

제19회 임베디드SW경진대회

개발완료보고서

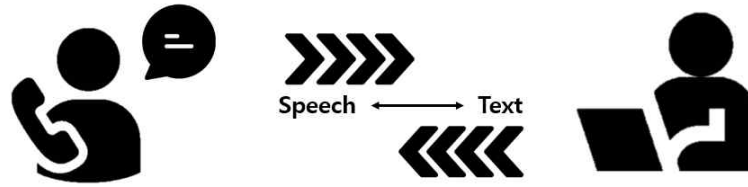
[자유공모]

□ 개발 요약

팀 명	VVS
<p>The diagram illustrates the VVS system architecture. It shows a flow between a Server, a Provider (phone icon), a Recipient (phone icon), a Raspberry Pi 4, and a User Interface. The Server handles STT (Text-to-Speech) and TTS (Speech-to-Text) processes. The Provider sends voice to the Recipient. The Recipient sends voice to the Raspberry Pi 4, which then sends it to the User Interface. The User Interface sends text back to the Recipient. The Recipient also sends text to the Server, which then sends voice back to the Recipient. A legend indicates that blue arrows represent voice (음성) and red arrows represent text (텍스트).</p>	
작품명	VVS(Voice Visualization Service)
작품설명 (3줄 요약)	청각장애인들을 위한 실시간 음성통화 시스템이다. 통화 환경에서 자체 개발한 STT와 TTS 서버를 통해 청각장애인도 실시간으로 음성을 상대방에게 보낼 수 있고, 텍스트 감성분석을 거쳐 본인의 감정을 음성으로도 표현할 수 있다.
소스코드	https://github.com/kusw1006/2021ESWContest_free_1049
시연동영상	https://www.youtube.com/watch?v=hyqGVYRIW48

□ 개발 개요

○ 개발 작품 개요



- 청각장애인들의 통화 접근성 향상을 위해 음성을 전달하는 상대방과 실시간 음성통화가 가능한 시스템을 개발하였다. 각자의 통화 환경에 맞게 시스템을 구축하였으며, 통화 상대방은 별도의 구현 없이 평소처럼 음성을 활용하여 대화하고, 서비스 대상자는 VVS를 통해 텍스트로 대화한다. 음식점의 특성을 고려하여 유선전화에 특화했다.
- 상대방의 음성이 사용자에게 텍스트로 변환되어 전달되고, 사용자가 작성한 답변 텍스트는 상대방에게 음성으로 변환되어 다시 전달된다. 텍스트의 감성을 분석하여 해당 감성에 맞는 음성으로 변환되어, 감정이 실린 목소리로 상대방에게 전달할 수 있다.
- 변환을 담당하는 하드웨어는 Raspberry Pi 4B+를 이용한다.

○ 개발 목표

- 1) Speech-To-Text(STT) 기술을 음식점 환경에 맞게 최적화된 방법으로 개발한다.
 - 1-1) 소음이 많은 환경에서 실시간으로 음성을 텍스트로 바꾸어 사용자에게 제공한다.
- 2) Text-To-Speech(TTS) 기술을 해당 서비스에 맞게 개발한다.
 - 2-1) 텍스트를 음성으로 바꿔 통화로 전달할 수 있다.
 - 2-2) 감정분석 기술을 통해 보내는 음성에 감정을 표현할 수 있다.

○ 개발 작품의 필요성

- 현재 배달 관련 플랫폼 앱들이 많이 출시되고, 상용화되며 소외되던 청각장애인들이 배달 서비스를 편리하게 이용할 수 있게 된 것은 사실이다. 그럼에도 불구하고 주문 실수 또는 불만 사항 접수, 메뉴, 배송지 변경 등의 문제들이 통화를 통해 이루어져 아직 완전한 서비스 이용에 어려움을 겪고 있다. 특히 점주들의 경우 아직 유선전화를 사용하는 곳이 많아 문자보다는 통화로 의사소통 하는 경우가 훨씬 많고, 만약 문자로 문의하더라도 통화 환경에 비하면 실시간으로 즉각적인 반응을 얻기 힘들다. 본 팀은 이러한 불편함을 해소하기 위해 Voice Visualization Service VVS를 개발하였다.
- 현재 여러 기업에서 시행되는 배달 음성인식의 경우 기본적인 메뉴나 단골 메뉴를 지정하면 이를 인식하여 주문할 수 있는 시스템이다. 하지만 통화로 문의해야 하는 상황이라면, 구체적인 메뉴를 인식해야 한다. 신조어를 이용하거나, 기존 데이터베이스에 쓰이지 않는 이름들이 신메뉴로 자주 올라오기 때문에 기본적인 메뉴, 예를 들어 치킨, 피자 등의 포괄적인 메뉴만으로는 제대로 된 음성인식 시스템을 이용할 수 없다. 실제로 여러 기업의 API를 사용했을 때 메뉴 인식에서 어려움이 많았다. 반면 VVS는 즉각적인 업데이트가 가능하다는 큰 장점으로 이러한 문제를 해결하고자 한다.

농인이 주로 사용하는 의사소통 방법은 “수어”

일상적인 의사소통에서 가장 많이 사용하는 언어는 ‘수어’라고 응답한 농인은 69.3%로 조사되었다. 농인의 제1언어가 ‘수어’임을 말해주는 결과이다. 가족과의 의사소통에서는 수어 사용 비율(42.7%)이 다소 낮게 나타났는데, 이는 가족 구성원 모두가 수어에 능숙한 것은 아니기 때문인 것으로 보인다.

생활 밀착 기관에서는 수어가 아닌 “필담”으로 주로 소통

주요 생활 밀착 기관인 관공서에서는 42.3%가, 금융 기관에서는 45.7%가 수어가 아닌 ‘필담’을 주로 활용한다고 응답하였다. 「한국수화언어법」에 한국수어는 국어와 동등한 자격을 가진 농인의 고유한 언어로 명시되어 있는 만큼, 농인이 차별 없이 일상생활을 영위할 수 있도록 제도적, 정책적 지원을 확대할 필요가 있음을 보여주는 결과이다.

<‘한국수어 사용 실태 조사’ 결과 발표>(사)한국농아인협회(2018. 04. 24)

- 현재 음성통화를 텍스트로, 텍스트를 음성으로 실시간 변환 서비스가 제공되지 않는다. 제공된다 하더라도 단방향이거나, 또는 실시간 진행이 힘들다. 수어를 사용해 영상통화로 진행하면 좋겠지만 위 조사결과 발표를 보면 알 수 있듯 일반인에게 수어 보급률이 낮을뿐더러 일반 통화, 특히 유선전화에 영상통화는 제공되지 않는 경우가 대부분이다. 청각장애인들 사이에서는 수어가 보편적으로 이용되나, 관공서 등 생활 밀착 기관에서는 텍스트로 주로 소통한다. 하지만 여기서도 불편한 점은 있다. 현재까지는 주로 통화를 이용한 서비스 안내가 더 보편적이다. 텍스트로 지원할 수 있지만, 빠르고 직관적인 전달이 힘들고, 특히 상대방이 텍스트를 계속 볼 수 없는 상황이라면 더 어려워진다. 특히 급격히 성장하는 배달문화에서 점주와의 통화 등 상대방이 텍스트로 전달하기 어려운 환경일 경우 자동으로 음성을 텍스트로, 텍스트를 음성으로 실시간 변환해주는 이 VVS를 활용한다면 더 원활하게 의사소통이 가능해진다.

여러 학술지나 통계에 따르면, 인간의 커뮤니케이션에서 사회적 의미의 60%는 비언어적인 요소가 매우 중요하다고 한다.

미국의 심리학자 앨버트 매리비언은 의사소통에서 상대방에게 영향을 주는 요소로 언어 7%, 청각 38%, 시각 55%를 꼽으며, 비언어적 요소가 커뮤니케이션에서 더 중요하다는 것을 강조했다. 또한 정신병리학자 자겐 루이스 역시도 “인간은 언어 이외의 기호를 대략 70만개나 사용하여 의사소통하고 있다”고 했다.

<[덴탈MBA]백 마디 말보다 효과적인 “상담동의”의 지름길, 비언어커뮤니케이션을 활용하라> 덴탈아리랑_박지연 대표(2019.06.27.)

- 텍스트만으로 의사소통을 진행할 경우 비언어적 표현이 어렵다. 청각장애인의 감정, 즉 미안함이나 분노, 그리고 즐거움 등을 표현하기 위해 이모지를 사용할 수 있겠지만, 위와 같은 통화 환경에서는 이를 전달하기 쉽지 않다. 특히 불만 사항을 접수하는 경우 주장의 강도가 높지 않거나, 발생 즉시 항의하지 않는다면 처리를 늦추는 프로세스를 가진 기업들이 대다수임을 다양한 유선전화 상담 아르바이트를 통해 경험했다. 이런 환경에서 청각장애인들은 알게 모르게 차별받고 있다. 2006부터 시행 중인 손말이음센터, 통화중 계서서비스의 경우 통화 중간 단계에서 중계사가 텍스트를 상대방에게 읽어 주는 역할을 한다. VVS는 중계사의 역할을 시스템에 대체할 수 있으므로 더 상용화가 쉽고, 청각장애인이 비장애인인 상대방과 더 간편하게 자유로운 통화를 할 수 있다.

□ 개발 환경 설명

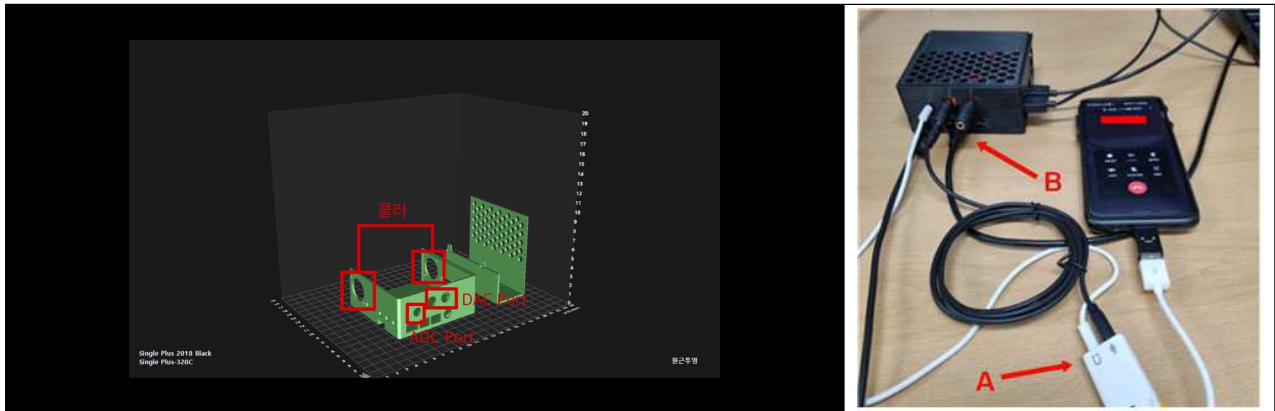
○ Hardware 구성

- Hardware는 사용자의 편의를 위해 최대한 간추리고자 했다. 사용자의 쉬운 접근성이 기기 구성의 우선적인 목표이므로 비전공자 청각장애인이 쉽게 시스템을 구동시킬 수 있도록 기기를 간소화시켰다. 기본 구성은 아래와 같다.

- Raspberry Pi 4B+
- HiFiBerry DAC+ ADC pro
- USB Virtual 7.1 채널 사운드카드

- 먼저 주체가 되는 Raspberry Pi 4에는 마이크 입력 회로가 없어 별도의 HiFiBerry 사운드 보드를 사용했다. 마이크 입력과 사운드 출력을 동시에 진행하기 위해 DAC+ 모델에 ADC가 결합된 제품을 이용했으며, 휴대폰에서 입력과 출력의 분리를 위해 USB 사운드 카드를 거친 뒤, 이 둘을 AUX 케이블로 연결했다.

○ Hardware 기능 (제어 방법 등 서술)



- 위 그림은 3D 프린터로 직접 제작한 VVS 하드웨어의 케이스 도면이다. 먼저 일반 Raspberry Pi 케이스로는 HiFiBerry를 모두 담을 수 없었다. DAC 포트 2개와 ADC 포트 1개가 들어갈 수 있어야 했고, 특히 두 보드의 발열 문제가 심했기 때문에 쿨러 설치가 필요했다. 따라서 위와 같은 케이스를 제작했다. 완성품은 오른쪽 사진과 같다.
- 스피커와 마이크의 구분을 위해 연결 방법이 중요하다. 먼저 핸드폰에 연결된 USB 사운드카드(A)는 마이크 포트에 HiFiBerry 보드(B)의 DAC 포트를 연결해야 하고, 스피커 포트에는 ADC 포트를 연결해야 한다. 이로써 핸드폰의 소리가 Raspberry Pi의 ADC 포트를 통해 들어갈 수 있고, 반대로 Raspberry Pi의 소리가 DAC 포트를 거쳐 핸드폰 마이크에 들어갈 수 있다.
- 사운드카드를 이용하는 이유는 하드웨어만을 이용해 핸드폰과 Raspberry Pi를 연결하기 위해서다. 각각의 마이크 입력과 스피커 출력을 교차시켜 AUX 케이블을 통해 실시간으로 받아야 하는데, 각각의 출력은 아날로그 형태이므로 이를 디지털로 변환시켜 줄 수 있는 ADC포트나 USB 사운드카드를 이용했다.

○ Software 구성

영역		구성
구현 기술		Kaldi를 활용한 한국어, 배달 및 서비스 제공 특화 STT 구현
		TCP/IP 통신을 활용한 서버/클라이언트 네트워크 구현
		RealTime STT 결과값 후보정을 위한 Spacing 및 rescore 구현
		텍스트에서 감정을 읽어내는 감정인식기 구현
		상황에 따라 말투 및 억양을 바꾸는 감정별 TTS 구현
		TKinter를 활용한 User Interface 구현
서버 클라이언트 구축	메인 서버	STT가 작동하는 메인 서버
	서브 서버	감정별 TTS가 작동하는 서브 서버
	클라이언트	띄어쓰기 보정 클라이언트
		User Interface Chatting 클라이언트
		음성 실시간 송수신 클라이언트

AM 파트

1. Clean Data Training (HMM-GMM)

- 음성데이터와 음소간의 정렬을 하는 단계로써, 최대한 음소의 특성을 잘 볼 수 있는 깨끗한 데이터로 음소와 음성데이터의 정렬을 진행한다.

