

<AI, BIG DATA 아카데미 15기>

AI Project 최종 보고서

“Hello MetaWorld”

VR기기를 활용한 AI 영어회화 교육 플랫폼

B반 4조

김민경 서형준

손유정 이영호

이찬희 임솔이

목차

1. 프로젝트 소개	3p
1-1. 추진배경 및 목표	3p
1-2. 메타버스와 교육	4p
2. 프로젝트 개요	4p
2-1. AI 챗봇 기능 설명	5p
2-2. 사용기술 소개	6p
2-3. Pipeline 개요	8p
3. 상세 기술 구현	9p
3-1. Unity	9p
3-2. Socket 통신	12p
3-3. Speech To Text	13p
3-4. NLP	16p
3-5. Model 생성	16p
3-6. Text To Speech	21p
4. 결론	24p
4-1. 구현 결과	24p
4-2. 기대효과 및 발전 방향	24p
4-2. 한계점 및 개선방안	25p
부록	26p
Learned lessons.	26p
References	??p

1. 프로젝트 소개

1-1. 추진배경 및 목표

한국인 영어 수준... '읽기'는 22위, '말하기'는 122위
한국사람 왜 영어회화 못 할까요?

일주일에 두 번, 2시간씩 일대일 영어회화 과외를 배운다면?

월 64만원

[참고1] 한국인 회화능력 수준 기사

[참고2] 영어회화 사교육 비용

본 AI 프로젝트는 오픽 시험으로 고민중인 조원 A 양으로부터 시작했다. 오픽 강의를 들어도 직접적인 회화보단 대화 스크립트를 달달 외우는 듯한 학습 방법이고, 원어민과 직접 대화하며 공부해보고 싶지만 일대일 회화교육은 매우 비싸다.

변인	구분	빈도	퍼센트
영어수준	상(1-3등급)	238	51.2
	중(4-6등급)	196	42.2
	하(7-9등급)	25	5.4
	합계(결측)	459(6)	98.8(1.2)
영어회화능력 자신감	높음	52	11.2
	보통	154	33.1
	낮음	259	55.7
	합계(결측)	465(0)	100(0.0)
중·고등학교 원어민 수업경험	경험없다	97	20.9
	1년 이하	108	23.2
	1-2년	129	27.7
	2-3년	71	15.3
	3년이상	60	12.9
	합계(결측)	465(0)	100(0.0)
영어권국가 체류경험	있다	69	14.8
	없다	396	85.2
	합계(결측)	465(0)	100(0.0)

[표1] 한국인 회화능력 수준

참고1의 기사에 따르면 토플 성적을 기준으로 세계 168개국 중 한국의 읽기능력은 22위로 상위권이였으나 말하기 실력은 122위에 그쳤다. 또한 표1¹에 따르면 영어 문법/읽기는 높은 수준을 갖지만 영어회화능력은 이에 미치지 못한다는 모습을 알 수 있다. 그럼에도 불구하고 아직까지 영어 회화 교육은 잘 이루어지지 않고 있기 때문에 누구나 쉽게 접할 수 있는 영어회화 교육시스템이 필요하다. 하지만 2016년 이후 상용화가 가속화된 VR, AR 등 최신 기술로 이루어진 디지털 러닝 시스템의 상용화는 미흡한 상태로, 10~20대로 이루어진 MZ세대 학생 및 대학생, 성인들의 필요를 충족하기에는 부족한 상황이다. 따라서 본 AI 프로젝트는 어디서나, 누구나, 언제든지 쉽게 접할 수 있는 체험형 영어회화 학습 프로그램을 구현하고자 한다.

¹ '원어민담당 영어회화 강의에 대한 대학생들의 인식 연구'(2016), 유범 광경섭

1-2. 메타버스와 교육

한국산업기술대, 메타버스 공학교육시스템 구축

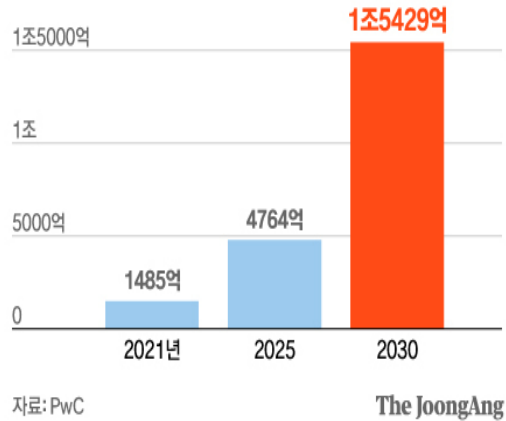
A | 임원지 기보 | © 양력 2022.06.06 15:15 | 100 댓글 1

| 혁신적인 미래 디지털 교육인프라 '퓨처VR랩' 모델 자체 개발



메타버스 공학교육시스템 '퓨처VR랩'에서 시뮬레이션 학습을 하는 모습 (사진: 한국산업기술대학교 제공)

