

졸업논문

제 목 : 딥러닝과 챗봇을 이용한 실시간 대화 서비스

지도교수 : 김 명 진

2023 년 11 월 14 일

한국외국어대학교 공과대학 정보통신공학과

학번 201802454 성명 이 광 호

학번 201802781 성명 이 재 환

학번 201801085 성명 김 준 석

청 구 및 승 인 서

제 목 : 딥러닝과 챗봇을 이용한 실시간 대화 서비스

대 학 : 공과대학

학과 : 정보통신공학과

학 번 : 201802454

성 명 : 이 광 호

학 번 : 201802781


성 명 : 이 재 환

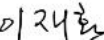
학 번 : 201801085


성 명 : 김 준 석

이 논문을 제출하오니 승인하여 주십시오.

2023년 11월 14일


성 명 : 이 광 호 (인) 

이 재 환 (인) 

김 준 석 (인) 

위 학생의 논문 제출을 승인함.

2023년 11월 14일

지도 교수 : 김 명 진 (인) 

목차

1. 서론	-1-
1.1 연구배경	-1-
1.2 연구목표	-5-
1.3 연구내용 및 방법	-6-
2. 관련기술동향 조사 및 분석	-7-
2.1 음성 스펙트럼 분석	-7-
2.1.1 Fourier transform(ft)	-7-
2.1.2 Mel Spectrogram	-8-
2.2 STT(Speech-to-Text)	-10-
2.3 TTS(Text-to-Speech)	-12-
2.4 대형언어모델(LLM)	-15-
2.4.1 GPT-3	-17-
2.4.2 SKT-KoGPT2.....	-19-
2.4.3 SKT-KoGPT-trinity.	-19-
2.4.4 KoGPT.....	-19-
2.5 노이즈 어텐션	-22-
3. 시스템 설계	-24-
3.1 사전 학습 구성도	-24-
3.2 서비스 구조도	-25-
3.3 음성 데이터 수집	-26-
3.4 음성 데이터 전처리	-27-
3.5 음성 데이터 대본 출력 및 text 파일 저장	-29-
3.5.1 구글 STT 라이브러리 설명.....	-29-
3.5.2 대본 출력 및 학습용 text 파일 생성	-30-

3.6 음성 합성 모델	-31-
3.6.1 Tacotron2	-31-
3.6.2 Tacotron	-33-
3.6.3 WaveNet	-35-
3.6.4 WaveGlow	-37-
4. 시스템 구현	-41-
4.1 음성 데이터 수집 전처리	-41-
4.1.1 음성 데이터 수집	-41-
4.1.2 음성 데이터 전처리	-42-
4.2 대본 생성 및 학습용 text 파일 생성	-44-
4.3 모델 학습	-45-
4.4 Flask 웹 서버 구현	-49-
4.5 안드로이드 앱 개발	-53-
4.5.1 로그인 화면	-54-
4.5.2 목록 화면	-55-
5. 실험 및 결과 분석	-62-
5.1 실험계획	-62-
5.2 실험결과	-63-
6. 결론 및 향후 연구 방향.....	-80-
7. 참고문헌	-82-

그림 목차

[그림 1.1-1] Amazon re:MARS 2022 캡처본	-1-
[그림 1.1-2] KT AI 보이스 스튜디오 서비스 구성도	-2-
[그림 1.1-3] 2017년 및 2021년 10세 단위 별 우울증 환자수 현황	-3-
[그림 1.1-4] 마이멘탈포켓-포키 서비스 화면	-4-
[그림 1.3-1] 앱 구조도	-6-
[그림 2.1.2-1] Mel-scale	-8-
[그림 2.1.2-2] Mel-spectrogram 예시	-9-
[그림 2.2-1] STT핵심 요소 기술	-10-
[그림 2.3-1] TTS 개요	-12-
[그림 2.3-2] Tacotron의 전체 구조	-15-
[그림 2.4-1] transformer 아키텍처의 다이어그램	-15-
[그림 2.5.1] 노이즈 어텐션 그래프 예시	-21-
[그림 3.1-1] 사전 학습 구성도	-23-
[그림 3.2-1] 서비스 사용 구조도	-24-
[그림 3.3-1] kaggle 데이터	-25-
[그림 3.4-1] 음성 파일 파형 예시	-28-
[그림 3.5.1-1] 구글 STT 설정	-30-
[그림 3.5.2.1] 학습용 text 파일 예시	-31-
[그림 3.6.1-1] Tacotron2 아키텍처 블록 다이어그램	-32-
[그림 3.6.3-1] WaveNet 아키텍처	-36-
[그림 3.6.3-2] WaveNet에서의 음성 데이터 생성	-37-
[그림 3.6.4-1] WaveGlow 아키텍처	-38-
[그림 3.6.4-2] Mean Opinion Scores	-40-
[그림 3.6.4-3] 추가 Mean Opnion Scores	-40-
[그림 4.1.2-1] 잡음제거 톨 이용 잡음 제거	-43-
[그림 4.1.2-2] 음성 파일 분할	-44-
[그림 4.2-1] 구글 STT 라이브러리를 이용, 분할된 wav파일의 대본 생성 코드	-45-
[그림 4.3-1] Tacotron2 학습환경	-46-
[그림 4.3-2] Tacotron main train loop 코드	-48-
[그림 4.3-3] WaveGlow main train loop 코드	-49-
[그림 4.4-1] 리눅스 웹 서버 환경	-50-
[그림 4.4-2] flask 웹 서버 코드	-52-
[그림 4.5-1] 안드로이드 앱 순서도	-53-
[그림 4.5.1-1] LoginActivity 코드	-54-

[그림 4.5.2-1] MainActivity의 onCreate() 함수	-55-
[그림 4.5.2-2] initModel 함수	-56-
[그림 4.5.2-3] DetailedActivity 일부 코드	-57-
[그림 4.5.2-4] DetailedActivity 일부 코드2	-58-
[그림 4.5.2-5] sendReq 함수	-59-
[그림 4.5.2-6] processText 함수	-60-
[그림 4.5.2-7] send 함수	-61-
[그림 5.2-1] KSS 데이터셋 학습 결과2	-64-
[그림 5.2-2] KSS 데이터셋 train 과정에서의 노이즈 attention 그래프	-65-
[그림 5.2-3] KSS 데이터셋 validation 과정에서의 노이즈 attention 그래프	-66-
[그림 5.2-4] KSS 데이터셋 inference 노이즈 attention 그래프	-66-
[그림 5.2-5] 김준석 데이터셋 학습 결과2	-68-
[그림 5.2-6] 김준석 데이터셋 train 과정에서의 노이즈 attention 그래프.....	-69-
[그림 5.2-7] 김준석 데이터셋 validation 과정에서의 노이즈 attention 그래프.....	-69-
[그림 5.2-8] 김준석 데이터셋 inference 노이즈 attention 그래프	-70-
[그림 5.2-9] 문재인 전 대통령 데이터셋 학습 결과2	-72-
[그림 5.2-10] 문재인 전 대통령 데이터셋 train 과정에서의 노이즈 attention 그래프.....	-73-
[그림 5.2-11] 문재인 전 대통령 데이터셋 validation 과정에서의 노이즈 attention 그래프.....	-73-
[그림 5.2-12] 문재인 전 대통령 데이터셋 inference 노이즈 attention 그래프.....	-74-
[그림 5.2-13] 최종 완성 애플리케이션 1	-76-
[그림 5.2-14] 최종 완성 애플리케이션 2	-77-
[그림 5.2-15] 최종 완성 애플리케이션 3	-78-
[그림 5.2-16] 최종 완성 애플리케이션 4	-79-

표 목차

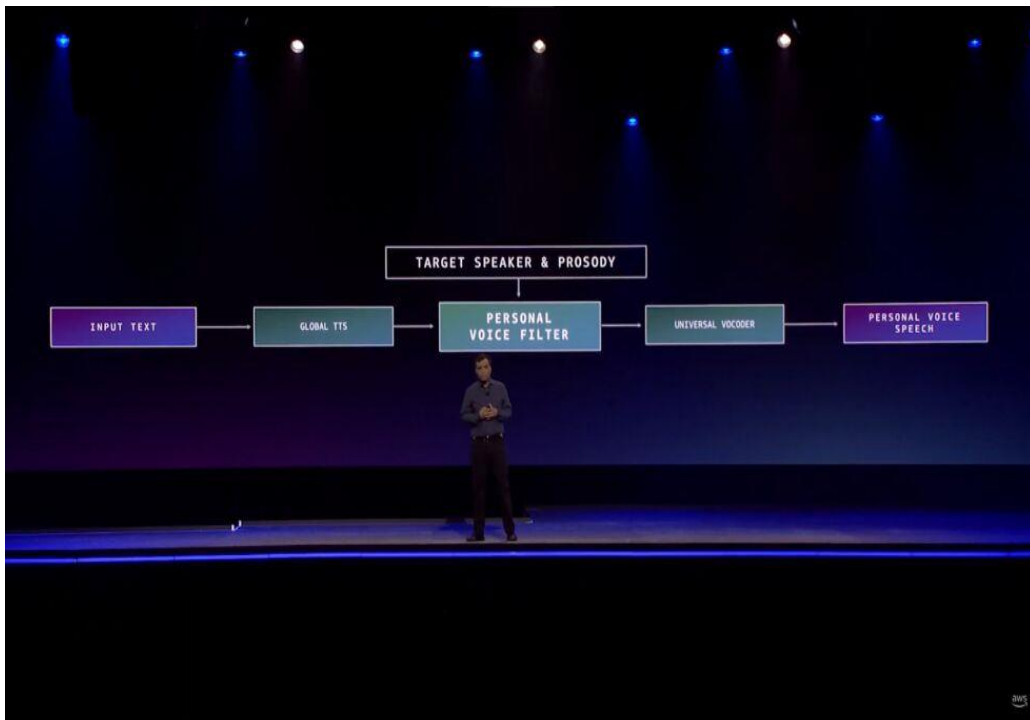
[표 2.4-1] GPT-3 학습 데이터.....	-18-
[표 2.4.4-1] KoGPT 파라미터	-21-
[표 2.4.4-2] 3개 모델 스펙 비교	-22-
[표 4.1.1] 수집 혹은 직접 녹음한 데이터 셋	-42-
[표 5.2-1] KSS 데이터셋 학습 결과1	-63-
[표 5.2-2] 김준석 데이터셋 학습 결과1	-67-
[표 5.2-3] 문재인 전 대통령 데이터셋 학습 결과1	-71-

1. 서론

1.1 연구 배경

기술력이 발전함에 따라 다양한 기업들은 직접적인 만남이 현실적으로 힘든 유명인이나 이미 더 이상 만날 수 없는 고인의 목소리를 음성 합성 기술을 이용하여 원하는 문장을 읽게 만드는 서비스를 선보이고 있다. 대표적인 예시로는 Amazon re:MARS 2022에서 공개한 고인의 목소리를 들을 수 있는 서비스와 30개의 예시 문장을 녹음하는 것으로 자신의 목소리와 닮은 AI 음성을 만들어주는 KT의 AI 보이스 스튜디오 그리고 유명한 유튜버나 연예인의 목소리를 이용하여 노래를 부르는 diff-svc(Singing Voice Conversion)서비스 등이 있다.

관련 예시1. Amazon re:MARS 2022



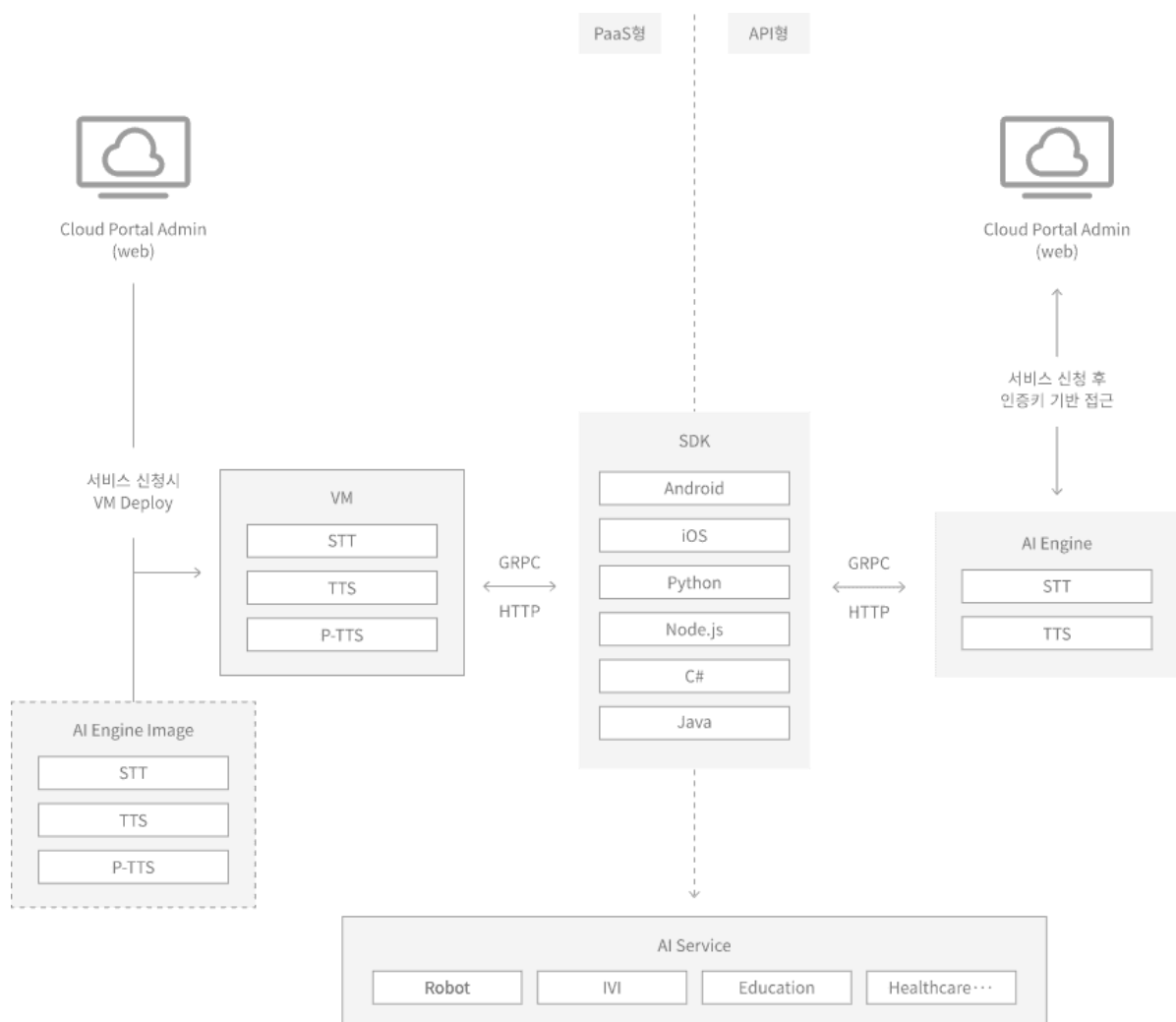
[그림 1.1-1] Amazon re:MARS 2022 캡처본

출처:

https://www.youtube.com/watch?time_continue=62&v=yG2PjzUknsU&embeds_referring_euri=https%3A%2F%2Fbrunch.co.kr%2F%40iotstlabs%2F252&source_ve_path=Mjg2NjlsMTM5MTE3LDEzOTExNywzNjg0MiwyMzg1MQ&feature=emb_title

2022년 6월 21일부터 6월 24일까지 라스베이거스에서 열린 Amazon re:MARS 2022 콘퍼런스에서는 폴란드 그단스크에 위치한 아마존의 TTS 연구소에서 개발한 새로운 기술을 선보였다. 이는 음성 비서 알렉사가 손자의 할머니 음성을 재현하는 기술로 적은 분량의 오디오 샘플을 이용하여 할머니 음성을 만들어 낸다. 개발된 시스템은 텍스트를 말소리로 변환하는 1차적인 작업을 진행한 후, 오디오 샘플을 이용하여 훈련된 보이스 필터를 통해 억양이나 음성, 강세 등에서 목표 발화자와 유사하게 들리도록 조정한다. 하지만 이 기술은 실시간으로 말을 주고받으며 상호 대화를 할 수 없다는 한계점이 있다.

관련 예시2. KT AI 보이스 스튜디오

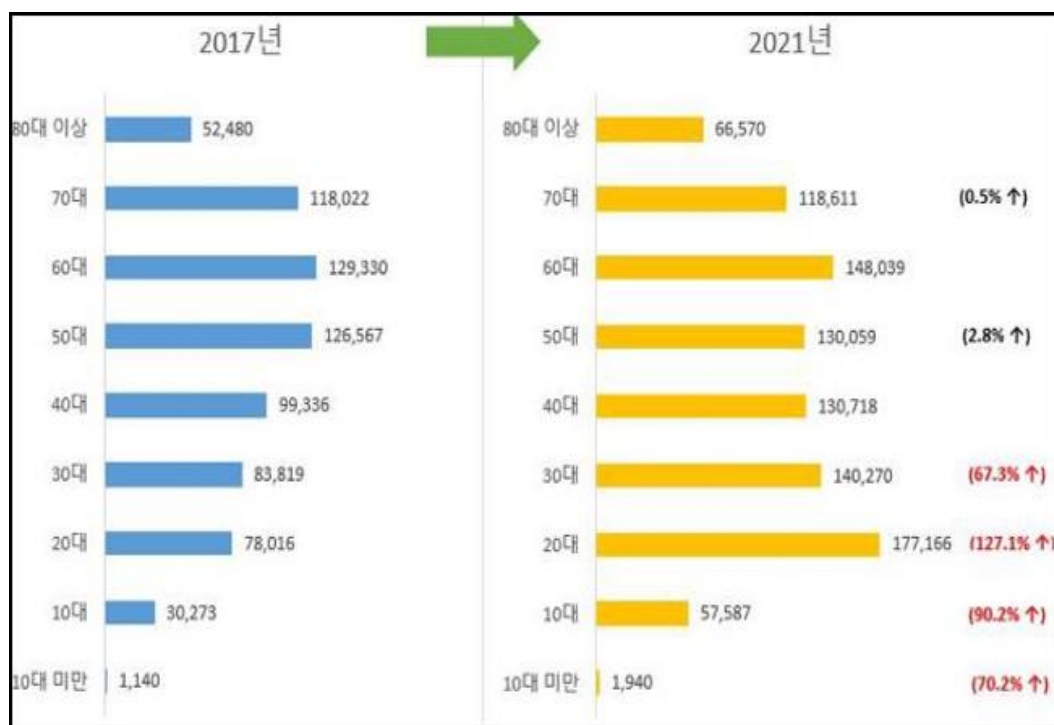


[그림 1.1-2] KT AI 보이스 스튜디오 서비스 구성도

출처: KT AI 보이스 스튜디오 서비스, https://cloud.kt.com/product/aiapi/genie_voice/

KT의 AI 보이스 스튜디오에서 제공하는 KT의 인공지능 보이스의 차별점은 '감정 더빙'이다. 목소리를 합성할 때 목소리의 크기와 톤 등을 통하여 사용자의 감정을 분석하여 합성한다. 원하는 스크립트를 선택하고 주어진 30개의 문장을 읽은 후 귀여움, 친절함, 다정함 등 원하는 분위기를 선택하면 AI가 녹음된 감정에 맞는 목소리를 합성하여 출력하는 방식이다. 또한 KT의 인공지능 보이스는 감정 구현뿐만 아니라 외국어 기능 또한 지원한다. 한국어만 녹음해도 중국어, 스페인어, 일본어 영어 등 다양한 언어로 말하는 자신의 목소리를 들을 수 있다.

앞서 설명한 알렉사와 KT AI 보이스 스튜디오 서비스들은 실시간으로 대화서비스를 진행하지 못한다는 점에서 한계점이 발생한다. 또한 이 서비스들은 주어진 문장을 단순히 읽어내는 것에 그친다. 본 연구는 음성 합성 딥러닝 기술에 대규모 언어 모델을 추가적으로 도입하여 단순히 기존에 입력된 문장을 읽을 뿐만이 아니라 실시간으로 대화할 수 있도록 하는 음성 대화 서비스를 제안한다.

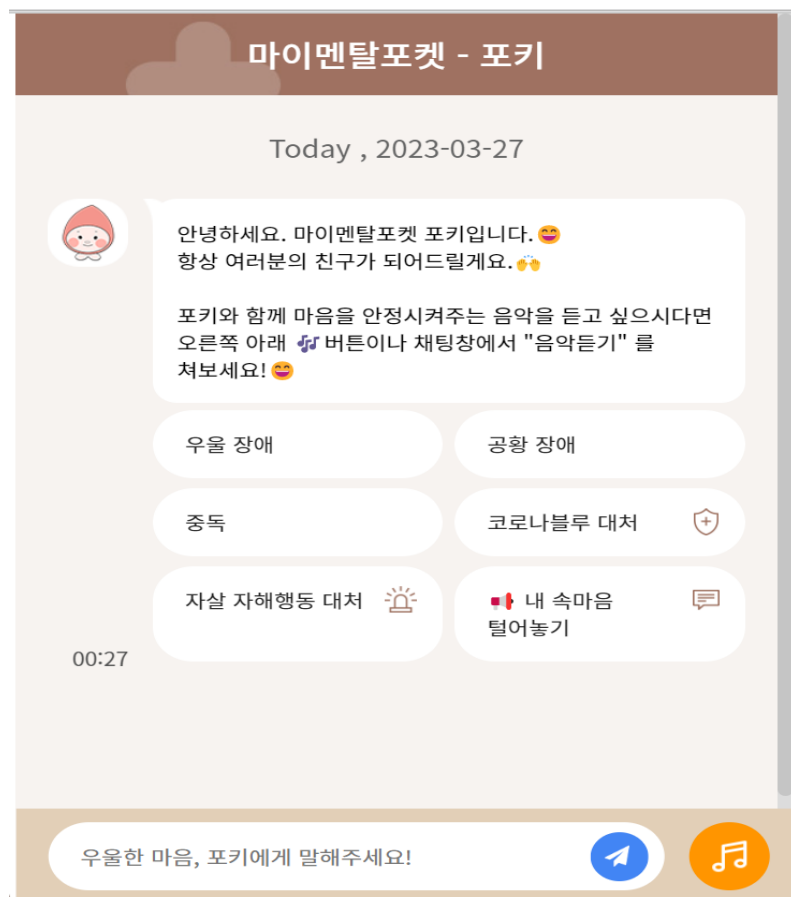


[그림 1.1-3] 2017년 및 2021년 10세 단위 별 우울증 환자수 현황

(건강보험심사 평가원이 2017년~2021년 동안 분석한 우울증과 불안장애 진료 통계 분석 결과)

출처: <https://www.medicaltimes.com/Main/News/NewsView.html?ID=1148087>

현대 사회에서는 기술의 발전으로 인하여 우리는 물질적 풍요는 늘었지만 그에 반하여 정신적 피로가 증가하고 있다. 근무환경, 거주지, 인간관계 등이 다양해지면서 정신적 피로와 함께 우울증이 증가하고 있는 것이다. 특히 어린이 보호, 간병인, 독거 노인, 1인 가구와 같이 직업이나 거주 환경 특성상 정신적 피로를 크게 겪거나 소통 기회가 적은 대상들은 우울증을 겪을 확률이 매우 높다. 이러한 사람들에게 AI 서비스를 제공하여 실시간으로 대화 상대 역할을 해준다면 심리적 고통을 완화함으로써 정신적 피해를 경감시키고 자살 예방과 같은 기대 효과를 누릴 수 있다.



[그림 1.1-4] 마이멘탈포켓-포키 서비스 화면

실제 이와 관련된 기술로 우울증과 심리 상담 역할을 수행하는 텍스트 기반 AI기술인 마이멘탈포켓 서비스가 존재하지만 이 서비스는 트리형 챗봇 구조를 가지고 있어 정해진 답변만 출력할 수 있으며 진단에 있어서는 보조적인 역할만 수행이 가능하다. 그에 반해 본 연구에서는 음성합성 기술을 이용한 실시간 대화가 가능한 서비스를 제공하고자 한다.

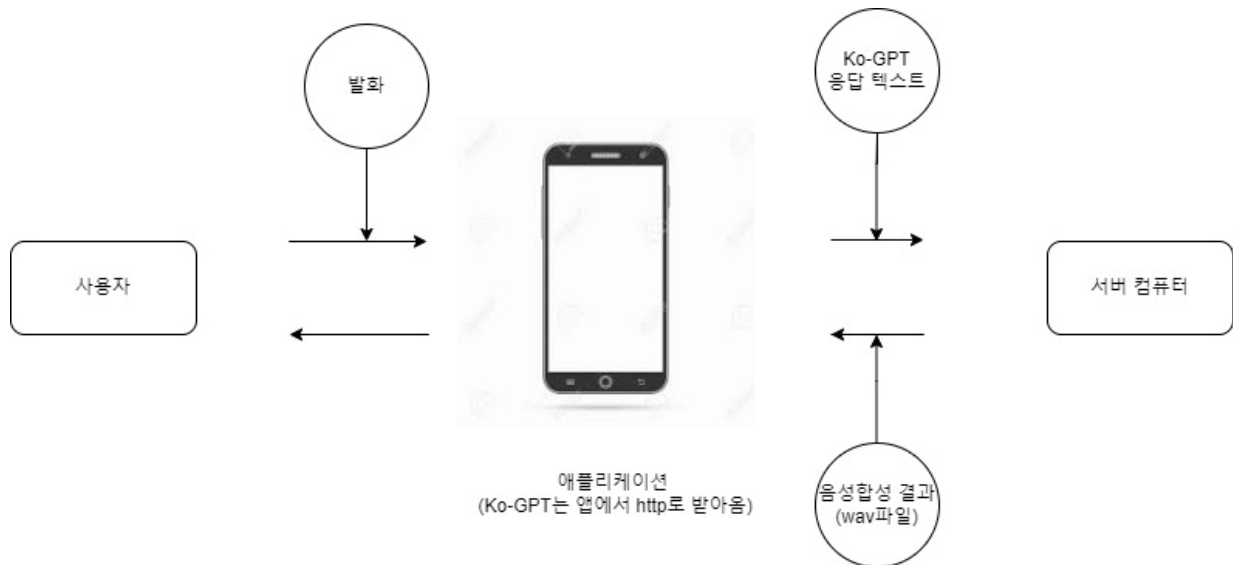
1.2 연구 목표

본 연구의 최종 목표는 원하는 사람의 목소리가 담긴 녹음 파일을 이용하여 Deep-learning 시킨 음성합성 모델과 대규모 언어모델을 바탕으로 실시간 대화가 가능한 서비스를 제공하고자 한다. 서비스 이용자에게 원하는 사람의 음성 녹음 파일을 제공받아 음성 합성 모델 Tacotron2의 학습을 진행한다. 서비스를 이용하는 사람은 안드로이드 어플리케이션을 통하여 듣고자 하는 사람의 목소리를 선택한 후 발화를 진행하게 되면 그 발화는 Speech-to-Text를 통해 텍스트 문자로 변형되어 언어 모델인 Kogpt의 서버로 전달된다. Kogpt는 적절한 응답을 생성한 후 이를 음성합성 모델 Tacotron2에게 전달하여 음성으로 변환한 후 이용자에게 어플리케이션을 통하여 출력하는 것으로 실시간 음성대화 서비스를 구현하고자 한다.

1.3 연구 내용 및 방법

본 연구에서는 3개의 음성 데이터셋을 이용하여 합성된 음성의 음질을 비교한다. 음성 합성의 음질은 overfitting의 발생 여부, 만약 발생하였다면, overfitting이 발생한 이유에 대해 분석을 진행하며, 오버 피팅이 발생한 경우, 합성된 wav파일의 음질과 attention 그래프를 비교하며, 어떤 이유에서 오버 피팅이 발생했는지 분석한다.