이때 샘플링 주파수에 따라 음성데이터의 특징벡터가 많이 달라지므로, 유선전화의 샘플링레이트로 일부 변환하여 유선전화상의 음성 인식률을 높였다.

- 사람의 음소 발음 특성은 주로 1kHz이하의 구간에서 돋보이기 때문에, 학습 시 MFCC Feature Vector를 추출하여 사용하며, 주변 음소의 영향을 고려하지않는 MonoPhone(단일음소) 학습을 진행한다. 이후 주변 음소의 영향 및 다양한 발화자의 특징을 정규화하여 TriPhone(삼중음소) 학습을 진행하며 이때 Delta+delta-delta, LDA-MLLT(Linear discriminant Analysis), SAT(Speaker Adaptive Training), fMLLR(Feature Space Maximum Likelihood Linear Regression) 순으로 정렬 알고리즘을 이용하여 음소의 확률분포를 학습한다.

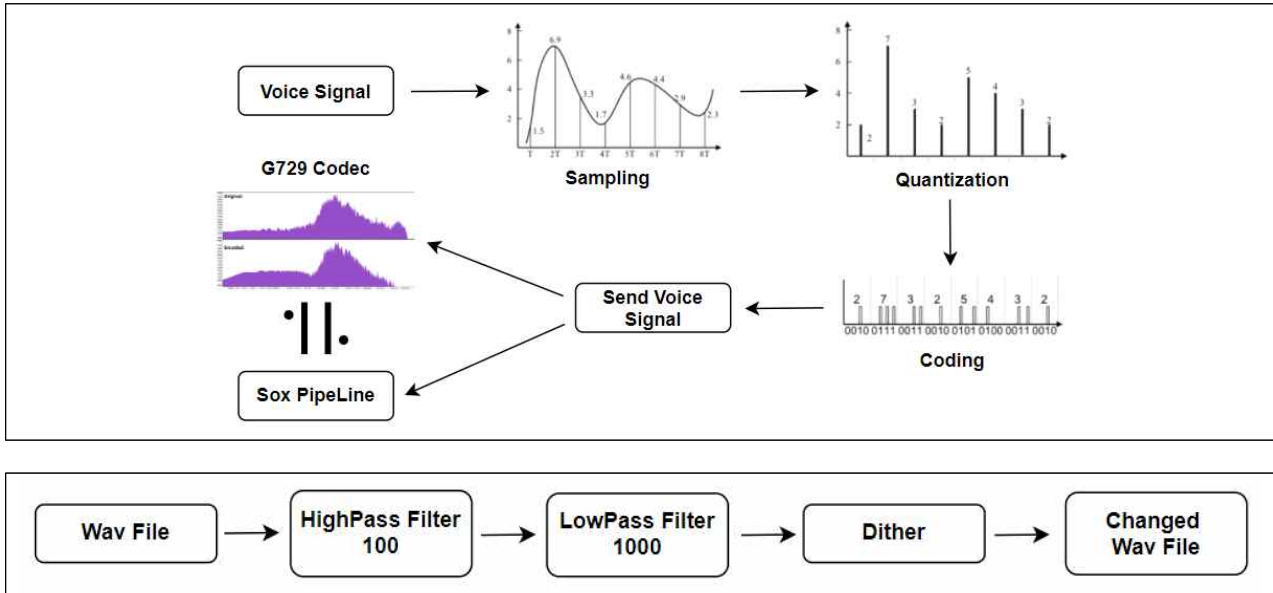
- 음향데이터의 크기에 따라 Maximum PDF(확률분포의 최대 개수)와 Maximum Gaussian을 조절하여 최적화하여 학습하였다.

2. MultiCondition Data Training(DNN)

- RIR(Room Impulse Responce)음향 데이터셋을 이용하여 음식점 뿐만 아니라 다양한 환경의 잔향(Reverb)을 음성 데이터셋의 파이프라인에 추가하고 이때 SNR(Signal to Noise Ratio)도 다양하게 적용하여 음소특징을 잃지않으며, 다양한 소음환경에 대해 적응할 수 있도록 하였다. 또한 소음추가 이외에도 코덱으로 인한 유선전화의 소리특성을 파이프라인에 추가하였으며, 뒤에 서술하였다.

- 이후 새롭게 제작된 음성 데이터 파이프라인에서 MFCC를 다시 추출하고 이전 Clean Data 학습을 통해 얻은 정렬정보를 토대로 DNN(Deep Neural Network)을 통해 확률분포를 재학습한다. 이때 사용된 모델은 1D-Convolution Layer + BatchNorm 이며 Activation Function으로는 relu를 사용하였고 데이터 양에 따라 Epoch를 조절하여 학습하였다.

3. 유선전화의 소리특성 추가



- 전화 음성 데이터는 발화자의 말을 표본화하고 양자화 한 후 부호화 하는 과정을 거쳐 전달 된다. 그리고 청취자는 이 부호화 된 코드를 코덱을 활용해 음성으로 복원 후 전달받게 된다. 이 코덱과 가장 유사한 방식으로 음성 데이터를 구성하고 전화 음성에 최적화된 학습 데이터를 확보하기 위해 Sox PipeLine에서 Low, Highpass Filter와 Dither 과정을 거쳐 음성 학습 데이터를 변형하였다.

LM 파트

1. 말뭉치

말뭉치		line
모두의 말뭉치	일상 대화 말뭉치 2020 v1.1	870,682
	구어 말뭉치 v1.1	20,871,188
AI Hub + selenium (x3)	한국어 음성(Kspon-speech)	252,150
	소상공인-고객 관련	2,007,735
합계		24,001,755

- Language Model 학습을 위해서는 양질의 말뭉치가 필요하며, 말뭉치에 포함된 단어, 주로 사용되는 어체에 따라 특정 domain에 대한 인식률이 향상된다. 배달 특성 상, 대화에는 주소와 메뉴 등의 단어가 자주 등장한다. 때문에 구어체 AIHub 및 모두의 말뭉치에서 제공하는 말뭉치에 더하여 추가적으로 selenium을 이용한 가게리뷰, 배달리뷰의 web scrapping을 진행해서 우리 domain에 특화된 말뭉치를 수집했다. 특히 고객 질의 응답, 음식점과 관련된 말뭉치의 경우는 LM학습시에 여러번 넣어서 해당 말뭉치에서 포함하는 단어들의 빈도수를 높여 인식에 더욱 용이하게 했다. 최종 문장 수는 약 2400만 줄이다.

2. Carpa

- LM의 학습결과로 n-gram arpa file이 생성된다. 서버 PC 성능 상 4-gram arpa file을 fst형태로 만들어 사용할 때가 가장 좋지만, 용량이 매우 크고 path 탐색에 오랜 시간비용이 들기 때문에 이를 직접적으로 이용하기 어렵다. 때문에 FST 자체는 가벼운 모델로 만들어 디코딩 결과를 빠르게 생성하고 const-arpa로 변환한 4-gram arpa file을 이용하여 Rescoring을 진행해 정확도를 향상시켰다.

3. HCLG

- AM과 LM을 조정하며 다양한 모델을 만들어 학습을 진행했다.

model	test1	test2	test3	test4	test5
AM	kspon 300시간	kspon 300시간을 유선전화 코덱에 맞게 변형			
LM	4-gram	3-gram small	3-gram medium	4-gram small	4-gram medium
성능	전화 음성을 인식하지 못함	전화 음성을 인식하지만 문맥이 맞지 않음	전화음성을 인식하고 문맥에 맞춤		
시간	19s	3s	4s	5s	13s

원본	이천오년 당시 박근혜 한나라당 대표와 직전 비서실장 유 의원은 동고동락 하던 사이였다
test1	하지만 [5] ~ [10] 발 분 요약 할 때는 맵 각지에 앞뒤 미션 [10] 월 단원 잡았다
test2	이 (천) 오 년 당시 박근혜 한나라당 대표 와 직접 비서실장 유 의원은 동고동락 하던 삶 이었다
test3-5	이 (천) 오 년 당시 박근혜 한나라당 대표 와 직전 비서실장 유 의원은 동고동락 하던 사이 었다

4. task domain 확장

- 신조어 및 새로운 배달메뉴가 계속해서 나오므로, 음성 인식 domain의 확장은 필수적이고, 신속해야한다.

4-1) run_task.sh

- task domain의 arpa를 제작한 뒤 srilm tool을 이용하여 interpolation하여, domain에 특화하여 LM을 구축하는 방법이나, 한번 interpolation이 진행되면 최적화 및 bias가 걸려 다른 domain의 arpa와 더 이상 interpolation을 진행할 수 없게 된다.

4-2) extend vocab demo

- interpolation의 단점을 해결하기 위해, top L.fst, G.fst가 nonterminal하게 동작할 수 있도록 모르는 단어에 대해 기존 <UNK> 심볼 대신 #nonterm:unk 심볼을 추가하여 해당 부분에 task domain G.fst를 결합할 수 있도록 전처리했다. 그 후, 제작한 top HCLG.fst를 가지고 각 task domain에 맞는 새로운 단어들로 L.fst, G.fst를 제작한 뒤, HCLG.fst로 묶고, make-grammar-fst Kaldi binary 파일을 이용하여 main HCLG.fst와 new HCLG.fst를 빠르게 결합하여, domain에 맞게 탈부착이 가능한 기술을 개발하였다.

- 이때 L.fst는 따로 g2p(Grapheme to Phoneme)를 이용하여 발음을 생성하지 않고, 한국어 발음 분류 table에 맞게 발음을 변형한 뒤 data driven 방식으로 형태소를 분석하는 morfessor tool을 이용하여 형태소를 분석하여 기존의 단어집과 비교하여 새로운 단어가 있을 때 해당 단어에 대해 새로운 Lexicon을 만들고 결합할 수 있도록 shell script로 제작해놓았다.

다화자 다감정 TTS (Text To Speech)

1. Text2Mel

1-1) Tacotron2

- Text2Mel을 진행해주는 Seq2Seq모델의 구조를 기반으로 하는 Neural Network이다.

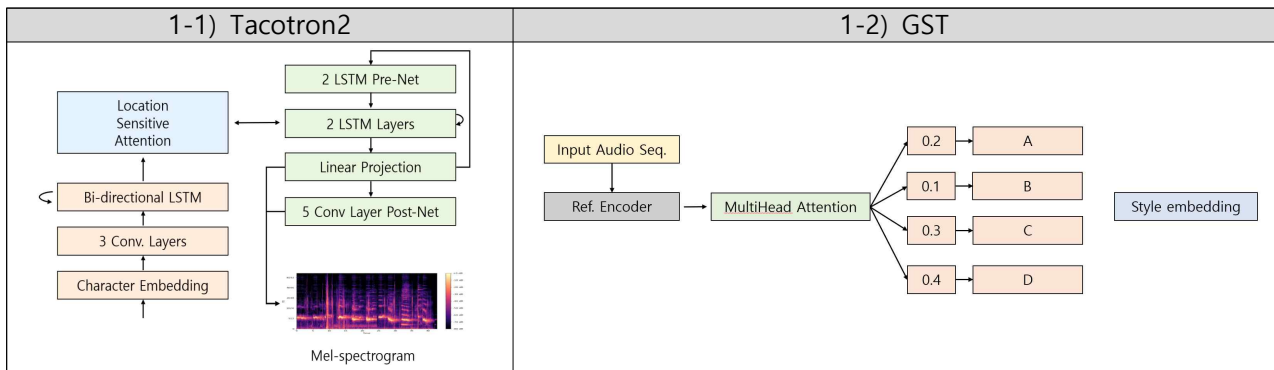
문자 임베딩을 Mel-Spectrogram에 맵핑하는 반복적인 구조로 이루어져있으며, Encoder와 Decoder를 연결하는 Location-Sensitive Attention이 있다. 이때 Decoder에서 AutoRegressive RNN을 포함하는데 이와 같은 이유로, 추론 속도가 떨어지는 특징을 가진다.

1-2) GST

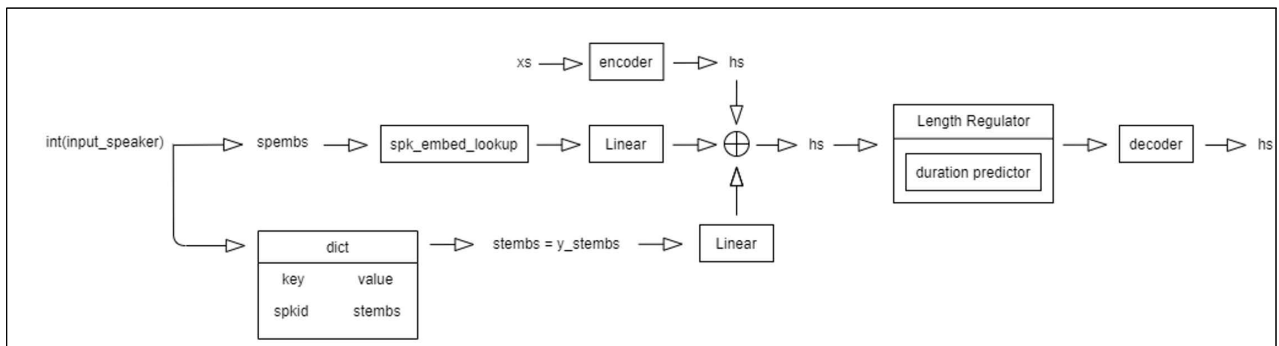
- 앞서 Tacotron2를 기반으로 GST(Global Style Token)으로 화자 및 감정에 대한 벡터를 추출한다. GST 구조는 다음과 같다.

- 다양한 길이의 오디오 입력의 운율을 Reference Encoder(참조 인코더)를 통해 고정된 길이의 벡터로 변환한다. 이후 학습 과정에서 Tacotron 인코더단의 출력과 concatenate하여 Tacotron 모델의 Attention의 입력으로 사용하여 목표 문장의 Style을 Embedding을 한다.