단위: 달러



[참고3] 메타버스 교육 적용사례

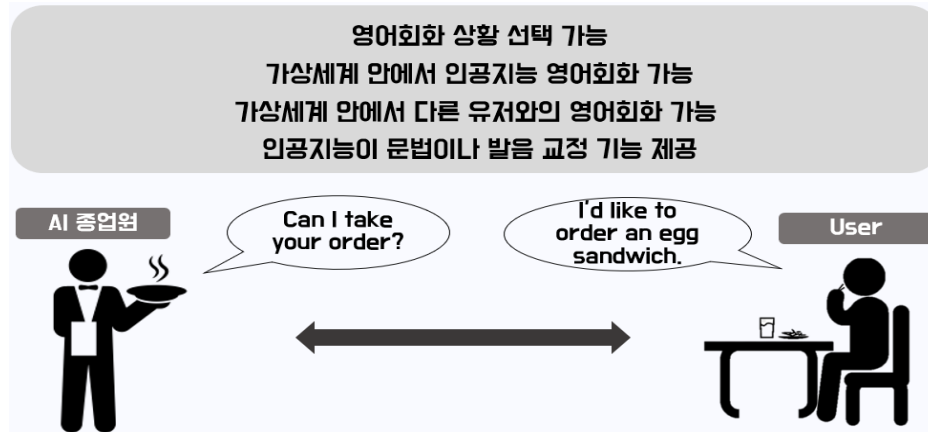
[그림1] 메타버스 시장 규모

메타버스는 '가상', '초월' 등을 뜻하는 영어 단어 '메타'(Meta)와 우주를 뜻하는 '유니버스'(Universe)의 합성어로, 현실세계와 같은 사회·경제·문화 활동이 이뤄지는 3차원의 가상세계를 가리킨다. 시장규모의 급성장 코로나19 사태가 맞물리며 비대면 교육이 중요시되는 현 상황에 교육계 또한 메타버스로 시선을 돌리고 있다. 참고3의 기사와 같이 대학에서도 메타버스를 이용해 비대면 온라인 실시간 강의를 하고자 하며 특히 비대면 교육으로 인해 학업성취도 저하가 크게 나타난 초등교육에서도 메타버스를 적용하기 위한 노력 중이다.

2. 프로젝트 개요

본 AI 프로젝트는 메타버스 안에서 학습 시스템을 구축하는 것을 중점으로 가상환경에서 영어 회화 체험을 목표로 한다. 기존의 챗봇 시스템은 단순히 정해진 규칙에 맞춰서 메시지를 입력하면 발화를 출력해주거나, 상대방의 발화를 분석하여 인공지능에 가까운 발화를 내놓는 수준에 그친다. 하지만 본 프로젝트는 다양한 상황 중 원하는 환경을 선택해 영어회화를 체험해 볼 수 있고, 인공지능을 통해 문법, 발음 등을 교정하도록 한다.

실제 대만 HTC에서 2달간 진행한 연구 결과, 가상환경 사용 시 표현 자신감이 상대적으로 10배 가량 높아지고 외국어 학습 효과를 2.5배 높일 수 있다고 분석했다. 이러한 연구 결과를 바탕으로 가상환경에서 실제 발화를 통한 영어회화 교육 콘텐츠를 구현하기로 결정했다. 나아가 사용자의 선택에 따라 인공지능뿐만 아니라 다른 사용자와 1:1로 대화할 수 있도록 하는 메타버스 세계를 구축하여 영어 학습에 대한 지속성과 능동성을 강화하도록 한다.



[그림2] 프로젝트 개요

2-1. AI 챗봇 기능 설명

본 프로젝트에서 구현하고자 하는 AI는 대화 참여자로서 적절한 답변을 제공해줌과 동시에 문법 및 발음 교정 기능을 제공하게 된다. 이를 위해서 VR기기를 통해 사용자가 발화를 하면 이에 대한 적절한 답변을 출력해주는 기능을 구현한다.