그 후 각 음성 합성 모델을 이용하여, 서비스를 제공해주는 애플리케이션을 개발한다.



[그림 1.3-1] 앱 구조도

[그림 1.3-1]은 본 연구에서 결과물로 제공된 앱의 작동과정을 표현한 그림이다. 사용자의 발화를 앱으로 받으면 앱과 http통신으로 연결된 Ko-GPT 모델을 이용하여, 사용자 발화에 대한 응답 텍스트를 생성한 후 그 응답 텍스트를 서버 컴퓨터로 넘겨 서버 컴퓨터에 들어있는 음성 합성 모델을 통해 결과물 wav 파일을 다시 핸드폰으로 전송해 앱 상에서 wav파일을 재생하는 형식으로 진행된다.

연구는 데이터셋에 중점을 두고 진행한다. 이유는 음성 합성에서 이용할 데이터에서 가장 중요한 부분은 방대한 데이터 셋과 음성 데이터의 품질이기 때문이다. 따라서, 방대한 양의 전문 성우 녹음을 이용하여 합성을 진행 후, 일반적인 사람의 목소리가 담긴 상황을 가정하여, 조원 한 명의 목소리를 합성하고, 마지막으로 유명인의 목소리가 담긴 녹음 파일을 제공하는 상황을 가정하여 합성을 진행한다. 그 후 각각의 모델에서 train, validation, test 과정에서 합성된 wav파일을 청취하고, 2.5절에서 설명할 노이즈 어텐션 그래프와 wav파일을 비교하여 합성이 잘된 경우와 아닌 경우를 비교하는 방식으로 진행한다.

2. 관련기술동향 조사 및 분석

2.1 음성 스펙트럼 분석

음성 처리 분야에서 기본이 되는 음성 스펙트럼 분석은 음성 신호를 다양한 주파수를 가지는 주기함수들로 분해하여 보여주는 방법인 스펙트럼을 이용하여 신호를 분석하는 것을 말한다. 모든 음향 신호는 단순파들의 합으로 표현할 수 있는데, 이를 스펙트럼으로 표현함으로써 어떤 단순파들의 합으로 이루어져 있는지 분석이 가능하다. 이 스펙트럼은 푸리에 변환을 통해 얻을 수 있다.

2.1.1 푸리에 변환

푸리에 변환(Fourier Transform)은 주파수 영역으로 신호를 변환하는 수학적 기법 중 하나이다. 이 변환이 사용되는 주된 목적은 시간 영역에서의 신호를 주파수 영역으로 변환하여 신호의 주파수 성분을 분석하는 것이다.

푸리에 변환은 다양한 분야에 응용되는데, 신호 처리, 통신, 영상 처리, 오디오 분석 등에서 주파수 성분을 분석하거나 필터링하는 데 자주 사용된다[1].

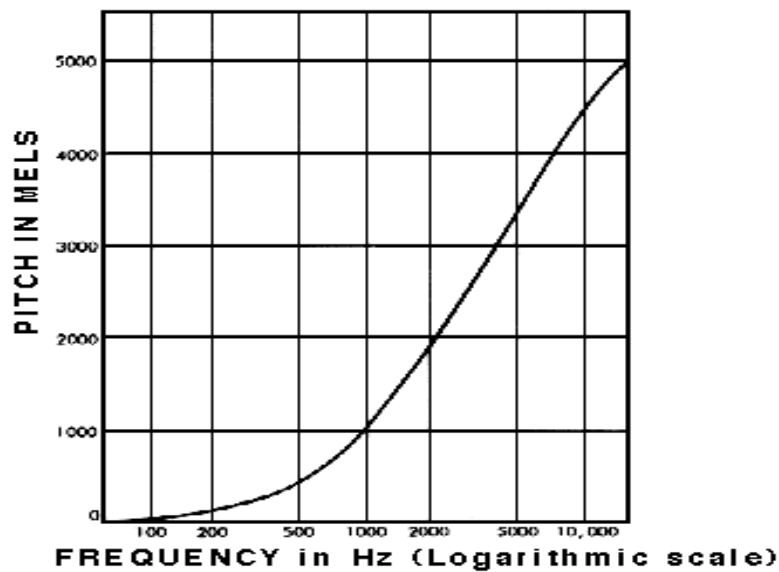
음성 신호를 분석하기 위해서는 시간에 따른 주파수 변화를 분석하는 것이 중요하다. 하지만 음성 신호에 푸리에 변환을 사용할 경우 시간 정보가 손실되어 각각의 주파수 성분이 언제 존재하는지 알 수 없어 음성 처리에 이용하는 데 적절하지 않다. 시간 정보를 보존하기 위해서는 STFT(Short Time Fourier Transform)를 이용한다. STFT란 아주 짧은 시간 단위로 신호를 잘라서 푸리에 변환을 해준 것을 말한다. 그리고 변환된 Spectrum을 모두 모은 것을 Spectrogram이라고 한다.

2.1.2 Mel Spectrogram

인간은 1kHz 이하에서 민감한 청각적 특성을 가진다. 사람은 높은 주파수보다 낮은 주파수에서의 차이를 더 쉽게 감지할 수 있는데 예를 들어 10,000Hz와 10,500Hz의 차이는 거의 구분할 수 없지만 500Hz와 1000Hz의 차이는 쉽게 구분할 수 있다.

Mel-spectrogram은 푸리에 변환을 통해서 주파수 영역으로 변환시킨 음성 신호를 위와 같이 인간이 더 쉽게 주파수의 변화를 인식할 수 있는 대역으로 변환한 것을 의미한다. 그리고 이에 맞춰 변환한 주파수 단위를 mel-scale이라고 부른다. Mel-scale로 변환하기 위해서는 주파수에 대하여 (3)과 같은 수학적 연산을 수행해야 한다[2].

$$\text{Mel}(f) = 2595 \log \left(1 + \frac{f}{700} \right) \quad (3)$$

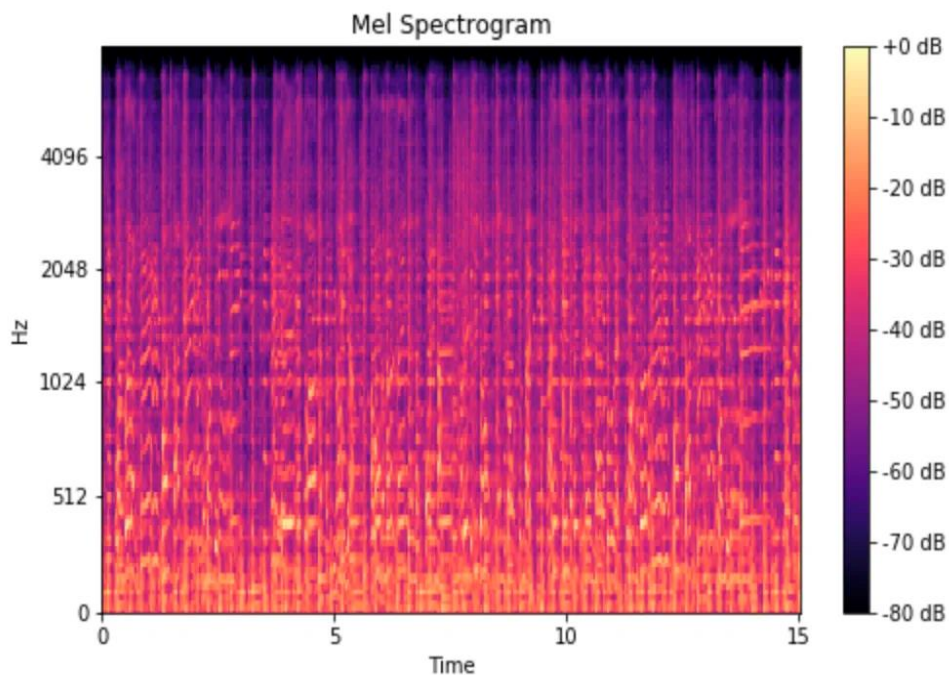


[그림 2.1.2-1] Mel-scale

출처: <https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53>

주파수 단위를 공식 (3)을 사용하여 mel 단위로 변환한 spectrogram을 mel-spectrogram이라고 한다. [그림 2.1.2-1]은 mel-scale과 주파수의 관계를 나타낸 그래프이다.

또한 음성데이터를 raw data 그대로 사용하게 되면 데이터 용량이 너무 커지고 파라미터가 너무나 많아진다. 그리고 사람들은 앞서 말했듯이 사람은 음성 신호에서 높은 주파수는 낮은 주파수보다 제대로 인식하여 받아들이지 못하기 때문에[2] 위 두가지 이유로 음성 신호를 분석할 때는 mel-spectrogram을 사용한다.

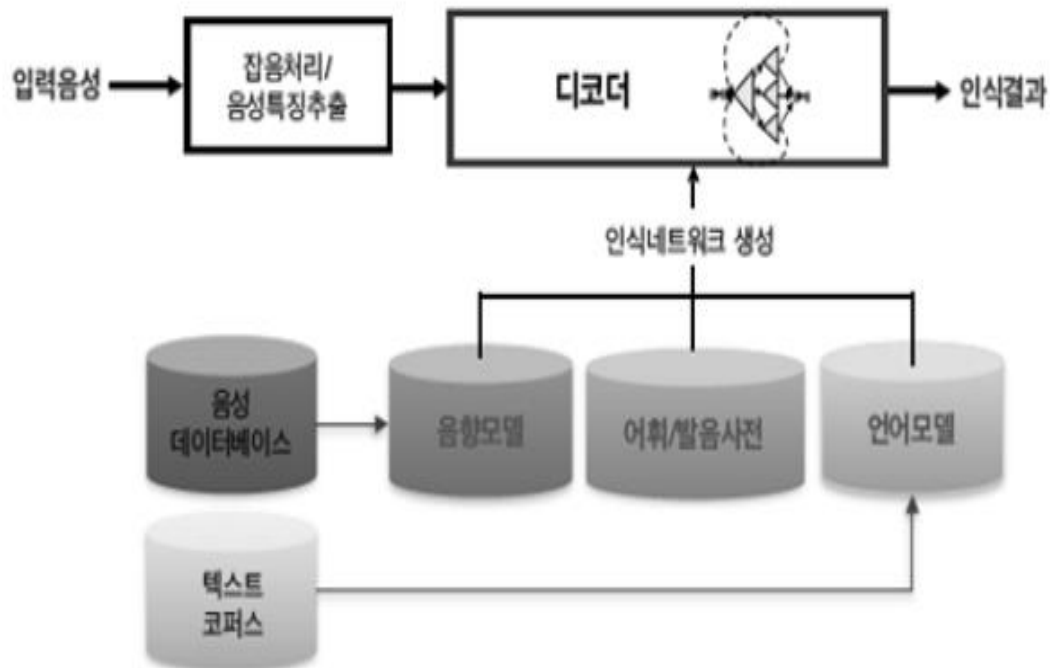


[그림 2.1.2-2] Mel-spectrogram 예시

출처: <https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53>

[그림 2.1.2-2]는 mel-spectrogram의 예시이다. Mel-spectrogram을 보면 가로는 시간, 세로축은 주파수에 관한 정보가 들어있는 것을 확인 가능하며, 위 mel-spectrogram은 뒤에 3.6.4절에서 설명할 WaveGlow의 input으로 사용된다.

2.2 STT



[그림 2.2-1] STT핵심 요소 기술

출처: <https://woongsin94.tistory.com/333>

STT(Speech-To-Text)는 사람의 음성을 텍스트 데이터로 추출하는 기술이다. STT를 위해 사용되는 데이터는 언어학 관점과 음향학 관점으로 나뉘어 볼 수 있다. 언어학 관점에는 어휘, 문법, 문맥 등 모델링에 필요한 언어 데이터가 있으며 음향학의 관점에는 발화자, 노이즈, 공간 등의 환경적인 데이터가 있다[3].

STT는 또한 2가지의 단계로 나눌 수 있다. [그림 2.2-1]을 참고하면, 음성/언어 데이터로부터 인식 네트워크 모델을 생성하고 그 모델의 결과와 발화자의 음성을 토대로 인식 결과물 즉, 음성의 대본을 생성하는 단계로 나뉜다.

STT의 엔진은 언어 데이터와 음성의 사전 지식을 사용하여 음성 신호로부터 문자 정보를 출력하고 이를 디코더(Decoder)라는 이름의 STT 알고리즘을 통하여 해석한다. 학습 단계 결과인 언어 모델과 어휘/발음 사전, 음향 모델을 이용하여 디코딩 단계에서는 입력된 특징 벡터를 모델과 스코어링(Scoring)을 비교하여 단어 열을 최종적으로 정하게 된다.

STT는 대부분 확률 통계 방식인 HMM(Hidden Markov Model)기반으로 이루어져 있다. 2010년대 이후부터는 딥러닝 기반인 HMM/DNN 방식을 통하여 단어 인식 오류를 개선하는 것으로 20%의 성능 향상이 이루어졌다[4].

최근에는 Sequence-to-Sequence 방식의 RNN(Recurrent Neural Network) 기반으로 성능과 속도 면에서 뛰어난 결과를 가져왔으며 End-to-End 학습 방식의 발전으로 음성 인식면에서도 오디오 특징을 입력하는 것으로 글자와 단어들을 출력으로 하는 단일 함수를 학습할 수 있게 되었다. 이처럼 다양한 학습법이 발달하여 STT의 성능은 향상됨과 동시에 다양한 서비스와 플랫폼에서 접할 수 있는 기술이 되었다[5].

STT의 경우, 사용자가 제공한 음성파일의 대본을 생성하기 위해 사용한다. STT는 사용자의 대화를 실시간으로 텍스트로 번역해주는 모델이다. STT는 2.3에서 설명할 TTS 그리고 본 연구에서 이용할 음성 합성 모델인 tacotron2, waveglow의 모델 학습 파일을 만들기 위해 사용된다. 음성 합성 모델은 기본적으로 학습에 이용되는 대본을 통해, wav파일에서 글자의 발음과 대본에서 단어의 관계를 학습하는 방식인데, 본 연구에서는 음성파일의 대본이 없다는 상황을 가정 즉, 사용자가 원하는 사람의 목소리 파일만을 모델에 제공해주는 상황이기 때문에, STT를 통해 직접 대본을 추출하는 과정이 필요하여 STT를 본 연구의 학습파일을 만드는 과정에서 이용하기로 하였다.

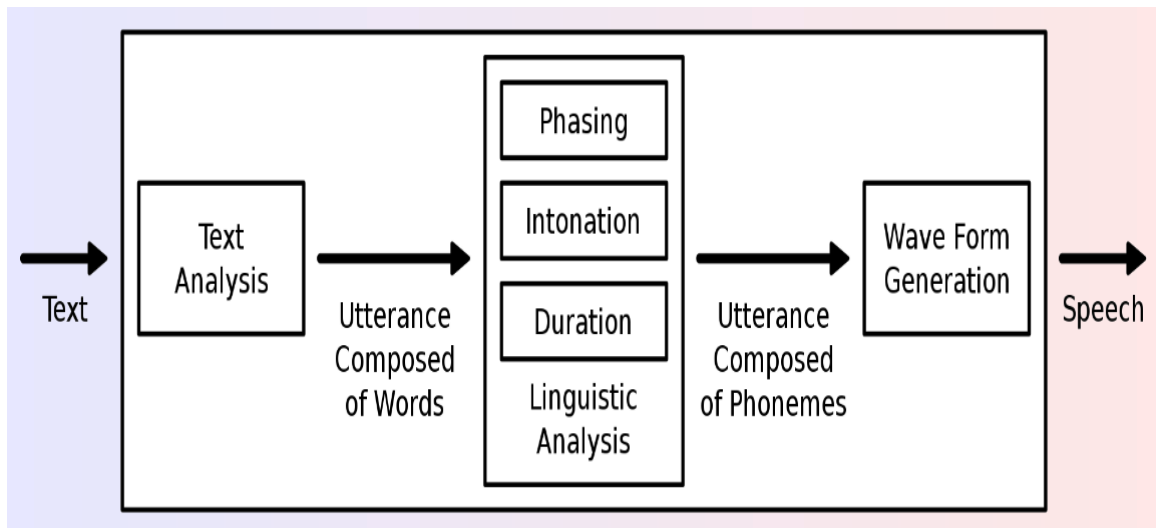
현재 사용 가능한 STT API 종류로는 10가지가 존재한다. Google 클라우드 음성 API(이하 구글 STT), IBM Watson Speech to text, Microsoft Azure Bing 음성 API, Amazon Transcribe, Amazon Polly, VoxSigma api, Twilio 음성인식, Speechmatics ASR, Nexmo Voice Api, Naver Clova 총 10개이며, 본 연구에서는 구글 STT API를 사용하기로 하였다.

구글 STT API를 사용하는 이유로는 크게 3가지로 첫 번째, 원하는 옵션을 선택할 수 있다. 학습 파일 상황에 맞춰 옵션을 추가하거나 제거할 수 있어, 유연성이 올라갈 수 있을 것이라고 판단했다.

두번째 이유로는 실시간으로 사용이 가능하기 때문이다. 기획한 서비스(애플리케이션)은 사람(서비스 이용자)로부터 사람의 목소리가 들어간 wav파일을 제공받고 그 사람의 목소리로 학습시켜 대화를 시키는 서비스이기 때문에, 실시간으로 사용이 가능하다는 점도 중요하다고 판단하였다.

마지막 이유는 오디오의 외부 잡음에 대해 안정적이라는 점이다. 외부 잡음에 안정적이라는 말은, 잡음이 있어도 일정 수준의 성능이 나온다는 의미로 이번 연구에서 STT라이브러리를 쓰는 이유는 결국 사용자로부터 받은 오디오 파일의 번역본 즉 대본을 작성한다는 점이다. 따라서 사용자 wav파일의 잡음이 많으면 번역이 제대로 되지 않는 경우가 발생하며, 그에 따라 음성 합성의 품질 또한 낮아질 것으로 생각했다. 따라서 오디오 노이즈에 안정적이라는 옵션 또한 중요하다고 생각했다. 위 세가지 이유로 이번 연구에서는 구글 STT라이브러리를 사용한다[6].

2.3 TTS



[그림 2.3-1] TTS 개요

출처: https://ko.wikipedia.org/wiki/%EC%9D%8C%EC%84%B1_%ED%95%A9%EC%84%B1

TTS(Text-To-Speech)는 디지털 장치나 컴퓨터가 사람의 언어인 텍스트를 인식하고 이를 인위적으로 사람의 소리로 합성하여 발음할 수 있게 하는 기술이다. 텍스트 음성 변환 시스템은 프론트엔드와 백엔드로 구성되며 프론트엔드는 사용자가 입력으로 준 텍스트를 분석하여 기호화된 언어 표현을 생성한다. 본 연구에서는 사용자의 입력 텍스트는 STT를 통해 생성된 음성 파일의 대본을 이용하며 기호화된 언어 표현으로는 언어의 시간 영역과 주파수 영역을 분석한 mel-spectrogram을 사용한다. 백엔드는 합성된 음성의 파형을 내보낸다.

프론트 엔드는 텍스트의 토큰화, 정규화 등으로 불리기도 하는데, 이 단계에서는 각 단어를 발음 기호로 변환시킨 후 단어나 문장, 텍스트 속어, 문장 등 운율 단위로 분할한다. 운율 정보와 발음 기호를 조합하여 기호화된 언어 표현 즉 본 연구에서는 mel-spectrogram을 생성하여 백엔드로 내보낸다.

백엔드는 프론트엔드가 내보내는 결과를 통하여 더 자연스러운 음성을 생성하기 위해 운율 등의 음성을 조정하여 실제 음성 데이터를 출력한다. 이때 음성의 특성이 정해져 있어 음성 합성 소프트웨어에 따라 음성이 원래 사람의 목소리와 조금 다르게 생성될 수 있다.

TTS의 초기 시스템은 음성의 품질과 자연스러움에 한계가 있었지만 최근 딥러닝과 같은 AI 기술의 발전에 의하여 사람의 목소리와 구분이 힘든 정도의 수준까지 발전했다. TTS는 오디오 북, 음성 비서, 자동 전화 응답 시스템 등 다양한 분야에서 활용되고 있으며 미래에는 감정 표현 능력의 향상과 함께 더욱 더 많은 언어를 지원할 것으로 예상된다.

음성 합성 시스템은 초기에 규칙 기반 방식을 주로 사용했는데 이에는 포먼트 합성기(1세대)와 linear predictive coding(선형 예측 부호화) 기반의 음성 합성기 등이 있다. 세대가 지남에 따라 합성된 음성의 발음은 더 정확해졌지만, 합성된 음성이 자연스럽지 않다는 한계점이 존재하여 규칙 기반 방식은 점점 인기가 없어졌다. 하지만 컴퓨터의 메모리 증가와 연산 속도 증가로 데이터 기반의 방식이 3세대부터 많이 연구되었다.

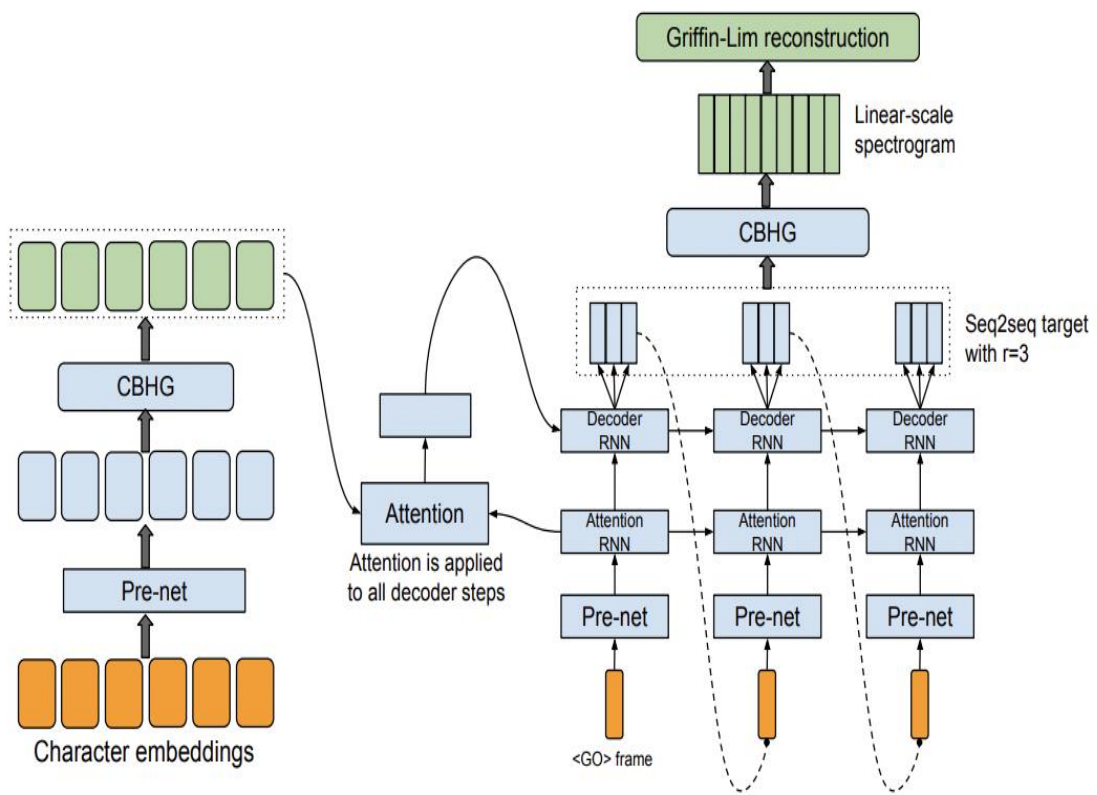
음성합성 분야는 굉장히 오래되었고 여러가지 방식이 존재한다. 그 중에서 상용화되어 있는 주류 방식에는 Unit-selection방식과 Statistical Parametric방식이 있다. Unit-selection 방식은 음편선정 방식이라고도 불리며 음편들을 짧은 단위로 저장한 후 합성음을 출력할 때 텍스트에 해당하는 것들을 선택하여 연결하여 출력한다. 음질은 실제 사람의 음성을 녹음한 것을 사용하기 때문에 훨씬 좋다는 장점이 있지만 많은 데이터를 사용한다는 점과 연결한 두 음편 사이에 경계가 있는데 이 경계가 부자연스러우며 주어진 문장을 이용하여 발화를 진행할 때 항상 똑같은 발화만을 하게 된다는 문제가 존재한다.

이러한 문제점 및 한계점을 극복하고자 제시된 방식이 바로 Statistical Parametric방식(통계적 파라미터 방식)이다. 이 방식에는 마르코프 모델(HMM)기반인 TTS 시스템(HTS, HMM-based speech synthesis)가 있으며 화자의 목소리 변환 및 파라미터를 조절하는 방식을 사용하여 음성 합성을 할 때 감정이 들어가는 등 다양한 음성합성이 가능하다.

또한 최근 심층 신경망(deep neural network)을 활용한 연구가 결과가 성능을 크게 향상시켜 음성 인식, 기계 번역 등 다양한 분야에서 나타나면서 DNN 기반의 연구결과들이 음성 합성에도 등장하고 있다.

현재 Statistical Parametric 방식 중 딥러닝을 사용한 방식이 엄청난 성과를 보여주었으며 그 중 딥러닝 방식에는 Tacotron, Wavenet 등이 있다. Tacotron은 Acoustic Model로 text를 mel-spectrogram으로 바꿔주는 역할을 수행하며 Wavenet은 Vocoder로 실제 음성인 Waveform 형태로 mel-spectrogram을 바꿔준다[6].

Tacotron은 Encoder-Decoder 형식의 모델로 문자열을 입력하면 프레임 단위로 그에 해당하는 mel-spectrogram을 출력해준다[7].



[그림 2.3-2] Tacotron의 전체 구조

출처: <https://seastar105.tistory.com/157>

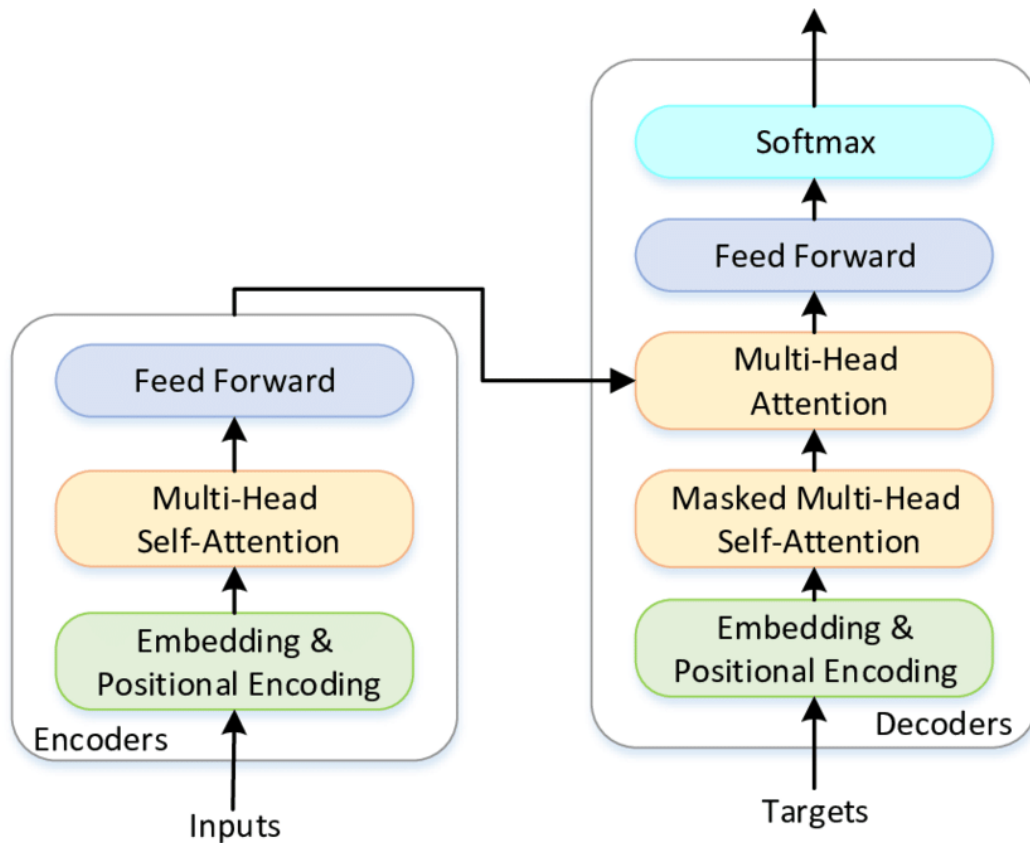
[그림 2.3-2]은 Tacotron 모델의 구조를 보여준다. Encoder 부분은 pre-net, CBHG 스택으로 이루어지며, 스펙트로그램을 파형으로 변환하는 단계에서는 신경망 기반이 아닌 Griffin-Lim 알고리즘을 이용한다. Tacotron은 end-to-end 모델로 audio-text pair(3장에서 후술할 학습 txt 파일)만 있으면 학습이 가능하다. 하지만 단점 또한 존재하는데 합성 속도 적인 측면에서는 느리다는 문제점이 존재한다. 그 이유는 auto-regressive한 방식으로 frame 별로 생성하기 때문이다. Alignment를 직접 학습하는 모델 특성상 generalization의 성능이 훈련시의 alignment가 좋으면 향상되지만 alignment를 시각화 했을 때 중간이 반복되거나 끊어지기 쉽다는 단점 또한 존재한다.

