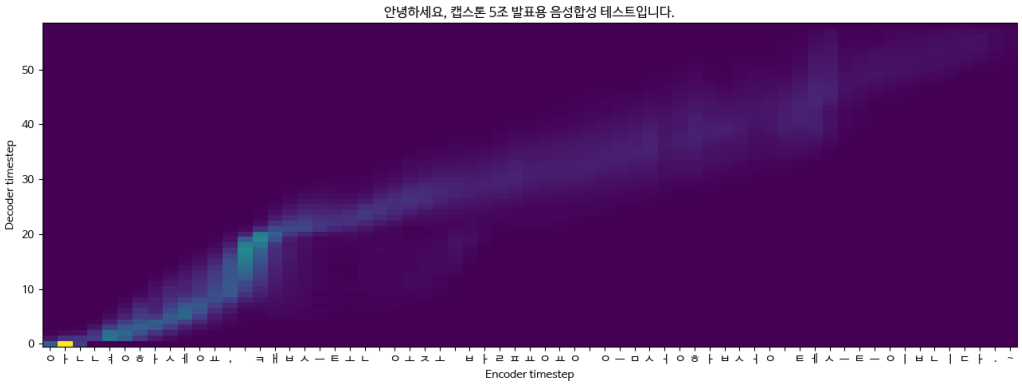
실험 7. ③번 데이터의 1-5 번 발화자의 데이터와 ②번 데이터를 함께 Multi-Speaker Model 로 310K step 학습하였다. 학습 결과 Loss 가 심하게 진동하며 제대로 감소하지 않고, 트레이닝셋에 포함된 문장은 잘 생성했지만, 포함되지 않은 문장은 제대로 생성하지 못했다. 이는 데이터양이 한 발화자에게 편향되어 있기 때문으로 추정한다.



실험 7-Train loss

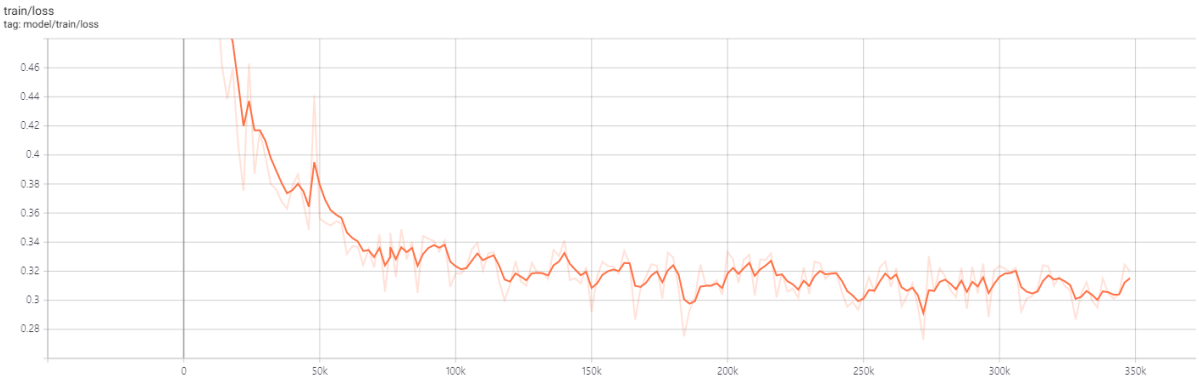


실험 7-트레이닝셋 포함 문장 테스트

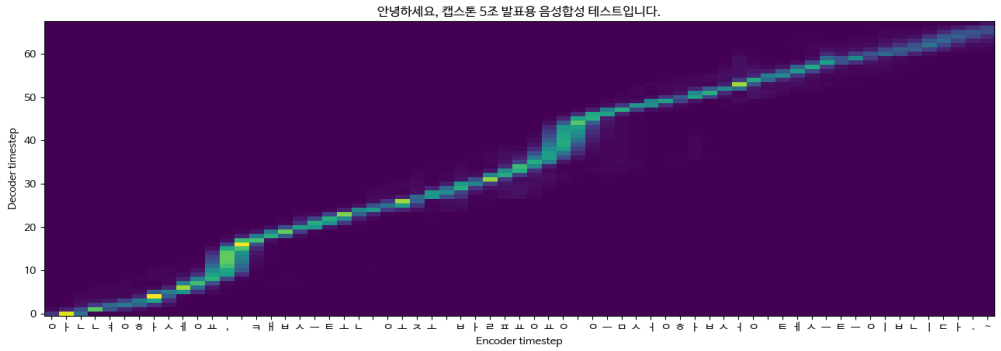


실험 7-새로운 문장 테스트

실험 8. ③번 데이터셋 1-20 번 발화자의 데이터를 Multi-Speaker Model 로 350K step 학습하였다. 학습 결과 새로운 문장으로 테스트한 음성이 생략된 음절 없이 잘 합성되었지만, 한 발화자 당 데이터양의 부족과 스마트폰 환경 음성의 품질로 인해 음성이 기계음 같이 들리는 한계점이 있었다.



실험 8-Train loss



실험 8-새로운 문장 테스트

실험 1~8 의 결과로 보아, Single-Speaker Model 은 Multi-Speaker Model 보다 한 사람당 많은 양의 데이터를 필요로 하며 여러 사람의 학습이 서버에서 이뤄져야하는 서비스 특성 상 서버의 부하가 심해질 수 있으므로 Multi-Speaker Model 을 사용해 최소 3 명의 데이터를 같이 학습시키기로 결정하였다. 또한, 해당 오픈소스 모델이 Pre-trained Model 에 대해 발화자 수를 유동적으로 변화시키지 못하므로 Pre-trained Model 은 사용하지 않기로 결정하였다. Learning rate 등의 Hyperparameter 값은 기존 값들로 유지하고, 학습 step 은 실험 도중 200K step 부터 테스트 결과물에 차이가 없다는 것을 확인해 최대 300K step 진행되도록 했다. 다양한 테스트셋을 확보하지 못해 7 시간, 100 문장(0.5 시간)이란 극단적인 데이터 양으로 실험을 진행해 어느 정도 양의 데이터가 학습을 시작하기에 충분할 지는 적절히 판단하기 어려웠다. 하지만, 실험 8 의 결과로 5 분*20 명=60 분이란 데이터 양이면 새로운 문장의 음성을 어느정도 생성할 수 있다고 생각되어 프론트엔드에서 데이터 양이 1 시간 쌓였을 때 학습 시작 요청을 하도록 설정하였다.

- 최종 모델 학습

위 실험 결과를 바탕으로 팀원 3 명 각각 스마트폰 환경에서 녹음 데이터를 만들고, 각 녹음에 대치되는 텍스트 데이터를 다음과 같이 만들어 사용하였다.