사용자의 음성(Wave file)을 인식한 VR 기기가 음성파일을 생성한다. 이를 글(Text file)로 변환하여 챗봇 시스템에 입력하면 챗봇은 해당 질문에 대한 답변(Text file)을 생성해준다. 챗봇이 제공한 답변은 다시 음성 파일(Wave file)로 변환하여 VR 헤드셋으로 출력하면 사용자가 들을 수 있게 된다. 다음은 이러한 프로세스를 구현하기 위한 AI 종업원의 기능을 정리한 것이다.

(1) AI 음성 발화 기능

AI 웨이터가 사용자의 발화에 대한 적절한 답변을 음성으로 제공하는 기능이다. 이때 적절한 답변이란, 학습시킨 영어 대화 데이터에서 가장 자연스러운 대화를 만들어주는 것이다. 본 프로젝트에서는 다양한 상황에 대한 답변을 제공하도록 구성했다.

(2) 문법 교정 기능

사용자의 발화 내용 중 부적절한 문법을 교정해주는 기능이다. 이를 말상자에 출력하여 사용자가 틀린 문법을 교육할 수 있도록 한다.

(3) 발음 확인 기능

사용자의 발화에 대한 정확한 발음을 들려주는 기능이다. 가상환경 내에서 특정 버튼을 클릭하면 정확한 발음에 대한 음성을 출력하도록 한다.

2-2. 사용 기술 소개

(1) Unity

Unity는 3D 및 2D 비디오 게임 개발환경을 제공하는 게임 엔진이자 상호작용이 가능한 가상현실 콘텐츠 제작을 위한 통합 제작 도구이다. 2005년 Apple 세계 개발자 회의에서 최초 공개되었으며, 2016년 기준 전 세계 게임엔진 시장의 45%를 차지하는 등 현재까지도 가장 인기있는 제작 도구이다. 본 프로젝트에서는 메타버스 내 환경 및 그래픽 구축에 활용되었다.