하지만 우리가 TTS 시스템으로 Tacotron에 기반한 end-to-end를 사용한 이유는 end-to-end 합성 방식은 음향 모델링 파트, 텍스트 분석 파트, 음성 합성 파트에 대하여 기존의 방식과 달리 전문적인 지식 없이도 구현이 가능하며 진입 장벽이 낮다는 장점이 있기 때문이다. 이를 통해 훈련에 있어 효율적으로 진행이 가능하다. 이는 다양한 음성 합성에 있어 텍스트 분석 파트를 사용하지 않아 적용이 용이하며 문자의 조합에 있어 평소에 자주 쓰지 않는 우리에게 있어 낯선 문자가 있어도 모델링이 가능하다는 장점이 존재하기 때문이다[7].

2.4 대형언어모델(LLM)

대형 언어 모델 또는 거대 언어 모델은 수많은 파라미터를 보유한 인공지능망으로 구성되는 언어모델이다. 자기 지도 학습이나 반자기지도학습을 사용하여 레이블링되지 않은 상당한 양의 텍스트로 훈련된다. LLM 은 2018 년 즈음에 모습을 드러냈으며 다양한 작업을 위해 수행된다. 대규모 언어 모델의 작동방식은 크게 토큰화, 트랜스포머 모델, 프롬프트의 3 가지로 나뉘고 있다.

토큰화는 자연어 처리의 일부로 일반 인간 언어를 기계 시스템이 이해할 수 있는 시퀀스로 변환하는 작업을 말하며 여기에는 각 토큰에 숫자 값을 할당하고 빠른 분석을 위해 인코딩하는 작업이 수반된다.



[그림 2.4-1] transformer 아키텍처의 다이어그램

출처: https://www.researchgate.net/figure/Simplified-diagram-of-the-Transformer-model_fig2_344911225

트랜스포머 모델은 BERT 나 GPT 와 같은 언어 모델의 기반이 되는 모델이다. 이 모델은 2017 년에 [8] 논문을 통해 소개되었으며 self-attention 이라는 매커니즘을 도입한다. 트랜스포머 모델은 이 self-attention 을 이용하여 순차적으로 데이터를 검사하여 어떤 단어가 서로 뒤따를 가능성이 높은지 관련 패턴을 식별하는 신경망의 일종으로 각각 다른 분석을 수행하여 어떤 단어가 호환되는지 결정하는 계층으로 구성된다[9].

[그림 2.4-1]에서 볼 수 있듯이 원래 논문에서 소개된 transformer 아키텍처는 encoder 와 decoder 의 메인 파트로 나뉘며 BERT 모델은 encoder 부분만 사용하며, GPT 는 decoder 부분만 사용한다. BERT 는 의미, 구문론적 언어 정보를 이해하기 위해 사용되며, 보통 임베딩 벡터의 형태로 출력이 생성된다. 따라서 BERT 는 텍스트 분류와 같은 작업에서 사용된다. 반면에 GPT 는 다음 단어를 생성하는데 사용되는 모델이다. 따라서 GPT 의 출력은 확률적 수치가 포함된 단어들이다.

이러한 transformer 를 이용한 GPT 와 같은 대규모 언어 모델은 AI 챗봇 기술을 가능하게 하는 요소이며 본 연구에서는 사용자와의 자연스러운 대화 즉, 사용자의 발화에 따른 적절한 응답을 생성하기 위해 이러한 대규모 언어 모델을 채택하게 되었다.

2.4.1 GPT-3

생성적 사전 학습 변환기 3(Generative Pre-trained Transformer 3), GPT-3 는 OpenAI 에서 만든 딥러닝을 이용한 대형 언어 모델이다. 이전 버전인 GPT-2 와 같이 비지도 학습과 생성적 사전 학습(generative pre-training)기법, transformer 를 적용해 만들어졌으며, 이 아키텍처는 2048 개의 토큰 길이 컨텍스트와 1,750 억 개의 파라미터 크기를 가진 디코더 전용 변환기 네트워크이다. 이 모델은 attention 매커니즘을 사용하며 이 attention 매커니즘은 모델이 다음 토큰을 예측할 때 가장 관련성이 높은 입력 텍스트의 한 부분에 선택적으로 집중할 수 있게 도와준다. 이렇게 하여 모델은 이전 토큰을 기반으로 다음 토큰이 무엇인지 예측하도록 훈련된다[10].

Dataset	Number of tokens	Proportion within training
Common Crawl	410 billion	60%
WebText2	19 billion	22%
Books1	12 billion	8%
Books2	55 billion	8%
Wikipedia	3 billion	3%

[표 2.4-1] GPT-3 학습 데이터

[표 2.4-1]은 GPT-3 훈련에 사용된 데이터셋 구성을 보여준다. GPT-3 에 대한 pre-training으로 사용된 데이터셋의 60%는 4,100억 byte-pair-encoded 토큰으로 구성된 Common Crawl 의 필터링된 버전이며, 다른 소스는 전체의 22%를 나타내는 WebText2 의 190 억 토큰, 8%를 나타내는 Book1 의 120 억 토큰, 8%를 나타내는 Book2 의 550 억 토큰, 3%를 나타내는 Wikipedia 의 30 억 토큰이다[10].

2020 년 6 월 11 일, OpenAI 는 사용자가 이 새로운 기술의 장점과 한계를 경험하는 데 도움을 주기 위해 사용자 친화적인 GPT-3 API 에 대한 액세스를 요청할 수 있다고 발표했다. 2022 년 1 월 27 일, OpenAI 는 GPT-3 의 fine-tuned 된 버전인 InstructGPT 가 API 에서 사용되는 기본 언어 모델이라고 발표했다. OpenAI 에 따르면 InstructGPT 는 지시를 더 잘 따르고, 조작된 사실을 더 적게 생성하며, 다소 덜 독성이 있는 콘텐츠를 생산함으로써 사용자 의도에 더 잘 맞는 콘텐츠를 생산한다[10].

InstructGPT 는 사람이 작성한 명령어 데이터 세트에 대해 훈련된 모델이다. 이 훈련을 통해 InstructGPT 는 요청되는 내용을 더 잘 이해하고 보다 정확하고 관련성 있는 출력을 생성할 수 있다. InstructGPT 는 다음과 같은 활용성을 지닌다.

- 자연어로 제공되는 instruction 을 따를 수 있다.
- 자연어로 질문하는 질문에 대답하는 데 사용될 수 있다.
- 명령을 따르고 질문에 답할 때 GPT-3 보다 더 정확하고 적절하다.
- 고객 서비스, 교육, 자동화 등 다양한 용도로 사용할 수 있다.

2022 년 3 월 15 일, OpenAI 는 API 에서 "text-davinci-002" 및 "code-davinci-002"라는 이름으로 편집 및 삽입 기능을 갖춘 GPT-3 및 Codex 의 새로운 버전을 제공했다. 이 모델들은 이전 버전보다 더 유능한 것으로 나타났으며 2021 년 6 월까지의 데이터를 이용하여 학습되었다. 2022 년 11 월 28 일, OpenAI 는 text-davinci-003 을 선보였다. 2022 년 11 월 30 일, OpenAI 는 이 모델들을 "GPT-3.5" 시리즈에 속한다고 언급하기 시작했고, GPT-3.5 시리즈의 모델에서 fine-tuned 된 ChatGPT 를 출시했다. OpenAI 는 GPT-3 에 GPT-3.5 를 포함하지 않는다[10].

상기한 GPT-3 계열 모델들은 자연어처리 분야에서 혁신적인 성능의 모델이지만 영어 기반의 task 만 수행할 수 있다. 본 연구는 한국어 기반 대화 시스템 제작에 목표를 두고 있기 때문에 한국어 응답을 생성할 수 있는 모델이 필요하다. 이를 위해 GPT-3 계열 모델을 한국어 데이터셋을 이용하여 fine-tuning 하는 작업을 고려해보았으나 시간적 그리고 물리적 여건상 불가능하다고 판단하여 이미 제작된 한국어 데이터셋 기반 모델을 모색하는 것으로 대체하였다.

GPT를 기반으로 한 한국어 모델은 활발한 연구가 진행 중이며 발표된 모델이 많지 않다. 본 연구는 발표된 모델 중 다음과 같이 3개의 모델을 추려 비교 분석을 하였다.

2.4.2 SKT-KoGPT2

SKT에서 발표한 한국어로 학습된 오픈소스 기반 GPT-2 모델인 KoGPT2는 질문에 대한 응답 생성, 문장 완성, 챗봇 등 한국어 해석이 필요한 여러 애플리케이션의 머신러닝 성능을 향상시키기 위해 만들어졌다. KoGPT2는 챗봇 구축, 텍스트 감성 예측, 텍스트 분석 기반 응답 생성에 사용될 수 있다[11].

KoGPT2는 부족한 한국어 성능을 극복하기 위해 40GB 이상의 텍스트로 학습되었다. 여기에는 한국어 위키 백과, 모두의 말뭉치 v1.0, 청와대 국민청원 등의 데이터셋이 포함된다. Vocabulary 크기는 51,200이며 매개변수 개수는 1억 2천 5백만이다. 또한 대화에 자주 쓰이는 이모티콘, 이모지 등을 추가하여 해당 토큰의 인식 능력을 올렸으며, <unused0> ~ <unused99> 등의 미사용 토큰을 정의하여 필요한 task에 따라 자유롭게 정의해 사용할 수 있게 했다는 특징이 있다[12].

2.4.3 SKT-KoGPT-trinity

이 모델은 2021년 5월에 SKT에서 발표한 GPT-3 기반 대규모 언어모델이다. 이 모델을 훈련시키기 위해 SKT에서 만들어진 대규모 데이터셋 KoDAT으로 학습했으며, 350억 토큰으로 72,000 스텝을 거쳐 훈련된 모델이다. Vocabulary 크기는 KoGPT2와 마찬가지로 51,200이며 매개변수 개수는 12억이다. 이 모델은 한국어 텍스트로만 훈련되었으며 해당 언어에 대한 분류, 검색, 요약, 생성에 특화되어 있다[13].

2.4.4 KoGPT

KoGPT는 카카오 브레인에서 개발된 Vocabulary 크기 64,512, 매개변수 개수는 61억을 가지는 GPT-3 기반 한국어 대규모 언어 모델이다. 기계 독해, 기계 번역, 작문, 감정 분석, 질의응답, 대화 등 높은 수준의 언어 과제를 해결할 수 있다. 사용자는 KoGPT에게 과제 내용을 간단히 설명하고 질문하는 것으로 특정 과제에 해당하는 답변을 얻을 수 있으며, 파라미터 값을 조정해 요구사항에 더 잘 부합하는 결과를 유도할 수 있다. KoGPT는 적은 양의 예시 데이터로도 사용자의 요구사항을 빠르게 학습시키는 Few-shot learning을 지원한다[15].

Few-shot learning 은 기존에 학습한 지식을 재사용하면서 새로운 작업에 적은 양의 데이터로 학습하는 방법이다.

KoGPT 모델의 입력으로 주어지는 프롬프트(prompt)는 과제에 대한 설명, 예시 데이터, 입력으로 구성된다. 예시 데이터는 KoGPT 가 과제 수행 시 참고할 입력과 결과 예시이며, 입력은 KoGPT 가 처리해야 할 과제를 의미한다.

KoGPT 모델에서 조정 가능한 파라미터에는 다음과 같이 3 가지가 있다.

파라미터	설명	범위
토큰(max_tokens)	답변 문자열 길이를 조정	0 ~ 2048
온도(temperature)	각 후보 토큰의 선택 확률의 분포를 결정	0 ~ 1.0
상위 확률(top_p)	후보 토큰의 개수를 결정	0 ~ 1.0

[표 2.4.4-1] KoGPT 파라미터

[표 2.4.4-1]에서의 토큰은 KoGPT 가 의미를 이해할 수 있는 문자열의 최소 단위이며, 프롬프트와 결과를 합해 최대 2048 토큰의 문자열을 다룰 수 있다.

후보 토큰은 KoGPT 가 답변 구성에 사용할 토큰의 집합을 말한다. 온도 값이 높을수록 가능성이 다른 후보 토큰 간의 선택 확률 차이가 줄어든다. 즉, KoGPT 가 더 다양한 후보 토큰을 선택 대상으로 고려하고 더 창의적이고 다양한 답변을 구성하게 된다. 반대로, 온도 값이 낮을수록 가능성이 높은 후보 토큰을 주로 선택해 고정적인 답변을 구성하게 된다.

KoGPT 는 답변 구성에 사용할 토큰들을 추려낼 때 가장 가능성이 높은 토큰부터 각 후보 토큰의 가능성을 합한 수치가 상위 확률을 넘지 않도록 수를 제한하여 선정한다. 상위 확률이 높으면 후보 토큰의 수가 많아져 더 창의적인 답변을 구성하게 되고, 반대로 상위 확률이 낮으면 후보 토큰 수가 적어져 고정적인 답변을 구성하게 된다[14].

앞서 살펴본 3 개의 모델의 스펙을 표로 정리하면 아래[표 2.4.4-2]와 같다.

모델	개발	학습데이터	Vocabulary	Parameters
SKT-KoGPT2	SKT	위키피디아, 뉴스, 나무위키, 네이버 영화 리뷰, 한국어 Common Crawl (152M)	51,200	125M
SKT-KoGPT- trinity	SKT	Ko-DATA 데이터셋 (1.2B)	51,200	1.2B
KoGPT	Kakaobrain	200B	64,512	6B

[표 2.4.4-2] 3 개 모델 스펙 비교

본 연구에서는 3 개의 모델 중 가장 학습 데이터 양이 많고 단어 사전의 크기, 매개변수 크기가 큰 카카오 브레인의 KoGPT 모델을 선택하였다.

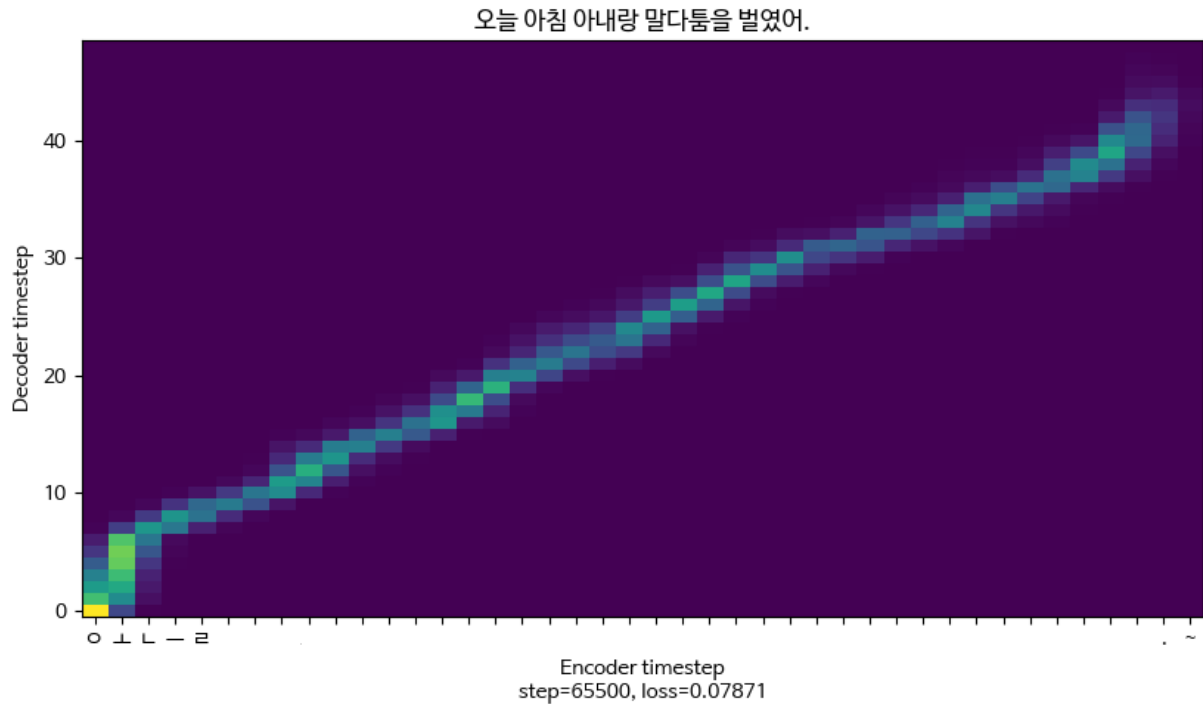
모델 사용은 카카오에서 제공하는 KoGPT REST API 를 활용하여 모델에 입력을 제공하고 출력을 받아오는 방식으로 진행하였다.

앞서 살펴본 3 개의 모델(skt-kogpt2, skt-kogpt-trinity, kogpt)은 다른 언어 모델과 마찬가지로 특정 프롬프트에 어떻게 반응할 지 사전에 예측하기 어렵고 따라서 경고 없이 공격적인 내용이 발생할 수 있다는 단점을 가지고 있지만 본 연구에서는 시스템 구현에 초점을 맞췄기 때문에 해당 문제는 고려하지 않는다.

2.5 노이즈 어텐션

어텐션(Attention)은 말 그대로 문장에서 중요한 부분에 더 집중하자는 뜻에서 시작된 개념이다. 이 개념이 도입되기 이전에는, 입력 시퀀스에 대한 모든 정보를 고정된 크기의 벡터로 인코딩 후 이를 디코더로 전달하는 seq2seq방식을 이용하였는데, 이러한 방식의 경우 두가지 문제점이 발생한다. 첫 번째로, 입력 시퀀스를 항상 고정된 크기의 벡터로 압축하기 때문에, 이 과정에서 정보 손실이 발생하는 문제점, 그리고 두 번째로 입력 시퀀스의 길이가 길어지면 RNN에서 기울기 소실 문제가 발생한다. 이러한 문제점을 해결하기 위해 어텐션 기법은 인코더의 모든 hidden state를 디코더로 전달하게 된다. 어텐션의 목적은 결국 단어들의 alignment를 예측하는 것으로 즉 현재 time step에서 들어갈 단어를 예측하는 것이다. 따라서 현재 time step에서 들어갈 확률이 높은 단어에 더 높은 가중치를 주는 방식으로 계산하며, 부여된 가중치를 바탕으로, 전달받은 모든 hidden state 벡터와 모든 문장 내 단어 벡터간의 유사도를 내적으로 계산 후 가장 높은 유사도를 가지는 벡터를 현재 time step의 벡터로 두는 방식이다.

음성 합성 분야에서는 딥러닝 모델을 통해 음질이 우수하고 자연스러운 음성을 합성하기 위한 연구가 활발하게 진행되고 있는데, 이들은 대부분 기계번역, 필기 생성, 캡션 자동완성 등의 분야에서 우수한 성능을 보인 바 있는 어텐션 기반 순환 시퀀스 생성 모델의 구조를 갖는다. 위 문단의 어텐션의 특성을 이용하여, 음성 합성 모델에서 어텐션은 입력 시퀀스(sequence)와 출력 시퀀스 간 대응 관계를 추정하고 이를 이용해 인코더가 추출한 벡터를 디코더로 전달하는 역할을 한다. 즉, 음성 합성 모델에서 어텐션은 출력 음성에서 각 음소의 위치를 결정하는데 중요한 역할을 담당한다. 따라서 어텐션이 부적절하게 추정될 경우 출력 음성의 발음이 잘못되거나 특정 부분이 반복, 혹은 생략되는 오류가 발생하게 된다.[15]



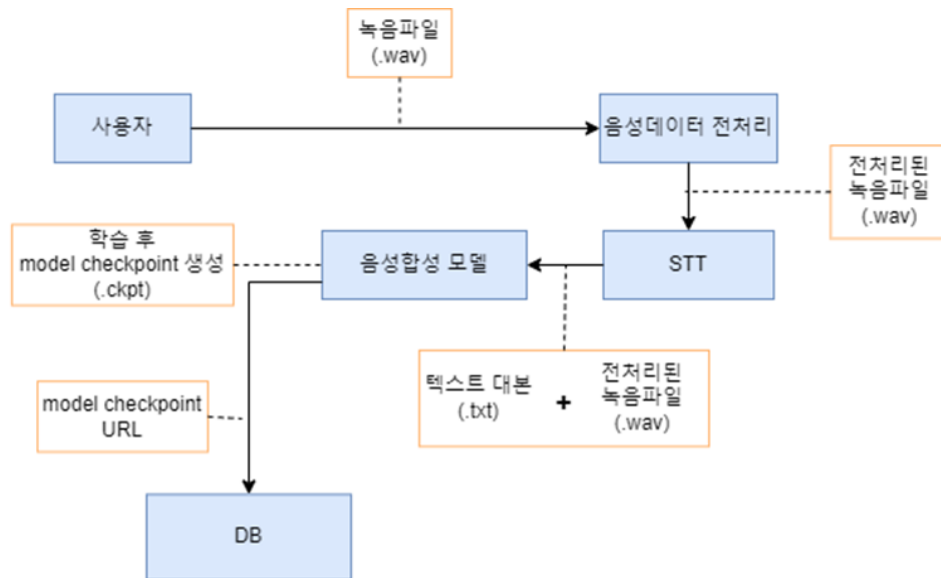
[그림 2.5.1] 노이즈 어텐션 그래프 예시

[그림 2.5.1]은 출력 음성의 음소 위치를 결정하는 노이즈 attention 그래프의 예시이다. 시간이 지남에 따라 음성이 출력되는 형태이므로 음성 합성에서 정상적인 노이즈 어텐션 그래프는 우상향의 그래프 형태를 가지게 되고, 각 음성의 음소 위치가 Encoder timestep에서 각 위치에 배치 되어있는 모습을 확인할 수 있다.

[그림 2.5.1]을 예시로 음성 합성이 정상적으로 잘 된 경우에는 그래프가 일정한 선의 형태를 가지고 색도 명확하지만 정상적으로 합성이 잘 안된 경우 즉, 어텐션이 부정확하게 추정되는 경우에는 위 그래프의 마지막 Encoder time step 부분의 그래프처럼 일정한 선의 형태가 아닌 색이 벌어지고 그래프의 두께가 두꺼워지는 현상이 발생한다. 또한 단어들의 alignment가 제대로 매칭이 안된 것을 확인 가능하다. 본 연구에서는 이러한 노이즈 어텐션 그래프의 성질을 통해 각 모델이 학습이 잘 되었는지 판단하고, 만약 잘 되지 않았다면, 그 이유를 분석한다.

3. 시스템 설계

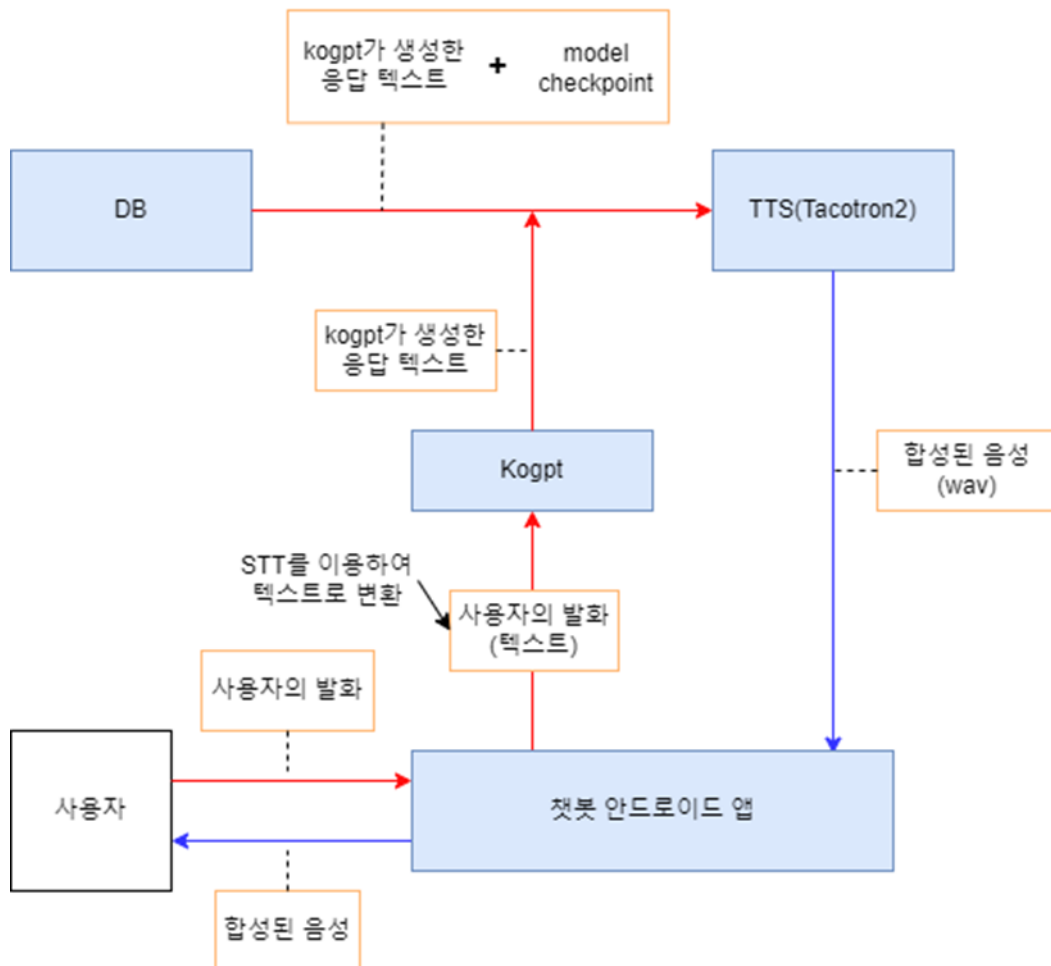
3.1 사전 학습 구성도



[그림 3.1-1] 사전 학습 구성도

[그림3.1-1]은 전체 시스템의 사전 학습 부분을 보여준다. 사전 학습의 경우 먼저, 사용자가 wav파일을 전달해주면, 음성파일을 전처리한다. 전처리 과정은 잡음을 제거한 후 잡음을 제거한 음성파일을 일정한 규칙으로 분할하는 음성파일 분할의 순서로 진행된다. 전처리 과정 후 구글 STT 라이브러리를 통해 분할한 음성파일 즉 세그먼트의 대본을 STT를 통해 추출하여, 전처리된 음성 세그먼트 wav파일의 위치와 이름, 즉 파일에서의 절대 경로와 그 음성 파일의 대본을 한 줄씩 txt파일로 저장을 한다. 그 후 학습용 txt파일과 wav파일을 음성 합성 모델을 통해 학습을 진행한다. 음성 합성 모델로는 tacotron2와 Waveglow를 이용하였으며, 학습 효율과 데이터 파일의 양 부족을 해결하기 위해, 12시간 정도의 KSS 음성 데이터를 학습시킨 후 다른 목소리는 KSS 목소리 데이터를 바탕으로 사전 학습된 모델을 fine-tuning한다. 모델 학습 시 500번째마다 checkpoint를 저장한 후 모델의 체크포인트 경로(URL)를 DB(MySQL)에 저장한다.