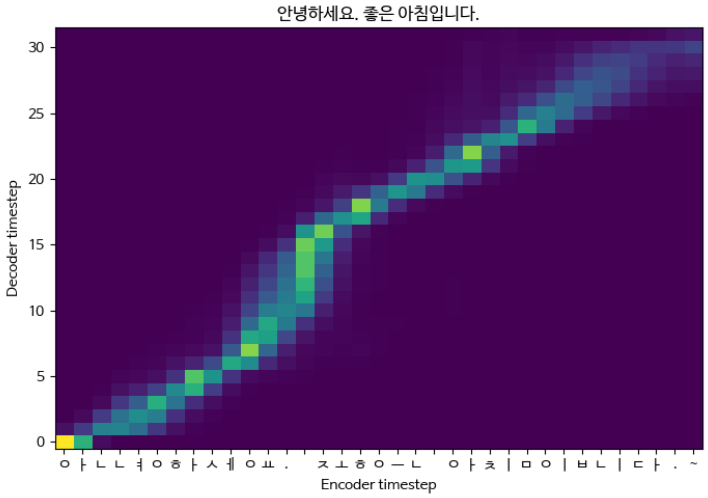
-이준협 데이터: 대본

-이민희 데이터: 대본 50% STT 50%

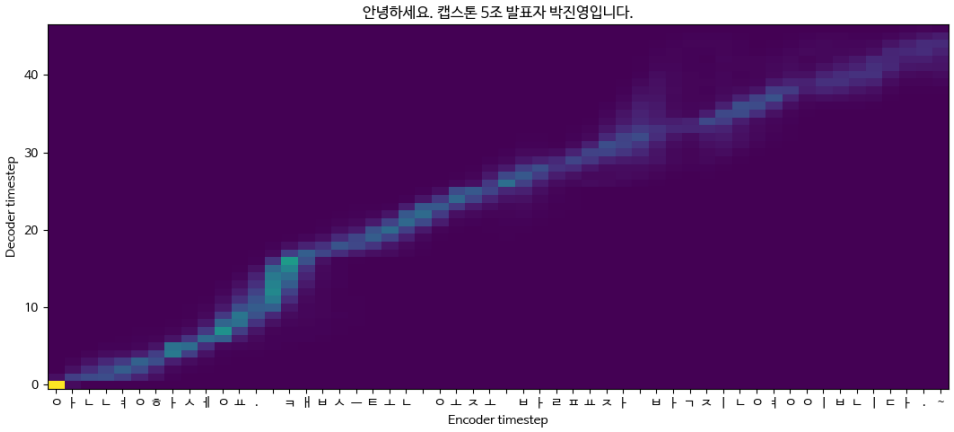
-박진영 데이터: STT

1) **각 500문장 녹음**: 직접 녹음 데이터를 수집하는데 생각보다 오랜 시간이 소모되어 우선적으로 데이터양을 각 500문장으로 하여 Multi-Speaker Model 학습을 시작하였다. 학습 결과, 새로운 문장을 생성하는데 있어 트레이닝셋에 포함되어 있는 단어들로 구성된 문장은 잘 생성해냈지만, 새로운 단어로 조합된 문장은 제대로 생성해내지 못하였다.

최종 모델 학습 1-Train loss



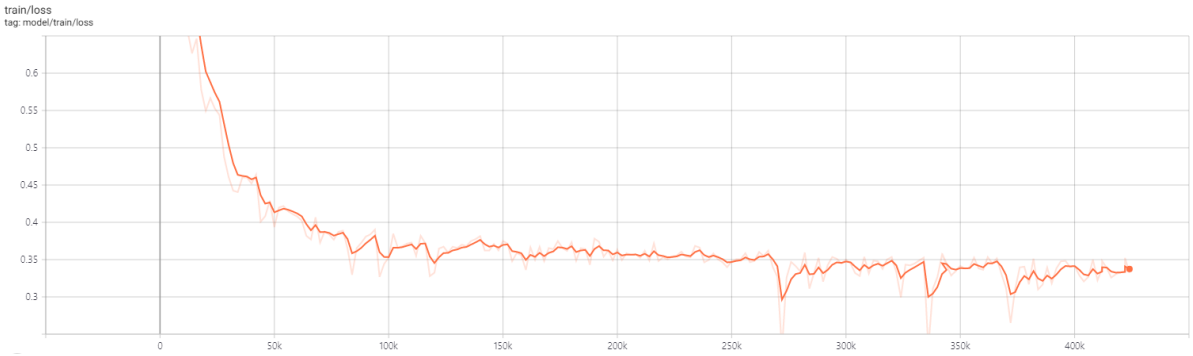
최종 모델 학습 1-트레이닝셋 포함 단어로 이뤄진 문장



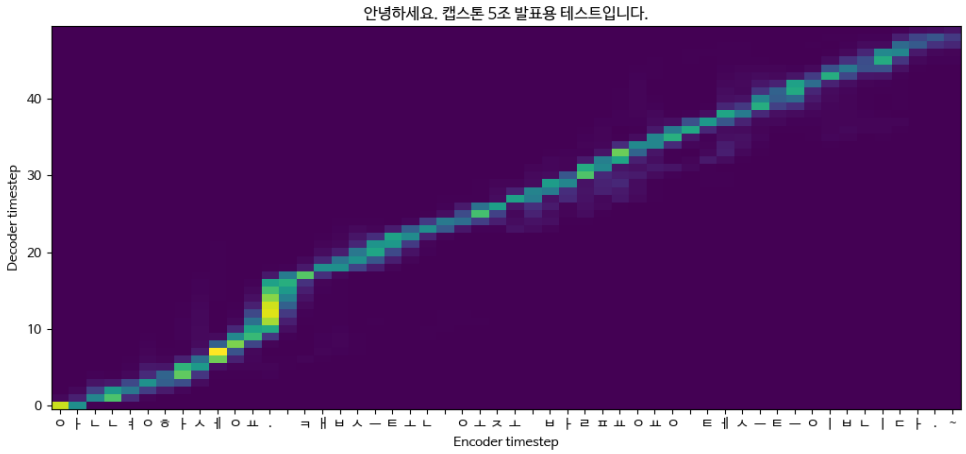
최종 모델 학습 1-새로운 문장 테스트

2) **각 1000문장 녹음**: 각 500문장의 양으로 완전히 새로운 문장들을 생성해내는데 한계가

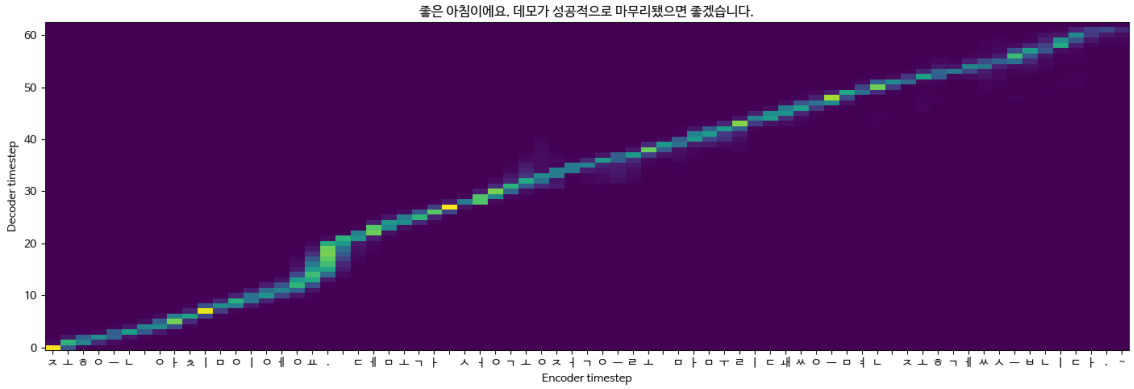
있다고 판단해 추가로 500문장을 녹음, 즉 각 1000문장(약 1시간 가량)의 데이터를 만들어 전처리까지 진행한 후 새로 학습을 진행하였다. 300K step 진행 결과 기계음 같이 들리는 문제로 예정보다 더 많은 415K step을 진행했지만, 스마트폰 환경의 소음이 낀 데이터, 대본이 아닌 STT로 부정확한 트레이닝 데이터, 전처리 후 기계음 같은 노이즈가 추가됨으로 인해 최종 모델에서 합성되는 음성이 기계음 같이 부자연스럽게 들리는 문제를 완전히 해결하지 못하였다. 하지만 트레이닝셋에 포함되지 않은 단어들로 이루어진 완전한 새로운 문장도 잘 생성해내는데 성공하였다.