이후의 Style Embedding Vector는 Text2Mel 모델을 사용할 때 임베딩 된 Character와 함께 add 또는 concatenate하여 Style에 맞는 Mel-spectrogram을 제작하는데 사용된다.

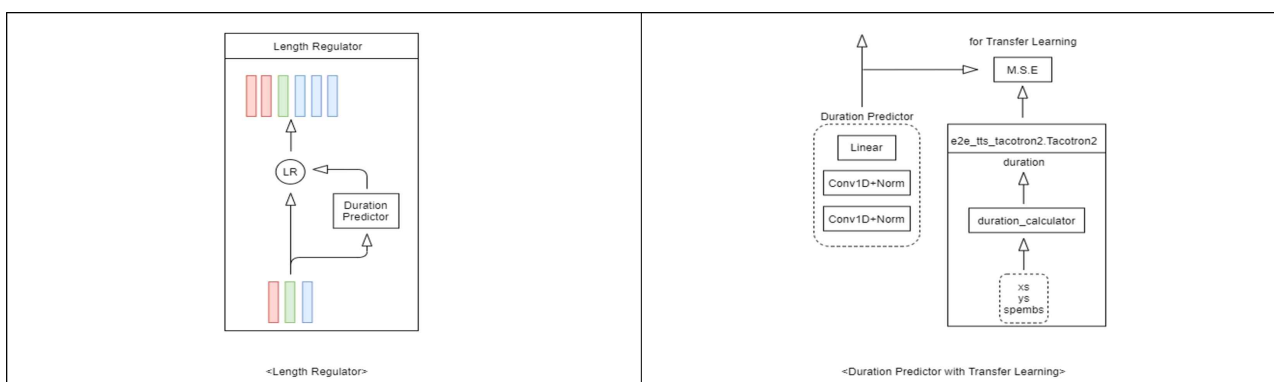


1-3) FastSpeech



- FastSpeech 모델은 Text2Mel 작업을 위한 Neural Network이며, 같은 작업을 하는 Tacotron2는 Regressive한 구조를 가지는데 비해 FastSpeech는 Non-autoRegressive한 구조를 가져 훨씬 더 빠른 inference가 가능해 해당 모델을 사용하기로 하였다.

전체 구조는 위의 그림과 같으며, Tacotron2-GST를 통해 학습하였던 Style Vector를 가져와 Character Embedding Vector와 add연산을 진행하여 추론에 사용한다.



- 학습은 위와같이 Fast Speech 모델의 Duration Predictor 부분에서 기존 Tacotron2의 Phoneme-Melspectrogram 정렬 정보(duration)를 가져와 label로 이용해 전이학습을 진행한 모델을 이용하였다.

2. Mel2Wave

2-1) Parallel WaveGAN

- 자기회귀 방식이 아닌, 비자기회귀 방식을 이용하여 GPU의 병렬가속연산 기능을 잘 활용한 보코더로써, 기존의 다른 네트워크들 보다 더 빠른 학습시간과 추론시간을 가지며, 대동소이한 결과를 내주어 Transformer 모델과 함께 음성합성에 최근 많이 사용되는 모델이다.

- <https://github.com/qkadldpdy>의 Pretrained model을 사용하여 기존 FastSpeech모델과 결합해 음성합성에 사용하였다.

3. 결과

- FastSpeech-ParallelWaveGAN의 음성합성 성능 평가를 위하여 팀원 4명을 포함해 추가로 6명의 학생을 섭외하여 총 10명의 학생이 평가하였으며 결과는 다음과 같다.

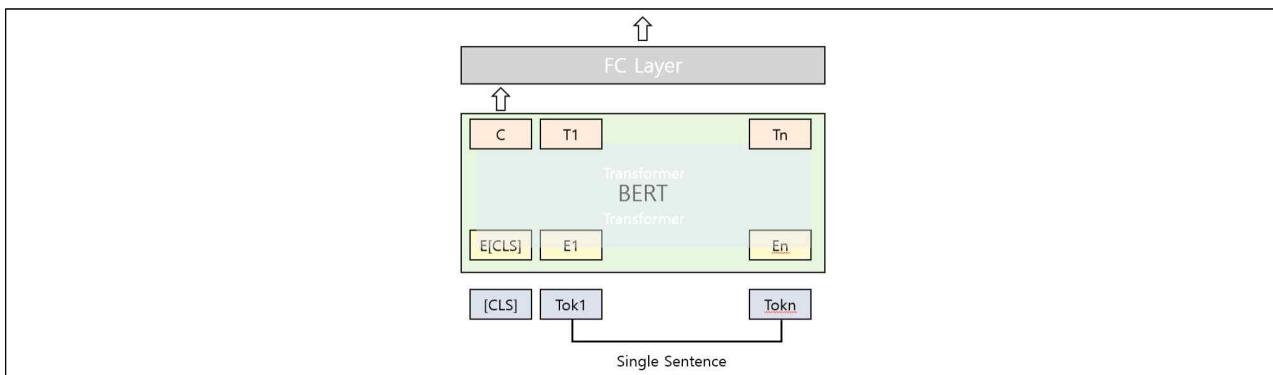
감정	MOS	감정	Accuracy
Neutral	4.2 ± 0.2	Neutral	92%
Anger	3.9 ± 0.23	Anger	81%
Sadness	3.8 ± 0.3	Sadness	85%
Happiness	3.9± 0.21	Happiness	93%

- 감정은 최대 감도로 설정하여 테스트를 진행하였으며, 측정방식은 Confusion Matrix를 이용하였다.

감성분석

1. koBERT Sentimental Classifier

- SKTBrain이 제작한 한국어 BERT (koBERT)를 이용해 BERT의 첫 Class Label을 받아 Fully Conected Layer를 한층 쌓은 간단한 구조로 구성하였으며, 가장 높은 디렉스를 감정과 매핑하여 감정을 분류할 수 있도록 하였다.



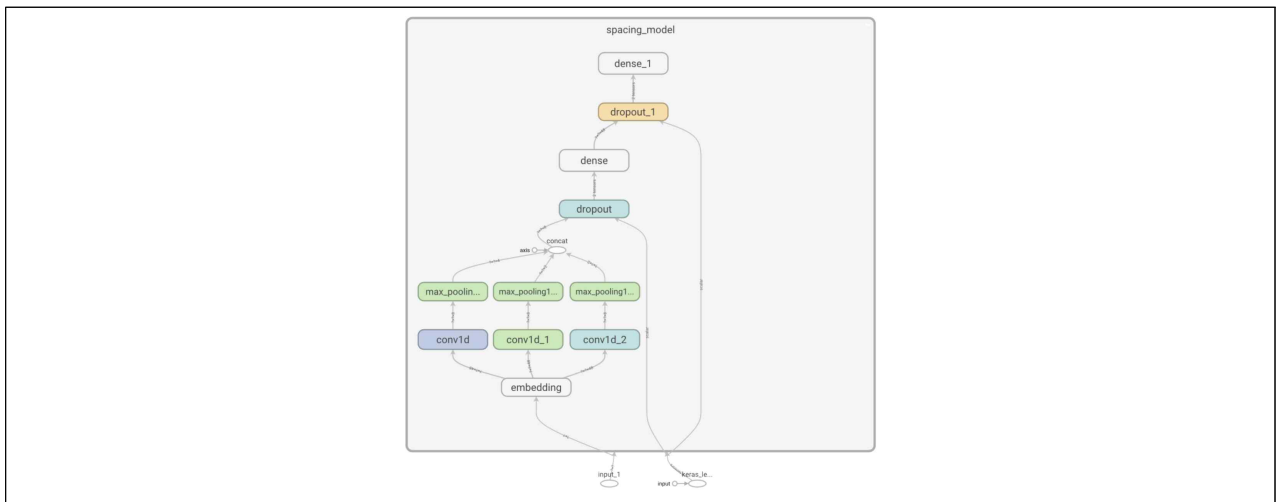
- 데이터셋은 AIHUB의 감성대화 말뭉치(27만 문장)의 60가지 감정을 3가지로 분류하고 (분노, 슬픔, 기쁨), 부족한 평서문은 국립 국어원 모두의 말뭉치의 구어체(2천만 문장 중 14만 문장)을 이용하여 학습하였다. 이때 Trainset : Testset = 0.9 : 0.1로 하였으며, Accuracy는 75~89% 정도가 나온다.

띄어쓰기(Spacing) 모델

1. CNN 모델 이용

- STT를 형태소 단위로 진행하기 때문에 최종 디코딩된 문장이 나왔을 때 띄어쓰기 오류가 많이 발생했기에 띄어쓰기 보정을 위한 방안을 탐색했고 이 딥러닝 모델을 사용하게 되었다.

- 우리 도메인에서는 경어체보다는 구어체와 배달관련 용어를 많이 쓰기 때문에 이에 맞는 corpus를 이용하여 학습시켰다. 사용한 corpus로는 국립국어원 모두의 말뭉치에서 제공하는 일상대화말뭉치 2020, 구어 말뭉치와, AIHub에서 제공하는 소상공인 고객 주문 질의 텍스트 (2020), 한국어대화(2018), 민원 콜센터(2020)가 있다. 25,000,000개의 문장으로 구성되어있으며 22,500,000 (Train set) / 2,500,000 (Test set)으로 이루어져있다.



- 모델 구조는 위의 그림과 같다. Bi-LSTM구조를 CNN으로 교체하면 비슷한 정확도에 더 빠른 성능을 얻을 수 있기 때문에 CNN 구조를 선택했다. Embedding 후 Conv1D- MaxPool1D (그림은 3개지만 2~10으로 조절 가능) 결과물을 concat한 후 Dense Layer 2번 통과한다.

- 학습 절차는 다음과 같다.

1. Index Character: Dataset 음소를 빈도 순으로 저장하여 상위 4996개 매핑
2. Vocab 제작: Index Character에 padding, bos, eos, unk 4개의 인덱스도 추가
3. 띄어쓰기가 잘 되어있는 한국어 문장에 BOS, EOS 태그를 추가한 뒤 띄어쓰기를 랜덤하게 삭제/ 추가하고 그를 원래의 문장으로 복구하기 위한 label생성. 0.5의 확률로 공백 삭제, 0.15의 확률로 공백 추가
4. Character Indexing 후 model 통과

- 학습된 모델을 문장에 적용했을 때 각 음절마다 0, 1, 2의 라벨을 갖게되는데 0은 현상유지, 1은 띄어쓰기 추가, 2는 띄어쓰기 삭제를 뜻한다. 예를들어, '나 는배고프다'라는 문장은 '0210000'으로 매칭되고 '나는 배고프다'로 수정된다.

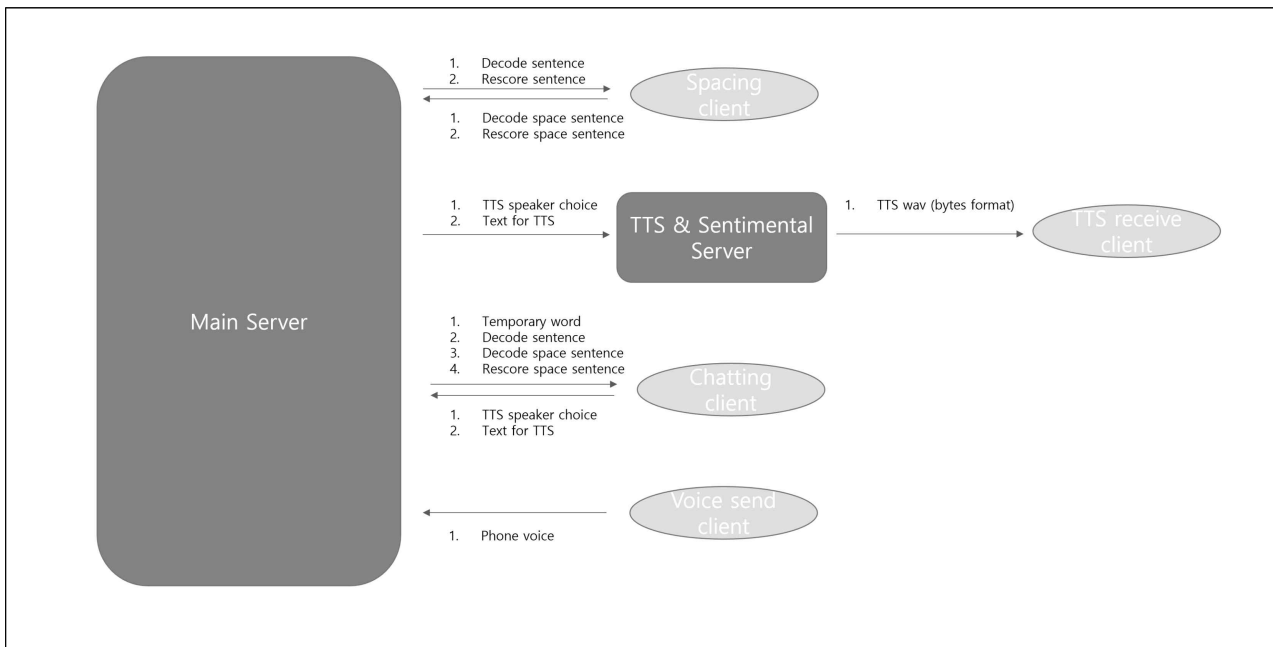
- 학습한 모델의 성능은 다음과 같다.

case	정확도
공백을 모두 제거한 문장	0.9442
모든 음절마다 공백을 추가한 문장	0.9539

정확도는 약 0.95이며, Intel(R) Xeon(R) CPU E5-2640 v3 환경에서 한 문장당 추론에 0.088ms를 소요했다.

○ Software 설계도

TCP 연결 구조



- 전체적인 Software 설계도는 위와 같다. main server는 TTS & Sentimental server, Spacing client, Chatting client, Voice send client 와 연결되어있으며, TTS & Sentimental server는 TTS receive client 와 연결되어 있다. 이 서버와 클라이언트들은 각 기능이 서로 종속적이기 때문에 연결 순서에 유의해야 한다.