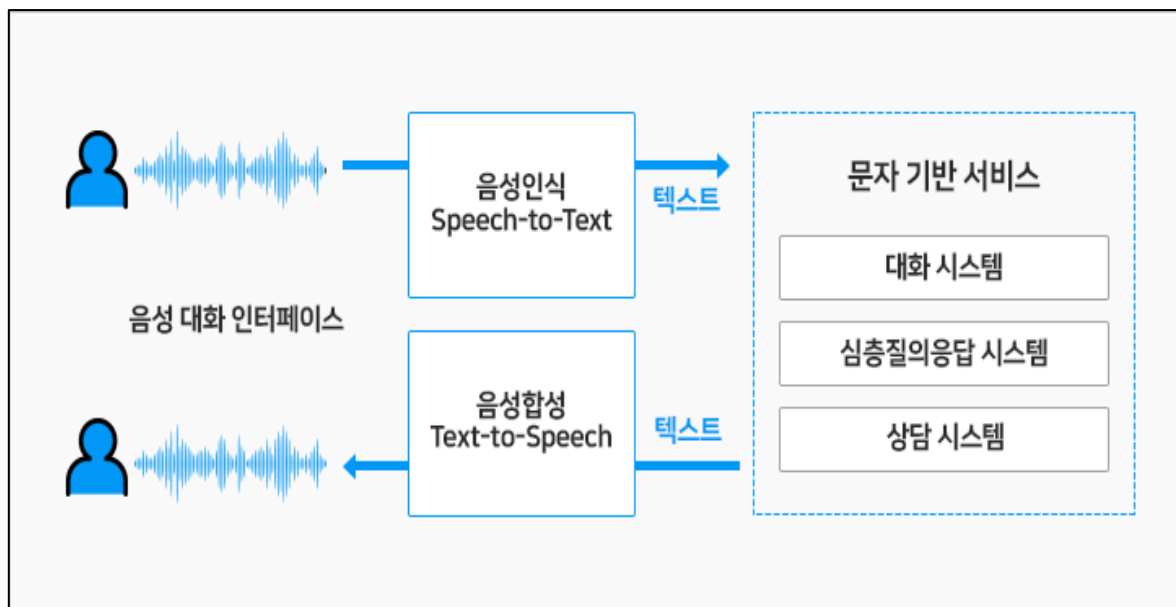


[그림 3] Unity 로고

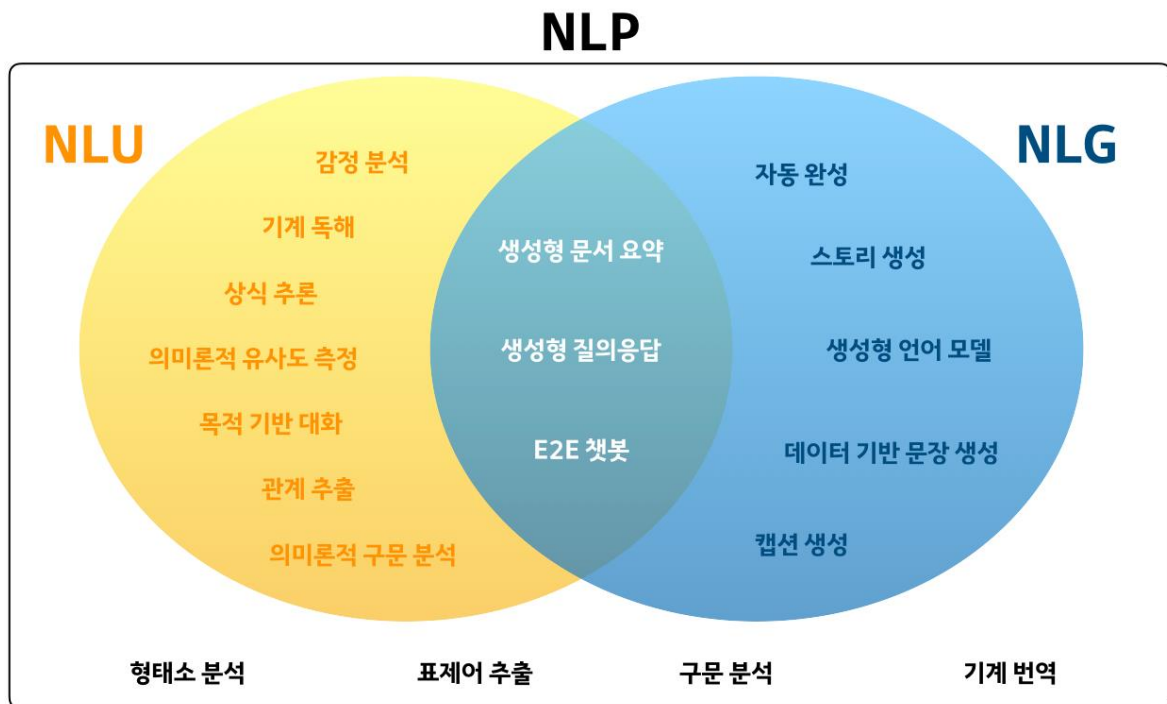
(2) Socket 통신

Unity와 python 간 텍스트 및 음성 정보를 주고받기 위한 데이터 전송 방식이다. 쌍방향 통신이기 때문에 Unity와 Python은 정보 전송의 주체 여부에 따라 Client와 Server로 작용할 수 있다. Unity에서 Speech to text 기술을 거친 텍스트가 python으로 전송되며, 이를 기반으로 구현한 AI 종업원의 음성이 Python에서 Unity로 전송된다.

(3) 음성언어 처리 기술



[그림4] 음성 처리 구조



[그림5] NLP 구성

STT(Speech to Text)

입력된 음성을 텍스트로 변환하는 기술로 인공청각이라고도 한다. 사람의 음성 대화 인터페이스를 통해 텍스트 데이터로 추출해 내는 것이다. 1963년 IBM이 음성으로 인식한 영단어를 텍스트로 전환해준 기술을 시작으로 현재까지 여러 플랫폼 및 서비스에 적용되고 있다. 본 프로젝트에서는 사용자의 발화내용을 VR기기로 입력받아 텍스트로 변환할 때 사용한다.