최종 모델 학습 2-Train loss



최종 모델 학습 2-새로운 문장 테스트



최종 모델 학습 2-긴 문장 테스트

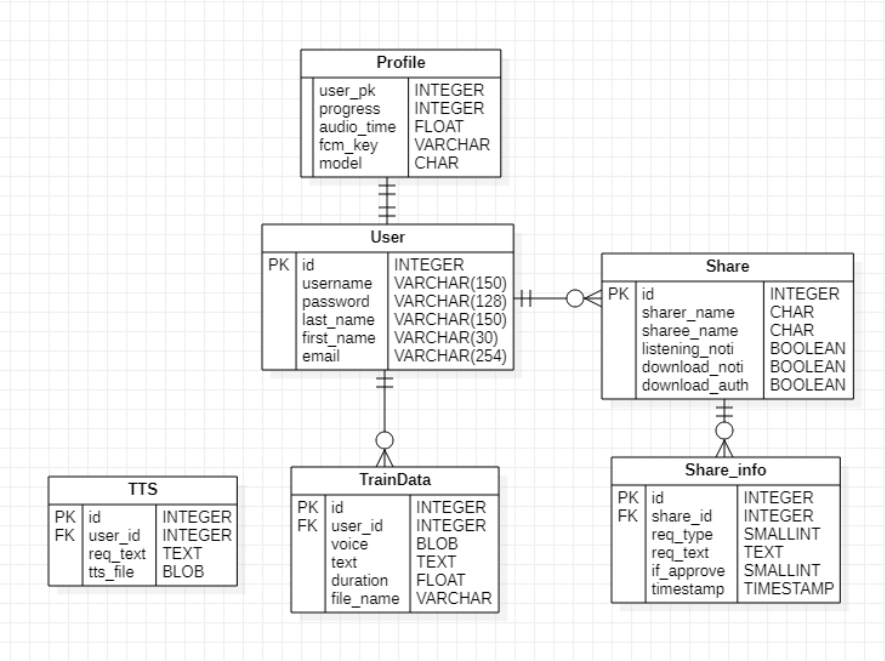
4.3. 서버

– 구현 방법

전처리 모듈과 TTS 모듈을 사용하기 위해 Python 서버인 Django 를 사용했다. 또한, 프론트엔드인 안드로이드와 연결해주기 위해서 API Server 로 구축해야 했기 때문에 Django-rest-framework 를 사용해주었다. 파일 업로드와 TTS 학습에 오랜 시간이 소요되기 때문에 Celery 를 사용해 비동기 방식으로 처리했으며, 프론트엔드 작업자와 효율적으로 작업하기 위해 API 문서화를 진행했다.

– 데이터베이스 설계

데이터베이스는 MariaDB 를 사용해 Django 와 연결해주었으며, 다음과 같이 필요한 Table 들을 설계하였다.



– 제공 기능

- 회원가입 및 로그인: Username과 비밀번호는 각 6자리, 8자리 이상으로 제한하고, Username과 Email 필드는 중복을 허용하지 않는다.
- FCM key 업로드: 알림 보낼 때 필요한 FCM key값을 받아 Profile에 저장해 놓는다.
- 모델 학습 진척도, 수집된 데이터 양 조회: 진척도는 퍼센트(%)로, 데이터 양은 초(sec) 단위로 제공해준다. 학습 진척도는 학습이 종료되었을 때의 최종 Loss와 데이터 양으로 계산했으며, 데이터양이 1시간이 넘는다면 loss*100(%)로, 데이터양이 1시간이 넘지 않는다면 1시간에 대한 비율을 loss에 곱해 계산해주었다.
- 모델 사용 가능 여부 조회: 어플리케이션에서 사용자가 자신의 모델을 갖고 있는 지 여부에 따라 버튼 비활성화 등 작동을 한다. 따라서 사용자 Profile에 사용가능한 자신의 모델이 있는지 여부를 반환해준다.
- 자신의 모델 공유 및 삭제: 공유해주고 싶은 사용자의 Username으로 자신의 모델을 공유해줄 수 있다. 공유 시 기본 알림, 허락 권한은 True로 설정되며 삭제 시 공유 모델 사용 기록 또한 모두 삭제된다.

- 자신이 공유해준 목록 조회: 프론트엔드에서 자신이 공유해준 모델의 권한을 변경하거나 삭제할 때 조회한다.
- 자신이 공유해준 모델의 사용 기록 조회: 자신이 공유해준 모델을 공유 받은 사람들이 사용한 목록으로, 요청한 텍스트, 청취인지 다운로드인지를 확인할 수 있다.
- 자신이 공유 받은 목록 조회: TTS 테스트를 할 때 사용가능한 모델 목록을 조회할 때 이 기능을 사용한다.
- 자신이 공유 받은 모델의 사용 기록 조회: 자신이 공유 받은 모델을 사용해 어떤 텍스트로 테스트를 했는지, 청취인지 다운로드인지를 확인할 수 있다. 또한, 다운로드 시 허락 상태가 응답 전인지 승인/거절 여부를 확인할 수 있다.
- 공유 정보의 알림 설정, 다운로드 권한 수정: 모델 주인이 자신이 공유해준 모델의 사용 알림, 다운로드 권한 설정을 수정할 수 있다.
- 공유 모델 사용 시 알림 보내기: 청취 시 알림만 보내지며, 다운로드 허락 권한 설정이 True이라면 모델 주인에게 다운로드 허락 요청을 보내게 된다.
- 공유 모델을 통한 TTS 음성 다운로드 허락 승인/거절: 자신이 공유해준 모델을 사용한 TTS 음성 다운로드 허락 요청에 대해 승인/거절을 선택해 응답할 수 있다.
- 통화 녹음 데이터 업로드: 통화 녹음 데이터는 비동기 작업으로 공백구간 분리, 소음 전처리, STT 단계를 거쳐 저장된다.
- 추가 녹음 데이터 업로드: 추가 녹음 데이터는 비동기 작업으로 소음 전처리 단계만을 거쳐 어플리케이션으로부터 전달받은 텍스트와 함께 저장된다.
- 모델 학습 시작 요청: 어플리케이션에서 수집한 데이터 총 양이 1시간 이상일 때 요청을 보내게 되며, 요청 보낸 사람이 3명이 넘었을 때 비동기 작업으로 학습이 진행되며, 학습이 완료됐을 때 사용자에게 학습 완료 알림을 보낸다.
- TTS 음성 테스트: 요청된 Username을 가진 User의 Profile에 저장된 모델을 사용해 요청 받은 텍스트로 음성을 합성하고 어플리케이션에서 재생, 다운로드 할 수 있게 서버에 저장된 음성 파일의 주소를 반환한다.

4.4. 프론트엔드

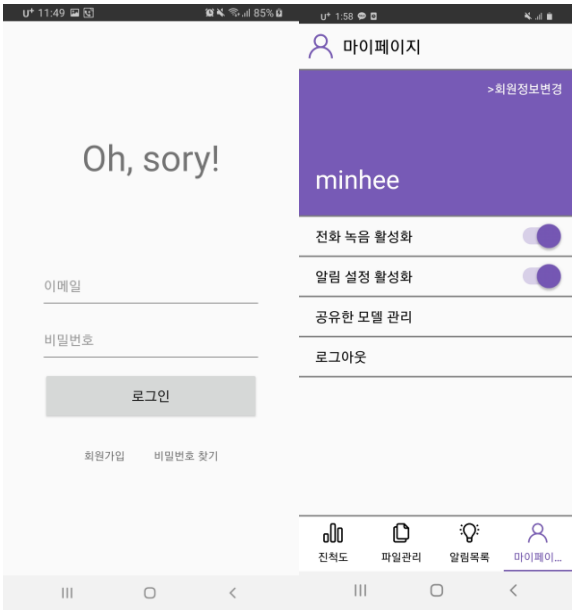
통화 시 녹음 음성을 획득할 수 있고, 사용자들이 쉽게 사용하기 위해 Android Studio 를 사용하여 어플리케이션을 제작하였다.

- **로그인**

처음 어플리케이션을 사용하는 사람도 앱의 취지와 사용법을 알 수 있도록 Viewpager adapter 를 사용해 전반적인 설명을 띄워주었다. 화면들을 스와이프 한 후 시작하기 버튼을 누르면 로그인 창으로 넘어가게 된다.

다음 사진은 로그인, 회원가입 화면이다. 한 번 로그인을 하면 로그인 정보를 Preference 에 저장 해놓기 때문에 자동 로그인 하도록 구현해 놓았으며 로그아웃을 원할 시 마이 페이지 화면에서 버튼 클릭으로 로그아웃을 할 수 있다. 아이디 별로 서버에 고유의 모델이 생성되기 때문에, 원한다면 로그인 로그아웃을 통해 아이디를 변경해 한 어플리케이션으로도 다양한 모델을 만들 수 있다.

회원가입 시 아이디와 이메일을 중복된 값으로 설정하지 않도록 구현하였다. 아이디는 서로 모델을 공유할 때, 유저를 구분하기 위해 사용되기 때문에, 한 계정 당 아이디와 이메일이 유일한 값으로 관리하는 것이 좋을 것이라 판단하였다. 유효한 입력을 받았는지, 모든 Edittext 창에 값이 들어와 있는 지 등을 앱 상에서 확인 후, 서버에 회원가입을 요청하면 서버에서 중복되는 값이 있는지 검색한 후 response code 로 상황에 맞는 값을 반환한다. 앱은 그 값에 따라 작동하도록 구현하였다. 중복된 값이 있다면 무슨 값이 중복인지 Toast 를 띄워주고, 올바른 정보일 땐 회원가입을 완료한다.