○ Software 기능

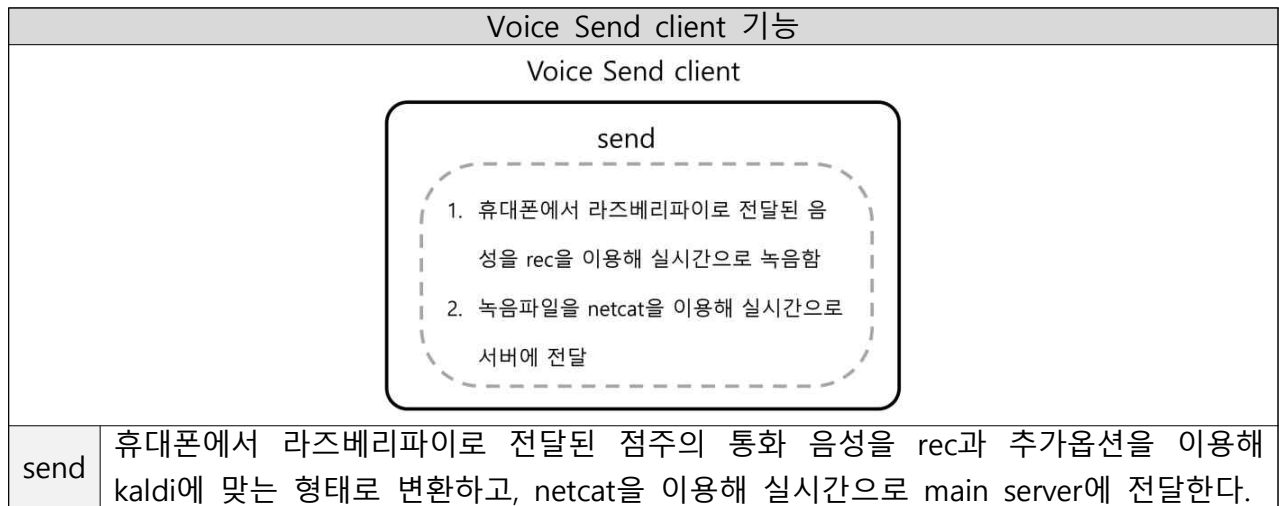
문장 토큰

Token	의미
T{num}	decode된 {num}번째 문장
C{num}	띄어쓰기 보정된 decode 문장
R{num}	Rescore된 {num}번째 문장
D{num}	띄어쓰기 보정된 Rescore 문장

- Software 기능에 대해 알아보기 전, 문장 앞에 붙어있는 각 토큰의 의미에 대해 먼저 알아보자. 채팅 서버에서 같은 문장에 대하여 입력을 여러 번 받기 때문에 서버로부터 받은 문장이 어떠한 과정을 거쳐서 전달된건지, 몇 번째 문장인지 확인해야한다. 때문에 T, C, R, D 토큰으로 각 문장의 종류를 구별했으며 문자의미는 위의 표에 나와있다. 여기서 num은 몇 번째 문장 인지를 나타내는 숫자다.

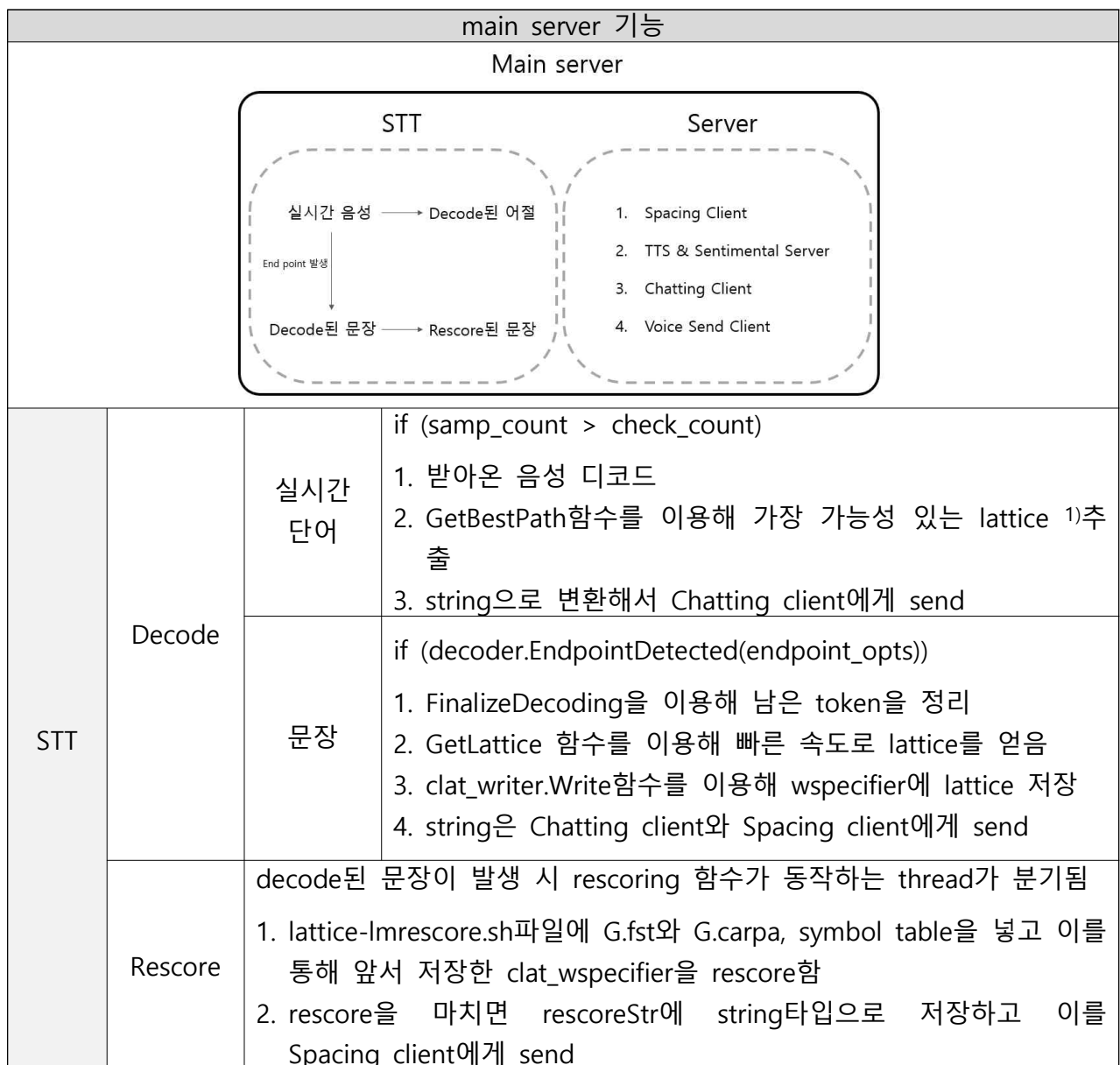
Voice Send client

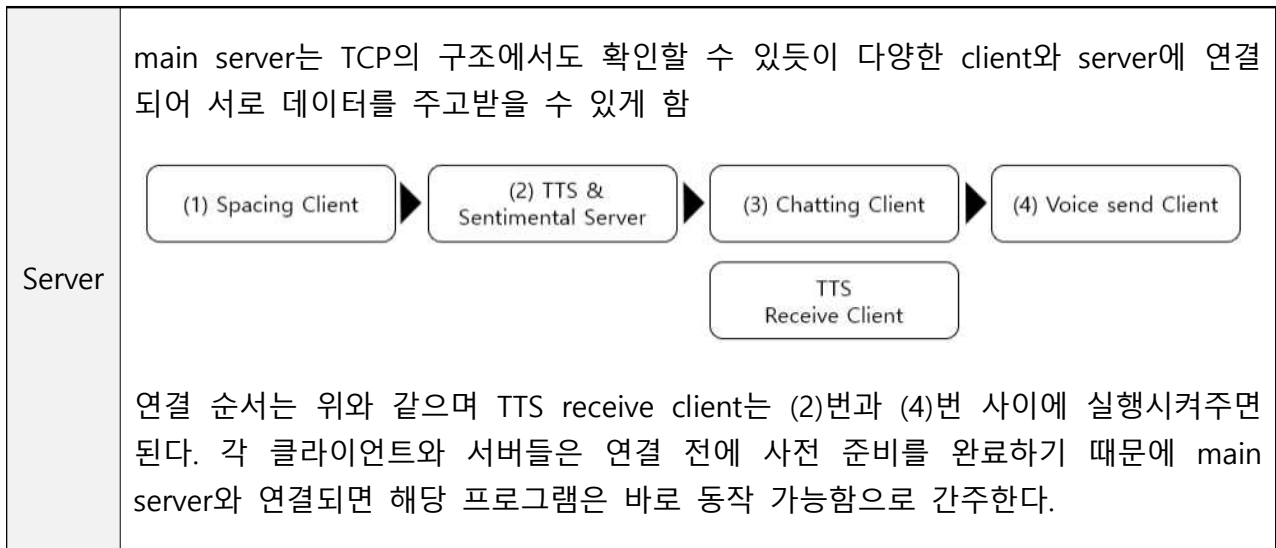
실시간으로 받는 음성을 rec를 통해 main server에 전달한다.



main server

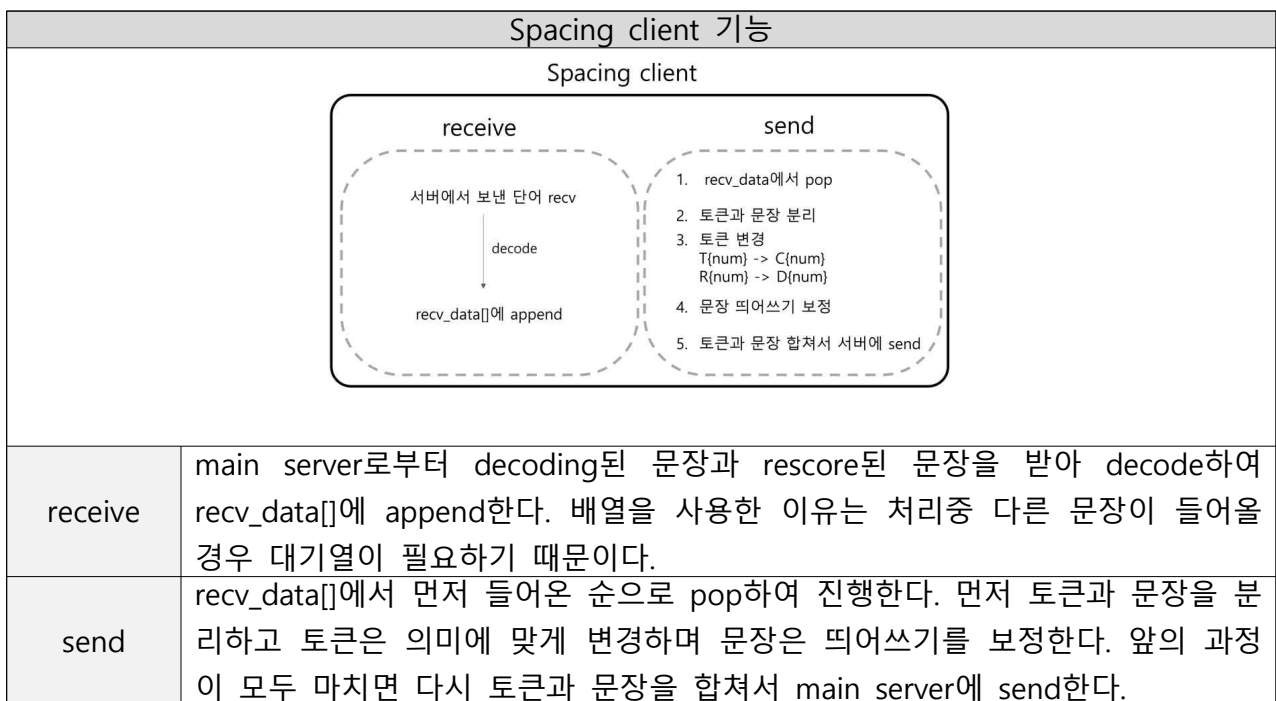
- main server의 주요 업무는 두 가지인데 STT와 서버 동작이 그에 해당한다.





Spacing client

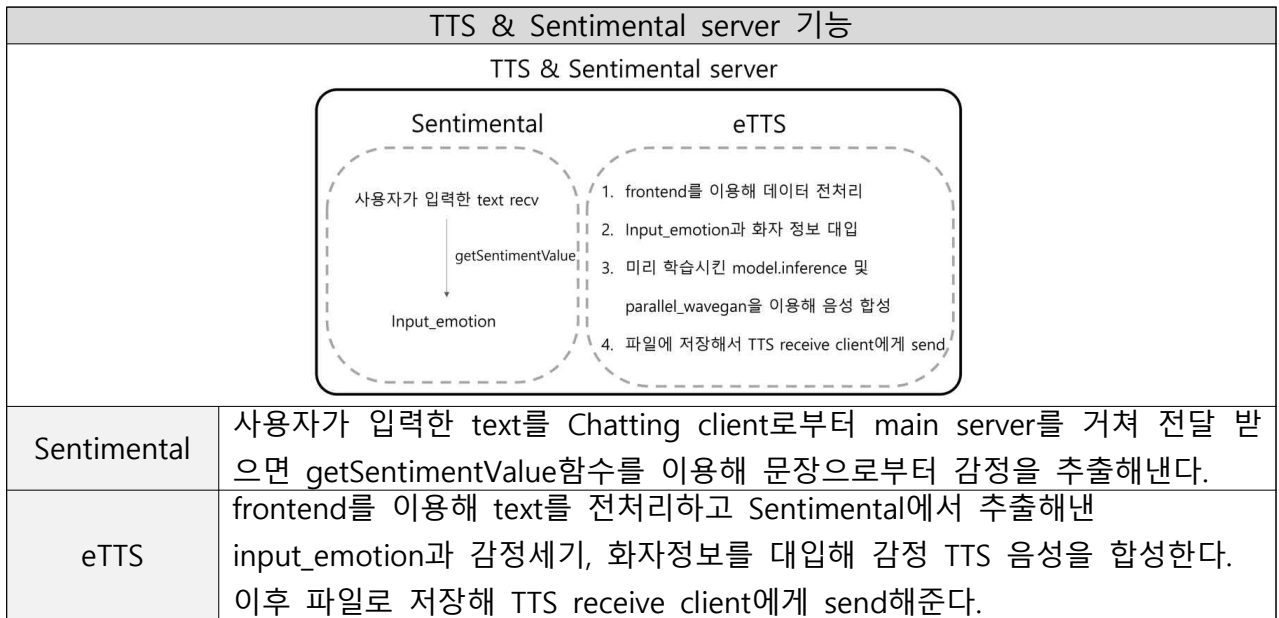
- 띄어쓰기 클라이언트는 딥러닝을 통해 학습된 모델을 이용하여 문장에서 띄어쓰기 오류를 찾아 보정해주는 프로그램이다. 주 기능은 receive와 send가 있다.