3.2 서비스 구조도

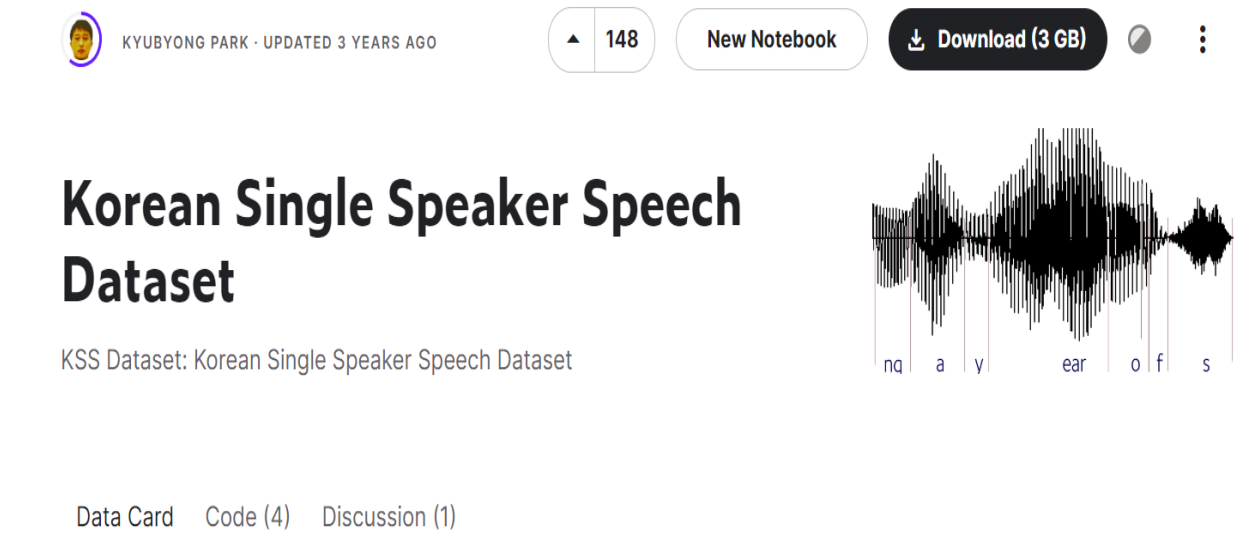


[그림 3.2-1] 서비스 사용 구조도

[그림 3.2-1]은 서비스 사용 구조도의 그림이다. 서비스는 안드로이드 앱을 통해서 제공된다. 사용자가 발화를 하면, 그 발화의 내용이 앱을 통해 대형언어모델인 KoGPT로 넘어가게 되며, KoGPT가 생성한 답변이 Flask 웹 서버로 이동되어, 서버에 내장된 모델로 그 음성을 합성하고 합성된 음성 wav파일이 안드로이드 앱으로 전송되어, 사용자에게 재생되는 방식으로 진행된다.

3.3 음성 데이터 수집

학습에 사용된 음성 데이터는 3개이다.



[그림 3.3-1] kaggle 데이터

[그림 3.3-1]은, kaggle이라는 데이터 사이트에서 제공하는 Korean single speaker speech dataset의 그림이다. 위 데이터 셋은 kaggle에서 제공하는 12시간 분량의 음성 데이터로, sample rate는 22050hz, 오디오 클립 수는 12,853개로 이루어져 있으며, 위 데이터로 음성 합성 모델을 학습 후, fine-tuning 방식으로 다른 음성 데이터들을 학습하는 방식을 진행하기 위해, 데이터 양을 다른 음성보다 많은 것으로 채택하였다.

두번째 데이터는 직접 녹음한, 음성 파일이다. 녹음실에서 소설책을 읽는 방식으로 진행되며, 음성 데이터 길이는 약 4시간 sample rate는 마찬가지로 22050hz이다.

세번째 데이터는 이미 잘 알려진 사람으로, 문재인 전 대통령의 연설, 공약 기자회견 영상 자료들을 이용하였다. Sample rate는 22050hz이며, 총 데이터 길이는 약 5시간이다.

3.4 음성 데이터 전처리

전처리 과정은 두 방법으로 이루어진다. 음성 데이터 잡음 제거 그리고 음성 데이터 분할을 통해 전처리를 진행하였다.

첫번째로, 잡음 제거의 경우 잡음이 음성에 섞여 있는 경우, 학습에 지장이 생기기 때문에 진행한다[16]. 단, 잡음의 종류만 보면, 자동차, 세탁기 등 생활 잡음과 혹은 다른 사람과의 대화의 경우, 그 사람만의 음성을 뽑아내는 것은 많이 힘든 작업이기에 잡음 제거의 경우는 시중에 존재하는 음성 편집기인 wavepad를 이용하여 진행한다.

두번째 전처리로는 음성 데이터 분할 과정이다. 음성 데이터를 300 ~ 1000kb의 세그먼트로 분할하는 과정을 거친다. 음성파일을 똑같은 크기로 나누지 않는 이유는, 동일한 간격으로만 자르게 될 경우, 잘린 세그먼트의 맨 앞 그리고 맨 뒤가 음성이 잘리거나 자연스럽게 지 않은 경우가 발생하여, 이 경우 학습에 지장이 갈 수 있기 때문이다.



[그림 3.4-1] 음성 파일 파형 예시

따라서 음성데이터는 파형의 진폭을 없는 곳을 기준으로 분할한다. 그 후 300 ~ 1000kb의 파일만 이용한다. 300kb 이하의 녹음 파일은 단어 하나 혹은 글자 하나만 녹음되어 있는 경우가 많아 학습에 영향이 없거나 부정적인 영향을 미치는 수준으로 타 논문, 혹은 프로젝트에 나와 있으며, 1000kb이상의 음성 세그먼트의 경우 뒤에서 설명할 예정인 구글 STT라이브러리의 경우 받아 드릴 수 없다는 오류가 발생하며, 그리고 tacotron2의 경우에도 1000kb이상의 음성 세그먼트는 다른 음성 세그먼트에 비해 학습량이 많아 다른 데이터를 학습하는데 영향을 줄 수 있기 때문이다[17].

음성 세그먼트 분할의 경우 파이썬의 pydub라이브러리에 AudioSegment 함수를 이용하여 분할한다.

3.5 음성 데이터 대본 출력 및 text 파일 저장

3.5.1 구글 STT 라이브러리 설명

2장 STT 라이브러리 설명에서도 언급했듯이 원하는 옵션 선택 가능, 실시간 사용 가능 그리고 오디오 노이즈에 안정적이라는 세가지 이유로 구글 STT 라이브러리를 사용하기로 하였다.

```
def speechtext(audiodata, channels, sample_width, frame_rate):
    client = speech.SpeechClient()
    audio = speech.RecognitionAudio(content = audiodata)
    config = speech.RecognitionConfig(
        encoding = speech.RecognitionConfig.AudioEncoding.LINEAR16,
        sample_rate_hertz = frame_rate,
        audio_channel_count = channels,
        language_code = 'ko-KR',
        model='default',
        use_enhanced=True,
        enable_automatic_punctuation = True,
        enable_word_time_offsets=True
    )
```

[그림 3.5.1-1] 구글 STT 설정

위 [그림 3.5.1-1]은 구글 STT설정에서 원하는 옵션을 선택할 수 있는 코드이다. language_code는 대본을 생성할 언어를 선택하는 옵션이다. model은 사용할 모델을 선택하는 옵션이며, use_enhanced은 추가 노이즈 개선으로, 오디오 음질을 조금 더 명료하게 만들어준다. 본 연구에서 사용할 언어는 한국어이므로 language_code는 한국어를 의미하는 ko-KR로 설정하였으며, model 옵션은 한국어의 경우 현재 default 모델만 가능하기에 default로 사용했다. use_enhanced는 오디오 개선 모드로 인식 정확도를 위해 조금이라도 노이즈를 제거해 오디오 음질을 조금이라도 개선시키고자 True로 설정했다. enable_word_offsets 옵션은 음성 인식 시간 정보를 반환하는 옵션으로 대본 작성 시간을 알아보기 위해 True로 설정했다.

위 코드를 통해 google API 서비스 센터에서 원하는 시간에 실시간으로 구글 STT 서비스를 이용할 수 있다.

3.5.2 대본 출력 및 학습용 text 파일 생성

잡음 제거 및 음성 세그먼트 분할이 완료된 파일들은 google STT API를 통해 대본을 생성한다. Tacotron2와 WaveGlow는 학습 정보 파일의 형식이 지정되어 있고, 그 형식에 맞추어 저장하기 위해 생성한 대본은 wav파일의 절대 경로와 같이 한 줄로 만들어 학습용 text 파일을 생성한다.

학습용 text파일 형식은 “음성파일위치 | 스크립트”로 이루어져 있으며, 예시를 들면, [그림 3.5.2.1]처럼 ../2.wav | "국민의 열망은 결코 멈추지 않을 것입니다" 와 같은 식으로 생성한다.

```
문재인/wav1/sound 101.wav|존경하는 국민 여러분.대통령으로서 무거운 짐을 내려놓습니다.
문재인/wav1/sound 102.wav|그동안 과분한 사랑과 지지로 성원해주신 국민 여러분께.무한한 감사의 말씀을 드립니다.
문재인/wav1/sound 103.wav|저는 이제 평범한 시민의 삶으로 돌아가
문재인/wav1/sound 104.wav|국민 모두의 행복을 기원하며 성공하는 대한민국의 역사를 응원하겠습니다.
문재인/wav1/sound 105.wav|지난 오 년은 국민과 함께 격동하는 세계사의 한복판에서 연속되는 국가적 위기를.해초는 식이었습니
다.
문재인/wav1/sound 106.wav|힘들었지만 우리 국민들은 위기 앞에 하나가 되어주셨습니다.
문재인/wav1/sound 107.wav|대한민국은 위기 속에서 더욱 강해졌고 더 큰 도약을 이뤘습니다.
문재인/wav1/sound 108.wav|대한민국의 국격도 높아졌습니다.대한민국은 이제 선진국이며 선도국가가 되었습니다.
문재인/wav1/sound 109.wav|우리 국민은 참으로 위대합니다.
문재인/wav1/sound 110.wav|저는 위대한 국민과 함께한 것이 더없이 자랑스롭습니다.
문재인/wav1/sound 111.wav|저의 퇴임사는.위대한 국민께 바치는 헌사입니다.
문재인/wav1/sound 112.wav|국정농단 사건으로 헌정질서가 무너졌을 때 우리 국민은 가장 평화적이고 문화적인 촛불집회를 통해,
문재인/wav1/sound 113.wav|그리고 헌법과 법률이 정한 탄핵이라는 적법절차에 따라 정부를 교체하고 민주주의를 다시.이렇게 하
세웠습니다.
문재인/wav1/sound 114.wav|전 세계가 한국 국민들의 성숙함에 찬탄을 보냈습니다.
문재인/wav1/sound 115.wav|우리 국민은 위기를 겪고 있는 세계 민주주의더 이상 앞으로 나아가지 못한 것은.우리의 의지와 노력
```

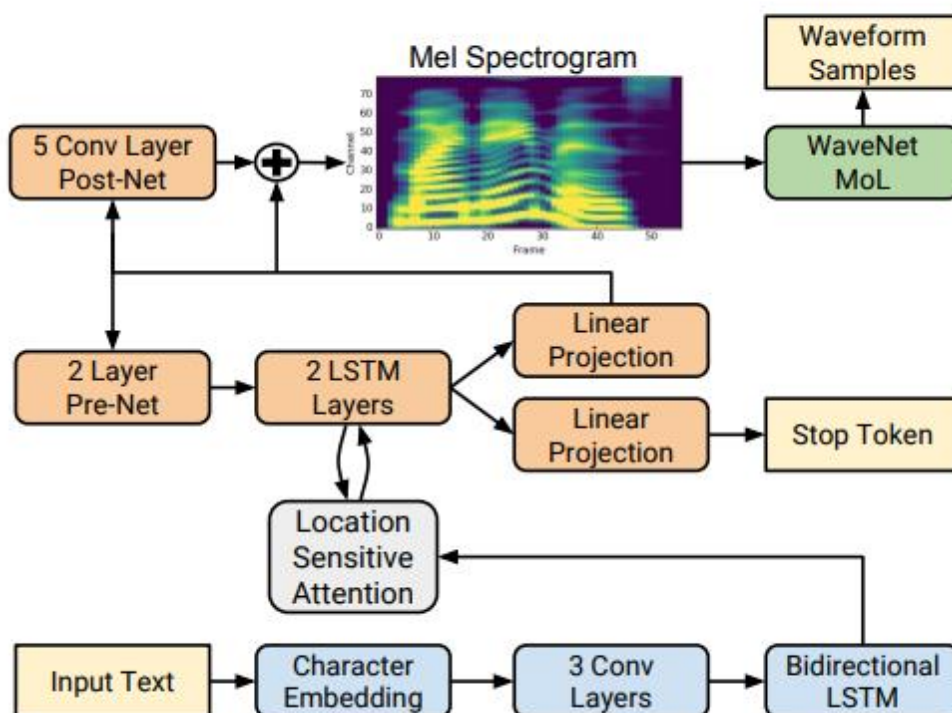
[그림 3.5.2.1] 학습용 text 파일 예시

3.6 음성합성모델

3.6.1 Tacotron2

Tacotron2 는 2018 년, 구글 딥마인드에서 제시한 TTS 모델이며, 문자 임베딩을 mel-scale 의 스펙트로그램으로 매핑시키는 seq-to-seq 기반 네트워크(Tacotron)에 WaveNet 모델이 추가된 형태이다. WaveNet 은 vocoder 의 역할을 하며, 스펙트로그램으로부터 시간영역 파형을 합성해낸다.

Tacotron2 는 Tacotron1 에 비해 다음 두 가지 면에서 발전된 형태를 띄고 있다. 첫번째로, Tacotron1 은 vocoder 로 Griffin-Lim 알고리즘과 Inverse short time fourier transform 을 사용한 반면, Tacotron2 는 Wavenet 을 이용하여 음성 합성의 품질을 향상시켰다. 두번째로, Tacotron1 은 encoder 로 CBHG 스택을, decoder 에서는 GRU recurrent 레이어를 사용했지만, Tacotron2 는 encoder 와 decoder 모두 vanilla LSTM, convolution layer 를 사용하여 상대적으로 단순한 레이어로 구성하여 모델을 간소화시켰다.



[그림 3.6.1-1] Tacotron2 아키텍처 블록 다이어그램

[그림 3.6.1-1]은 앞서 설명한 음성 합성 모델 Tacotron2 의 구조를 보여준다. 이 모델은 두 개의 구성요소로 구성된다. 첫번째는 입력 문자 시퀀스로부터 mel-spectrogram 프레임들의 시퀀스를 예측하는 sequence-to-sequence 특성(feature) 예측 네트워크이며, 두번째는 예측된 mel-spectrogram 프레임으로부터 시간 축의 파형을 생성해내는 WaveNet(기존 WaveNet 의 약간 수정된 버전)이다[18].

Mel-spectrogram 이란 음성 신호나 음악 신호와 같은 오디오 데이터의 주파수 변화를 시각화하는 기술 중 하나이며, 주파수 스케일을 인간의 청각에 가깝게 조정한 스펙트로그램을 말한다. 즉, 음성 인식에 중요한 저주파 대역의 특징은 부각시키는 반면, 마찰음이나 기타 노이즈의 비율이 높은 고주파 대역은 덜 부각시킨다.

WaveNet 은 실제 인간의 음성과 거의 비교할 수 없을 정도의 오디오 품질을 자랑하지만, WaveNet 에 대한 입력은 정교한 텍스트 분석 시스템과 강력한 사전(발음 가이드)과 함께 상당한 전문 지식을 필요로 한다. 이때, Tacotron 은 앞서 말한 WaveNet 을 위한 언어적, 음향적 특징의 생산을 하나의 neural network로 대체함으로써 복잡한 feature engineering 없이 데이터로 직접적인 학습이 가능하여 전통적인 음성 합성 파이프라인을 단순화시켜준다[19].

Tacotron2 의 학습을 진행할 때는 epoch 을 크게 설정하고 학습하되, 학습 중간마다 생성된 checkpoint 로 음성을 생성하여 직접 들어보고 학습을 중단할지 말지 결정한다.

3.6.2 Tacotron

Tacotron 은 sequence-to-sequence 아키텍처를 가지며, 문자 시퀀스로부터 스펙트로그램을 생성해내는 역할을 한다. Tacotron 은 encoder 와 decoder, 그리고 attention 으로 구성되며, encoder 는 문자 시퀀스를 hidden feature representation 으로 변환하고, 이를 decoder 가 스펙트로그램을 예측하는 데 사용한다.

입력 문자 시퀀스는 512 차원 문자 임베딩 과정을 거치며, 3 개의 convolution 계층 스택을 통과한다. 각 convolution 계층은 5×1 의 shape 을 가지는 512 개의 필터를 가진다. 각 필터는 5 개 문자 단위로 연산하며, batch 정규화, ReLU 활성화 함수를 거친다. 마지막 convolution 계층의 출력은 512 개의 unit 을 가지는 하나의 양방향 LSTM(Long Short Term Memory) 계층을 통과하여 인코딩된 특성을 생성한다.

Encoder 의 출력은 attention 네트워크에 의해 decoder 의 각 출력 단계를 위한 고정길이의 context vector 로서 사용된다. Tacotron2 에서는 additive attention 에서 확장된 형태인 location-sensitive attention 을 사용한다[18].

Location-sensitive attention 은 이전 decoder time step 에서 생성한 결과뿐만 아니라, 이전 step 에서의 attention 가중치 즉, 어떤 정보에 집중했는지에 대한 정보를 추가로 사용하는 메커니즘이다. 이는 모델이 입력을 순차적으로 일관되게 처리하게 함으로써 입력 시퀀스의 일부분이 반복되거나 누락되는 것을 방지할 수 있다.

$$e_{i,j} = w^T \tanh(Ws_{i-1} + Vh_j + b) \quad (4)$$

위 수식(4)은 additive attention 연산 수식을 나타낸다. 여기서 h 는 양방향 RNN encoder 로부터 생성된 순차 표현을 나타내며, s_{i-1} 는 recurrent 뉴럴 네트워크(eg. LSTM, GRU)의 $i - 1$ 번째 상태(state)를 나타낸다. 위 수식에 단어의 위치 정보를 추가하기 위해 이전 단계에서 생성된 alignment 정보, α_{i-1} 의 각 위치 j 로부터 k 개의 벡터, $f_{i,j} \in \mathbb{R}^k$ 를 추출한다. 그리고 이 벡터들을 attention 연산에 식(5)와 같이 사용한다[19].

$$e_{i,j} = w^T \tanh(Ws_{i-1} + Vh_j + Uf_{i,j} + b) \quad (5)$$

Attention 값은 입력과 위치 특성을 128-차원의 hidden representation 으로 변환 후 계산된다. 위치 특성은 32 개의 31x1 1D convolution 필터로 계산된다.

Decoder 는 자기회귀 반복 신경망(autoressive recurrent neural network)으로서 인코딩된 입력 시퀀스로부터 mel-spectrogram 을 예측하는 역할을 한다. 이전 time step 에서의 예측값은 먼저 256 개의 hidden ReLU 유닛을 가지는 fully-connected 레이어 2 개로 구성된 pre-net 을 통과하게 되고, 이 pre-net 의 출력과 attention 컨텍스트 벡터가 결합되어 1024 개의 유닛을 가지는 단방향 LSTM 두 층의 입력으로 주어진다. 이렇게 LSTM 출력과 attention 컨텍스트 벡터가 결합된 입력은 정답 스펙트로그램 프레임을 예측하기 위해 linear transform 과정을 거친다. 마지막으로, 예측된 mel-spectrogram 은 5 개의 convolution 계층으로 이루어진 post-net 을 통과하게 된다. 각 post-net 계층은 512 개의 5x1 필터들과 배치 정규화 계층으로 구성되며 마지막 계층을 제외하고 모두 tanh 활성화 함수를 가진다.

추가로, decoder LSTM 의 출력과 attention 컨텍스트 벡터의 결합된 값은 상수로 변환되어 sigmoid 활성화 함수를 통과하여 출력 시퀀스의 종료 확률을 예측한다. 이 종료 확률을 나타내는 'stop token'이라고 불리는 토큰은 추후 추정에서 쓰이며 모델이 동적으로 출력 생성 종료 시점을 결정하도록 해준다. 구체적으로, 이 확률이 0.5 의 임계점을 넘어가면 생성을 종료한다.

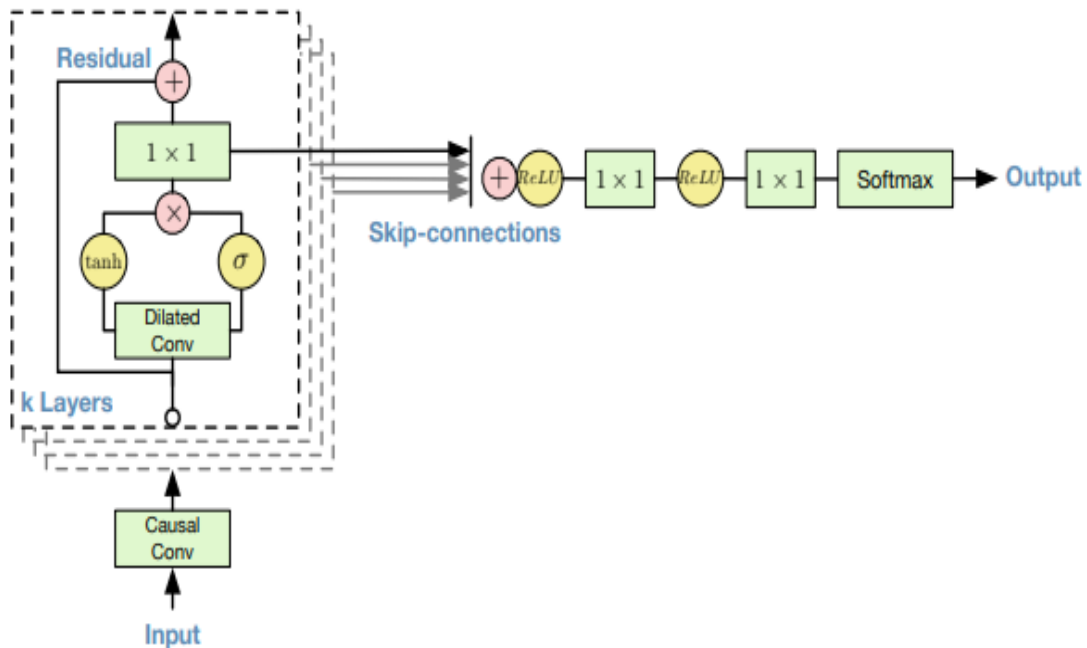
Tacotron 네트워크의 convolution 계층들은 0.5 비율의 dropout 을 이용하여 규제를 하며, LSTM 계층들은 0.1 비율의 zoneout 을 이용하여 규제를 한다[18].

Tacotron 의 loss 는 Tacotron 으로 생성한 mel-spectrogram 과 정답 mel-spectrogram 의 차이를 계산한 MSE(Mean Squared Error)와 Tacotron 으로 생성한 종료 확률과 실제 종료 여부와의 차이를 계산한 BCE(Binary Cross Entropy)로 구성된다.

3.6.3 WaveNet

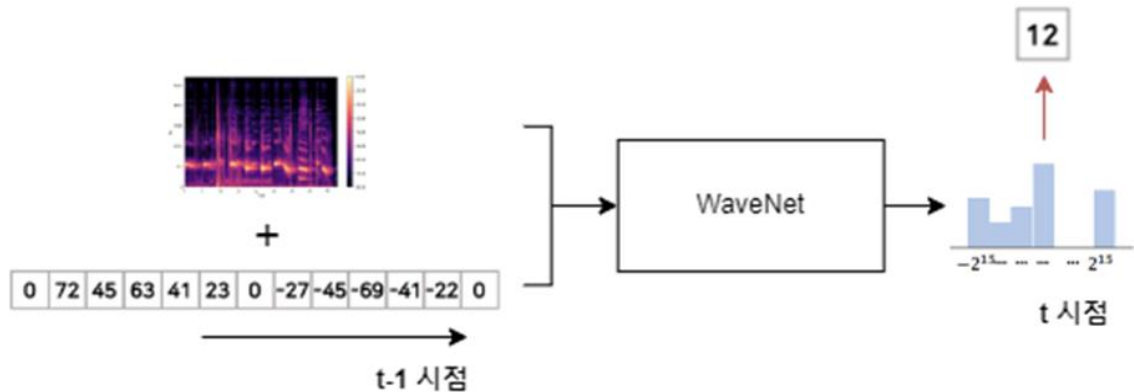
WaveNet 은 딥마인드에서 개발한 음성 생성 모델이다. WaveNet 은 Conditional Generative 모델로서, 입력 데이터에 따라 다양한 조건에서 음성을 생성할 수 있도록 설계되었다. 또한 WaveNet 은 자기회귀(autoregressive) 모델로서, 과거 시점까지 생성된 음성 데이터와 mel-spectrogram 에 기반하여 한 시점 뒤의 특정 음성의 등장 확률을 조건부 확률 분포를 통해 추출하는 확률론적 모델이다. 학습 시에는 ground truth 데이터와 비교하여 각 특정 음성의 생성 확률 분포를 최대화하는 방식으로 학습한다. WaveNet 의 아키텍처는 기본적으로 딥러닝의 순차적 구조인 순환 신경망(RNN)이나 장단기 메모리(LSTM)를 사용하지 않고, convolution network 를 사용한다.

WaveNet 은 causal convolution 을 사용하여 시간적인 순서를 지키면서 미래의 정보를 사용하지 않도록 한다. 또한 dilated convolution 을 사용하여 점점 더 넓은 receptive field 를 갖게 되면서 멀리 떨어진 입력 정보를 고려하면서도 별도의 RNN 을 사용하지 않는다는 특징이 있다.



[그림 3.6.3-1] WaveNet 아키텍처

WaveNet 은 [그림 3.6.3-1]과 같이 30 개의 residual 블록 스택과 skip connection 으로 구성된다[20]. 각 residual 블록에서 추출된 출력이 skip connection 으로 합쳐지고 이를 최종 output 으로 활용한다. 이 output 은 아래 그림처럼 앞서 설명한 한 시점 뒤의 특정 음성의 등장 확률 즉, $-2^{15} \sim 2^{15}$ 사이의 디지털 신호 값들의 확률 분포를 이용하여 가장 확률이 높은 신호 값을 선택하는 방식이다.



[그림 3.6.3-2] WaveNet 에서의 음성 데이터 생성

WaveNet 의 loss 로는 WaveNet 으로 생성한 음성 파형과 실제 음성 파형의 차이를 계산한 Negative log-likelihood loss 를 사용한다.