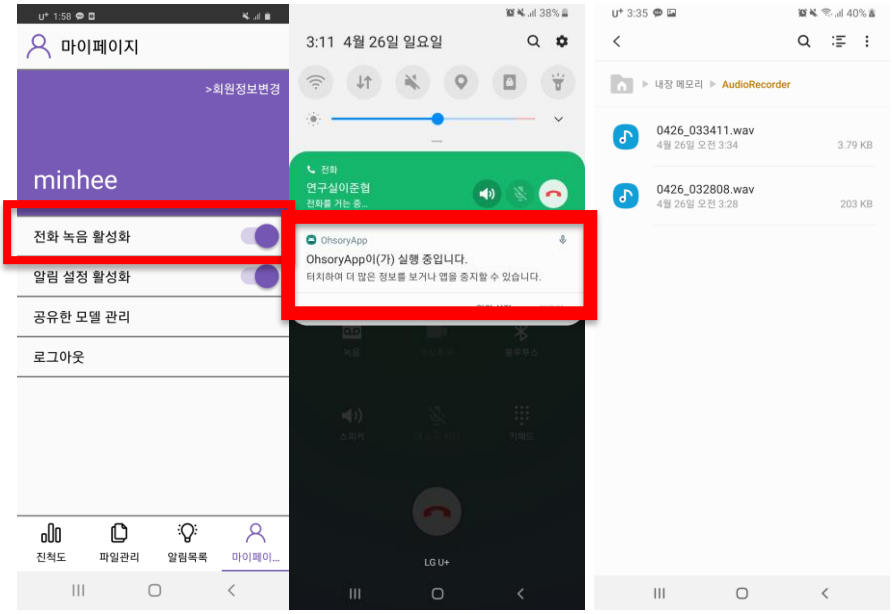
- **통화 오디오 녹음**

사용자가 스마트폰으로 전화하는 상황을 인식해, 사용자의 목소리를 녹음하여 데이터셋을 축적하는 기능이다. 밑의 왼쪽 사진과 같이 마이 페이지에서 '전화 녹음 활성화' 토글 버튼을 통해 기능을 컨트롤 할 수 있다.

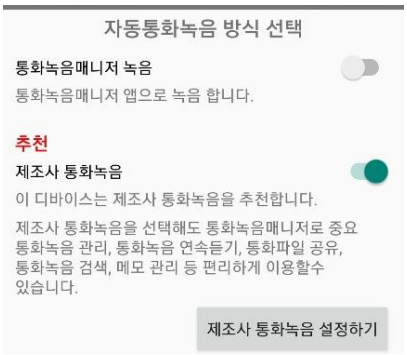
기능을 활성화 해놓으면, PhoneStateListener 를 사용해 앱이 사용중인 상태가 아니라도 핸드폰의 전화 상태 변화 이벤트를 감지할 수 있다. 전화 중인 상태이면 상단에 notification 을 띄워 마이크에서 소스를 가져와 그것을 녹음하는 service 를 foreground service 로 동작하도록 하였다. 이는 사용자가 앱이 실행중인 것을 알 수 있게 해주고, 안드로이드 정책에 의해 background service 는 원치 않는 리소스 사용 제한의 영향을 받기 때문이다.

사용자만의 목소리를 녹음 즉, 마이크로 들어오는 소리(VOICE_UPLINK)만 녹음을 하여, 사람 목소리 구분을 위한 추가적인 작업을 피하고, 원하는 데이터만 모으도록 하였다. 하지만 안드로이드 보안 정책에 의해 버전 10 에서는 전화 시 마이크에서 소스를 아예 가져올 수 없었고, 버전 10 미만에서는 전화 녹음이 불법인 곳이 있어 VOICE_UPLINK 로 설정시 소스를 가져올 수 없었고, VOICE_CALL(VOICE_UPLINK + VOICE_DOWNLINK)을 사용할 경우 VOICE_UPLINK 값에만 접근할 수 있어 위와 같이 VOICE_UPLINK 를 사용해 원하는 대로 작동하도록 구현하였다.

녹음된 전화 음성 파일은 로컬 저장소에 전화를 한 시간을 이름으로 저장되어 있다가, 앱 실행 시 네트워크에 연결이 된 경우에 서버로 보내고 로컬에서 삭제를 한다. 이렇게 구현한 이유는 전화 시 항상 네트워크에 접속 되어있지 않을 수도 있을 것이며, 전화 종료 시 notification 을 사용자가 지우지 않아도 없어지게 하기 위해서 바로 foreground service 를 종료하기 때문에 서버로 보내는 작업까지 바로 진행하기엔 무리가 있을 것이라고 판단하였기 때문이다.



시중에 나와있는 안드로이드 버전 10 에서도 작동하는 통화 녹음 앱들은 어떻게 구현을 해 놓았는지 참고해보았는데, 아래와 같이 시스템 기본 전화 앱에 있는 '통화 녹음' 기능을 활성화 하도록 권유하며, 이 설정 화면으로 넘겨주는 버튼을 제공하는 방법이 일반적이었다. 하지만 이와 같은 방법은 통화 자체를 녹음하여, 상대방의 목소리까지 녹음이 되어 사람 목소리 구분 기능이 없는 우리의 앱에는 부적합하다고 판단하여, 추후에 버전 10 에도 작동하는 앱을 제공하기 위해선, 사람 목소리 구분 기능을 추가하거나, 보안 정책이 바뀌게 되면 그에 맞춰 다시 개발을 하는 식으로 진행해야 할 것이다.

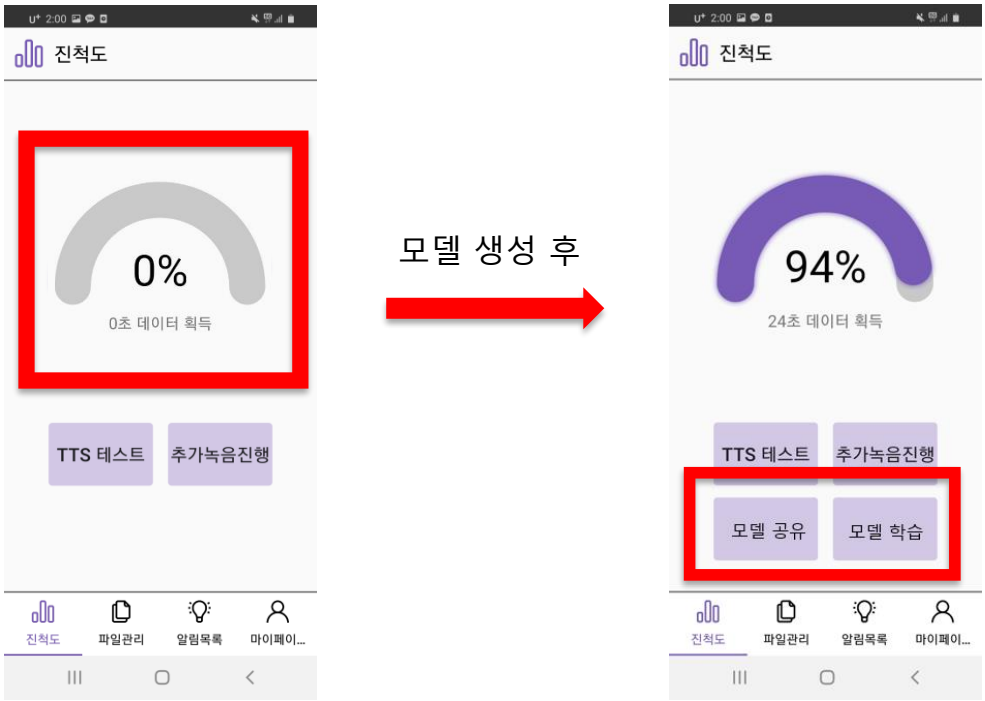


- 음성 합성 진행 현황 / 모델 학습

왼쪽 사진은 모델이 없는 사용자의, 오른쪽 사진은 모델이 있는 사용자의 메인 화면이다. Progress bar 와 Textview 를 사용하여 서버에서 받아온 모델의 진척도, 현재까지 모은 데이터의 총 시간 정보들을 보여준다.