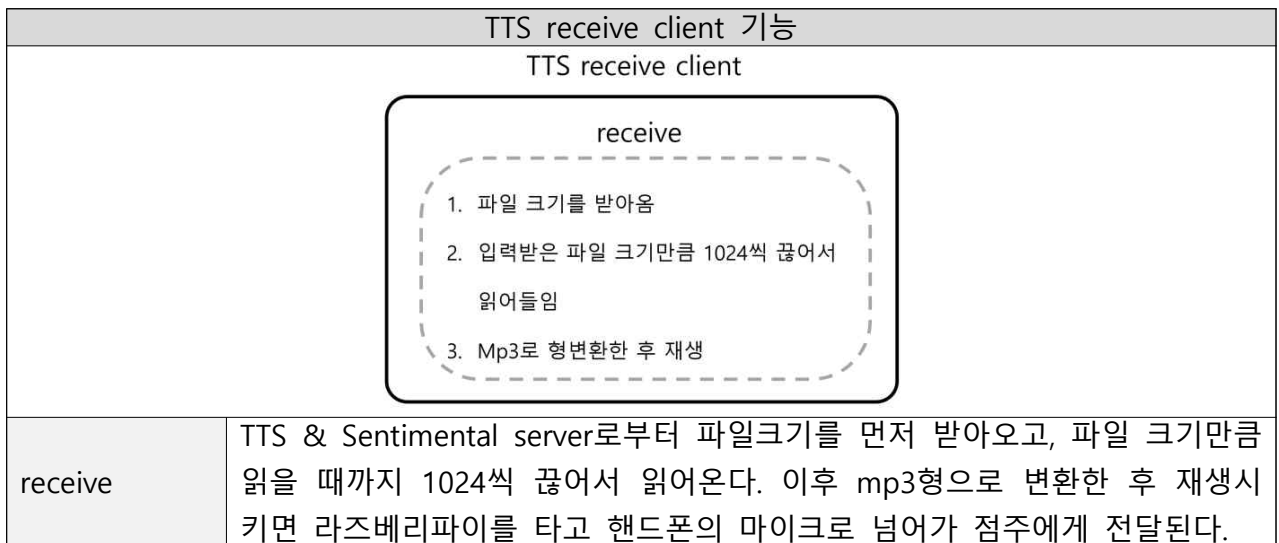
TTS & Sentimental server

- TTS & Sentimental server는 딥러닝으로 학습한 TTS model과 Sentimental model을 기반으로 감정 TTS를 실행하는 프로그램이다. 주 기능은 Sentimental과 eTTS가 있다.

1) Lattice는 결정 격자 구조를 의미하며 여기서는 단어 간의 전이확률을 포함하고 있는 집합을 의미한다.

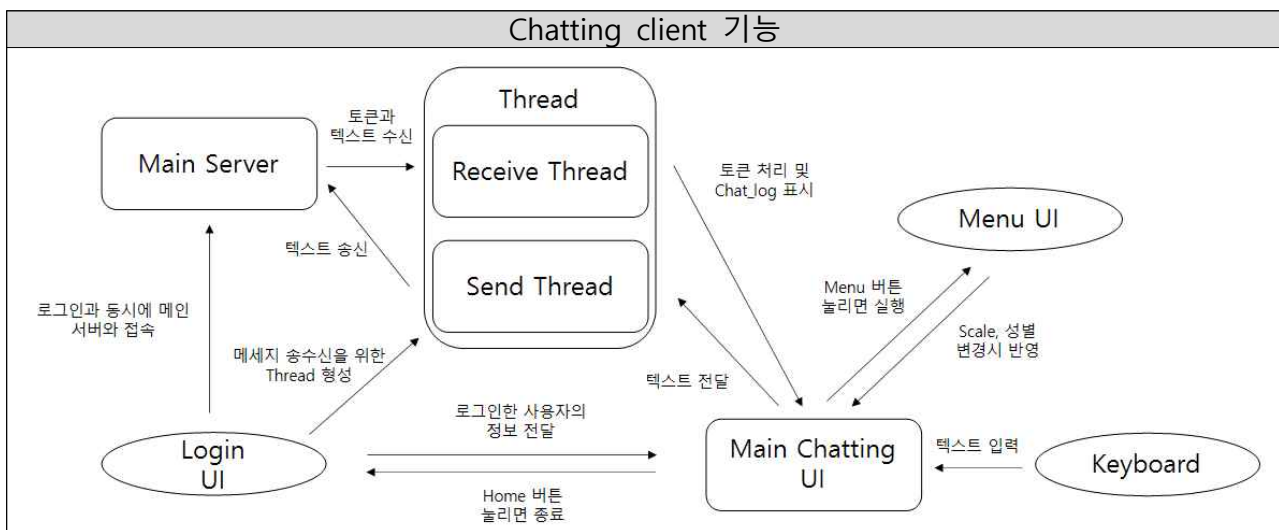


TTS receive Client



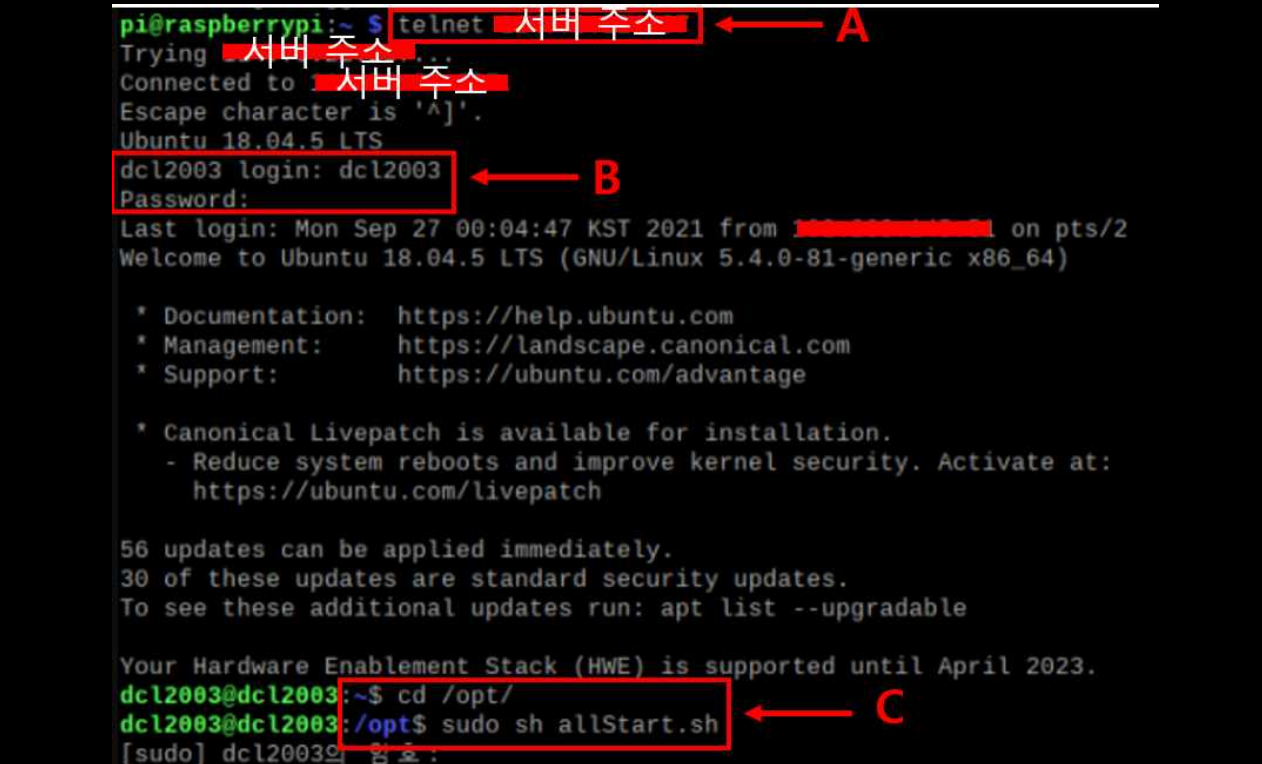
Chatting Client

- chatting client는 유저가 사용하게 되는 UI를 띄워주는 클라이언트이다.



Main Server	Login이 진행된 후에 Client와 TCP로 연결이 된다. Receive Thread와 Send Thread와 텍스트를 송수신한다.
Login UI	Email과 Password를 입력받아 기존 사용자와 일치하는지 여부를 확인하고 Login 성공시 Sever에 접속하고 Thread를 실행한다.
Receive Thread	Main Server로부터 토큰과 텍스트를 받아와 토큰에 따라 분리하고 Main Chatting UI에 Chat log를 띄운다.
Send Thread	Main Chatting UI에서 전달받은 텍스트를 Main Server에 전송한다.
Main Chatting UI	Thread가 받아온 텍스트의 토큰을 분석해서 해당 자리에 표시되고 keyboard에서 텍스트를 입력받는 Main Interface이다. 좌측의 Home 버튼을 누르면 종료된다.
Menu UI	TTS Scale과 성별을 바꿀 수 있고 user Information을 확인할 수 있다. 성별과 scale 조정은 실시간으로 반영되며 이후 다시 접속해도 유지된다.
Keyboard	텍스트를 영어로 입력받아 한글로 변환한다.

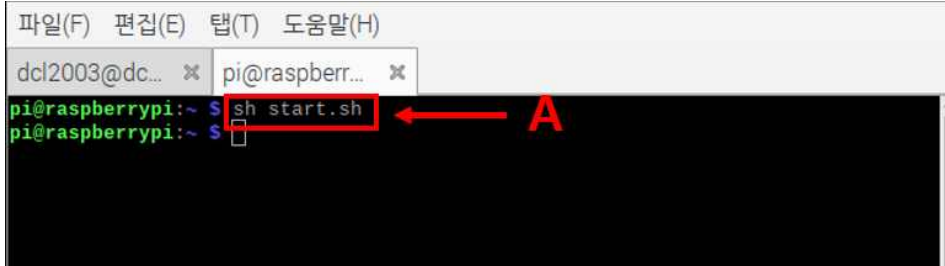
○ 프로그램 사용법 (Interface)

server setting	
 <pre> pi@raspberrypi:~\$ telnet 서버주소 Trying 서버주소... Connected to 서버주소. Escape character is '^]'. Ubuntu 18.04.5 LTS dcl2003 login: dcl2003 Password: Last login: Mon Sep 27 00:04:47 KST 2021 from [redacted] on pts/2 Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-81-generic x86_64) * Documentation: https://help.ubuntu.com * Management: https://landscape.canonical.com * Support: https://ubuntu.com/advantage * Canonical Livepatch is available for installation. - Reduce system reboots and improve kernel security. Activate at: https://ubuntu.com/livepatch 56 updates can be applied immediately. 30 of these updates are standard security updates. To see these additional updates run: apt list --upgradable Your Hardware Enablement Stack (HWE) is supported until April 2023. dcl2003@dcl2003:~\$ cd /opt/ dcl2003@dcl2003:/opt\$ sudo sh allStart.sh [sudo] dcl2003의 암호: </pre>	
기호	설명
A	telnet \${서버주소} : 서버에 원격으로 접속한다.
B	로그인할 ID와 Password를 입력한다.
C	cd /opt/ : /opt폴더로 이동한다. sudo sh allStart.sh : allStart.sh 파일을 실행한다. passwd 입력이 필요하다.

```
LOG (online2-tcp-nnet3-decode-faster[5.5.958-1539-57f8d]:Accept():online2-tcp-nnet3-decode-faster.cc:711) Accepted connection from: 114.70.22.237
LOG (online2-tcp-nnet3-decode-faster[5.5.958-1539-57f8d]:main():online2-tcp-nnet3-decode-faster.cc:423) @@@@@@@@@@@@@@@@@@1111@@@@@@@@@@@@@@@@@
LOG (online2-tcp-nnet3-decode-faster[5.5.958-1539-57f8d]:main():online2-tcp-nnet3-decode-faster.cc:438) @@@@@@@@@@@@@@@@@@2222@@@@@@@@@@@@@@@@@
LOG (online2-tcp-nnet3-decode-faster[5.5.958-1539-57f8d]:Accept():online2-tcp-nnet3-decode-faster.cc:691) Waiting for client...
2021-09-27 21:55:32.483144: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcudart.so.11.0
using cached model
using cached model
using cached model
sentimental load
Now ready to synthesize!
화자 번호를 입력해주세요. *Option* 여자: 0~4, 10~14 / 남자: 5~9, 15~19
감정 세기를 입력해주세요. *Option* 0.5(약하게), 1.0(적당하게), 2.0(세게)
LOG (online2-tcp-nnet3-decode-faster[5.5.958-1539-57f8d]:Accept():online2-tcp-nnet3-decode-faster.cc:711) Accepted connection from: 114.70.22.237
LOG (online2-tcp-nnet3-decode-faster[5.5.958-1539-57f8d]:main():online2-tcp-nnet3-decode-faster.cc:423) @@@@@@@@@@@@@@@@@@1111@@@@@@@@@@@@@@@@@
LOG (online2-tcp-nnet3-decode-faster[5.5.958-1539-57f8d]:main():online2-tcp-nnet3-decode-faster.cc:438) @@@@@@@@@@@@@@@@@@2222@@@@@@@@@@@@@@@@@
LOG (online2-tcp-nnet3-decode-faster[5.5.958-1539-57f8d]:Accept():online2-tcp-nnet3-decode-faster.cc:691) Waiting for client...
connect success
```

connect success가 뜰 때까지 잠시 대기한다.