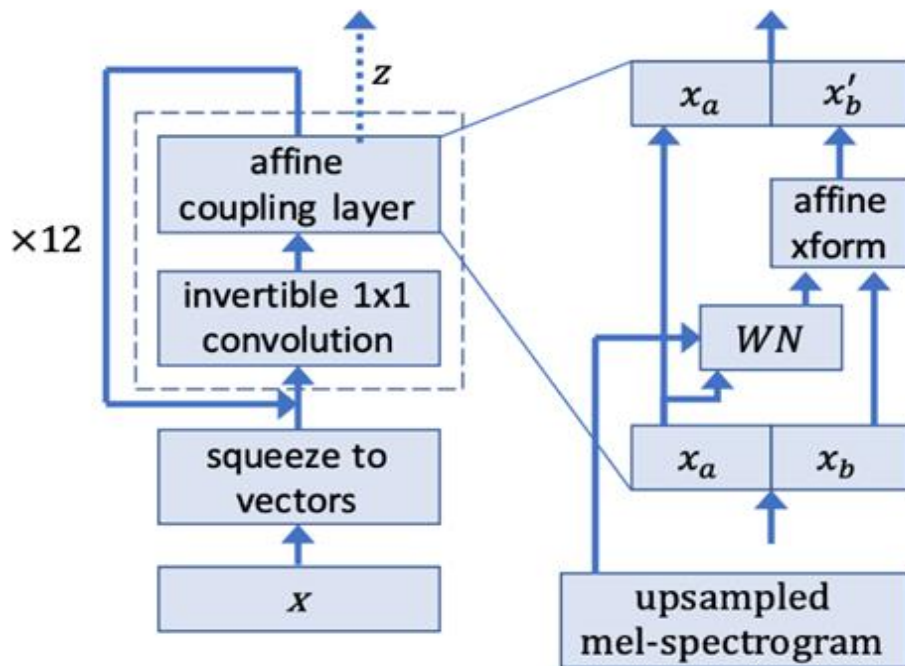
Tacotron2 에서는 mel-spectrogram 으로부터 음성 파형을 생성해 내기 위해 WaveNet 의 수정된 버전을 사용한다. 기존 구조와 동일하게 30 개의 residual block 이 사용되었고, conditioning stack 에서는 기존의 3 개 대신 2 개의 upsampling 계층을 사용하였다. 또한, softmax 계층을 이용하여 이산 값을 예측하는 방식 대신 PixelCNN++와 Parallel WaveNet 의 방식에 기반하여 10 개 mixture of logistic 분포(MoL)를 활용하여 24 kHz 에서 16 bit 의 샘플을 생성한다[19]. 이러한 접근은 기존의 방식보다 품질이 좋은 오디오를 합성할 가능성이 높다.

WaveNet은 autoregression 즉, 이전 시점의 모든 예측 값을 모델에 다시 입력하면서 이전에 입력한 값을 이용해 다음 값을 예측하는 방식이기 때문에 각 입력 샘플마다 연산량이 매우 많다. 따라서 다른 음성 생성 모델들에 비해 합성된 음성의 품질은 높지만 음성합성 속도가 느리다는 단점이 있다. 본 연구에서 구현하고자 하는 서비스는 실시간 대화 서비스이므로 kogpt 로부터 생성된 텍스트로부터 음성을 합성해내는 데 소요되는 시간을 단축하여 빠르게 음성을 합성할 수 있는 모델이 필요하다.

3.6.4 WaveGlow

WaveGlow 는 NVIDIA 에서 개발한 음성 합성 모델이다. WaveGlow 는 Generative Adversarial Networks(GANs)이나 Variational Autoencoders(VAEs)와 같은 기존의 음성 합성 방법과 다른 방식으로 작동한다. WaveGlow 는 Flow 기반 생성 모델을 기반으로 한다. Flow 기반 모델은 확률적 역변환을 사용하여 입력 데이터를 임의의 다른 공간으로 매핑하는 함수를 연속적으로 적용하여 출력 데이터를 생성한다. 이 변환 함수는 서로 독립적이기 때문에 각 변환 함수는 병렬로 계산될 수 있으며 이는 고성능 GPU 에서 매우 빠르게 음성을 합성할 수 있는 이점을 제공한다.

WaveGlow 는 convolution 과 deconvolution 레이어를 사용하기 때문에 병렬 처리 장치에서 기존의 WaveNet 모델보다 효율적인 학습과 빠른 속도의 추론이 가능하다. 또한 하나의 네트워크로 구성되어 있어 구현이 쉽고 구조가 단순하다는 장점이 있고 하나의 비용 함수(likelihood loss)만 사용하여 훈련 프로세스를 단순화하였다.



[그림 3.6.4-1] WaveGlow 아키텍처

위 그림에서 볼 수 있듯이, WaveGlow 는 8 개의 audio sample 을 그룹화하여 벡터로써 처리(squeeze)한 후 이 벡터들을 invertible 1x1 convolution, affine coupling layer 로 구성되는 여러 개의 'steps of flow'를 거치게 된다[20].

WaveGlow 는 zero mean spherical Gaussian 분포로부터 샘플링하여 음성으로 합성해낸다.

$$z \sim \mathcal{N}(z; 0, I)$$

$$x = f_0 \circ f_1 \circ \dots \circ f_k(z) \quad (6)$$

식(6)과 같이 normal distribution 에서 latent vector z 를 sampling 하여 뽑아낸 후 이를 normalizing flow 방법을 통해 원하는 data x 로 변환시킨다. 여기서는 mel-spectrogram 이 x 가 가지는 distribution 의 조건이 된다.

WaveGlow 는 다음 식(7) 같이 negative log-likelihood 를 최소화하는 방향으로 학습을 진행한다.

$$\text{Log } p_\theta(x) = \log p_\theta(z) + \sum_{i=1}^k \log |\det(J(f_i^{-1}(x)))|$$

$$z = f_k^{-1} \circ f_{k-1}^{-1} \circ \dots \circ f_0^{-1}(x) \quad (7)$$

여기서 첫번째 항은 spherical Gaussian 의 log-likelihood 이다. 이 항은 변환된 sample 의 l_2 norm 에 penalty 를 부여한다. 두번째 term 은 f^{-1} 변환의 jacobian 의 determinant 를 포함한다[21].

WaveGlow 의 모델 구조는 Glow 와 유사하지만 Glow 에서의 NN() 대신에 WaveGlow 에서는 WN() 모듈이 존재하고 이 모듈에 condition 으로 upsampled mel-spectrogram 이 들어가게 된다. WN()은 WaveNet 모듈이지만, WaveNet 처럼 causal convolution 을 이용하지 않고 일반적인 convolution 을 이용한다. 음성 합성 모델에 대한 평가 점수는 MOS(mean opinion score)을 통해 나타내는데 MOS 는 전화통화 성능 평가에 주로 사용되는 통계에 의한 주관적 평가 기법으로 선별된 여러 명의 평가자가 품질을 테스트 하는 방법이다 대략적으로 MOS 의 경우 4~5 사이가 고품질 3.5~4 정도 되면 자연스러운 통화 수준 3~3.5 정도되면 약간의 품질저하가 발생한다고 본다[21].

WaveGlow 논문[21]에서 제시한 Griffin-Lim, WaveNet, WaveGlow 의 실험을 통한 음성 합성 품질 점수(MOS, Mean Opinion Score)는 아래 [그림 3.6.4-2]와 같다.

Model	Mean Opinion Score (MOS)
Griffin-Lim	3.823 ± 0.1349
WaveNet	3.885 ± 0.1238
WaveGlow	3.961 ± 0.1343
Ground Truth	4.274 ± 0.1340

[그림 3.6.4-2] Mean Opinion Scores

System	MOS
Parametric	3.492 ± 0.096
Tacotron (Griffin-Lim)	4.001 ± 0.087
Concatenative	4.166 ± 0.091
WaveNet (Linguistic)	4.341 ± 0.051
Ground truth	4.582 ± 0.053
Tacotron 2 (this paper)	4.526 ± 0.066

[그림 3.6.4-3] 추가 Mean Opinion Scores

위의 [그림 3.6.4-2]와 [그림 3.6.4-3]은 MOS score 에 관한 지표이며, MOS 는 앞서 설명한대로 주관적인 평가 지표이기 때문에 두 실험 결과의 차이가 발생하게 된다.

실험에는 13,100 개의 오디오 클립으로 구성된 LJ speech data 를 사용하였으며, 22,050KHz 의 sampling rate 을 사용하였다. 또한 원본 오디오로부터 FFT 사이즈 1024, hop 사이즈 256, 윈도우 사이즈 1024 의 mel-spectrogram 을 추출하여 WaveNet 과 WaveGlow 네트워크의 입력으로 제공하였다. Griffin-Lim 은 mean opinion score 의 기준점으로서 비교 대상으로 추가하였다.

위 점수표에 따르면, 합성된 샘플들의 MOS 는 절대적인 척도에서는 비슷하지만, 세 모델 모두 실제 오디오의 MOS 점수에는 못 미치는 결과를 보였다. 이 중에서 WaveGlow 가 가장 높은 MOS 를 가지고 있지만, 논문에 따르면, 약 1000 개의 샘플을 수집한 후에는 모든 모델이 유의미한 차이 없이 비슷한 점수를 보였다[21]. 반면, 음성 합성 속도 면에서는 WaveGlow 가 WaveNet 보다 월등히 빠르다. 논문에 따르면, WaveNet 은 0.11KHz 의 속도로 음성을 합성한 반면, WaveGlow 는 10 초 발화를 합성하는데 520KHz 의 속도로 합성하였다. 이는 500KHz 의 속도로 합성한 Parallel WaveNet 보다 약간 더 빠른 속도를 보였다[20].

이와 같은 결과를 바탕으로 본 연구에서는 WaveNet 대신 합성 속도 측면에서 더 좋은 성능을 보인 WaveGlow 모델을 채택하였다.

4. 시스템 구현

4.1 음성 데이터 수집 및 전처리

4.1.1 음성 데이터 수집

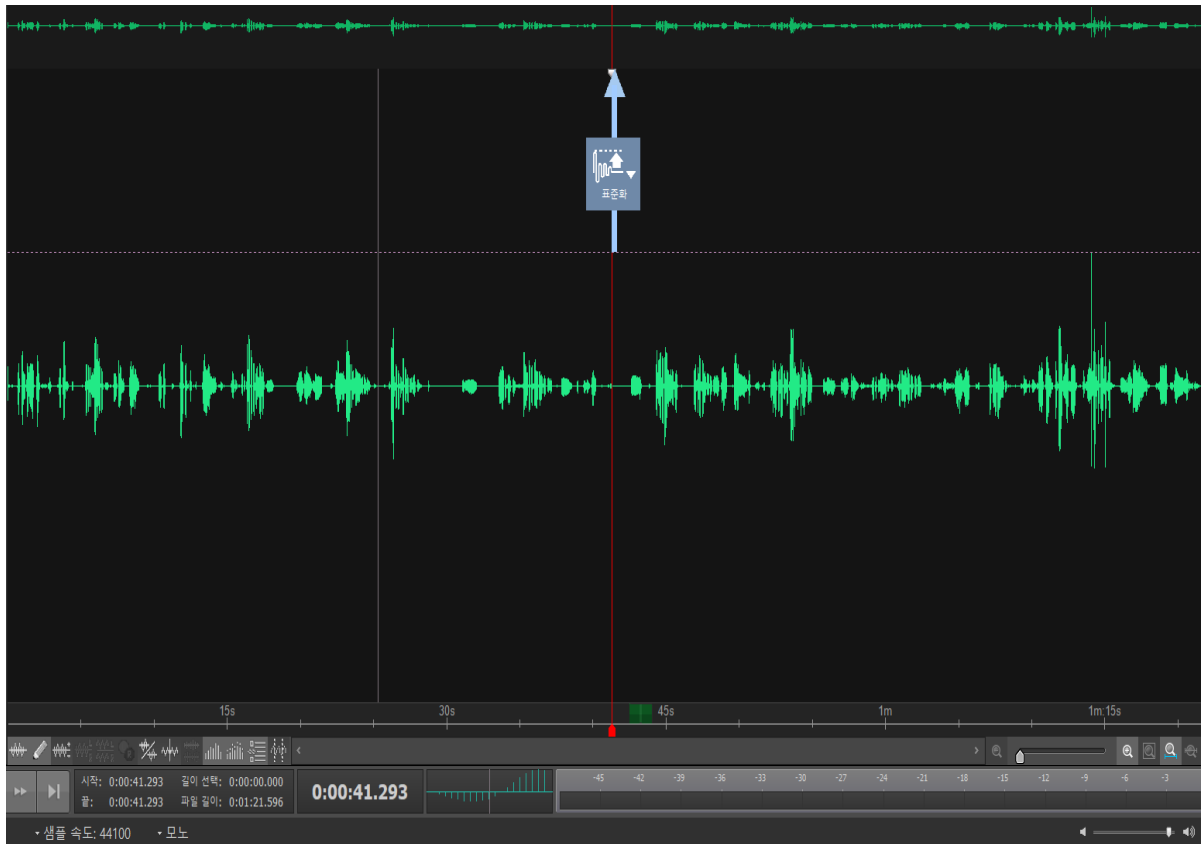
	KSS	김준석(직접 녹음)	문재인 전 대통령
설명	전문 여성 성우 녹음	소설책 녹음	연설, 공약 영상 이용
데이터 길이	12시간	4시간	5시간
Sample rate	22050 hz	22050 hz	22050 hz
분할된 오디오 클립 수	12,853개	2,218개	4,386개

[표 4.1.1] 수집 혹은 직접 녹음한 데이터 셋

[표 4.1.1]은 본 연구에서 사용하는 데이터 셋에 대한 설명이다. 데이터 셋은 총 3개이며, KSS에서 제공하는 전문 여성 성우 녹음, 그리고 조원 중 한 명이 소설책을 읽으며, 스튜디오에서 직접 녹음한 데이터 셋, 그리고 공인 문재인 전 대통령의 연설, 공약 영상 등을 이용한 데이터 셋이다. KSS의 경우 다른 데이터 셋에 비해 양이 많은데, 이 이유는 문재인 데이터와 김준석 데이터의 경우 본 연구기간에 모을 수 있는 데이터 양의 한계가 존재하여, 데이터 양의 부족으로 인한 오버피팅을 줄이고자, KSS데이터를 먼저 음성 합성을 한 후 그 모델 위에 다른 목소리들을 추가 학습시키는 방법 즉 fine tuning을 시키기 위하여, 다른 데이터 셋보다 양을 많은 것을 선택했다.

4.1.2 음성 데이터 전처리

위에서 설명한 대로 음성 학습의 진행을 위해선, 잡음이 최대한 작아야 한다. 따라서 카메라 셔터 소리 외부 소음 등을 제거하는 전처리 과정이 필요하다. 단 모든 잡음을 제거하는 과정은 많이 어렵고 시간도 부족하여 잡음 제거의 경우 이미 유료 음성 서비스 앱인 wavepad를 이용해 잡음을 제거한다.



[그림 4.1.2-1] 잡음제거 툴 이용 잡음 제거

잡음 제거가 완료된 음성 파일들은 STT가 1000kb이상의 파일을 받아드리지 못하기 때문에, 1000kb이하로 음성파일을 몇 개의 세그먼트로 분할하는 과정을 거치기 위해 리눅스 서버에 저장시켜 두었다. 잡음을 제거한 음성 파일들은 원활한 학습을 위해 [그림 4.1.2-1]에서처럼 진폭이 존재하지 않는 부분 즉 공백을 기준으로 음성 데이터 분할을 진행한다.

```
def get_audiosegment(file):
    audio_segments = []
    audio = AudioSegment.from_wav(file)
    min_silence_len = 500
    silence_thresh = -50
    segments = split_on_silence(audio, min_silence_len, silence_thresh)
    for s in segments:
        audio_segments.append(s)
    return audio_segments

for j in range(len(audiosegments)):
    filename = base_path + wav_name + str(wav_count) + ".wav"

    save_str = "moon/" + wav_name + str(wav_count) + ".wav"
    new_lines.append(save_str)
    seg = audiosegments[j]
    seg.export(filename, format = 'wav')
    wav_count += 1
print("{} 저장완료".format(file))
```

[그림 4.1.2-2] 음성 파일 분할

[그림 4.1.2-2]은 pydub 라이브러리의 AudioSegment 함수를 이용하여 음성 파일을 분할하는 코드이다. 그 후 음성 세그먼트 파일의 이름이 중복되는 것을 피하기 위해, 분할된 세그먼트에 고유 숫자를 추가하여 리눅스 서버에 저장한다.

4.2 대본 생성 및 학습용 text 파일 생성

```
def speechtext(audiodata, channels, sample_width, frame_rate):
    client = speech.SpeechClient()
    audio = speech.RecognitionAudio(content = audiodata)
    config = speech.RecognitionConfig(
        encoding = speech.RecognitionConfig.AudioEncoding.LINEAR16,
        sample_rate_hertz = frame_rate,
        audio_channel_count = channels,
        language_code = 'ko-KR',
        model='default',
        use_enhanced=True,
        enable_automatic_punctuation = True,
        enable_word_time_offsets=True
    )
    response = client.recognize(config = config, audio = audio)
    text = ""
    for result in response.results:
        text += result.alternatives[0].transcript
    return text.encode("utf-8")

def get_script_by_wav(audio_segments):
    script = []
    segment = []
    for s in audio_segments:
        audio_data = s.raw_data
        f_size = len(audio_data)
        if f_size/1000 <= 1000:
            with tempfile.NamedTemporaryFile(suffix=".wav", delete=False) as f:
                s.export(f.name, format="wav")
                with wave.open(f.name, "rb") as wave_file:
                    channels = wave_file.getnchannels()
                    sample_width = wave_file.getsampwidth()
                    frame_rate = wave_file.getframerate()
                    frames = wave_file.getnframes()
                    audio_data = wave_file.readframes(frames)
                    sp = speechtext(audio_data, channels, sample_width, frame_rate)
                    if sp:
                        script.append(sp.decode("utf-8"))
                        segment.append(s)
    return script, segment
```

[그림 4.2-1] 구글 STT 라이브러리를 이용, 분할된 wav파일의 대본 생성 코드

잡음 제거 및 음성 세그먼트로 분할이 완료된 wav 파일에 대해, 구글 STT 라이브러리를 이용하여 wav 파일의 대본을 생성한다.

4.3 모델 학습

NVIDIA-SMI 440.64.00				Driver Version: 440.64.00		CUDA Version: 10.2	
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile Uncorr. ECC		
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute M.	
0	GeForce RTX 2070	Off	00000000:01:00.0	On			N/A
0%	42C	P8	13W / 215W	7902MiB / 7981MiB	0%	Default	
Processes:							
GPU	PID	Type	Process name	GPU Memory Usage			
0	1074	G	/usr/lib/xorg/Xorg	72MiB			
0	1982	G	compiz	128MiB			
0	4315	C	python	7697MiB			

[그림 4.3-1] Tacotron2 학습환경

음성 학습 모델의 학습 환경은 [그림 4.3-1]에 나타난 바와 같다. 운영체제는 Ubuntu 16.04, 그래픽 카드는 8GB 메모리의 RTX 2070를 사용하였으며, 16GB RAM을 사용하였다.

각 모델의 하이퍼파라미터의 경우 Tacotron는 epoch을 500, 배치 크기를 32로 설정하였으며, 학습률(learning rate)을 1e-3로 설정하였다. WaveGlow는 epoch을 500, 배치 크기를 12(배치 크기를 작게 설정한 이유는 12를 초과하여 설정할 경우 cuda out of memory error가 발생하기 때문이다)로 설정하였으며, 학습률을 1e-4로 설정하였다.

Tacotron는 KSS 데이터셋의 경우 매 1000번째 train step마다 검증을 수행하도록 설정했으며, 김준석, 문재인 데이터셋의 경우 500번째 train step마다 검증을 수행하도록 하였다. Tacotron의 경우 검증을 수행한 후 즉, 매 1000번째 train step마다 그 시점의 모델의 가중치를 저장한 체크포인트 파일을 생성하며, WaveGlow의 경우 매 500번째 train step마다 체크포인트 파일을 생성하여 저장한다. WaveGlow의 경우 검증을 수행하지 않는다.

본 연구에서는 Tacotron의 경우 train loss값이 0.10 이하에서 더 이상 줄어들지 않는다고 판단될 때, WaveGlow의 경우 train loss 값이 -10 이하에서 더 이상 줄어들지 않는다고 판단될 때, 학습을 중단시키는 방식으로 진행하였다.

본 연구에서는 먼저 KSS 데이터셋을 이용하여 Tacotron과 WaveGlow의 학습을 진행하였다. KSS 데이터의 총 12853개의 오디오 클립 중 약 80%인 10282개를 학습용 데이터, 약 10%인 1285개를 검증용(validation), 나머지 1286개를 테스트용으로 사용하였다.

김준석 데이터셋의 경우 총 2218개의 오디오 클립 중 약 80%인 1774개를 학습용 데이터, 약 10%인 221개를 검증용, 나머지 223개를 테스트용으로 사용하였다.

문재인 전 대통령 데이터셋의 경우 총 4386개의 오디오 클립 중 약 80%인 3508개를 학습용 데이터, 약 10%인 438개를 검증용, 나머지 440개를 테스트용으로 사용하였다.

김준석, 문재인 전 대통령 데이터셋의 경우 KSS 데이터셋에 비해 더 적은 양의 데이터를 가지고 있었기 때문에, 두 데이터셋의 경우 KSS를 학습시킨 Tacotron과 WaveGlow 모델을 fine-tuning시키는 방식으로 학습을 진행하였다.

```

for epoch in range(epoch_offset, hparams.epochs):
    print("Epoch: {}".format(epoch))
    for i, batch in enumerate(train_loader):
        start = time.perf_counter()
        for param_group in optimizer.param_groups:
            param_group['lr'] = learning_rate
        model.zero_grad()
        x, y = model.parse_batch(batch)
        y_pred = model(x)
        loss = criterion(y_pred, y)
        reduced_loss = loss.item()
        loss.backward()
        grad_norm = torch.nn.utils.clip_grad_norm_(
            model.parameters(), hparams.grad_clip_thresh)
        optimizer.step()
        if not is_overflow and rank == 0:
            duration = time.perf_counter() - start
            print("Train loss {} {:.6f} Grad Norm {:.6f} {:.2f}s/it".format(
                iteration, reduced_loss, grad_norm, duration))
            logger.log_training(
                reduced_loss, grad_norm, learning_rate, duration, iteration)
        if not is_overflow and (iteration % hparams.its_per_checkpoint == 0):
            validate(model, criterion, valset, iteration,
                    hparams.batch_size, n_gpus, collate_fn, logger,
                    hparams.distributed_run, rank)
        if rank == 0:
            checkpoint_path = os.path.join(
                output_directory, "checkpoint_{}".format(iteration))
            save_checkpoint(model, optimizer, learning_rate, iteration,
                            checkpoint_path)
    iteration += 1

```

[그림 4.3-2] Tacotron main train loop 코드

[그림 4.3-2]은 Tacotron 모델의 학습(train) 코드를 나타낸다. 이 코드는 모델을 학습시키고, 학습 과정에서의 손실(loss) 값을 출력하고 체크포인트를 저장하는 등의 작업을 수행한다. 이 코드에서는 train_loader 데이터 로더를 이용하여 미니배치를 순회하면서 학습을 진행한다.

이 코드에서는 'model.zero_grad()'를 통해 모델의 기울기(gradient)를 초기화한다. 이는 이전 미니배치의 기울기가 영향을 미치지 않도록 한다. 또한, 'criterion()' 손실 함수를 통해 예측값 y_pred와 실제 타겟 y 사이의 손실(loss)을 계산한다.

```
loss_values = []
for epoch in range(epoch_offset, epochs):
    print("Epoch: {}".format(epoch))
    for i, batch in enumerate(train_loader):
        model.zero_grad()
        mel, audio = batch
        mel = torch.autograd.Variable(mel.cuda())
        audio = torch.autograd.Variable(audio.cuda())
        outputs = model((mel, audio))
        loss = criterion(outputs)
        reduced_loss = loss.item()
        loss.backward()
        optimizer.step()
        print("{}:\t{:0.9f}".format(iteration, reduced_loss))
        loss_values.append(reduced_loss) # Append loss value to the list
        if (iteration % iters_per_checkpoint == 0):
            if rank == 0:
                checkpoint_path = "{}/waveglow_{}".format(
                    output_directory, iteration)
                save_checkpoint(model, optimizer, learning_rate, iteration, loss_values,
                               checkpoint_path)
        iteration += 1
```

[그림 4.3-3] WaveGlow main train loop 코드

위 그림은 WaveGlow 모델의 학습(train) 코드를 나타낸다. 마찬가지로 train_loader 데이터 로더를 이용하여 미니배치를 순회하면서 학습을 진행한다.

'mel, audio = batch'는 미니배치 데이터 'batch'에서 멜 스펙트로그램(mel spectrogram)과 오디오 데이터를 추출하는 역할을 한다. 그리고 'outputs = model((mel, audio))'를 통해 입력 데이터인 멜 스펙트로그램과 오디오를 모델에 전달하여 오디오 출력 값 'outputs'를 얻는다. 이후 'criterion()'손실 함수를 통해 모델 출력 값 'outputs'와 실제 값 사이의 손실(loss)을 계산한다.

4.4 flask 웹 서버 구현

웹 서버는 리눅스 환경에서 구현하였으며, 모델 학습에 이용한 컴퓨터와 동일하다.

NVIDIA-SMI 440.64.00 Driver Version: 440.64.00 CUDA Version: 10.2									
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr.	ECC		
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute	M.		
0	GeForce RTX 2070	Off	00000000:01:00.0	On			N/A		
0%	42C	P8	13W / 215W	7902MiB / 7981MiB	0%		Default		
Processes:									
GPU	PID	Type	Process name				GPU Memory		
							Usage		
0	1074	G	/usr/lib/xorg/Xorg				72MiB		
0	1982	G	compiz				128MiB		
0	4315	C	python				7697MiB		

[그림 4.4-1] 리눅스 웹 서버 환경

OS는 Ubuntu 16.04 그래픽 카드는 RTX 2070, GPU 메모리는 8GB의 환경에서 서버를 구현하였다.

서버에서 하는 일은 크게 세가지로 이루어져 있다. 첫번째, 음성 합성 모델을 불러오는 것, 두번째 안드로이드와 통신하는 것. 세번째, KoGPT의 응답 text를 불러온 모델을 통해 합성하여, 안드로이드로 다시 전송하는 역할을 한다.

```
global curPath

global synthesizer

print('curPath: ', curPath)
if curPath == "":
    synthesizer = Synthesizer()
    synthesizer.load(load_path, 1, None)
    curPath = load_path
    return jsonify({'message': 'model initialized successfully'})
elif curPath == load_path:
    return jsonify({'message': 'model initialized successfully'})
elif curPath != load_path:
    # new_synthesizer = Synthesizer()
    # new_synthesizer.load(load_path, 1, None)
    synthesizer.close()
    synthesizer = Synthesizer()
    synthesizer.load(load_path, 1, None)
    # synthesizer = new_synthesizer
    curPath = load_path
    return jsonify({'message': 'model initialized successfully'})
```

```

@app.route('/text', methods = ['GET', 'POST'])
def get_text():
    text = request.data.decode('utf-8')
    text = get_clean_text(text)
    print(text)
    if text:
        audio = synthesizer.synthesize(
            texts=[text],
            base_path="samples",
            speaker_ids=[0],
            attention_trim=False,
            isKorean=True)[0]
        wav_filename = "out.wav"

        return send_file(wav_filename, as_attachment=True)
    return jsonify({"error": "No text"})

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)

```

[그림 4.4-2] flask 웹 서버 코드

[그림 4.4-2]는 웹 서버 코드이다. GET, POST 를 통해, 안드로이드 앱과 HTTP 통신을 진행한다.