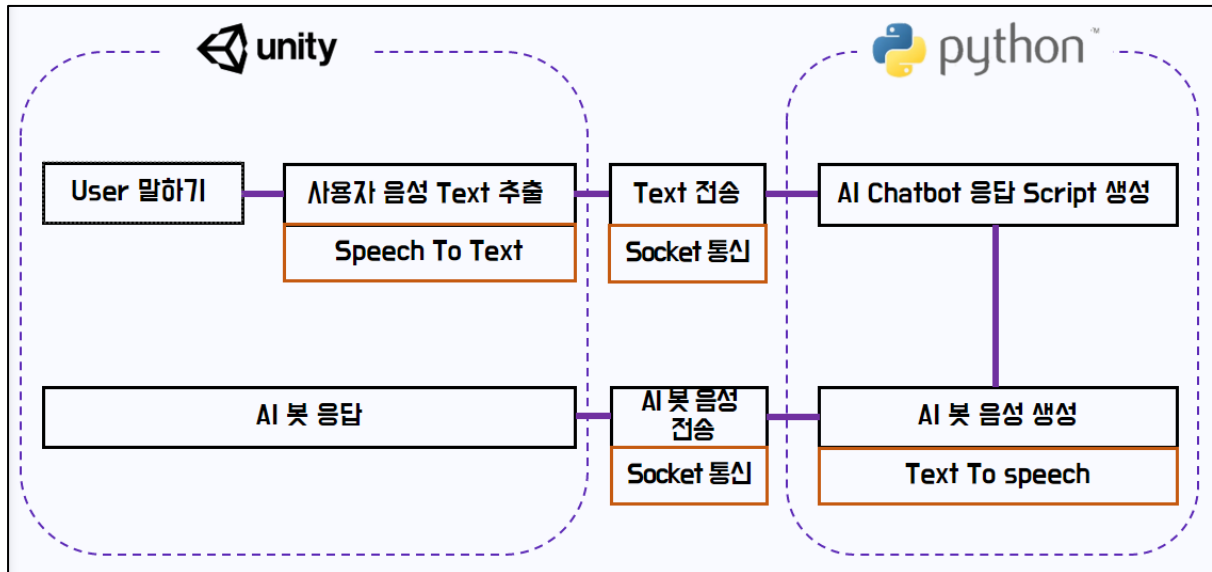
NLP(Natural Language Processing)

NLP는 자연어 처리 기술이다. 자연어 처리란, 인간의 언어를 컴퓨터와 같은 기계에 가르쳐, 묘사할 수 있도록 연구하고 이를 구현하는 인공지능의 중요 분야 중 하나이다. NLP는 NLU(Natural Language Understanding)과 NLG(Natural Language Generation)로 구성되어 있는데, 자연어를 이해하고 만들어냄으로써 기계와 인간이 상호작용할 수 있도록 도와준다. 여기서 NLU란, 컴퓨터가 인간의 발화 의도를 이해하는 기술이고, NLG는 지식 기반이나 논리 형식과 같은 기계 표현에서 의미 표현으로 자연어 문장 생성 기술이다.

TTS(Text to Speech)

텍스트 파일을 음성으로 변환하는 음성 합성 기술이다. 분석 대상의 음성을 녹음하고 일정 단위로 분할하여 합성기에 입력한다. 이를 지시에 따라 필요한 단위만 다시 결합하여 인위적인 음성을 만들어낸다. 챗봇을 통해 생성된 답변을 음성으로 출력할 때 사용한다.