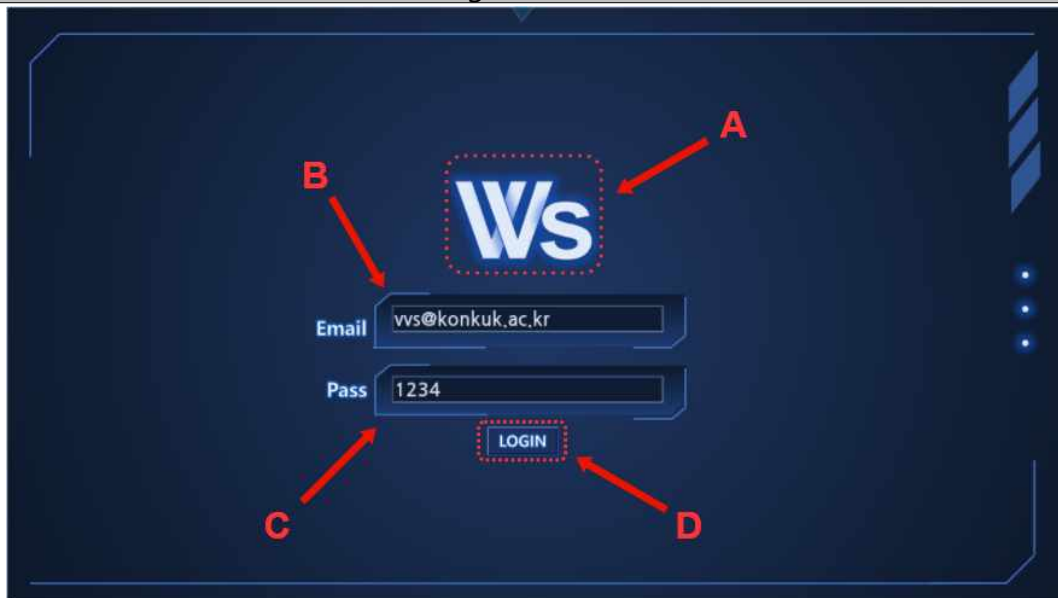
raspberry pi setting



기호	설명
A	sh start.sh

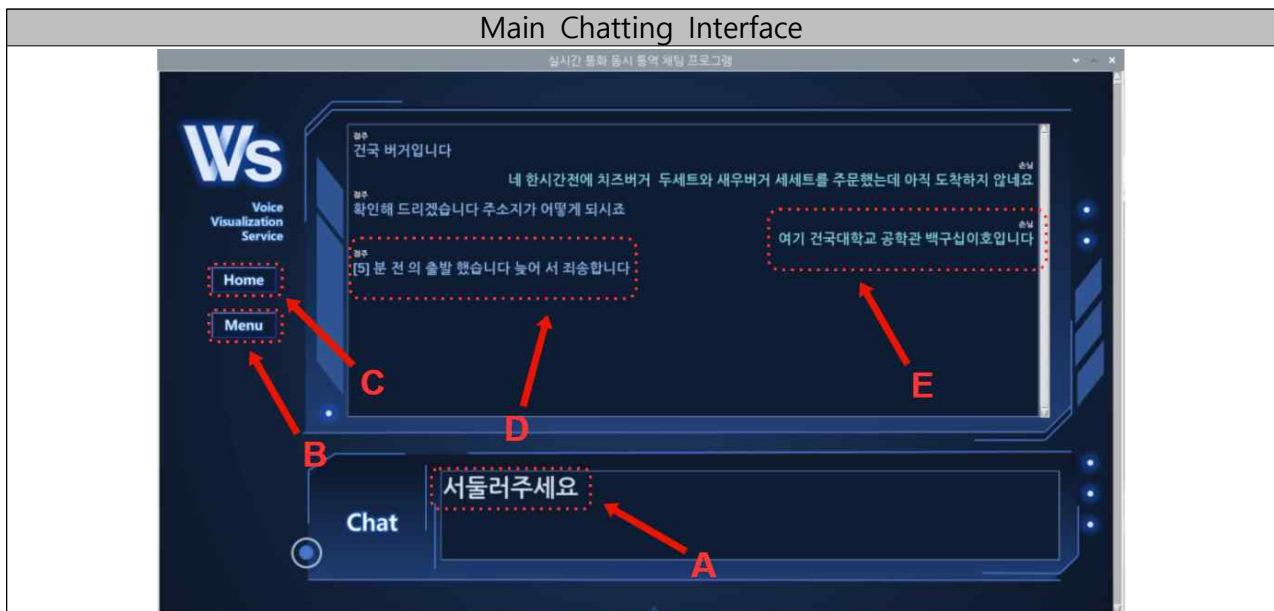
: Chatting client, TTS receive client, rec을 실행시키기 위해 start.sh파일을 실행한다.

Login Interface



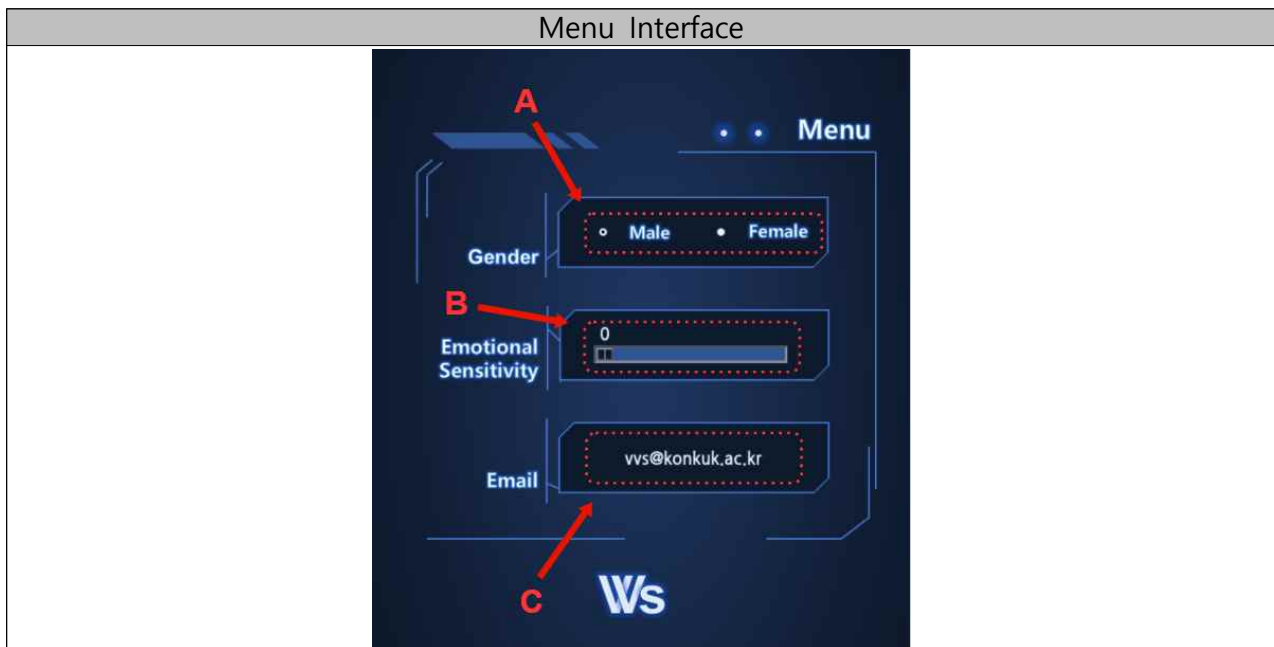
기호	설명
A	우리 팀의 로고 이미지이며 VVS에서 V 2개를 합쳐서 만든 로고이다.
B	사용자의 Email을 입력하는 공간이다.
C	사용자의 Password를 입력하는 공간이다.
D	Email과 Password를 모두 입력한 후 누르면 Login이 진행된다. Enter로도 접근할 수 있다.

사용자의 Email과 Password를 입력한 후 Enter를 입력하거나 Login 버튼을 누른다.



기호	설명
A	사용자가 텍스트를 입력하는 공간이다.
B	사용자가 성별, 감도를 설정하거나 Information을 열람할 때 사용하는 버튼이다.
C	사용자가 프로그램 사용을 종료하고 싶을 때 누르는 버튼이다.
D	점주의 말이 실시간 번역되어 좌측에 텍스트로 입력된다. 이때 처음에는 RealTime STT가 들어오고 이후 Decode된 STT, Spacing된 STT가 차례로 들어오고 마지막에 최종 Rescore STT가 들어오게 된다.
E	사용자가 입력한 텍스트가 우측에 입력된다.

점주가 말하는 텍스트가 좌측에 실시간으로 입력되면 하단 입력창에 키보드로 입력을 하면서 대화를 이어나가는 방식이다.



기호	설명
A	사용자가 TTS 성별을 설정할 수 있는 곳이다.
B	사용자가 TTS 감도를 설정할 수 있는 곳이다.
C	사용자의 Information을 확인할 수 있는 곳이다.

이곳을 통해 TTS 성별, 감도를 설정하고 Information을 확인할 수 있다.

○ 개발환경 (언어, Tool, 사용시스템 등)

- 사용PC

1. Work Station (Main)

CPU: Intel(R) Xeon(R) CPU E5-2640 v3

VGA: GTX1080Ti SLI

RAM: 32GB (+ Swap Memory 96GB)

HDD: SSD 1TB / HDD 3TB

CUDA 10.1

CUDNN 7.6.5

2. Work Station2 (Sub)

CPU: Intel(R) Xeon(R) CPU W-2223

VGA: RTX3080

RAM: 64GB (+ Swap Memory 128GB)

HDD: SSD 1TB / HDD 4TB

CUDA 11.1

CUDNN 8.0.5

- 사용언어

Perl Script, Shell Script, Python, C++

- 사용 라이브러리

ESPNet 0.7.0

Tensorflow 2.4.0

Pytorch 1.9.0

numpy >= 1.21.2

Pydub

koBERT

gTTS

Panda

pyaudio

Jamo

자세한건 [github](#)참고

□ 개발 프로그램 설명

○ 파일 구성

자세한 파일 구성도는 [github\(https://github.com/kusw1006/2021ESWContest_free_1049/\)](https://github.com/kusw1006/2021ESWContest_free_1049/)를 참고해주세요
VVS:

```
├─home
│   ├──Sentimental
│   ├──spacing
│   └─TTS
├─opt
│   ├──kaldi
│   │   ├──bin
│   │   ├──src
│   │   │   └─online2bin
│   │   └─tools
│   │       └─extras
│   ├──models
│   └─zeroth
│       └─s5
│           ├──data
│           │   └─local
│           │       └─lm
│           │           ├──buildLM
│           │           │   ├──_courpus_task_
│           │           │   └─_scripts_
│           │           └─_scripts_
│           ├──local
│           │   ├──chain
│           │   │   └─multi_condition
│           │   └─nnet3
│           │       └─multi_condition
│           ├──steps
│           │   ├──data
│           │   └─dict
│           └─utils
│               └─lang
└─raspberryPi
    ├──API
    │   ├──stt
    │   │   └─stt_test
    │   └─tts
    ├──src
    └─utils
```

○ 함수별 기능

main server

함수	기능
AdvanceDecoding	디코딩 가능한 객체에 더 이상 준비된 프레임이 없을 때까지 디코딩한다.
GetBestPath	여러 경로중 가장 가능성 있는 경로를 출력한다.
FinalizeDecoding	남은 token을 정리해 후에 GetLattice를 진행할 때 빠른속도로 동작할 수 있게 한다.
rescoring	<ul style="list-style-type: none"> - main 함수에서 한 문장이 decode되면 해당 문장이 몇 번째 문장인지 나타내는 id와 클라이언트 정보를 담고있는 clnt_socks, 학습을 통해 생긴 단어장인 symbol table을 구조체를 통하여 한번에 인자로 받고 main thread에서 분기하는 thread 함수다. - 필요한 변수를 선언한 후, /opt/zeroth/s5/local에 있는 test.sh파일을 불러온다. test.sh파일은 decode에서 발생한 lattice들을 utt와 함께 저장시켜 놓은 wspecifier를 rescore하는 파일이며, rescore 결과를 출력해준다. - rescoring 함수에서 popen으로 test.sh파일을 불러오면 rescore 결과 뿐만아니라 다양한 Log를 함께 전달받기 때문에 이중 utt값인 000을 포함하고있는 문장만 찾아서 rescoreBuf에 저장한다. - rescore 결과는 단어의 라벨로 출력되기 때문에 실제 문장을 얻기 위해서는 이 라벨에 해당하는 단어들을 단어장인 symbol table에서 찾아야한다. rescore 결과에서 라벨들을 추출하고 word_syms.Find 함수를 이용하여 라벨에 맞는 단어를 찾아낸다. 이 단어들을 R{num}토큰과 함께 최종적으로 rescoreStr에 string 형태로 append시키고 다시 char 형태로 변환시켜서 띄어쓰기 클라이언트에 보내준다.
recv_msg_space	<ul style="list-style-type: none"> - 띄어쓰기된 문장을 받아와서 채팅 클라이언트에게 보내줌 main 함수에서 새로 음성을 받아올 때마다 main thread에서 분기하여 병렬적으로 동작하는 함수로, rescoring함수와 같은 구조체를 인자로 받는다. - 필요한 인자를 선언한 후 띄어쓰기 클라이언트에서 문자열을 받아온다. 받아오는데 성공하면 ret에 1이 들어가게 되고 if문을 통해 채팅 클라이언트에게 해당 문자열을 전송해준다.
recv_msg_chat	<ul style="list-style-type: none"> - 채팅 클라이언트에게서 사용자가 입력한 텍스트를 받아와서 TTS서버에 보내줌 - void* recv_msg_space(void* _Package)와 함께 분기하며 인자도 같은 구조체를 받아온다. 사용자가 입력한 채팅을 채팅 클라이언트에서 전송해주면 이를 받아서 TTS 서버에게 넘겨준다.