모델을 불러오는 시간을 줄이기 위해, 미리 모델을 load 해 두는 pre-loading 방식을 이용하여 구현하였다. Kogpt와 안드로이드에서 text가 전송이 된다면, 그 text를 바탕으로 음성 합성을 진행하며, 다시 합성된 음성을 안드로이드로 전송해주는 역할을 한다.

4.5 안드로이드 앱 개발



[그림 4.5-1] 안드로이드 앱 순서도

위 그림은 안드로이드 앱의 순서도를 나타낸다. 로그인 화면(첫번째 화면)에서 사용자가 아이디와 비밀번호를 입력하면 대화 상대를 선택할 수 있는 목록 화면으로 넘어가게 된다(두번째 화면). 사용자가 대화를 원하는 상대를 목록에서 선택하게 되면, 해당 상대와 음성으로 대화할 수 있는 대화 화면으로 넘어간다(세번째 화면).

4.5.1 로그인 화면

```
public class LoginActivity extends AppCompatActivity {  
    3 usages  
    private EditText userEdt, passEdt;  
    2 usages  
    private Button loginBtn;  
  
    @SuppressWarnings("MissingInflatedId")  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.acitivity_login);  
        userEdt=findViewById(R.id.editTextTextPersonName);  
        passEdt=findViewById(R.id.editTextTextPassword);  
        loginBtn=findViewById(R.id.LoginBtn);  
        setVariable();  
    }  
    1 usage  
    private void setVariable(){  
        loginBtn.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View view) {  
                if (userEdt.getText().toString().isEmpty() && passEdt.getText().toString().isEmpty()) {  
                    Toast.makeText(context: LoginActivity.this, text: "please fill the form of login", Toast.LENGTH_SHORT).show();  
                } else if(userEdt.getText().toString().equals("android") && passEdt.getText().toString().equals("2018"))  
                    startActivity(new Intent( packageContext: LoginActivity.this, MainActivity.class));  
            }  
        });  
    }  
}
```

[그림 4.5.1-1] LoginActivity 코드

위 코드는 로그인 화면을 구현하는 코드이다. 로그인 버튼을 누를 경우 목록 화면(MainActivity)으로 넘어가도록 구현되었다.

4.5.2 목록 화면

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    binding = ActivityMain2Binding.inflate(getLayoutInflater());
    setContentView(binding.getRoot());
    int[] imageList = {R.drawable.red, R.drawable.blue, R.drawable.black};
    String[] nameList = {"kss", "moon", "junseok"};
    for (int i = 0; i < imageList.length; i++) {
        listData = new ListData(nameList[i], imageList[i]);
        dataArrayList.add(listData);
    }
    listAdapter = new com.example.project.ListAdapter( context: MainActivity.this, dataArrayList);
    binding.listView.setAdapter(listAdapter);
    binding.listView.setClickable(true);
    binding.listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
            Intent intent = new Intent( packageContext: MainActivity.this, DetailedActivity.class);
            intent.putExtra( name: "name", nameList[i]);
            intent.putExtra( name: "image", imageList[i]);
            Tacotron.initModel(nameList[i]);
            startActivity(intent);
        }
    });
}
```

[그림 4.5.2-1] MainActivity의 onCreate() 함수

위 코드는 대화 상대 목록을 표시하는 화면을 구현하는 코드이다. ArrayAdapter를 상속받은 ListAdapter를 이용하여 목록 뷰를 구성하였다. 항목 하나를 누를 경우 대화 화면으로 넘어가게 되며, 넘어가기 전 해당 대화 상대의 pre-trained Tacotron2 모델을 초기화하는 initModel 함수를 호출한다.

```

static void initModel(String name) {
    Retrofit retrofit = new Retrofit.Builder()
        .baseUrl(BASE_URL)
        .client(new OkHttpClient.Builder()
            .connectTimeout( timeout: 30, TimeUnit.SECONDS)
            .readTimeout( timeout: 30, TimeUnit.SECONDS)
            .writeTimeout( timeout: 30, TimeUnit.SECONDS)
            .build())
        .build();

    RetrofitService service1 = retrofit.create(RetrofitService.class);
    RequestBody requestBody = RequestBody.create(MediaType.parse( string: "text/plain"), name);
    Call<ResponseBody> call = service1.getModelInitialized(requestBody);
    call.enqueue(new Callback<ResponseBody>() {
        @Override
        public void onResponse(Call<ResponseBody> call, Response<ResponseBody> response) {
            if (response.isSuccessful()) {
                Log.d(TAG, msg: "model init onResponse: success - " + response.message());
            }
            else {
                Log.d(TAG, msg: "model init onResponse: failed - " + response.message());
            }
        }
    })
}

```

[그림 4.5.2-2] initModel 함수

initModel 함수는 대화 상대의 이름을 인자로 받으며, Flask 서버로 POST HTTP 요청을 보낸다. 이 요청에는 대화 상대의 이름이 body에 포함되어 전송된다. Flask 서버는 이 HTTP 요청을 받으면, 해당 대화 상대의 Tacotron2 모델을 메모리에 초기화시키는 로직을 수행하게 된다. HTTP 통신에는 안드로이드의 Retrofit 라이브러리를 사용하였다. Retrofit은 앱에서 서버와의 HTTP 통신을 효율적으로 구현하기 위한 오픈소스 라이브러리이다.

```

speechRecognizer = SpeechRecognizer.createSpeechRecognizer(context, this);
final Intent speechIntent= new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
speechIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
    RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
speechIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());
speechRecognizer.setRecognitionListener(new RecognitionListener() {
    @Override
    public void onReadyForSpeech(Bundle bundle) {}
    @Override
    public void onBeginningOfSpeech() {
        ViewGroup viewGroup = findViewById(android.R.id.content);
        View dialogView = LayoutInflater.from(context, DetailedActivity.this).inflate(R.layout.alertcustom, viewGroup, attachToRoot: false);
        alertSpeechDialog = new AlertDialog.Builder(context, DetailedActivity.this);
        alertSpeechDialog.setMessage("listening...");
        alertSpeechDialog.setView(viewGroup);
        alertDialog = alertSpeechDialog.create();
        alertDialog.setView(dialogView);
        alertDialog.show();
    }
}

```

[그림 4.5.2-3] DetailedActivity 일부 코드

위 코드는 음성으로 대화할 수 있는 대화 화면을 구현하는 DetailedActivity의 음성 인식 기능 구현 코드이다. SpeechRecognizer 객체를 생성하여 setRecognitionListener를 할당한다. 마이크 이미지를 누르면 SpeechRecognizer 객체의 startListening 함수가 호출되어 사용자가 발화할 경우 음성을 인식할 수 있게 된다.

```

@Override
public void onResults(Bundle bundle) {
    imageView.setImageResource(R.drawable.baseline_mic_24);
    ArrayList<String> arrayList=bundle.getStringArrayList(SpeechRecognizer.RESULTS_RECOGNITION);
    String utterance = arrayList.get(0);
    editText.setText(utterance);
    alertDialog.dismiss();
    if (!utterance.isEmpty())
        Kogpt.sendReq(utterance, findViewById(R.id.kogptResponse));
    else
        Log.d(TAG, msg: "the utterance is empty!");
}

```

[그림 4.5.2-4] DetailedActivity 일부 코드2

유저의 발화가 끝나 음성 인식이 종료될 때 호출되는 onResults 함수에서는 화면의 파란 박스에 해당 발화 텍스트를 표시한다. 그 후 해당 텍스트를 Kogpt 클래스의 sendReq 함수를 통해 카카오의 Kogpt 서버로 POST 요청을 보낸다.


```

static void sendReq(String utterance, TextView targetView) {
    prompt = prompt + "Q: " + utterance + " A: ";
    Log.d(TAG, "msg: "sendReq - prompt: [" + prompt + "]);
    Retrofit retrofit = new Retrofit.Builder()
        .baseUrl("https://api.kakaobrain.com/v1/inference/kogpt/")
        .addConverterFactory(GsonConverterFactory.create())
        .build();
    RetrofitService service1 = retrofit.create(RetrofitService.class);
    PostRequest req = new PostRequest(prompt, MAX_TOKENS, TEMPERATURE, TOP_P, N);
    Call<KogptPostResult> call = service1.createPost( contentType: "application/json", KEY, req);
    call.enqueue(new Callback<KogptPostResult>() {
        @Override
        public void onResponse(Call<KogptPostResult> call, Response<KogptPostResult> response) {
            if (response.isSuccessful()) {
                // UI 작업 가능
                KogptPostResult result = response.body();
                String kogptResponse = result.getGeneration().getText();
                String text = processText(kogptResponse);
                prompt += text;
                targetView.setText(text);
                Log.d(TAG, "msg: "cleaned utterance: " + text);
                Log.d(TAG, "msg: "onResponse: success - " + text);
                Tacotron.send(text);
            }
        }
    });
}

```

[그림 4.5.2-5] sendReq 함수

sendReq 함수도 마찬가지로 Retrofit 라이브러리를 이용하였다. 요청의 헤더에 포함되는 Kogpt 모델의 파라미터로 max_token = 12, temperature = 0.2, top_p = 1.0으로 설정하였다. 또한 프롬프트는 대화 형식의 응답을 받기 위해 "Q: 질문 A: 응답"의 형식으로 구성하였으며, 질문은 유저의 발화를 의미하고, Kogpt는 'A:'이후에 문장을 생성하게 된다. Kogpt가 응답을 생성하고 나서 다시 유저가 발화를 할 때 이전 프롬프트("Q: 질문 A: 응답"의 형태)에 추가하여 프롬프트를 구성하여 Kogpt가 이전 대화의 정보를 활용하여 이후의 응답을 생성하도록 하였다.

```

static String processText(String text) {
    int idx = text.indexOf('Q');
    String sliced;
    if (idx != -1)
        sliced = text.substring(0, idx);
    else
        sliced = text;

    return TextCleaner.getCleanText(sliced);
}

```

[그림 4.5.2-6] processText 함수

Kogpt는 프롬프트의 'A:'이후에 나올 문장을 예측한다. 하지만, 답변 문장 하나만 생성하는 것이 아닌 설정된 토큰 개수 범위 내에서 'Q: 질문 A: 응답'형식의 문장을 계속해서 생성해낼 수 있기 때문에 위와 같은 코드를 이용하여 Kogpt로부터의 응답을 받으면, 'Q'문자 이후의 텍스트는 잘라내어 'A:'에 해당하는 응답 문장 하나만 반환하도록 처리하였다.

Kogpt로부터 응답을 수신하면, 해당 응답 텍스트를 화면에 존재하는 유저의 발화 텍스트 박스 윗부분에 표시하도록 하였다. 그 후, Tacotron 클래스의 send 함수를 이용하여 Kogpt가 생성한 응답 문장을 Tacotron2 서버로 전송하여 음성으로 변환하도록 하였다.