2-3 Pipeline 개요



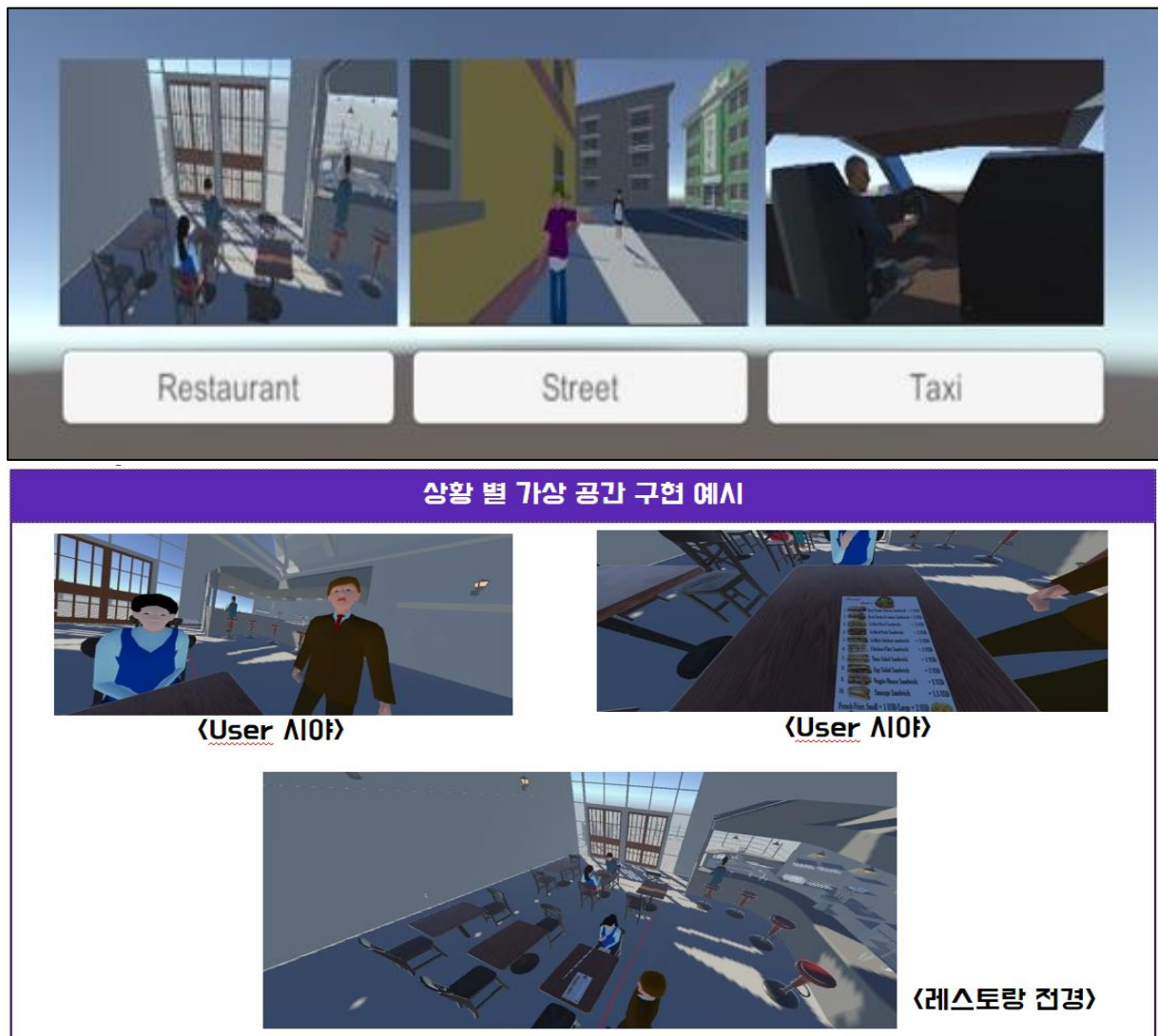
[그림6] Pipeline 개요

먼저 가상세계 안에서 사용자가 해당 상황에 대한 대화를 시작하면 이 음성을 STT(Speech To Text) 모델을 통해서 Text 파일로 추출한다. 추출한 Text는 socket 통신을 통해서 챗봇 시스템으로 전송된다. 이후 챗봇 시스템에서는 해당 대화를 가장 자연스럽게 이어주는 답변을 찾아 제공한다. 이렇게 제공된 답변은 TTS(Text To Speech)기술을 사용하여 Wave 파일로 변환 후 생성한 음성을 다시 socket통신을 통해서 Unity로 전송하여 사용자에게 출력해줌으로써 사용자는 자연스러운 대화가 가능하도록 한다.

3. 상세 기술 구현

3-1. Unity

3-1-1. 개발 환경



[그림7] Unity를 통해 구현한 가상환경

Unity를 이용하여 가상세계 환경을 구현했다. Unity는 C#을 기반으로 개발되었으며, Unity를 통해 구현한 가상세계를 VR기기인 Oculus Quest2에 연동하기 위해 가상세계를 VR, AR기기등과 상호 작용할 수 있도록 데이터를 변환시켜주는 라이브러리인 XR Interaction Toolkit을 사용하려 했으나

'Legacy XR is enabled' error로 인해 사용하지 못했다. 이를 대체하기 위해 Asset Store에서 Oculus Integration을 다운로드 받아 대신 사용하였다. 각각의 가상환경을 자세히 살펴보면 Asset store에서 다운로드 받은 object를 이용하여 각 공간을 생성한 후에, 캐릭터 별 각 관절을 조절하여 해당 상황에 맞는 사람들을 구현하였다. 더불어 애니메이션 효과를 삽입하여 역동적인 상황을 연출했다. 구현에 큰 도움을 준 애니메이션 Asset으로는 Idle MoCap이 있다.