Spacing client

recv	<p>main에서 띄어쓰기 모델을 준비하고 서버와의 연결을 마치면 분기되는 thread 함수이며 인자로 서버와 연결된 소켓 객체를 이용한다.</p> <p>함수가 실행되면 서버로부터 인코딩된 데이터를 1024바이트씩 받게되며 이를 디코딩해서 전역변수인 recv_data에 넣어준다</p>
get_inference_fn	<p>텐서의 " + "를 " "로 바꾼 후 split해서 byte_array에 가로방향으로 concat해준다. byte_array를 main에서 만든 vocab_table에 넣어 byte에 해당하는 string을 찾아준다. 미리 정의해준 model에 string을 넣고 가장 (확률이) 큰값의 인덱스를 반환하여 model_output에 저장한 후 byte_array와 model_output을 convert_output_to_string 함수에 전달한다.</p>
convert_output_to_string	<p>model_output의 값이 1이면 띄어쓰기를 추가해야하고 2이면 띄어쓰기를 제거하는 연산을 한다. 이때 strings_result는 tensorflow 배열이며 값은 바이트 타입이다.</p>
send	<p>recv(client_sock)과 같이 분기되는 thread 함수이며 인자로 소켓 객체와 inference함수를 넘겨받는다.</p> <p>함수가 실행되고 전역변수인 recv_data에 문장이 들어있다면 먼저 들어온 문장부터 pop 해서 target_str에 넣고 find함수를 이용해 토큰을 찾아준다. 이후 토큰은 token에 문장은 target_str에 분리하여 넣는다.</p> <p>target_str은 tf.constant를 이용해상수 텐서로 바꿔주고 inference함수를 이용해 띄어쓰기 보정한다. 이때 결과값인 result를 numpy배열 형태로 바꿔준 뒤 b"" 내부에 합쳐준 후 한 번에 decode해서 result_str을 얻어낸다.</p> <p>토큰은 식별을 위해 T->C, R->D로 바꿔야하므로 find함수와 replace함수를 이용해 띄어쓰기 토큰으로 바꿔준다. 마지막으로 토큰과 문장을 다시 합친 후 인코딩해서 서버에 보내준다.</p>

TTS & Sentimental server

frontend	<p>Text2grp.split_syllables(text)를 이용해 특수문자 or 띄어쓰기 or 초중종성을 new_c, new_g에 추가해준다. 공백문자는 띄어쓰기로 치환하며, 문장 끝은 . , ! ? 로 맞춰준다. 마지막으로 공백, table에없는문자, 그밖(한국어)에 해당하는 id를 <eos> 추가와 함께 idseq에 저장한다.</p>
recv	<p>main server와 연결해주며 1024바이트씩 끊어서 데이터를 받아온다. 받아오는데 성공하면 디코딩하여 recv_data에 append해준다.</p>
send	<p>음성을 빠른속도로 전달하기 위해 다른 포트를 열어서 TTS receive client와 연결한다. 연결에 성공하면 받아온 데이터의 가장 앞에 붙어있는 토큰을 분석해서</p>

	<p>화자의 성별, 감정 표현 정도를 정한다. 이후 getSentimentValue에서 미리 학습된 data를 바탕으로 text를 분석해서 가장 높은 확률을 가진 감정의 인덱스를 반환하면 해당 인덱스에 맞는 감정표현을 input_emotion에 넣어준다. 화자 및 감정에 대한 설정을 마치고 input_text를 frontend에 인자로 넣어서 전처리 해준 다음 model.inference를 통해 mel spectrogram을 추론한다. 이후 cmvn에 적용한 뒤에 가우시안 노이즈를 추가한 다음 parallel_wavgan의 입력으로 넣어 synthesis를 함으로써 음성을 만들어낸다. 마지막으로 이 결과물을 TTS receive client에게 전달하기 위해 numpy배열 형태로 파일에 저장한 뒤 해당 파일을 TTS receive client에게 전송한다.</p>
--	--

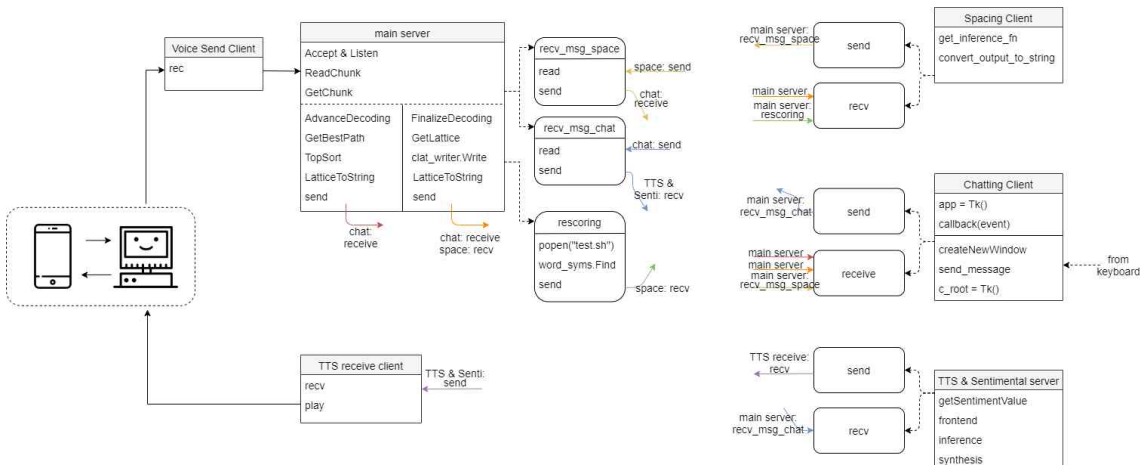
함수	기능
engkor	영어로 들어온 입력값을 한국어로 바꿔준다.
send	Textbox에 입력된 텍스트를 불러와 Main Server에 전송하고 Chat_log textbox에 채팅 형식으로 표시한다.
send_message	키를 맵핑하여 각 키에 해당하는 영어를 한국어로 바꿔준다.
receive	Main Server로부터 받아온 텍스트와 토큰을 토큰에 맞게 분리하고 그에 맞게 Chat_log textbox에 텍스트를 채팅 형식으로 표시한다.
login	Main Server에 접속하며 Thread를 실행하여 send, receive 함수를 Thread로 열어 놓고 프로그램을 실행하기 위한 추가적인 shell을 실행한다.
set_go_send	send thread가 동작할 수 있도록 한다.
scale_get	scale 값을 저장하고 전송한다.
set_menu	Menu Interface를 구성한다.
createNewWindow	Main Chatting Interface를 구성한다.
callback	enter 혹은 login 버튼을 눌렀을 때 login 여부를 결정한다.
Main	처음 login interface를 실행한다.

TTS receive client

- TTS server로부터 wav를 받아 음성으로 변환해서 play
- 데이터를 한번에 모두 받는 방식이 아니라 1024씩 끊어서 받기 때문에 먼저 TTS & Sentimental server로부터 파일의 크기를 받아온다. 이후 파일의 크기만큼 받을 때까지 1024씩 데이터를 받아오고 모두 받으면 mp3파일로 변환한 뒤 이를 play한다.

○ 주요 함수의 흐름도

주요 함수 흐름도

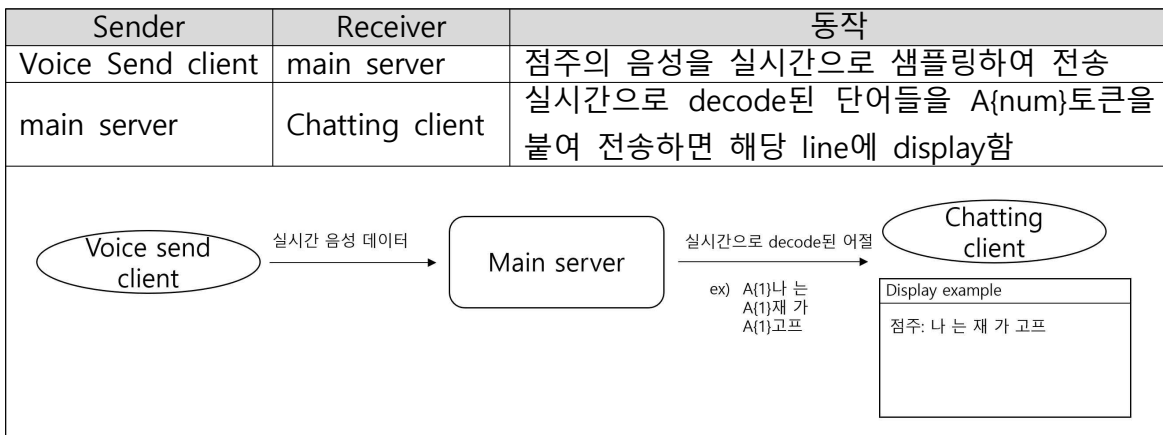


사각형: client & server

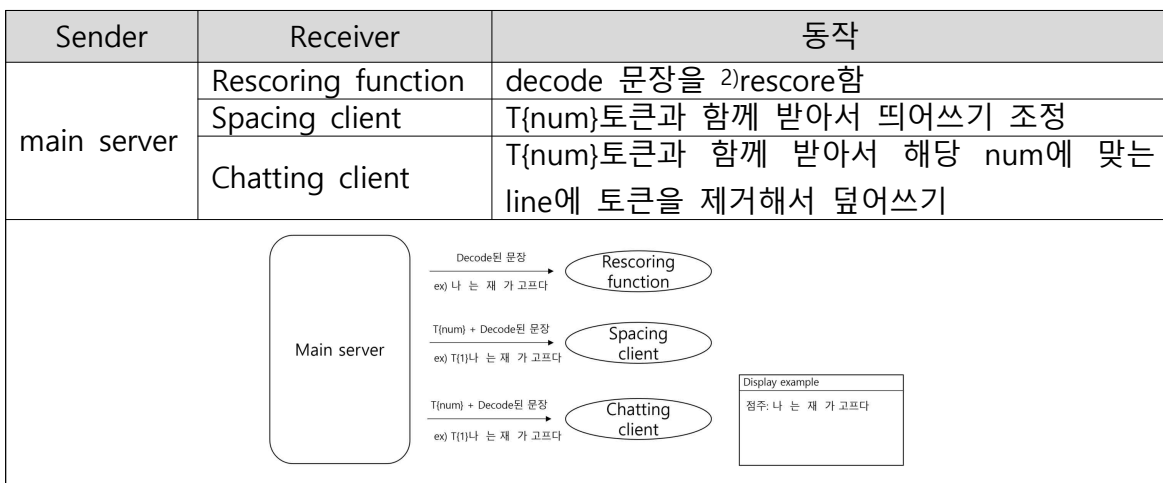
라운드 사각형: client & server의 thread 함수

색상 화살표: 같은 색의 화살표끼리 연결되어 있음

1. 실시간 단어 디코딩

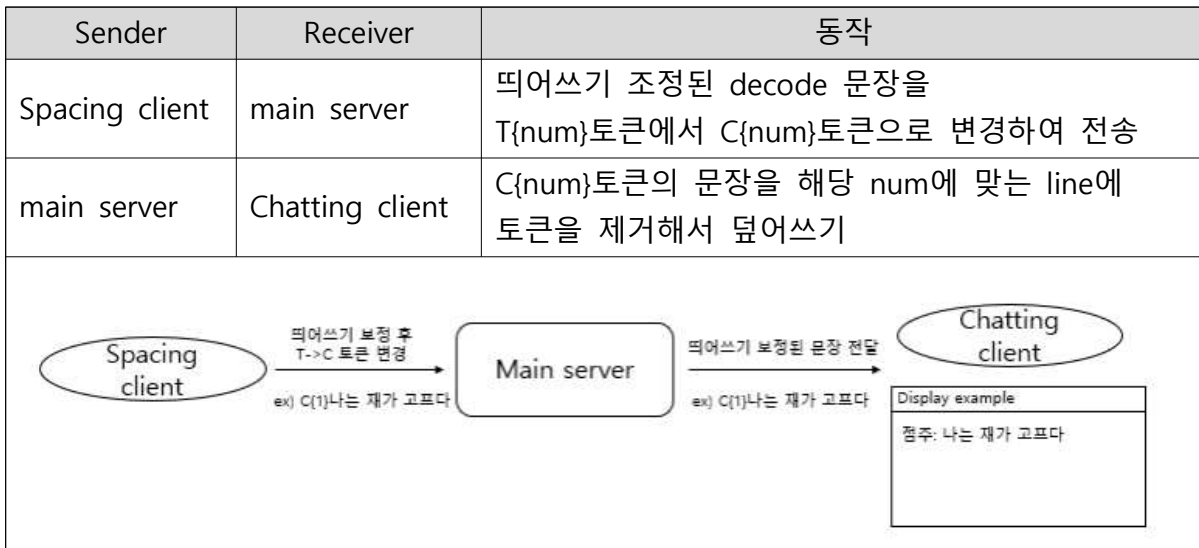


2. 디코딩된 문장

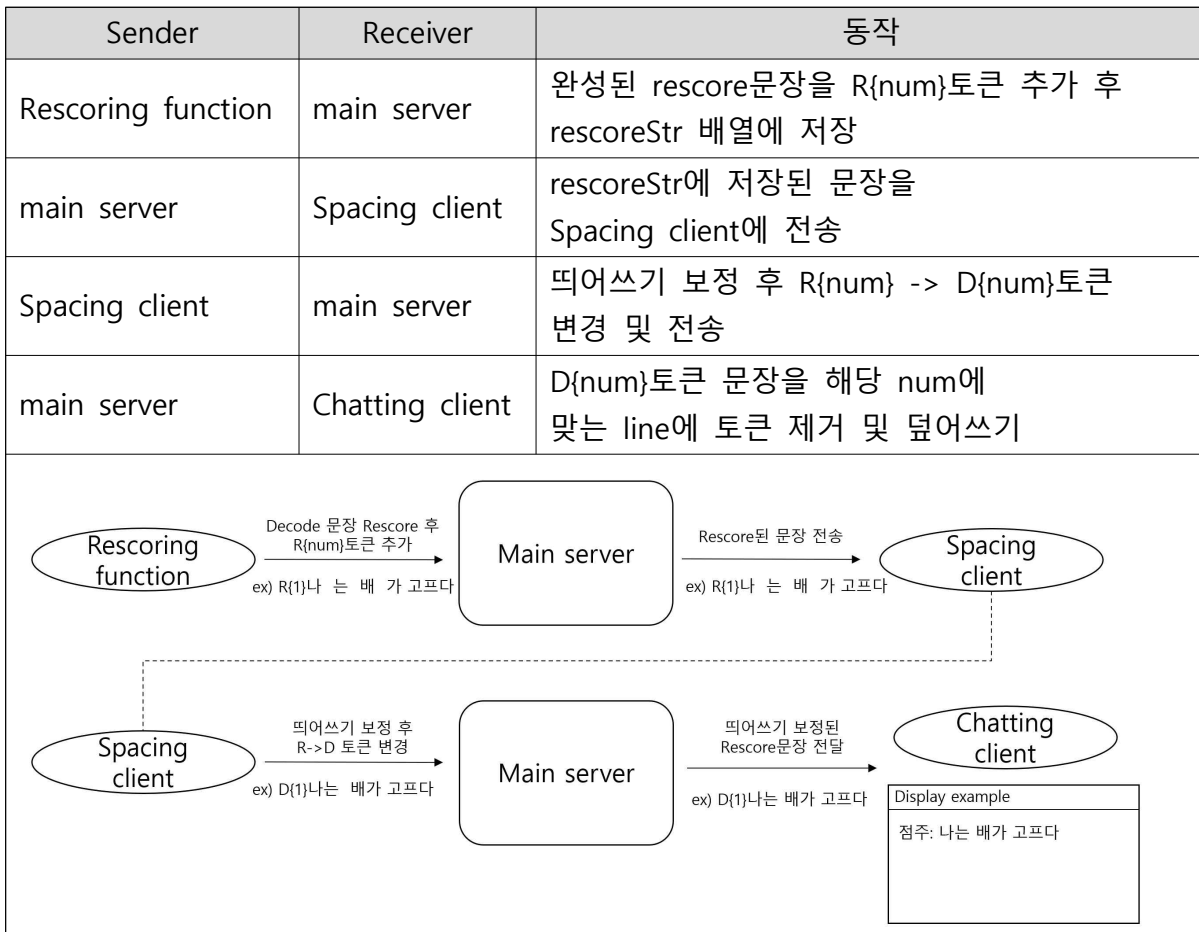


2) rescore란 기존 모델의 단어 매칭확률을 더 정확한 모델에서의 확률로 재계산 및 적용함을 의미한다.