```

static void send(String text) {
    Retrofit retrofit = new Retrofit.Builder()
        .baseUrl(BASE_URL)
        .client(new OkHttpClient.Builder()
            .connectTimeout( timeout: 30, TimeUnit.SECONDS)
            .readTimeout( timeout: 30, TimeUnit.SECONDS)
            .writeTimeout( timeout: 30, TimeUnit.SECONDS)
            .build())
        .build();
    RetrofitService service1 = retrofit.create(RetrofitService.class);
    RequestBody requestBody = RequestBody.create(MediaType.parse( string: "text/plain"), text);
    Call<ResponseBody> call = service1.getAudio(requestBody);
    call.enqueue(new Callback<ResponseBody>() {
        @Override
        public void onResponse(Call<ResponseBody> call, Response<ResponseBody> response) {
            if (response.isSuccessful()) {
                Log.d(TAG, msg: "onResponse: success - " + response.message());
                // UI 작업 가능
                String savedPath = saveAudioFile(response.body(), FILE_NAME);
                playAudio(savedPath);
            }
            else {
                Log.d(TAG, msg: "onResponse: failed - " + response.message());
            }
        }
    })
}

```

[그림 4.5.2-7] send 함수

send 함수는 음성으로의 변환을 원하는 텍스트를 인자로 받아 Retrofit을 이용하여 POST 요청을 생성하고 Flask 서버로 전송한다. 음성 데이터를 수신하게 되면, onResponse 함수를 통해 오디오를 파일로 저장하고 해당 파일을 playAudio 함수를 통해 재생한다.

5. 실험 및 결과 분석

5.1 실험 계획

사용자의 녹음 파일 길이는 가변적이기 때문에 제공되는 녹음 파일의 양이 적을 수도 있다. 하지만 음성 합성을 진행하는데 있어 높은 품질의 합성 결과를 얻기 위해서는 방대한 데이터 양과 높은 녹음 음성 품질이 필요하다. 따라서 제시한 서비스는 데이터양이 풍부하고 품질이 좋은 데이터셋으로 먼저 음성합성 모델을 학습시킨 후 사용자의 녹음 파일을 fine-tuning을 통해 추가 학습시켜 합성 품질을 향상시키고자 한다. KSS는 전문 성우가 일상적인 대화를 주제로 녹음을 진행한 12시간 분량의 데이터 셋을 제공해주고 있기 때문에 이에 적합하다고 판단하였다. 따라서 본 연구는 먼저 KSS를 이용하여 실험을 진행하여 그 결과를 분석한다.

사용자가 전문 성우가 아닌 일반적인 사람의 목소리가 담긴 녹음 파일을 제공하는 상황을 가정하기 위해 팀원 중 한 명인 김준석 학생의 목소리를 직접 녹음하여 두번째 실험 데이터셋으로 사용했다. 이 데이터셋을 이용하여 앞서 KSS로 학습시킨 모델을 fine-tuning을 통해 추가 학습시켜 실험을 진행한 후 그 결과를 통해 합성된 음성의 품질을 분석한다.

마지막으로, 사용자가 유명인의 목소리가 담긴 녹음 파일을 제공하는 상황을 가정하기 위해 문재인 전 대통령의 목소리가 담긴 음성을 수집하여 데이터셋을 제작하였다. 문재인 전 대통령을 선정한 이유는 다른 유명인에 비해 상대적으로 데이터 수집이 용이했기 때문이다.

따라서 본 연구에서는 KSS, 김준석 학생, 문재인 전 대통령의 3개의 음성 데이터셋을 이용하여 음성 기반 대화 시스템의 효용성을 실험한다. 실험은 다음과 같이 음성 합성 모델의 합성 품질, 합성 속도를 측정한다.

음성 합성 모델의 합성 품질 측정을 위해 각 데이터셋을 이용하여 음성 합성 모델을 학습한 후, train과 validation 과정에서 생성된 노이즈 attention 그래프를 비교하는 방식으로 overfitting 여부를 판단한다. 그리고 inference의 결과로 생성된 노이즈 attention 그래프를 바탕으로 임의의 문장에 대한 합성의 품질을 판단한다. Inference에 사용한 문장은 모두 “안녕하세요 저는 한국외국어대학교 학생입니다.”로 동일하다. 또한 학습의 결과로 생성된 wav 파일을 직접 청취하여 정성적 평가를 진행한다.

합성 속도는 사용자가 서비스를 이용하면서 발생하는 딜레이 즉, 사용자의 발화 후 응답 음성이 출력되기까지의 시간을 포함한다. 이를 측정하기 위해 Flask 서버와 Kogpt 서버를 연동하여 서비스를 구현한 안드로이드 앱에서 KSS, 김준석, 문재인 전 대통령 데이터셋을 사전 학습한 각 Tacotron2 모델을 이용하여 서비스 이용 테스트를 진행하였다. 테스트를 하는 과정은 다음과 같다.

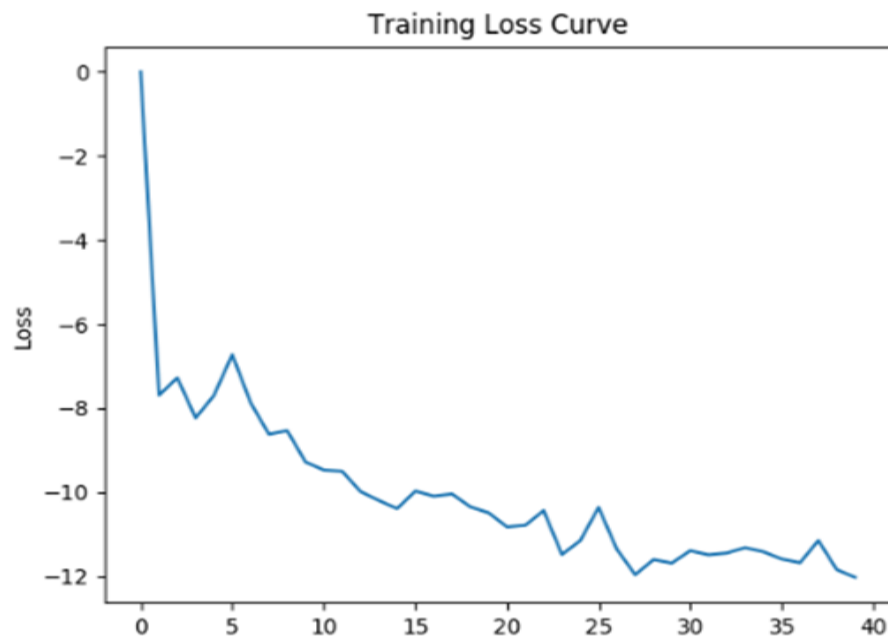
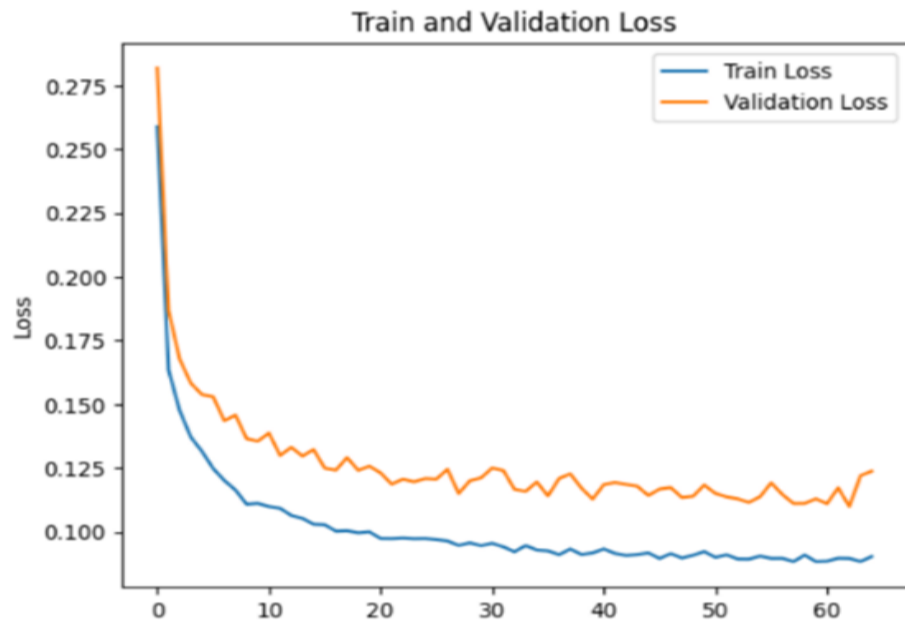
1. 앱에 로그인한 후, 대화상대 목록에서 KSS, 김준석, 문재인 전 대통령 항목 중 하나를 누른다.
2. 해당 모델이 초기화될 때까지 대기한다.
3. 마이크를 눌러 질문 문장을 발화한다.
4. 음성이 출력될 때까지 기다린다. 이때, 발화 후 음성이 출력되기까지의 시간(초)을 측정한다.

5.2 실험결과

	Tacotron	WaveGlow
Total train step	65000 steps	40000 steps
Train loss(MSE+BCE)	0.081	-12.096
Validation loss(Negative log-likelihood)	0.112	

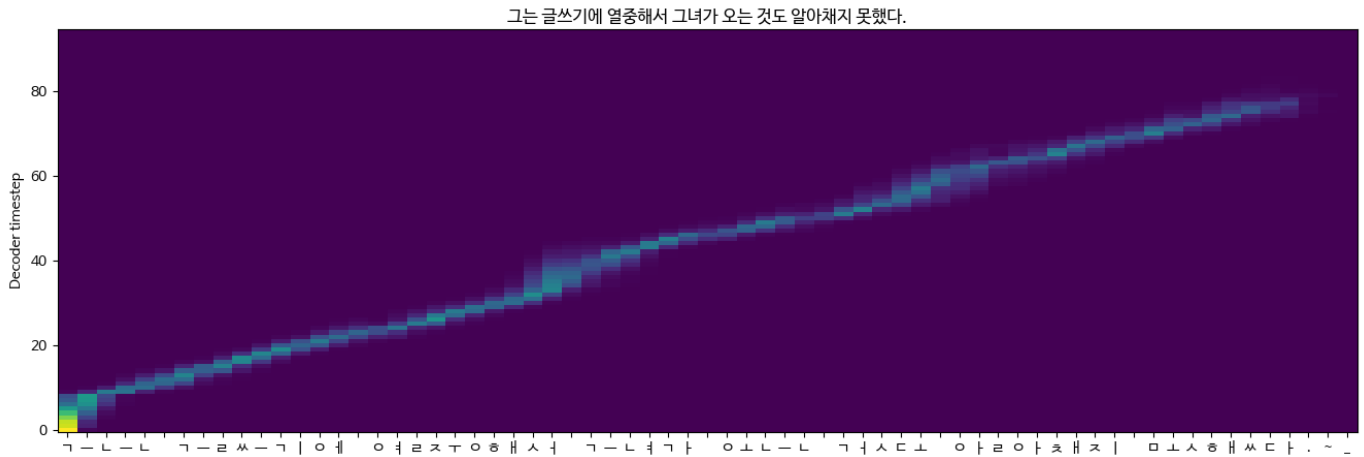
[표 5.2-1] KSS 데이터셋 학습 결과1

[표 5.2-1]는 KSS 데이터셋으로 학습시킨 두 모델의 학습 후 결과를 나타낸다. 학습에 소요된 시간은 Tacotron 모델 약 25시간, WaveGlow 모델 약 20시간, 총 약 45시간이며 Tacotron과 WaveGlow 각각 총 65000번, 40000번의 train step을 거쳤다. Tacotron의 loss값은 train loss가 0.081, validation loss가 0.112이며, WaveGlow의 train loss값은 -12.096로 나타났다.

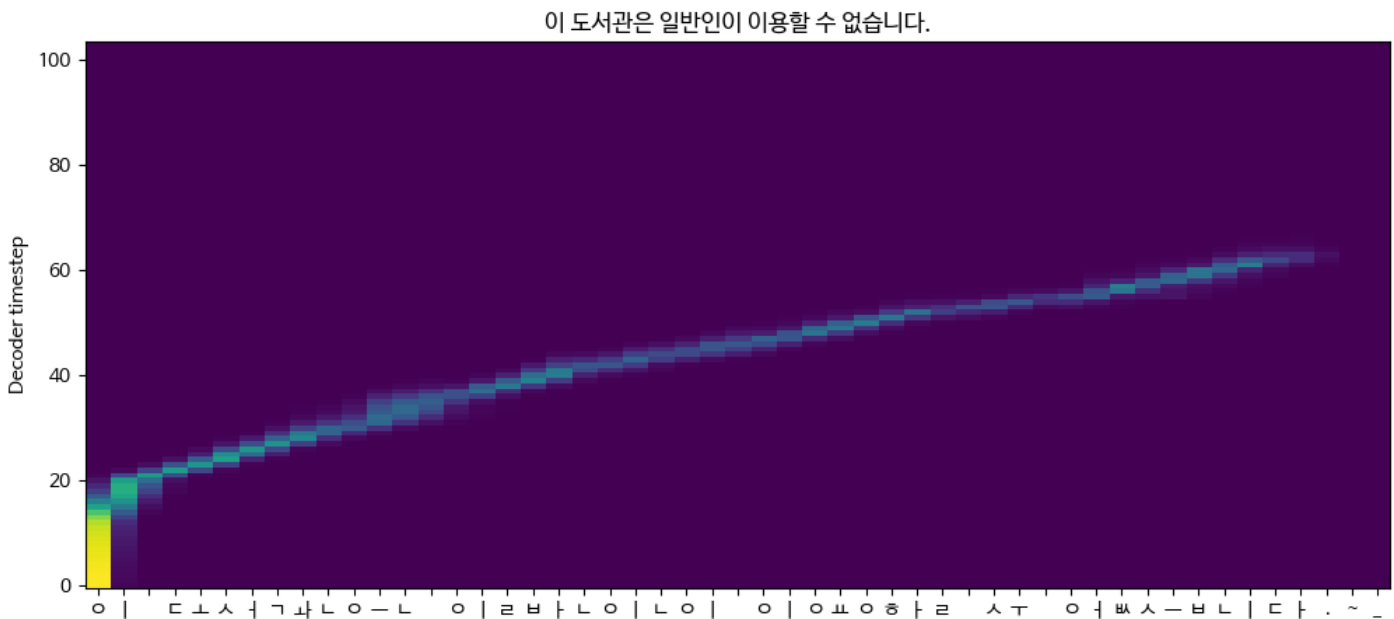


[그림 5.2-1] KSS 데이터셋 학습 결과2

[그림 5.2-1]은 KSS 데이터셋의 학습 후 두 모델의 loss 그래프를 보여준다. 위 그림에서 위쪽 그래프는 Tacotron의 loss 그래프, 아래쪽은 WaveGlow의 loss 그래프를 나타낸다.

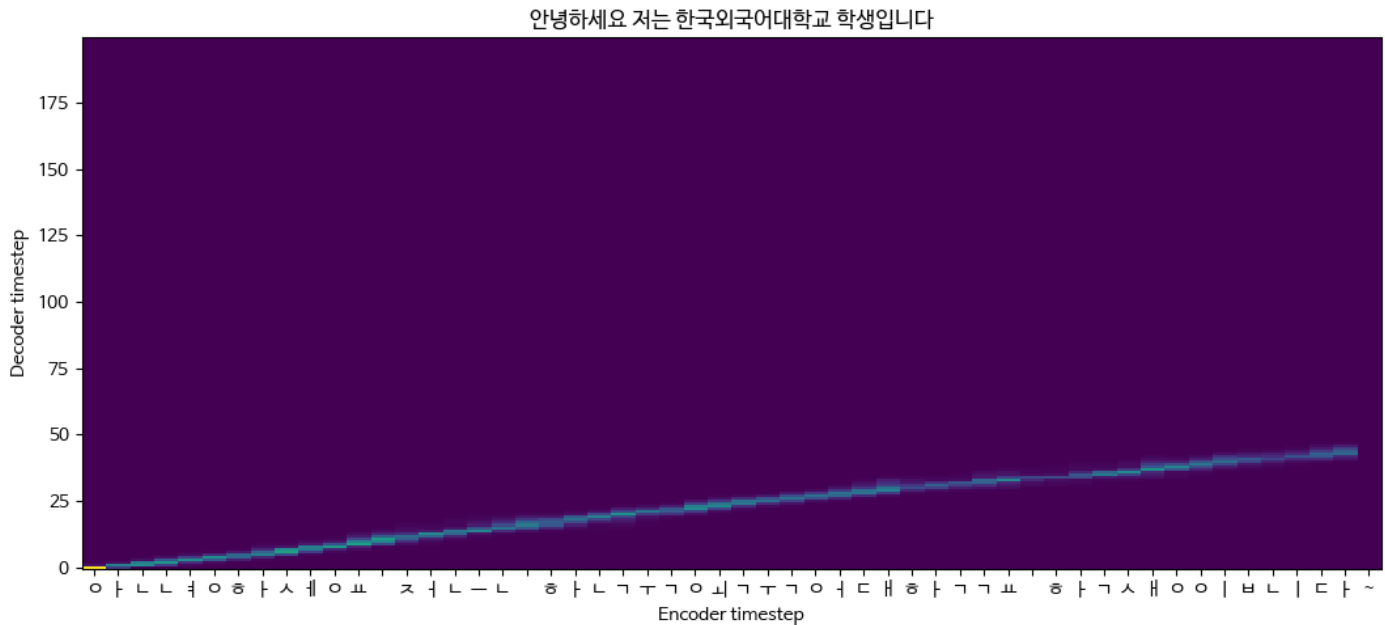


[그림 5.2-2] KSS 데이터셋 train 과정에서의 노이즈 attention 그래프



[그림 5.2-3] KSS 데이터셋 validation 과정에서의 노이즈 attention 그래프

위 두 [그림 5.2-2], [그림 5.2-3]에 표현된 그래프는 마지막 train step에서의 KSS 데이터셋 train, 그리고 validation에 대한 노이즈 attention 그래프를 보여준다. 두 그래프 모두 비슷한 두께와 색을 가졌음을 볼 수 있으며 직접 합성된 음성을 들어봤을 때, 음질이 깨끗하고 발음이 잘 들리는 것으로 보아 모델의 일반화가 잘 되었음을 확인할 수 있었다.



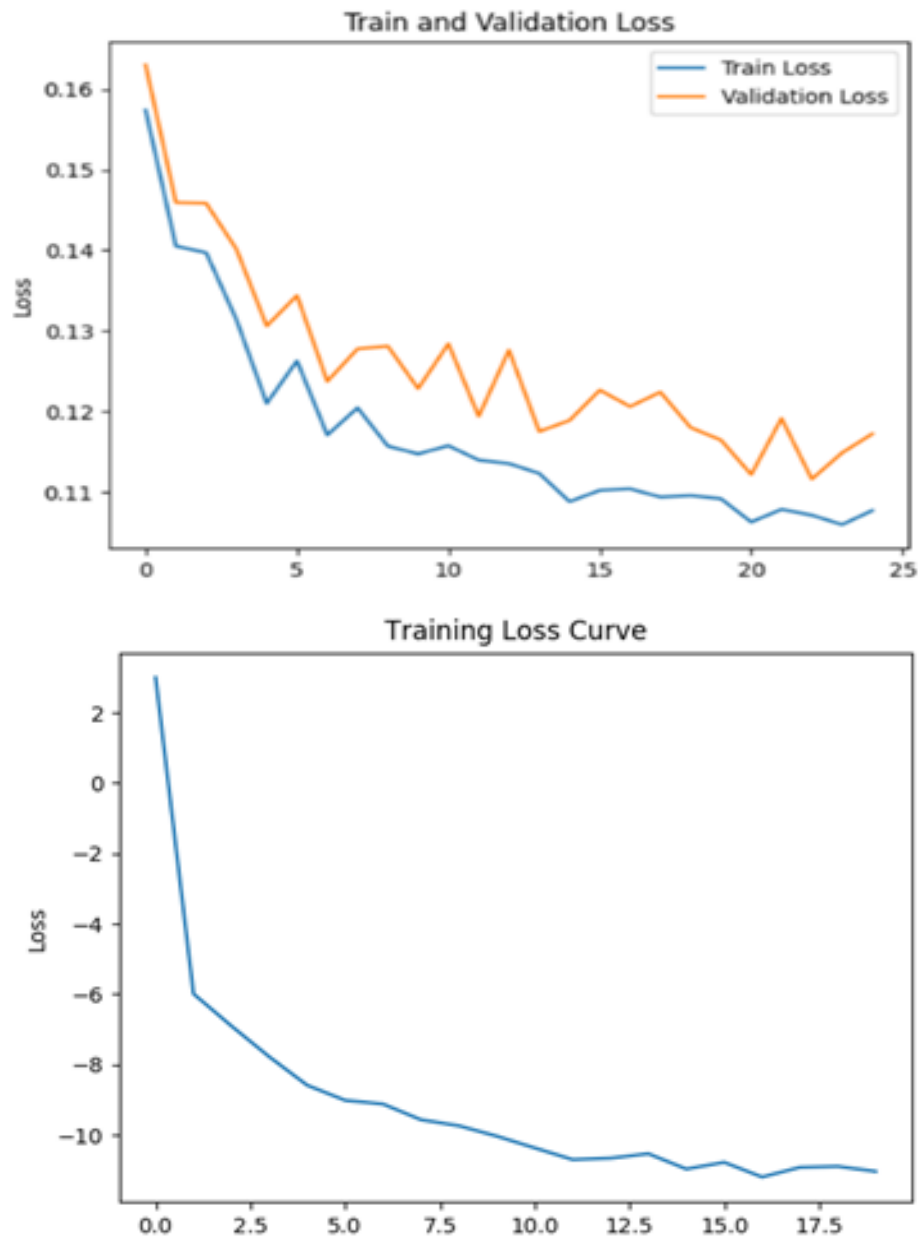
[그림 5.2-4] KSS 데이터셋 inference 노이즈 attention 그래프

[그림 5.2-4]은 “안녕하세요 저는 한국외국어대학교 학생입니다”라는 문장을 합성했을 때의 노이즈 attention 그래프이다. 보이는 것과 같이 일직선의 일정한 두께를 가진 모양이며 직접 합성된 음성을 들어봤을 때도 음성이 잘 합성되었음을 확인하였다. 위 결과를 토대로, 음성 합성이 잘 진행된 경우 어텐션 그래프의 모양은 우상향 그래프이며, 선의 두께는 일정하고, 각 문장의 음소 ㅇ, ㅏ, 등이 encoder timestep 에 맞게 잘 들어간 것을 확인 할 수 있다.

	Tacotron	WaveGlow
Total train step	25000 steps	20000 steps
Train loss(MSE+BCE)	0.105	-11.185
Validation loss(Negative log-likelihood)	0.116	

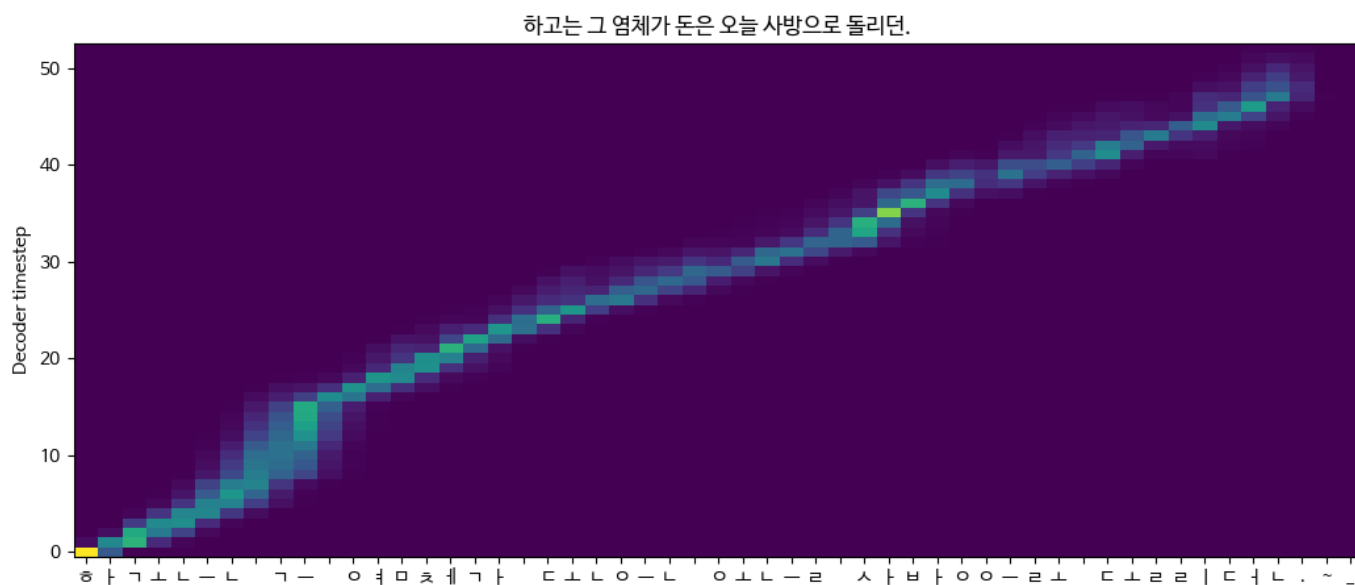
[표 5.2-2] 김준석 데이터셋 학습 결과¹

[표 5.2-2]는 김준석 데이터셋으로 학습시킨 두 모델의 학습 후 결과를 나타낸다. 이 데이터셋의 경우 앞서 KSS 데이터셋으로 학습시킨 Tacotron2 모델을 이용하여 fine-tuning을 진행하였다. 학습에 소요된 시간은 Tacotron 모델 약 10시간, WaveGlow 모델 약 10시간, 총 약 20시간이며 Tacotron과 WaveGlow 각각 총 25000번, 20000번의 train step을 거쳤다. Tacotron의 loss값은 train loss가 0.105, validation loss가 0.116이며, WaveGlow의 train loss값은 -11.185로 나타났다.

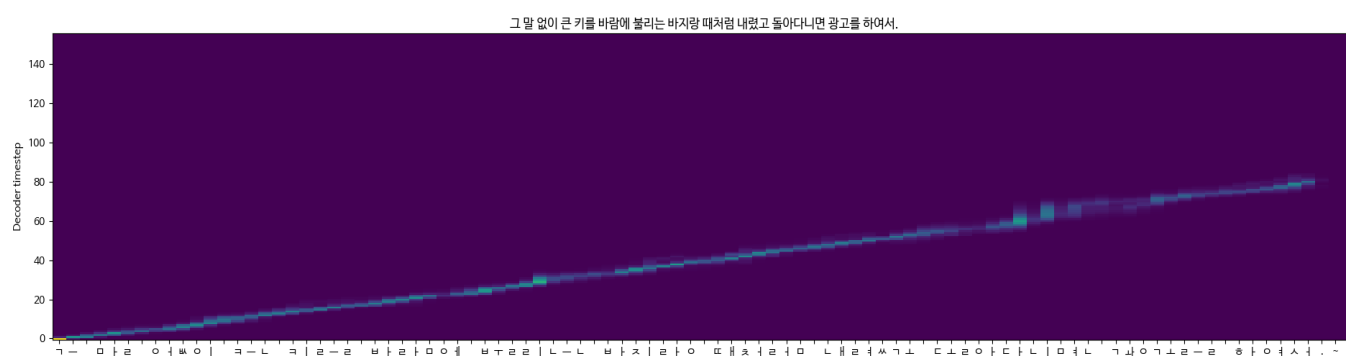


[그림 5.2-5] 김준석 데이터셋 학습 결과2

[그림 5.2-5]은 김준석 데이터셋의 학습 후 두 모델의 loss 그래프를 보여준다. 마찬가지로 위 그림에서 위쪽 그래프는 Tacotron의 loss 그래프, 아래쪽은 WaveGlow의 loss 그래프를 나타낸다.

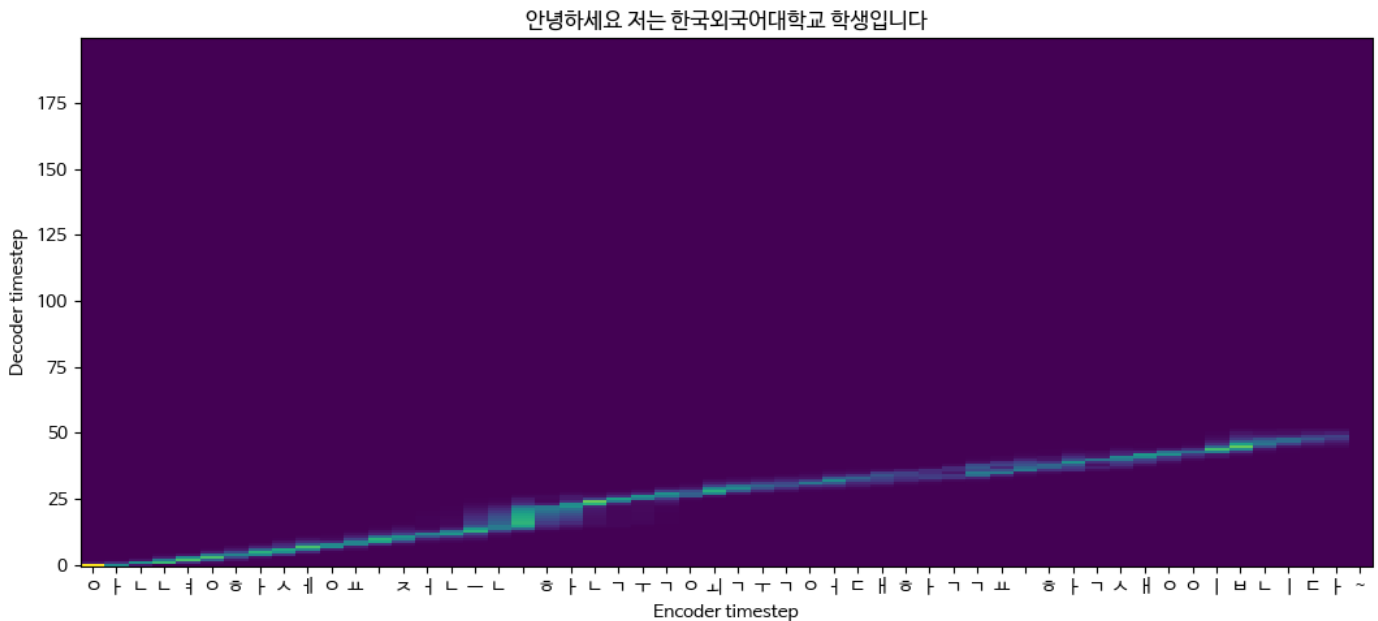


[그림 5.2-6] 김준석 데이터셋 train 과정에서의 노이즈 attention 그래프



[그림 5.2-7] 김준석 데이터셋 validation 과정에서의 노이즈 attention 그래프

[그림 5.2-6]과 [그림 5.2-7]은 마지막 train step에서의 김준석 데이터셋 train, 그리고 validation에 대한 노이즈 attention 그래프를 보여준다. Train 과정의 경우 색이 뚜렷한 모습을 볼 수 있지만 validation 과정의 경우 색이 비교적 옅은 모습을 볼 수 있으며 일정 부분에서는 노이즈가 많이 포함된 모습도 볼 수 있었다. 직접 합성된 음성을 들어봤을 때도 validation 데이터의 경우 KSS에 비해 음성 품질이 다소 떨어지는 모습을 확인하였다.



[그림 5.2-8] 김준석 데이터셋 inference 노이즈 attention 그래프

[그림 5.2-8]은 “안녕하세요 저는 한국외국어대학교 학생입니다”라는 문장을 합성했을 때의 노이즈 attention 그래프이다. 보이는 것과 같이 중간 부분에 색이 열리는 모습을 확인할 수 있다. 정성 평가에서는 KSS에 비해 음성 품질은 다소 떨어지지만 큰 노이즈 없이 음색이 잘 반영되었고 발음도 비교적 정확하게 들리는 것을 확인하였다.

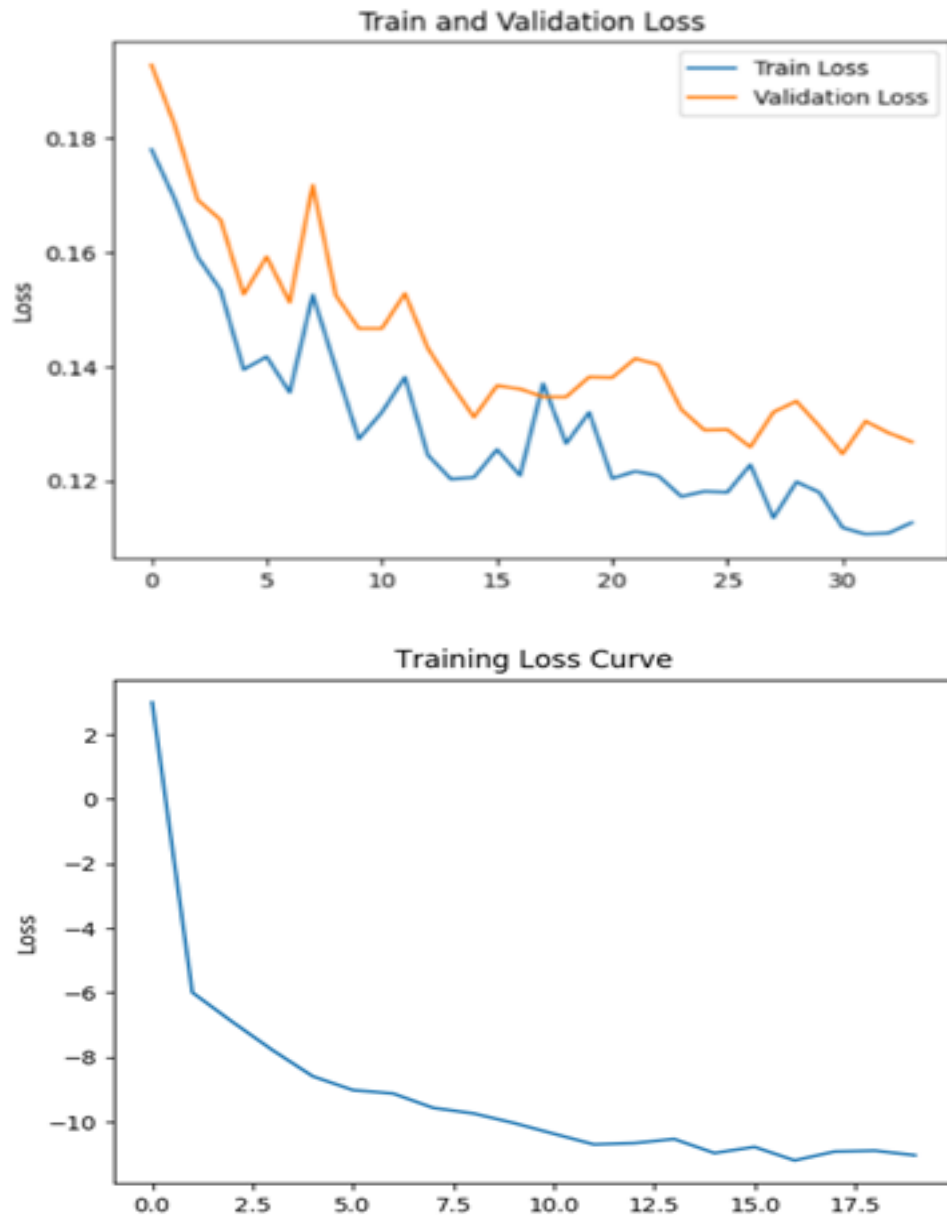
이를 위의 KSS데이터 셋과 비교한 결과를 살펴보면, 음성 합성이 비교적 정확한 부분은 우상향의 그래프의 형식과 선의 두께 또한 일정한 것을 확인하였다. 하지만 일부분 음성이 불안한 ‘전’과 ‘한’ 사이의 부분의 그래프 모양이 희미하고 번져있는 모습을 확인할 수 있으며, 이를 토대로 합성이 잘 안된 부분의 경우 그래프가 번지는 형태임을 확인할 수 있었다.

이러한 결과가 나타난 이유는, KSS와의 데이터 셋 양과 질의 차이에서 비롯된 것이라고 판단된다. 그 이유는 KSS데이터 셋은 약 12시간이고, 전문 성우가 일정한 문장을 읽기 때문에 데이터 셋 문장 전체로 보면 대부분의 한국어 음소들이 적당하게 들어가 있었지만 김준석 데이터 셋의 경우 데이터 셋 양의 크기는 약 2~3시간 정도이며, 데이터 셋을 모은 과정 역시 일상적인 대화가 아닌 소설 책의 내용이기 때문에, 학습되지 못한 음소가 있으며, 그 학습되지 못한 음소를 음성합성으로 생성하려 할 경우 부정확한 음성이 들리는 것으로 판단된다.

	Tacotron	WaveGlow
Total train step	34000 steps	30000 steps
Train loss(MSE+BCE)	0.109	-10.102
Validation loss(Negative log-likelihood)	0.132	

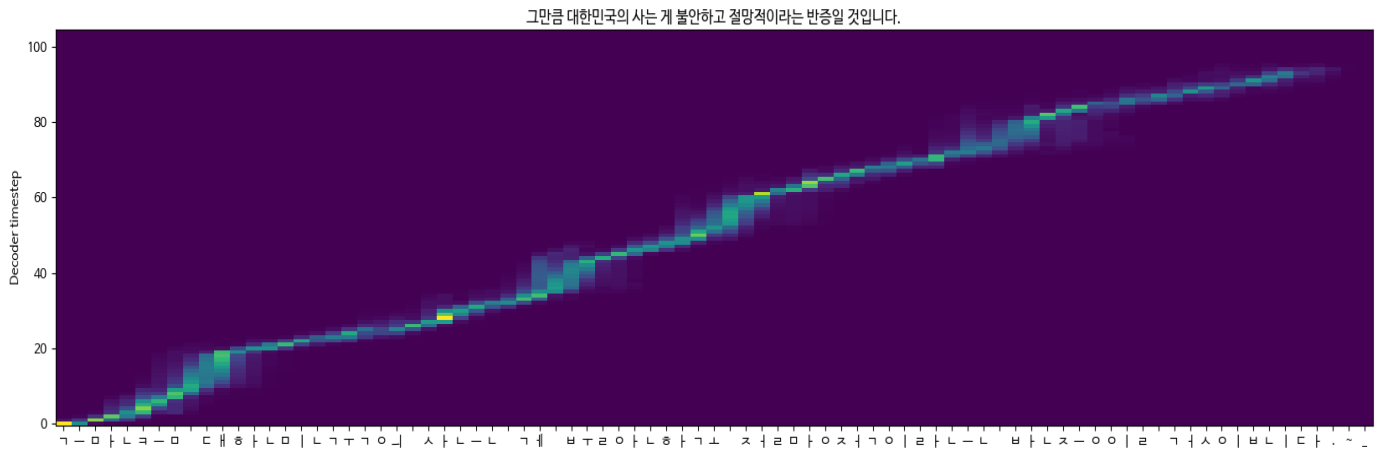
[표 5.2-3] 문재인 전 대통령 데이터셋 학습 결과¹

[표 5.2-3]는 문재인 전 대통령 데이터셋으로 학습시킨 두 모델의 학습 후 결과를 나타낸다. 김준석 데이터셋의 경우와 마찬가지로, 앞서 KSS 데이터셋으로 학습시킨 Tacotron2 모델을 이용하여 fine-tuning을 진행하였다. 학습에 소요된 시간은 Tacotron 모델 약 13시간, WaveGlow 모델 약 15시간, 총 약 28시간이며 Tacotron과 WaveGlow 각각 총 34000번, 30000번의 train step을 거쳤다. Tacotron의 loss값은 train loss가 0.109, validation loss가 0.132이며, WaveGlow의 train loss값은 -10.102로 나타났다.

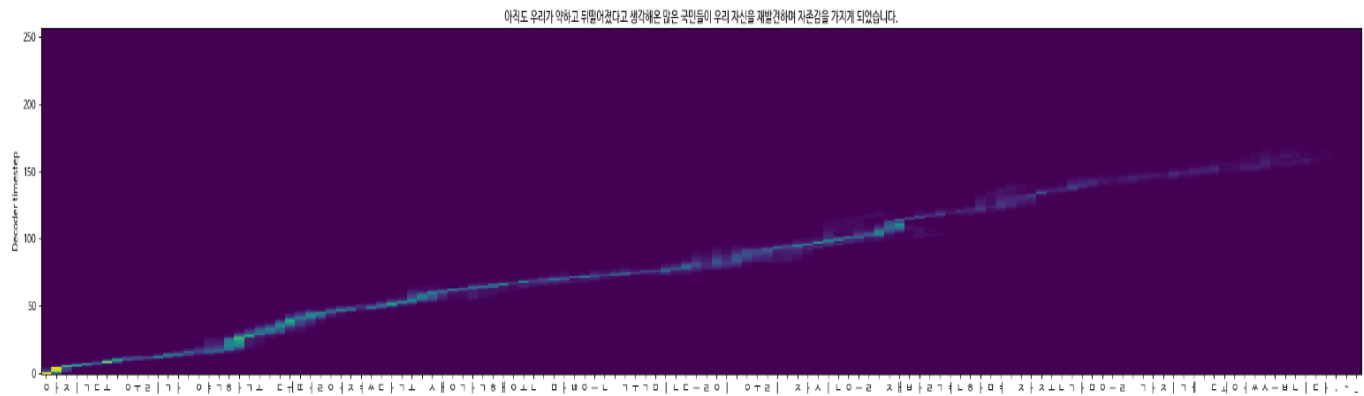


[그림 5.2-9] 문재인 전 대통령 데이터셋 학습 결과2

[그림 5.2-9]은 문재인 전 대통령 데이터셋 학습 시 두 모델의 loss 그래프를 보여준다. 마찬가지로 위 그림에서 위쪽 그래프는 Tacotron의 loss 그래프, 아래쪽은 WaveGlow의 loss 그래프를 나타낸다.