모델이 없는 사용자의 음성 데이터가 일정 양 이상까지 차면 모델을 만들만한 정도로 데이터가 수집되었다고 판단하여 자동으로 앱에서 서버로 모델 요청을 보낸다. 우리 앱에서는 우선 1 시간 이상의 데이터가 모이면 모델 학습을 요청한다.

모델 생성이 완료 되면 모델 결과에 따라 Progress bar에 값이 변경된다. 이 기능으로 사용자는 쉽게 진행 현황과 모델 성능을 확인할 수 있다. 또한 자신의 모델이 있으므로 하단에 '모델 공유' '모델 학습' 버튼이 활성화 된다. 모델 공유를 통해 자신의 모델을 공유하거나, 앱을 지속적으로 사용해 음성 데이터가 더 모인 경우 수동으로 모델 학습 요청을 보낼 수 있다.



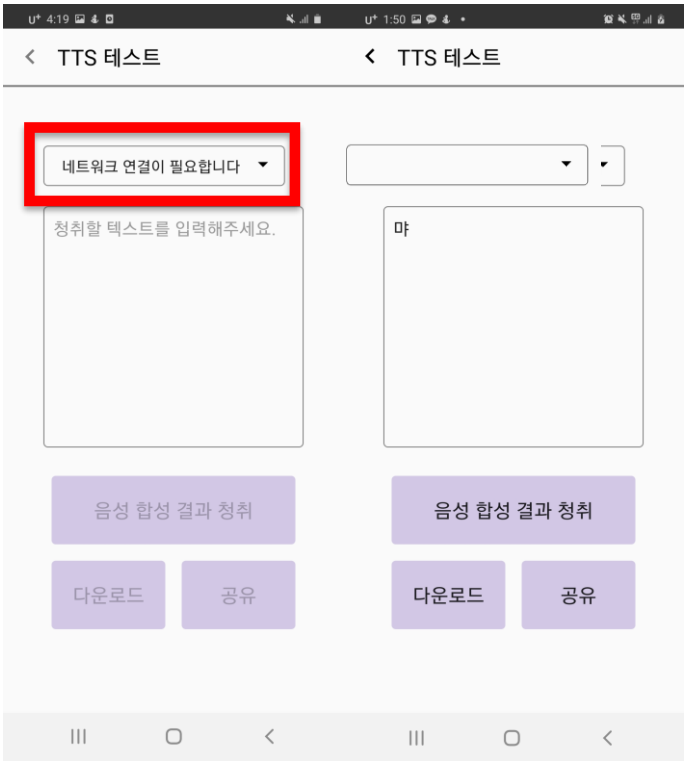
- TTS 테스트

사용자가 가진 모델들을 사용하여 원하는 문장을 입력하고 그에 대한 TTS 음성 파일을 청취/다운/공유 할 수 있다. 네트워크에 연결이 되지 않은 경우, 가지고 있는 모델이 없을 경우 Spinner에서 설명해주며 Edittext 와 버튼들이 비 활성화 되어있다. 네트워크 연결이 되면 사용자가 접근 할 수 있는 모델 목록을 서버로부터 받아와 Spinner에 띄워주며 모델을 선택 시 Edittext가 활성화되고, Edittext 창에 입력을 하면 '음성 합성 결과 청취 버튼'이 활성화 된다.

'음성 합성 결과 청취' 버튼을 클릭하면 선택된 모델과, 입력된 문장을 서버로 보내 TTS 합성을 요청하며 생성된 결과 음성이 있는 링크에 접근해 파일을 받아 읽고 결과를 MediaPlayer 를 사용해 들려주고 '다운로드' 버튼이 활성화된다. 보통 음성 파일을 들려줄 때는 다운로드를 하고 파일에 접근해서 읽는 방법을 주로 쓰는 것 같지만, 우리의 어플리케이션은 모델 주인이 다운로드 허용을 해줘야만 다운로드 할 수 있는 경우가 있으므로 File.createTempFile 를 사용하여 캐시에 저장하여 사용자는 파일에 접근할 수 없고, 들을 수만 있다.

'다운로드' 버튼을 누르면 'Ohsory/선택된 모델 주인 ID/요청 문장.wav'로 파일이 생성되고, 공유 버튼이 활성화된다. 요청 문장을 파일 이름으로 설정하면 중복 위험이 있을 것이라 판단하였지만, 같은 모델로 같은 문장 음성 파일을 생성하였을 때는 덮어씌워주는 것이 너무 많은 파일이 생성되지않고, 사용자의 의도에 맞게 업데이트하는 것이라 판단하여 위와 같이 구현하였다.

'공유' 버튼을 누르면 생성된 wav 파일 자체를 공유 할 수 있다. 이러한 TTS 테스트를 하다가, 또 다른 TTS 테스트 요청을 위해 다른 문장을 입력하고 '음성 합성 결과 청취' 버튼을 누르면 버튼들이 초기 상태가 되었다가 앞에서 설명한 것 과 같이 작동한다.



- **추가 녹음**

데이터 수집 속도가 느려서 추가적으로 데이터를 수집해 빠르게 모델 생성을 하고 싶거나, 모델의 성능을 높이고 싶은 사람은 '추가녹음진행' 버튼을 클릭해 추가 녹음을 진행할 수 있다.

예시 문장들 중에 사용자가 원하는 문장을 직접 선택해 녹음할 수 있다. 네트워크에 연결되지 않은 상황에서도 녹음할 수 있도록 로컬에 txt 파일을 저장해 놓는다. 앱을 처음 설치하고, 이 파일이 Ohsory 폴더에 없다면, 안드로이드 raw 에 txt 파일을 불러 한 줄씩 읽어 로컬에 저장한다. 왜 그냥 항상 raw 에서 불러와서 읽으면 되지 로컬에 따로 또 파일을 저장해야하나 의아할 수 있지만, 사용자가 중복된 문장을 녹음하지 않도록 한 번 녹음한 문장은 txt 파일에서 지워 나가며 관리하는데, raw 에 있는 파일은 읽을 수만 있기 때문에 로컬에 저장하도록 구현하였다.

로컬 파일에 계속 접근하는 것은 시간이 오래 걸리므로 맨 처음 Activity 를 띄울 때만 파일을 한 번 열고 한 문장씩 ArrayList 에 담아놓고 접근하였다. 또한 녹음을 해서 지워야 하는 문장을 그때그때 txt 파일을 열어서 수정하는 것이 아닌, ArrayList 의 값만 삭제 후 Activity 가 종료될 때 ArrayList 값이 변해 있을 때만 파일을 변경하였다.

Infinite RecyclerView 를 사용해서 원하는 방향으로 계속 스크롤이 되도록 하였고, 스크롤뿐만 아니라 좌우 버튼 클릭을 통해서도 문장을 선택할 수 있다. 녹음 버튼을 눌러 녹음을 시작하고 일시 정지 버튼을 눌러 녹음을 종료하면 오른쪽 사진과 같은 화면이 되는데 여기서 녹음 된 음성 파일을 중간 점검으로 들어볼 수 있고, 마음에 들지 않는 경우 재 녹음,

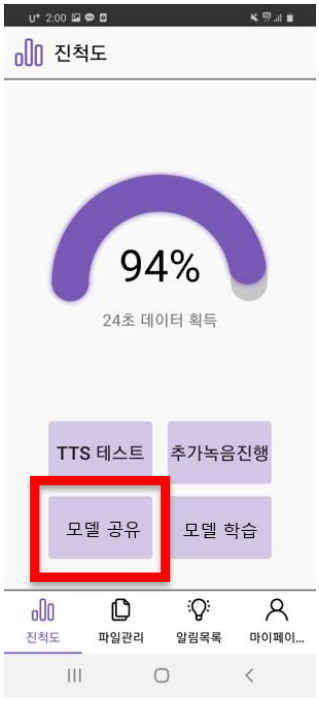
마음에 드는 경우 음성 파일을 저장할 수 있다. 저장 후 ArrayList 에서 문장이 삭제되고 RecyclerView 가 업데이트 된다.