3. 띄어쓰기 조정된 디코드 문장



4. 띄어쓰기 조정된 rescore 문장



5. TTS 과정

Sender	Receiver	동작
Chatting client	main server	입력된 텍스트를 main server에 전송

main server	TTS & Sentimental server	텍스트를 TTS & Sentimental server에 전송
TTS & Sentimental server	TTS receive client	Sentimental model의 감정 추출 추출된 감정을 나타내는 역양의 wav 파일을 생성해 byte 형식으로 전송
TTS receive client	Phone	byte 형식을 wav 파일로 변환하여 재생, 점주에게 전달

6. 사용자 인터페이스

Interface	작동 조건	동작
Login	코드 실행 시 팝업	사용자 정보와 입력이 일치 시 Login을 성공시키고 Main 서버와 연결
Chatting	Login 성공	서버로부터 받은 A, C, D, T 토큰을 Chatting Log로 띄우고 입력 텍스트 서버전송
Menu	Menu 버튼 작동	성별, Scale, Information을 변경할 수 있음


```

graph LR
    LoginUI(Login UI) -- "로그인 성공시 TCP 서버 접속" --> MainChattingUI(Main Chatting UI)
    MainChattingUI -- "Home 버튼 눌리면 종료" --> LoginUI
    MainChattingUI -- "Menu 버튼 눌리면 실행" --> MenuUI(Menu UI)
    MenuUI -- "Scale, 성별 변경시 반영" --> MainChattingUI
  
```

○ 기술적 차별성

추가 단어집의 빠른 결합

- (Github 참고) 원하는 Task domain에게 FineTuning³⁾이 쉽도록 설명 및 시스템을 구축하였다.
- 수백만 개의 각 Task domain 관련된 다양한 단어집들을 기존 시스템에 여러 번 추가가 가능하다. 현재 음향모델, 언어모델 시스템에서 시뮬레이션 결과 최대 24시간 이내에 수백만 개의 단어를 기존 시스템에 추가해 결합할 수 있음을 확인하였다. VVS와 비교했을 때 다른 API들은 몇 가지 단어에 대한 인식률을 높이는 기능은 있으나 별도의 Task domain에 특화된 음성인식을 만들기는 어려운 구조다.

뛰어난 유선전화 인식

- 기존 한국어 음성인식 시스템들은 모두 일반적인 휴대폰 마이크 환경에서의 인식을 목표로 하여 16kHz 이상으로 샘플링된 음성 데이터를 가지고 인식하지만, 유선전화에서는 4kHz로 샘플링되기 때문에 음성인식에 취약하다. 하지만 VVS는 다양한 Sampling Rate와 그 중 특히 전화선 Sampling Rate에 특화하여 학습시키므로 유선전화 인식에 뛰어난 성능을 보인다.

3) Fine Tuning이란 기존 학습된 모델을 기반으로 Architecture를 새로운 목적(나의 이미지 데이터에 맞게) 변형하고 이미 학습된 모델 Weights로부터 학습을 업데이트하는 방법을 의미한다.

Kaldi System 내의 실시간 Rescoring 개발

- 현시점까지 Kaldi System을 이용한 STT 모델 중 실시간으로 rescoring 하는 시스템은 찾아보기 힘들다. VVS는 실시간 분석에 사용하기 위해 인스턴스된 정보들을 그대로 가져와 multi threading 하는 방식을 이용했으며 이를 통해 앞에 들어온 정보들로부터 병렬적으로 rescoring을 진행하는 시스템을 개발하였다.
- 개발한 시스템을 통해 VVS는 실시간 rescoring을 진행할 수 있고, 프로그램이 실행되는 도중 잘못 인식된 단어나 문장이 있더라도 이를 확률에 기반, 고칠 수 있다. 실제 Google STT와 비교했을 때 '치즈 볼'이란 단어를 Google STT는 'CGV'로 인식 후 이를 고치지 못하는 반면, VVS는 인식을 어려워했으나 이후 rescoring 과정을 통해 '치즈 볼'로 제대로 인식한 것을 확인했다.

□ 개발 중 발생한 장애요인과 해결방안

Kaldi 구현

- 기존 interpolation 방식이 아닌 Phrase hint 방식으로 구현하는데 정보가 kaldi, zeroth 커뮤니티에도 없었다.
- 지나치게 데이터를 변형할 경우 음소에 대한 파형의 특징을 잡기 어려워져 HMM-GMM을 통한 음소-파형 alignment 학습이 어려울 수 있기 때문에 적절한 양을 조절하는데 어려움을 겪었고, 최대한 음소학습시 파형 변형이 크지않도록 multi condition 학습의 위치를 조절하는데 어려움을 겪었다.
 - 이를 해결하기 위해 음형을 보며, Clipping이 일어나지 않도록 노이즈를 추가할 때 SNR을 조절하였으며, 원래의 음소 파형의 특성을 잃으면 음소와의 alignment가 불가능해져, 최대한 노이즈 추가를 DNN정렬시에 처리할 수 있도록 순서를 최대한 조절했다.

Server 구현

- 기존 tcp decoder에서는 rescore를 사용할 수 없는 구조였고, 이를 구현한 reference자료 또한 존재하지 않았다.
 - 다른 decoder에서 사용하는 rescore 파일을 불러와서 thread를 통해 병렬적으로 실행시킴으로써 rescore를 구현함
- 먼저 들어온 문장이 rescore 되기 전에 다음 문장이 들어오면 rescore함수를 중복해서 사용하면서 변수값 변형이 일어났다.
 - 각 문장마다 id를 부여하고 thread와 결과를 저장하는 string을 배열로 선언하여 각 id에 해당하는 index에서만 동작하도록 했다.
- intel mkl 동적 라이브러리 버전 설정 오류가 계속해서 발생했다.
 - 호환되는 버전의 라이브러리를 설치하고 관련된 path설정을 모두 바꿈

HW & UI 구현

- 라즈비안에서 python tkinter 라이브러리를 사용할 때 textbox UI에서 한글이 써지지 않는 문제가 발생했다.
 - 오토마타를 활용하여 영어 문자로 들어오는 키 값을 맵핑 후 한국어의 음소 단위로 변환해주는 방식으로 해결하였다.
- 라즈베리파이 내장 AV Jack과 USB 사운드카드를 이용해 시도했으나 라즈베리파이에 마이크 입력회로가 내장되어 있지 않아 AV Jack으로 소리의 입력을 받아오지 못했다.
- Bluetooth를 이용했으나 쌍방향 소리 입출력을 핸드폰에서 받아오지 못하고 Bluetooth를 on/off 시키는 방식으로 해야 했다. 또한 라즈베리파이 내장 블루투스 모듈의 성능이 좋지 않아 지속적으로 연결이 끊기는 문제가 발생했다.
- 블루투스 dongle 4.0과 블루투스 송신기를 이용해 휴대폰의 소리를 라즈베리파이로 전달해봄. 블루투스만을 이용한 양방향은 불가능하고, 하울링 문제가 발생했다.
 - 이 부분들을 해결하기 위한 HiFiBerry DAC+ ADC pro 보드를 이용함.

□ 개발결과물의 차별성

API

- 무료, 실시간, 오프라인 구현이 가능하다.
 - : 음성인식을 사용하는 여러 기업에서 STT나 TTS API를 제공하나, 비용이 필요하고, 실시간이나 오프라인 구현이 어렵다는 단점이 있다. VVS는 이와 비교했을 때 무료로 이용할 수 있고, 실시간 동작과 오프라인 구현 모두 가능하다.
- 사용하는 domain에 빠른 특화 가능
 - : 배달 산업의 경우 신메뉴가 올라오는 속도가 빠르며 신조어의 사용 비율이 높다. 이에 따른 즉각적인 대응이 필요한데, API를 사용할 경우 이런 대응이 어렵다.
- 다양한 상용품에 준하는 성능
 - : Zeroth EE를 사용하는 기업들은 점점 늘어나고 있다. 하지만 유선전화를 이용한 한국어 STT 서비스 중에서는 현 시점까지 VVS가 유일하다.

Hear World Communication _ Captioned Speech-to-Text Telephones (2011)

- 양방향 통신
 - : Captioned STT Telephones는 2011년 미국에서 출시된 STT 전화기로, 유선전화를 통해 상대의 말을 텍스트로만 보여주는 제품이다. VVS는 여기에 사용자의 텍스트까지 상대방에게 음성으로 전달하는 TTS 기능을 통해 양방향으로 통신할 수 있다.
- 기존 휴대폰 이용 가능
 - : 유선전화는 점점 사라지는 추세다. 위 제품과 달리 VVS는 기존 휴대폰을 유지하며 별도의 시스템 지원 없이 사용할 수 있다.
- 한국어의 지원
 - : Caption STT Telephones는 미국 제품이다 보니 영어만을 지원한다. 영어와 달리 교착어에 속하는 한국어는 음성인식이 매우 까다롭다. VVS는 한국어에 특화된 제품인 만큼 한국어 사용자들도 이용하기 편리하다.

KT _ 인공지능 음성합성 '마음 톡' & '나를(narle)' 손말 영상통화 서비스

- 음성통화에서의 구현
 - : 영상통화를 기반으로 하는 위 두 서비스는 청각장애인이 입력한 문자가 음성으로 상대방에게 전달되거나, 아바타나 AR 이모티콘을 통한 수어 통화가 가능하다. 하지만 모르는 사람과의 통화나 영상을 이용하기 어려운 환경 등 음성통화를 해야 하는 상황에서도 이용할 수 있다.
- 텍스트의 감정 표현
 - : 영상통화보다는 텍스트가 여러 환경에 적용하기 편리하다. 하지만 텍스트만을 이용했을 때 여러 단점들이 있는데, 예를 들어 감정 표현이 있다. VVS는 문장분석 및 감정TTS를 통한 사용자의 '감정'까지 전달이 가능하다.
- 일반 요금제에서도 사용 가능
 - : '마음 톡'이나 '나를' 서비스의 경우 별도의 요금제 또는 앱이 필요하다. VVS는 제품만 있다면 일반 요금제의 통화 환경에서도 충분히 STT와 TTS를 구현한다.

□ 개발 일정

No	내용	2021年											
		6月			7月			8月			9月		
연구 및 세팅	딥러닝 및 NLP 연구												
	Kaldi System 분석												
	서버PC 세팅												
	Zeroth 구조 분석												
LM 제작 및 학습	1차 말뭉치 수집 및 정제												
	HCLG 1, 2차 학습 및 테스트												
	Phrase Hint System 개발												
	2차 말뭉치 수집 및 정제												
	HCLG 3, 4차 학습 및 테스트												
	HCLG 5차 학습 및 테스트												
AM 제작	디지털 신호처리 학습												
	음향 모델 최적화												
	음향 데이터(Kspon Speech) 이식												
TCP 서버 및 TTS STT 개발	API 기반 비교용 TTS, STT 클라이언트 제작												
	Tacotron2 기반 TTS 제작												
	띄어쓰기 클라이언트 제작												
	감정분석 클라이언트 제작												
	다화자, 다감정 TTS 클라이언트 제작												
	채팅 클라이언트 제작												
하드 웨어	블루투스 기반 하드웨어 제작												
	HifiBerry 기반 하드웨어 제작												
	하드웨어 외관 제작												
	시험 평가 및 테스트												
제작 마무 리	트러블 슈팅												
	영상 촬영 및 제작												
	최종 결과 보고서 작성												

□ 팀 업무 분장

No	구분	성명	참여인원의 업무 분장
1	팀장	이찬현	전체 업무 기획, 총괄 및 개발 업무 분배 kaldi 개발을 통해 특정 Task Domain 추가 기능 개발 서버 환경 개발 담당 보고서 작성
2	팀원	김한비	kaldi Language model 최적화 언어모델 관련 클라이언트 제작 및 최적화 띄어쓰기 시스템 구현 감성분석 시스템 구축 보고서 작성
3	팀원	안진혁	kaldi Acoustic Model 최적화 음향모델 관련 클라이언트 제작 및 최적화 딥러닝 TTS 클라이언트 제작 Chatting 클라이언트 개발 보고서 작성
4	팀원	신지혜	H/W개발 - 라즈베리파이 & 휴대폰 사운드 전송 시스템 구축 API 토대의 STT, TTS 구현 3D Printing - 케이스 제작 UI 개발 시연 영상 편집 보고서 작성