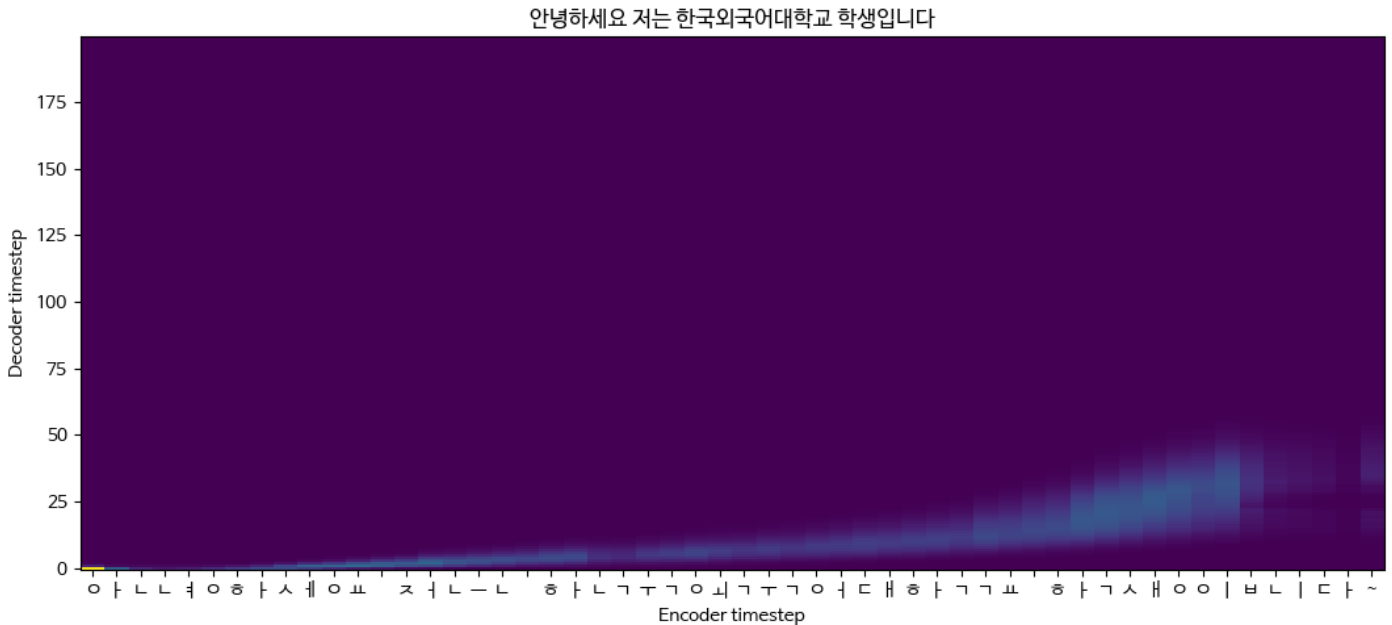


[그림 5.2-10] 문재인 전 대통령 데이터셋 train 과정에서의 노이즈 attention 그래프



[그림 5.2-11] 문재인 전 대통령 데이터셋 validation 과정에서의 노이즈 attention 그래프

[그림 5.2-10]과 [그림 5.2-11]은 각각 마지막 train step에서의 문재인 전 대통령 데이터셋의 train, 그리고 validation에 대한 노이즈 attention 그래프를 보여준다. Train 과정의 경우 validation보다 색이 뚜렷한 모습을 볼 수 있지만 중간 어구 사이마다 두께가 두껍고 색이 열리는 모습을 볼 수 있다. validation 과정의 경우 색이 후반부로 갈수록 열리는 모습을 볼 수 있으며 직접 합성된 음성을 들어봤을 때도 노이즈가 많이 추가되어 KSS 혹은 김준석 데이터셋에 비해 음성 품질이 떨어지는 모습을 확인하였다. 따라서 문재인 전 대통령 데이터셋의 경우 overfitting의 정도가 심한 것을 볼 수 있다.



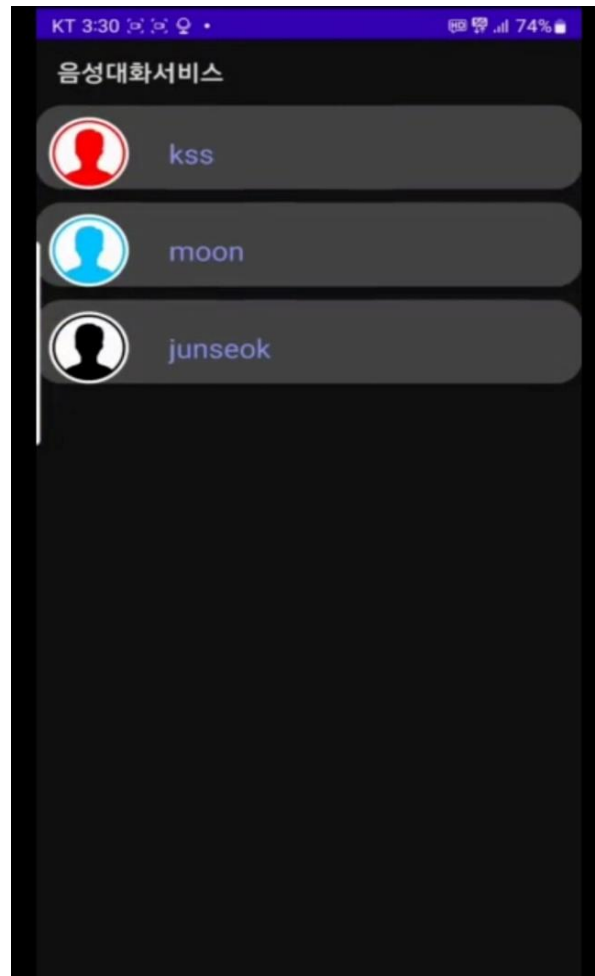
[그림 5.2-12] 문재인 전 대통령 데이터셋 inference 노이즈 attention 그래프

[그림 5.2-12]는 “안녕하세요 저는 한국외국어대학교 학생입니다”라는 문장을 합성했을 때의 노이즈 attention 그래프이다. KSS, 김준석 데이터셋과 다르게 음성 품질이 많이 떨어짐을 확인할 수 있다. 직접 합성된 음성에서 노이즈가 대부분인 음성을 들을 수 있었으며 발음 식별에 어려움을 겪었다.

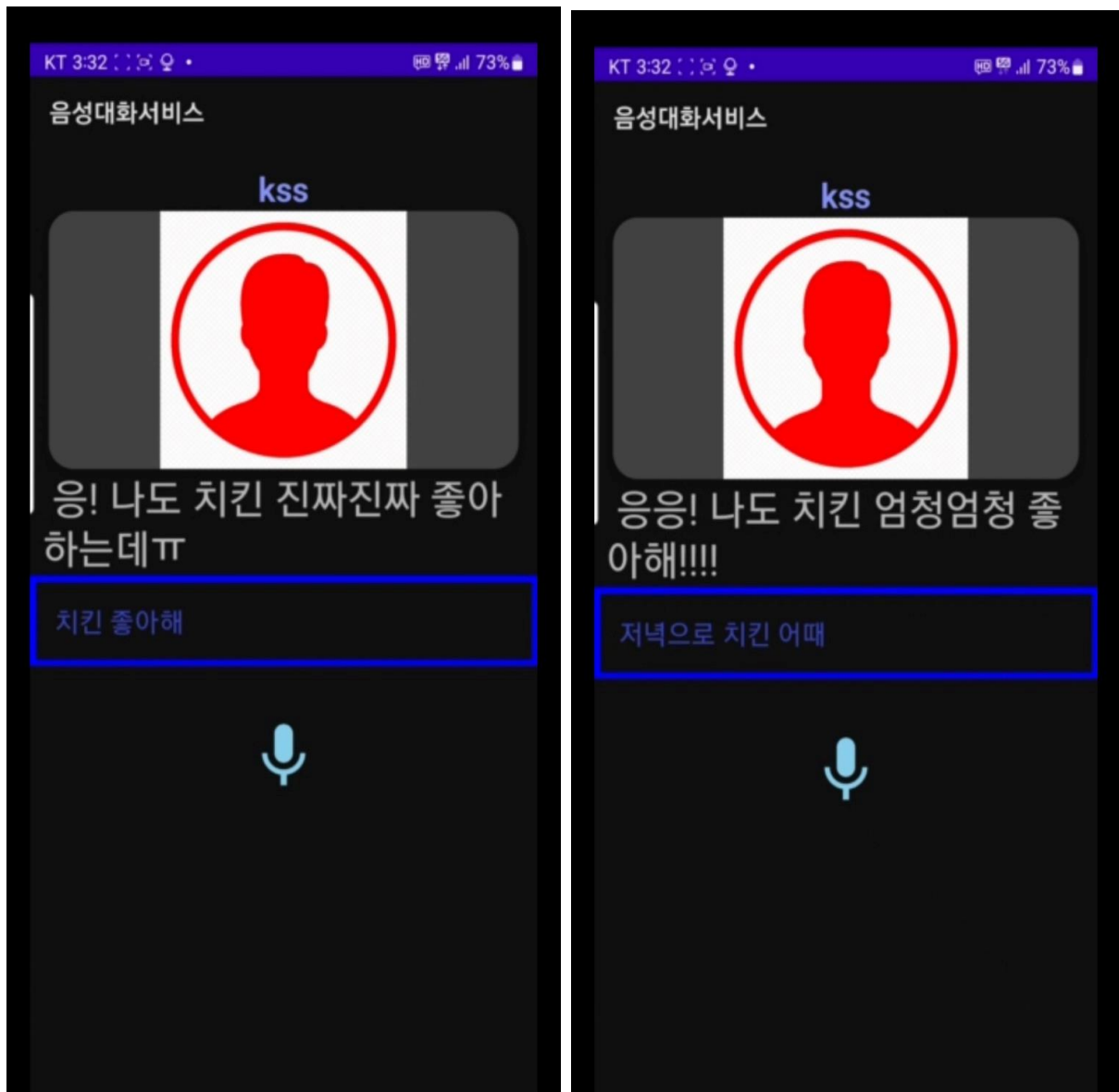
위 3개의 결과를 통해 모델의 마지막 체크포인트 파일을 이용하여 음성 합성을 진행했을 때 KSS, 김준석 데이터셋의 경우 화자의 음색이 비교적 잘 반영되어 자연스러운 음성이 합성됨을 확인할 수 있었으며, 문재인 전 대통령의 경우 노이즈가 많이 추가되는 모습을 보였다. 또한 음색 정보가 잘 반영이 안되고 많은 단어의 학습이 잘 안된 결과를 확인할 수 있었다.

위 결과를 토대로, 문재인 전 대통령 데이터셋의 경우 validation 과정에서는 중간 중간에 어눌하고 말이 잘 들리지 않는 문제가 발생하였으며, test 과정에서는 노이즈가 대부분인 음성을 습득하게 되었다. 이러한 문제가 발생한 이유는 김준석 목소리의 경우와 마찬가지로 데이터 셋의 문제라고 생각되는데, 문재인 대통령의 연설, 정치 유세 등 특정 주제에 편향된 주제의 대화 셋만 구할 수 있었고, 그에 따라 반복되는 단어만 지속적으로 나왔던 것으로 판단된다. 그에 따라 첫 번째 학습되지 못한 음운 음소의 수가 다른 데이터 셋에 비해 많았으며, 두 번째로는 train dataset에도 지속적으로 단어가 반복되다 보니, test dataset 안의 단어들만 학습이 되는 상황이 발생한 것으로 보인다. 그에 따라, 모든 모델 체크 포인트를 확인해본 결과, 18500번째 train step에서 저장된 model의 체크포인트가 오히려 성능이 가장 뛰어났으며, 일상적인 주제가 아닌 정치관련 이야기를 진행해 보았을 때, 소리가 더 잘 들리는 것을 확인할 수 있었다.

위 실험 결과를 정리하면, KSS 데이터 셋은 비교적 깔끔한 음성과 더불어 대부분의 음운 음소가 잘 학습된 모습을 확인할 수 있었으며, 김준석 데이터 셋은 데이터 셋 양과 질 부족으로 인한 일정 음소의 누락이 발생하여, 특정 발음이 잘 들리지 않는 문제가 발생했다. 마지막으로, 문재인 데이터 셋의 경우, 데이터 셋의 발화가 특정 주제에 대해 몰려있고, 따라서 특정 단어 혹은 특정 주제가 반복되어, 다른 데이터 셋보다 오버피팅이 심하고 그에 따라, 학습되지 못한 음소 수도 많다는 것을 알 수 있었다.

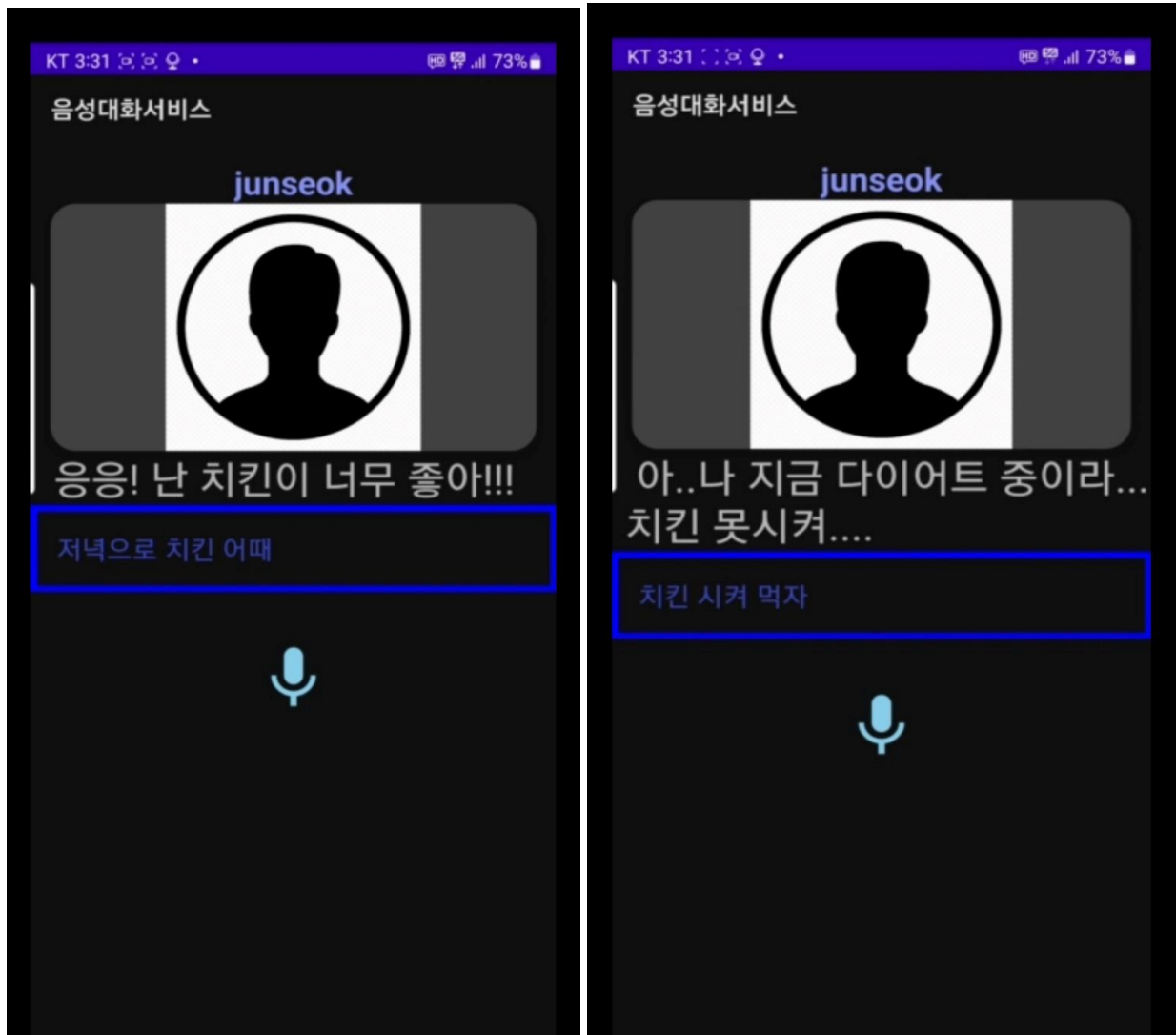


[그림 5.2-12 최종 완성 애플리케이션 1]

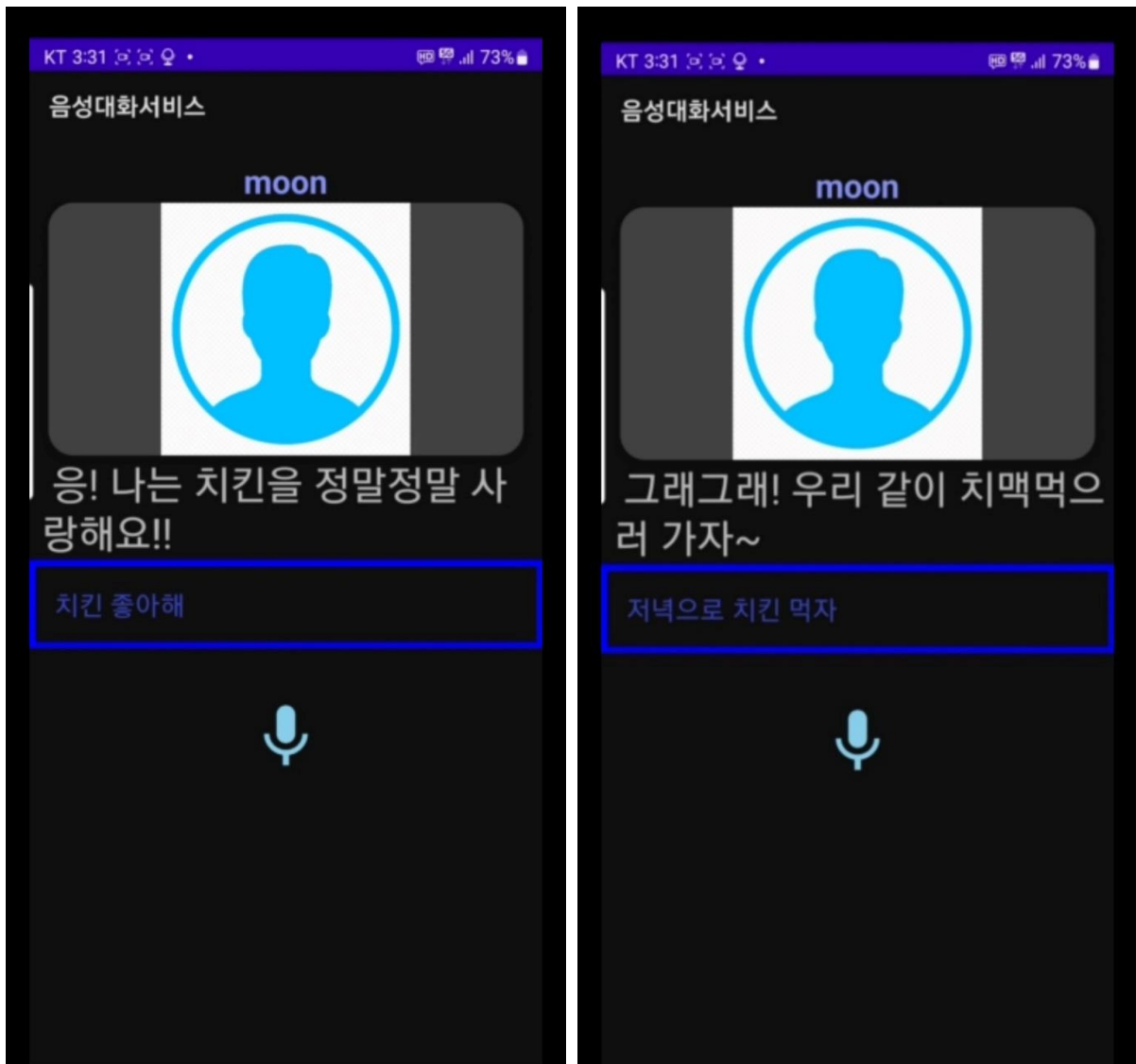


[그림 5.2-13 최종 완성 애플리케이션 2]

완성된 앱의 모습은 [그림 5.2-12], [그림 5.2-13]과 같다. 대화를 하고싶은 사람(목소리)를 선택하면 모델을 로딩한 후에, 대화를 할 수 있는 화면으로 이동한다. 마이크 버튼을 누르면 사용자의 대화를 듣고 STT를 이용하여, 파란 박스 안에 사용자의 발화를 스크립트로 변환하고 그 위에, KoGPT의 응답이 출력된다. 그 응답 text를 Tacotron2 학습 모델이 저장 되어있는 리눅스 서버로 전송하고, 서버에서 학습된 모델을 이용하여 음성 합성을 진행한 후, 완료된 wav파일을 다시 안드로이드 앱으로 전송하여 wav파일을 재생시키는 방식으로 대화 서비스가 완성된다.



[그림 5.2-14 최종 완성 애플리케이션 3]



[그림 5.2-15 최종 완성 애플리케이션 4]

또한 합성 속도를 측정하기 위한 과정에서 사용자의 발화 후 음성이 출력될 때까지의 시간을 재본 결과, 세 모델 모두 사용자의 발화 후 평균 약 20초의 딜레이가 발생하고 음성이 출력되었다.

이 딜레이 시간을 줄이기 위한 방안으로 모델을 preloading한 후 음성을 합성하도록 하였다. 즉, 유저가 질문을 발화하고 해당 발화에 대한 응답 텍스트를 Kogpt가 생성한 후 이 텍스트를 Tacotron2 모델이 음성으로 합성해낼 때, 합성할 때마다 모델을 초기화시키는 것이 아닌, 대화상대를 선택할 때 미리 해당 모델을 초기화 시킨 상태에서 이후의 음성 합성을 진행하도록 하였다.

해당 방식으로 위의 실험을 다시 수행한 결과, 세 모델 모두 사용자의 발화 후 평균 약 5초의 딜레이가 발생하여 15초 정도 감소한 결과를 보였다.

6. 결론 및 향후 연구 방향

본 연구에서는 음성 합성을 이용하여 원하는 사람과 대화할 수 있는 앱을 개발하였다. 서비스 측면에서는, 가장 핵심적인 기능인, 이제 들을 수 없는 사람과 대화하는 기능을 중점적으로 구현하였으며, 음성 합성 부분에 대해서는 최대한 합성 시간이 빠르고 합성 품질 또한 우수한 모델을 선정하는 방향으로 진행하였다.

음성 합성의 품질은 최대한 높이고, 학습 시간을 줄이기 위해 데이터 셋이 가장 많았던 KSS데이터 셋을 이용하여 fine-tuning을 이용하는 방식으로 학습을 진행하였다. 그 결과 KSS, 김준석 목소리 데이터 셋에서는 합성된 음성의 품질이 좋음을 확인하였으나, 문재인 전 대통령 녹음 파일의 경우 오버 피팅이 발생함을 확인하였다.

Overfitting이 생긴 이유로는 크게 두가지 이유로 나눌 수 있는데, 첫 번째 이유는 전처리 과정에서 음성 분할 시 공백을 기준으로 자르다 보니, 분할된 데이터 당 길이 차이가 너무 심해 데이터 간 불균형으로 생긴 문제로 생각하며, 전처리 과정을 일정한 길이로 자른 후 맨 앞과 맨 뒤에 백색 잡음과 불안정한 음성을 제거하는 방식으로 바꾸는 방향으로 전처리를 다시 하고 학습을 진행시킬 예정이다.

두번째 이유는 데이터셋 자체의 문제이다. 유명한 목소리를 하나 넣음에 따라, 문재인 전 대통령의 목소리를 합성하기로 결정 후 합성을 진행하였다. 이 때 사용한 모든 음성 파일은 정치 관련 연설, 혹은 새해 인사 관련 내용이다 보니 정치 관련 문장들만 삽입 되어있었다. 하나의 주제에 대해서만 반복해서 이야기하다 보니 겹치는 단어들이 많았고, 그에 따라 학습되지 못하는 음운이 다수가 존재하게 됐다는 것이 그 이유이다. 이 경우 처음 학습시킬 때 문재인 대통령 연설을 모두 학습데이터에 이용하였기 때문에, 추가 데이터 셋을 추가로 구하기는 어려운 상황이며, 따라서 추가적으로 다른 사람의 목소리를 학습시키는 방법으로만 해결이 가능 할 것 같다. 추가로 KSS데이터 셋 같은 경우에도 졸업 프로젝트 시연영상에도 나왔듯이, 치킨이라는 단어에서 불분명한 발음이 나온 것으로 보아, 데이터 셋으로 인한 학습되지 못하는 음운이 존재한다라는 것을 알 수 있었다.

또한, 이 앱에서 몇 가지 아쉬운 점과 한계점이 존재한다.

첫째, 음성 합성 모델에서의 한계점이다. tacotron1, tacotron2 + waveglow, WaveNet 세개의 모델을 비교하며, 성능은 비슷하고, 합성 속도는 다른 모델보다 더 빠른 tacotron2 + waveglow 모델을 이용하였으나, 모델 합성 시간의 소요 시간이 약 5초로 실시간 서비스라고 계획했던 것과는 다르게, 일정시간이 지나야 답변을 받을 수 있다는 한계점이 존재한다. 따라서 음성 합성 시간을 조금 더 줄일 수 있는 방안을 찾고, 이를 기준으로 서비스를 발전시킬 생각이다.

두번째 한계점은 서비스 상에서의 한계이다. 음성 합성을 진행하기 위해서는 원본 파일에 잡음이 최대한 적게 들어가야 음성 합성 결과가 잘 나오게 된다. 하지만, 원본 파일에서 잡음이 심한 경우 음성 합성 서비스를 제공할 수 없다는 단점이 존재하며, 이에 따라 서비스의 폭이 많이 제한되는 단점이 존재한다. 또한 핵심 기능인 원하는 사람과 대화하는 기능까지는 구현했으나, 원하는 사람의 음성을 학습시켜 서버에 올리는 기능은 안드로이드 앱에 추가를 하지 못하였다.

향후 연구는, 음성 합성을 바탕으로, 조금 더 적은 양의 데이터, 추가로 더 좋은 품질의 음성 합성이 가능하도록 하는 연구가 필요하다. 이런 연구를 바탕으로, 더 많은 사람들에게 다양한 서비스를 제공할 수 있으며, 이러한 방향의 연구 및 프로젝트는 실 생활에서 많은 사람들에게 도움을 줄 것으로 생각된다.

7. 참고문헌

- [1] 신동현. "3D 로그 멜-스펙트로그램에 dilated CNN과 attention based sliding LSTM을 적용한 음성 감정 인식." 국내석사학위논문 한양대학교 대학원, 2020. 서울
- [2] "Understanding the Mel Spectrogram", Medium, <https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53>
- [3] "STT(Speech-to-Text)", Tistory, <https://woongsin94.tistory.com/333>
- [4] 손원섭, 김응곤, "음성 인식을 이용한 자막 자동 생성 시스템 (Subtitle Automatic Generation System using Speech to Text)", 한국전자통신학회 논문지, pp 81-84, Aug 2021
- [5] 이건수, 김중연, "음성 기반 상담의 품질 평가를 위한 자동화 기법 (A Method of Automated Quality Evaluation for Voice-Based Consultation)", 한국 인터넷 정보학회 제 22권 2호, KSII, pp 70-71, Apr 2021
- [6] Choi Yeunju, Jung Youngmoon, Kim Younggwan, Suh Youngjoo, Kim Hoirin. An end-to-end synthesis method for Korean text-to-speech systems. *Phonetics Speech Sci.* 2018 제 10권 1호, pp 39-48, 2018
- [7] "음성 합성", 위키백과 우리 모두의 백과사전, https://ko.wikipedia.org/wiki/%EC%9D%8C%EC%84%B1_%ED%95%A9%EC%84%B1
- [8] A. Vaswani et al. "Attention Is All You Need", *Advances in Neural Information Processing Systems* 30, (NIPS), 2017
- [9] "대형 언어 모델," 위키백과 우리 모두의 백과사전, https://ko.wikipedia.org/wiki/%EB%8C%80%ED%98%95_%EC%96%B8%EC%96%B4_%EB%AA%A8%EB%8D%B8.
- [10] "GPT-3," Wikipedia, <https://en.wikipedia.org/wiki/GPT-3>.
- [11] "KoGPT2," SKT Open Source, <https://sktelecom.github.io/project/kogpt2/>.
- [12] "SKT-AI/KoGPT2", GitHub, <https://github.com/SKT-AI/KoGPT2>.
- [13] "skt/ko-gpt-trinity-1.2B-v0.5", Hugging Face, <https://huggingface.co/skt/ko-gpt-trinity-1.2B-v0.5>.

- [14] "KoGPT", 이해하기 | Kakao Developers 문서,
<https://developers.kakao.com/docs/latest/ko/kogpt/common>.
- [15] 양진혁, 김인중, "노이즈 어텐션을 통한 딥러닝 기반 음성 합성", 한국 정보 과학회, 한동대학교 정보통신대학원, pp 904-906, Jun 2019
- [16] 이성주, 강병옥, 정훈, 박전규, 이윤근, "음성인식 시스템을 위한 전처리 기술", 한국정보과학회 학술발표논문집, pp 966-968, Dec 2017
- [17] 손종옥, 이용주, 배건성 "전처리 기법에 따른 잡음 음성의 인식 성능 비교" 한국음향학회 하계학술발표대회 논문집 제 19권 1호, pp 31-34, 2000
- [18] J. Shen et al. "Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions", ICASSP, pp. 4779-4783, 2018
- [19] J. Chorowski et al. "Attention-Based Models for Speech Recognition", NIPS, 2015
- [20] R. Prenger, R. Valle and B. Cantanzaro, "WaveGlow: A Flow-based Generative Network for Speech Synthesis," ICASSP 2019, pp. 3617-3621, 2019
- [21] "MOS", 정보통신기술용어해설, http://www.ktword.co.kr/test/view/view.php?m_temp1=2056
- [22] Tacotron2 & Waveglow 오픈소스, <https://github.com/NVIDIA/tacotron2>