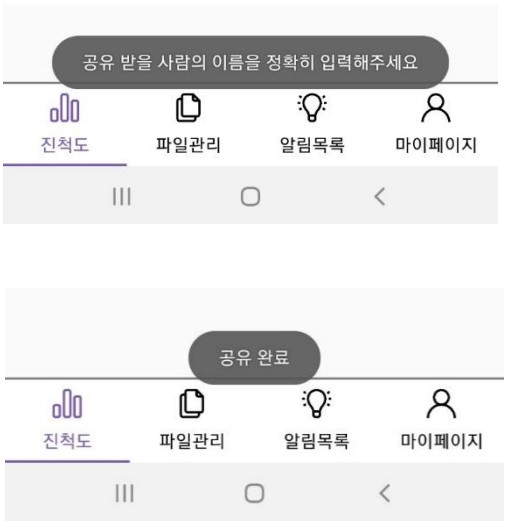
저장 시 네트워크에 연결 되어있으면 바로 서버로, 연결되지않는 경우 전화 녹음과 같이 로컬에 저장되었다가 앱 실행 시 보낼 수 있는 상황일 경우 서버로 전송하게 된다. 전화 녹음과 다른 점은 전화 녹음은 사용자가 무슨 문장을 말했는지 사전에 알 수 없어, 서버에서 STT 를 하여 수집한 음성과 문장을 쌍으로 데이터를 관리하는데, 추가 녹음 시엔 이미 문장이 정해져 있으므로 문장도 같이 서버로 전송한다.

사용자가 추가 녹음 기능을 통해 수집된 데이터의 양이 늘어난 것을 바로바로 확인 할 수 있는 것이 추가 녹음 화면의 필요성을 느낄 것으로 판단하여, 추가 녹음을 끝내고 메인 progress 화면으로 다시 돌아왔을 때 바로 업데이트 된 값으로 화면을 변경하도록 구현하였다.

- **모델 공유**

자신의 모델이 만들어진 사용자는 주변 사람들에게 자신의 모델을 공유할 수 있다. 공유 받은 사람은 접근 가능한 모델 목록이 업데이트 되어 TTS 테스트 창에서 공유해준 사람의 모델을 사용할 수 있다. 공유를 해줄 사람이 메인 화면에서 '모델 공유' 버튼을 누르면 하단 사진과 같은 dialog 가 뜨고, 공유를 해줄 사람의 UserName 을 입력, 자신의 모델이 사용될 때 어떠한 알림을 받을 지 알림 설정을 할 수 있다. 공유 받을 사람의 이름이 유효한 값이면 공유가 완료된다.

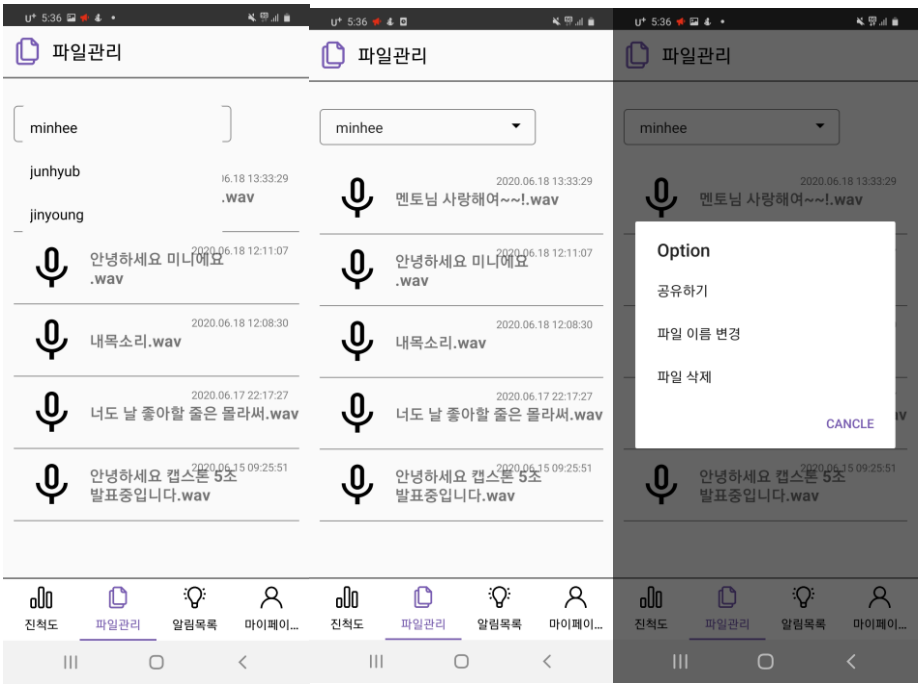




- **파일 관리**

우리 어플리케이션에 의해 생성되고, 다운로드 받아진 파일들을 앱 상에서 관리할 수 있는 화면이다.

Ohsory 폴더 안에 있는 폴더 들의 이름은 각 음성 파일의 모델 주인 ID 이기 때문에, Ohsory 폴더 안에 있는 폴더들의 이름을 불러와 Spinner 에 띄워준다. Spinner 를 선택해 그 폴더 안에 있는 음성 파일들의 이름을 RecyclerView 로 띄워주었다. 각 파일을 클릭 시 음성들을 들을 수 있으며, 롱 클릭 시 공유하기, 이름 변경, 파일 삭제 기능을 수행할 수 있다.



- **알림 목록**

모델을 공유할 때 설정해놓은 모델 알림 설정에 맞추어서 모델을 공유 받은 사용자가 공유 받은 모델을 사용하면 모델 주인에게 사용 알림이 가게 된다.

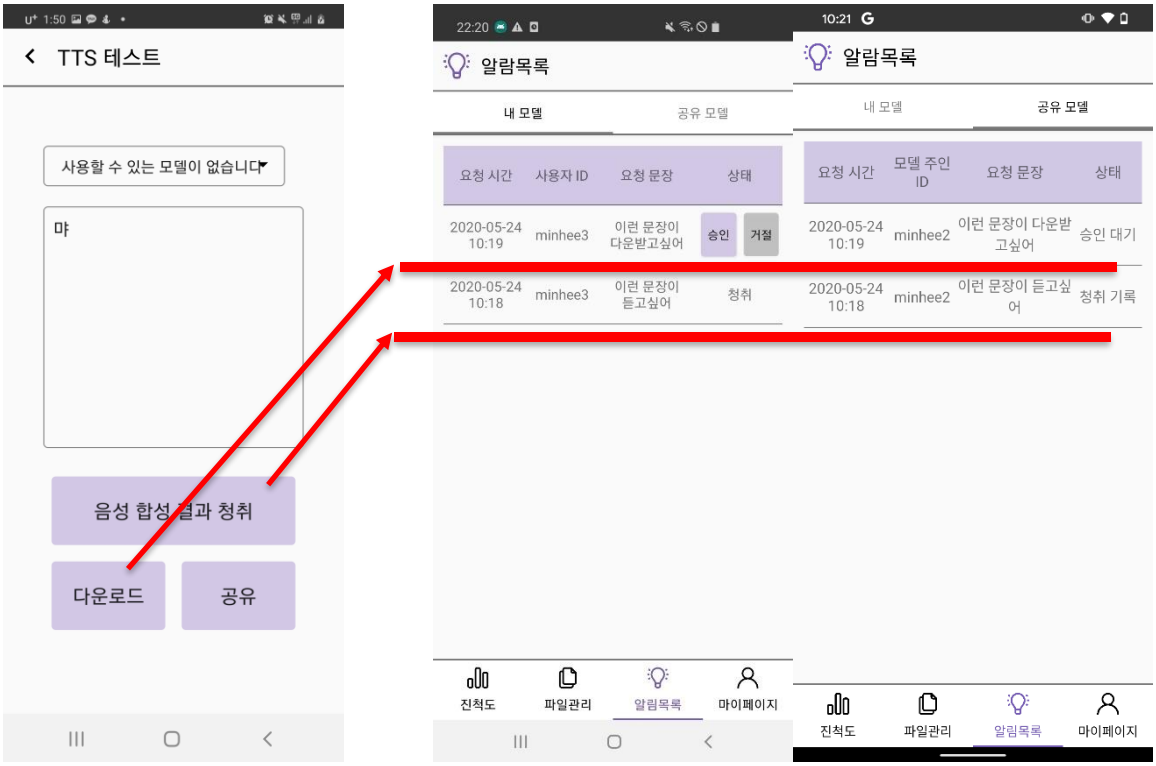
음성 합성 결과 청취 버튼을 누른 경우, 모델 주인에게 타이틀로 '공유 받은 사용자의 ID'의 모델 사용 알림 바디로 요청 문장 : '요청 문장' 인 notification 이 뜨게 되고, 다운로드 버튼을 누른 한 경우 타이틀로 '공유 받은 사용자의 ID'의 다운로드 요청 바디로 요청 문장 : '요청 문장' 인 notification 이 뜨게 된다. 이 알림창을 선택하면 우리의 어플리케이션이 실행된다.

앱을 실행하고 알람 목록 화면에 오면 지금까지 받은 알림을 확인 할 수 있다. 내 모델을 다른 사용자가 사용해서 받은 알림, 사용 내역을 '내 모델'에서, 공유 받은 모델을 사용한 내역과, 다운로드 요청의 현재 상태를 '공유 모델' 화면에서 확인한다

밑에 예시 화면들로 두 사용자 사이의 공유 interaction 을 설명한다. '모델 사용 알림 설정을 모든 알림 활성화, 다운로드 요청 시 허락 필수'로 설정한 사용자가 다른 사용자에게 모델을 공유한 상황이다. 하단바가 회색인 사람이 자신의 모델 공유를 해준 사용자, 하단바가 검정색인 사람이 모델 공유를 받은 사용자이다.

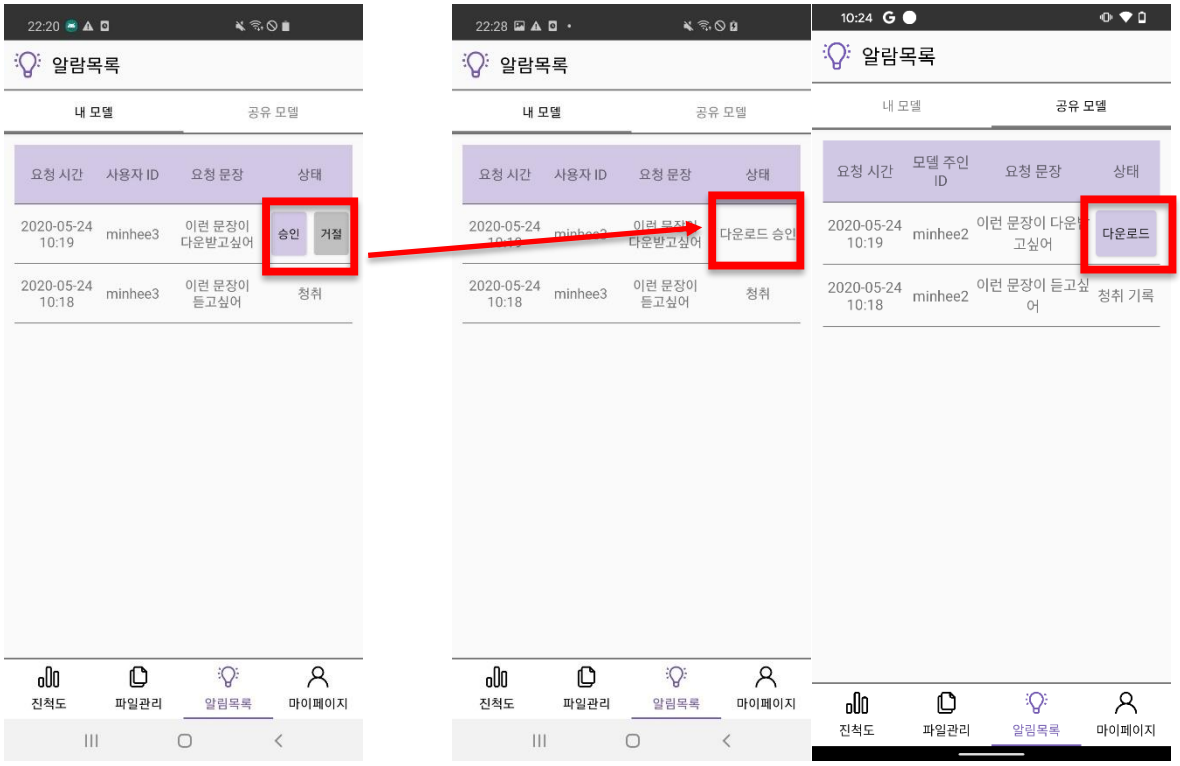
공유 받은 사람은 TTS 창에서 공유해준 사용자의 모델을 선택하고, 요청 문장을 입력 후 '음성 합성 결과 청취' 버튼을 누르면 모델 주인에게 알림을 보내며 바로 청취가 가능합니다. 하지만 다운로드는 허락이 있어야 하므로 모델 주인에게 알림을 보내고 '다운로드 요청을 하였습니다. 알람 목록을 확인하세요' Toast 창이 뜨며 다운로드는 되지않습니다. 알림 목록 화면을 확인하면 청취 기록을 볼 수 있고, 모델 주인의 허락을 받기 전에는 승인 대기 상태인 것을 확인 할 수 있다.

공유를 해준 사람은 내 모델 창에서 어느 시간에, 어떤 공유를 받은 사람이, 어떤 문장을, 청취하였는지, 다운로드 요청을 하였는지 확인 가능하고, 다운로드 요청이 온 경우 승인, 거절 버튼을 통해 컨트롤이 가능하다.



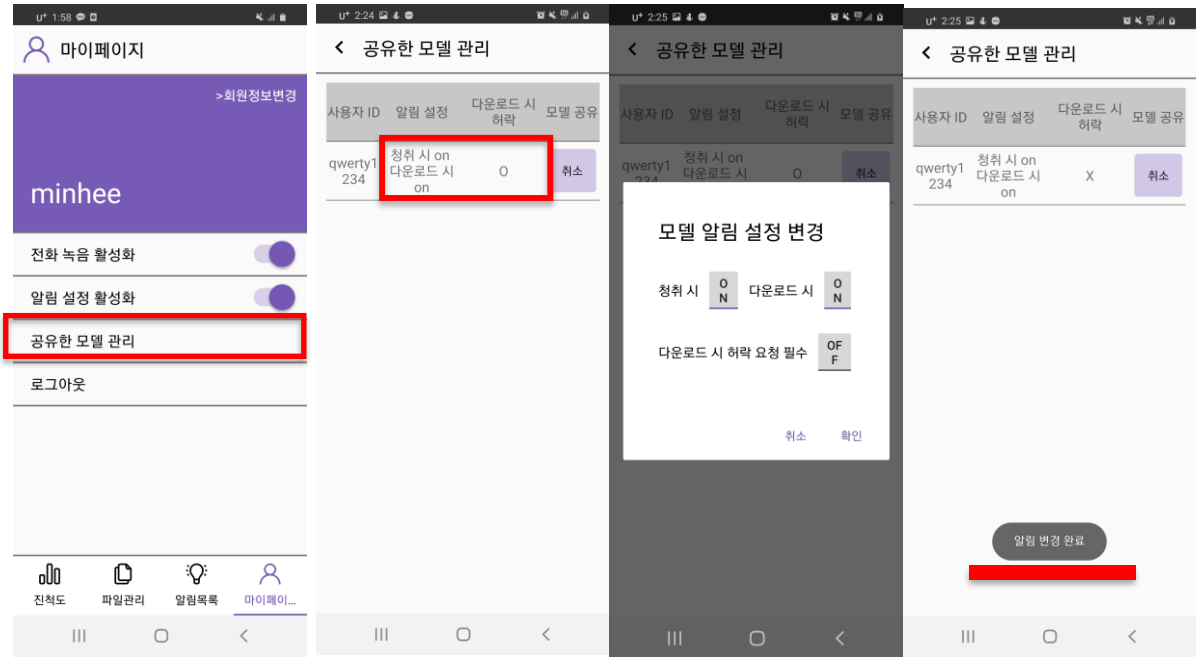
모델 공유를 해준 사람이 승인 버튼을 누르면 버튼 두개가 사라지고 다운로드 승인 하였다는 텍스트 상자로 변경되고, 다운로드 요청을 한 사용자의 화면도 다운로드를 할 수 있도록 활성화된 버튼을 확인할 수 있다. 만약 거절 버튼을 누를 경우, 다운로드 거절로 문구가 바뀌고, 요청을 한 사용자의 화면에 다운로드 버튼이 생기지 않는다.

다운로드 시 허락 기능이 꺼진 경우 알림은 똑같이 작동하나, 다음과 같은 허락 없이도 다운로드 요청을 할 사용자가 다운로드 버튼을 누르자마자 바로 다운로드를 할 수 있다.

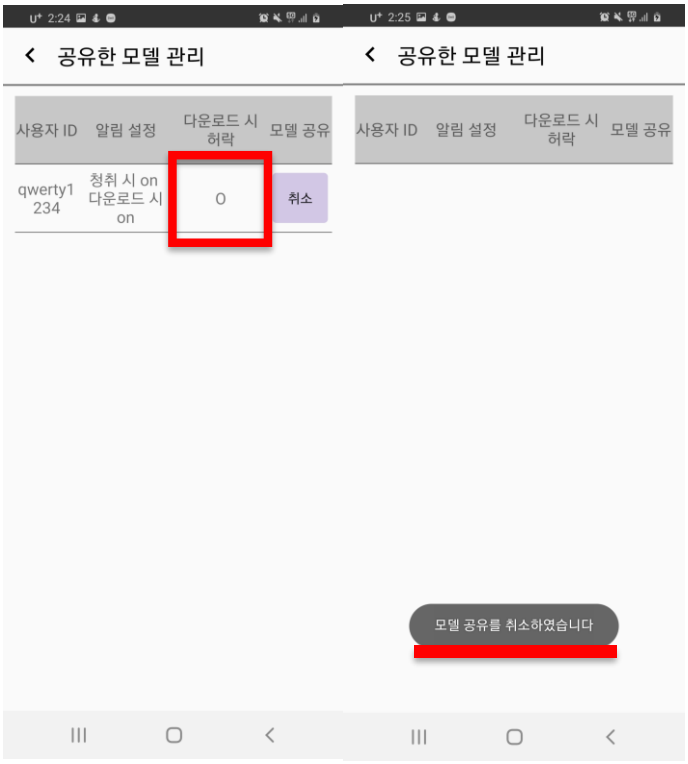


- **모델 관리**

마이 페이지 -> 공유한 모델 관리에서 다른 사용자에게 자신의 모델을 공유 해준 정보들을 확인하고 관리할 수 있다. 각 정보들을 서버로부터 받아와 RecyclerView 로 띄워주고 있고, 알림 설정, 다운로드 시 허락 정보 값들을 선택 시 각 모델을 공유할 때 설정했던 모델 알림 설정을 변경할 수 있다.



또한 모델 공유를 아예 취소하는 기능이 있다. 공유 기능을 그만 두고 싶은 사용자에 해당하는 RecyclerView Item 의 취소 버튼을 누르면 '모델 공유를 취소하였습니다' Toast 가 뜨게 되고 그 사용자는 이제 더 이상 자신의 모델에 접근할 수 없다. 추후 이 취소 기능을 실행 할 때 상대방이 자신의 모델을 사용하여 다운 받은 파일들도 다 삭제하도록 요청을 할지 말지 선택할 수 있고, 선택된 요청을 실행하도록 하는 기능도 추가되면 매우 좋을 것 같다.


공유한 모델 관리
사용자 ID
알림 설정
다운로드 시 허락
모델 공유
qwerty1234
청취 시 on
다운로드 시 on
0
취소
공유한 모델 관리
사용자 ID
알림 설정
다운로드 시 허락
모델 공유
모델 공유를 취소하였습니다

5. 프로젝트의 한계점 및 발전 가능성

5.1. 한계점

- **기계음이 많이 끼는 전처리 모델의 결과물 & 오래 걸리는 전처리 과정**

소음과 원음을 모든 상황에서 완벽하게 분리하는 모델은 존재하지 않기 때문에 어느 정도의 기계음은 용인이 되지만 우리 모델은 기대했던 소음 제거수준보다 아쉬운 부분을 보였다. 카페 소음이나 에어컨 소음, 도로 소음 같은 큰 소음같은 경우는 잘 제거해주었지만 약간 거슬리는 정도의 작은 소음은 제거하지 못하거나 큰소리를 제거해도 남아있는 경우가 대부분이었다. 데이터 학습 시 약 3dB 수준의 합성음만 생성하고 높은 수준의 합성음은 학습하지 않아서 발생한 것으로 유추된다. 전처리가 완벽하지 않다 보니 그 뒤로 진행되는 STT 모델과 TTS 모델에 영향을 끼쳤다.

또한 GPU 의 성능에 따라 모델의 속도가 다르지만 약 15~30 초 정도 걸린다. 사용자가 TTS 모델을 이용할 때 불편함을 느낄 정도의 시간이라 생각된다.

- **데이터 품질에 크게 영향받는 TTS 모델**

TTS 모델이 학습을 할 때 전처리가 완료된 음성 데이터와 해당 음성데이터의 STT 결과물의 쌍을 학습데이터로 사용한다. 이때 전처리모델이 가진 한계와 STT 모델과 VAD 의 부정확성에 의해 어느정도는 소음과 공백이 끼고 텍스트 쌍도 약간은 틀린 데이터쌍들이 만들어진다. 이러한 데이터들을 가지고 TTS 모델이 학습할 경우 학습시간을 아무리 늘려도 결과물의 음질이 좋지 않게 생성되는 한계점이 있으며 이를 해결하기 위해선 더 높은 수준의 연구가 필요할 것으로 보인다.

5.2. 발전 가능성

- **전처리 과정의 수준 향상**

프로젝트를 진행하면서 전처리 파트에서 부족했던 점들이 많이 느껴졌다. 소음 제거 모델을 학습 시킬 경우 학습 데이터를 만들 때 한가지 SNR 을 가지는 합성음만 만들지 말고 낮은, 적당, 높은 수준 이렇게 최소 세가지 SNR 을 가지도록 합성음을 만든다면, 더 자연스럽고 기계음이 거의 없는 음원파일을 생성하는 모델이 만들어질거라 예상된다. 추가적으로 전화 녹음 데이터가 많이 쌓이게 된다면 VAD 도 머신러닝을 통해 더욱더 정교하게 공백분리를 하도록 만들어 줄 수 있다. Speech-to-Text 모델도 Kakao 에서 무료로 제공하는 STT API 가 아닌 더 좋은 성능을 가지는 모델로 바꾸면 오답률이 적은 텍스트가 만들어 질 것이다. 이런식으로 전체적인 전처리 과정을 개선하면 정제되고 깔끔한 데이터쌍이 만들어질텐데, 이는 TTS 모델이 더 자연스러운 합성음을 만들 수 있도록 할 것이다.

- **합성음의 다양한 활용**

현재 우리 프로젝트는 합성음을 단순히 개인이 소유하거나 타인에게 공유하는 수준에서 사용하고 있다. 하지만, 다양한 플랫폼이 존재하는 이 시대에서 TTS 합성음의 활용 범위는 굉장히 넓다. 책의 text 파일을 넘겨주었을 경우 원하는 사람이 읽어주는 오디오 북을 만들 수 있으며, 본인만의 ai 비서를 만들 수도 있다. 앞서 말한 전처리 기술을 개선하여 더 수준 높은 TTS 합성음을 만드는데 성공하였을 경우 추후에 합성음을 활용할 수 있는 타 기업들과의 제휴도 노려봄직하다.

6. 업무 분담

- 박진영
  - 서버 설계 및 구축
  - 음성 합성 모델 구현

- 이민희
  - UI 디자인 및 구현
  - 앱 기능 구현

- 이준협
  - 데이터 수집 및 전처리
  - Speech-to-text

7. 개발 일정

(공통: 하늘색 / 이준협 : 노란색 / 박진영: 빨간색 / 이민희: 녹색)

	주차	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	제안서 제출 및 아이디어 정리									중간데모					최종데모	최종보고서
	구현 방법 학습 및 자료조사															
이준협	전처리 기술조사															
	데이터 수집															
	노이즈 제거(신호 처리)															
	데이터 전처리 (VAD)															
	노이즈 제거(머신러닝)															
	전처리(필터링)															
	speech-to-text															
이민희	문장 생성 자료 조사															
	UI 디자인/구현															
	앱(로그인 및 자동 녹음)															
	앱(서버 연동)															
	앱(공유/다운로드)															
	앱(진척도 표시 및 나머지 파트)															
	문장 생성															
박진영	음성합성 자료 조사															
	서버, DB 설계															
	서버, DB 구축															
	TTS 모델 학습															
	모델 모듈화															
	진척도 판단 알고리즘															
	테스트 및 성능 향상															

8. GitHub

https://github.com/celi1004/Capstone_Ohsory