3-1-2. VR 개발 흐름도

기존의 가상세계 제작 및 VR 기기 연계를 위해서는 Oculus Link라는 SDK를 활용해 Unity에서 가상환경을 개발해야 한다. 그래야만 자신이 구성한 가상세계를 VR을 착용하지 않고도 PC로 즉각적으로 확인이 가능해진다. 또한 윈도우가 설치되어 있는 PC에서 다음과 같은 일정 수준이상의 외장 그래픽 카드가 있어야만 VR기기와 연결하여 활용할 수 있다는 단점이 있다.

구성 요소 권장 사양	Oculus Link가 지원되는 GPU
프로세서 Intel i5-4590/AMD Ryzen 5 1500X 이상	NVIDIA Titan Z
그래픽 카드 아래 GPU 표 참조	NVIDIA Titan X
메모리 RAM 8GB 이상	NVIDIA GeForce GTX 970
운영 체제 Windows 10	NVIDIA GeForce GTX 1060 데스크톱, 3GB
USB 포트 USB 포트 1개	NVIDIA GeForce GTX 1060 데스크톱, 6GB
AMD GPU	NVIDIA GeForce GTX 1060M
AMD 200 Series	NVIDIA GeForce GTX 1070(전체)
AMD 300 Series	NVIDIA GeForce GTX 1080(전체)
AMD 400 Series	NVIDIA GeForce GTX 1650
AMD 500 Series	NVIDIA GeForce GTX 1650 Super
AMD 5000 Series	NVIDIA GeForce GTX 1660
AMD Vega Series	NVIDIA GeForce GTX 1660 Ti
	NVIDIA GeForce RTX 20-series(전체)

[그림8] Oculus Link 지원 권장 사양

본 프로젝트를 진행하는데 사용한 장치가 권장사양을 충족하지 못하여 에어링크 연결에 어려움이 있었기 때문에 유선 연결 방식을 선택했다. Oculus 장치와 PC를 연결하기 위해서는 USB 3.0 케이블이 필요하지만 소유하고 있지 않아 USB 2.0 케이블로 호환이 되는 것을 확인하고 이를 사용하기로 결정했다. 케이블 연결 후, APK파일을 VR기기에 전송하여 기기 내부에서 접속하는 방식을 채택하였다.

3-1-3. Unity를 활용한 VR 가상세계 개발의 어려움

Unity를 활용한 개발에 대한 정보 탐색에 큰 어려움을 겪었다. 대부분 내용이 게임 개발을 위한 강좌였기 때문에 본 프로젝트에 필요한 가상세계 구현에 도움이 되는 정보를 찾는 것에 어려움이 있었고, VR개발에 초점을 맞춘 강의가 부족하여 Youtube뿐만 아니라 외국 개발자 사이트에서 세부적인 내용은 별도로 찾아야 하는 어려움 역시 있었다. 또한 강의마다 VR기기 종류, Unity 버전이 모두 달랐기 때문에 본 프로젝트를 진행하는 데에 사용한 기기 종류와 버전에 맞는 강의를 찾는 것에 아주 큰 어려움을 겪었다.

Unity에서 생성한 가상공간 내에 사람을 구성하기 위해 Asset store에서 다운로드 받은 사용하였는데, 주로 게임에 적용할 수 있는 칼싸움, 제자리 뛰기, 총쏘기, 스프린트 등 일상생활에 적용하기 어려운 모션들만 존재했기 때문에 직접 관절을 조절하여 사용해야만 했다.



[그림9] Oculus Quest2

본 프로젝트에서 버튼 누르기처럼 기기와 상호작용할 수 있는 동작을 구현할 필요가 있었는데, 전기수와 같이 XR interaction toolkit을 사용하려 했지만 'Legacy XR is enabled' 오류가 발생했다. 이 오류를 해결하기 위해서는 재설치가 필요하다는 것을 알게 되었다. 프로젝트 진행을 위해 사용한 PC의 사양으로 인해 시간이 많이 소요되어 deadline까지 해결할 수 없다고 판단했기 때문에 Oculus integration을 asset store에서 다운로드 받아 사용하기로 결정하였다.

3-1-4. Unity를 활용한 VR 가상세계 개발 TIP 및 사용 Toolkit

- Unity-VR기기 Build Setting 안내 링크

https://www.youtube.com/watch?v=Hnoad3DM_pA&t=29s

- Oculus integration을 이용한 VR 상호작용 방법 소개 링크

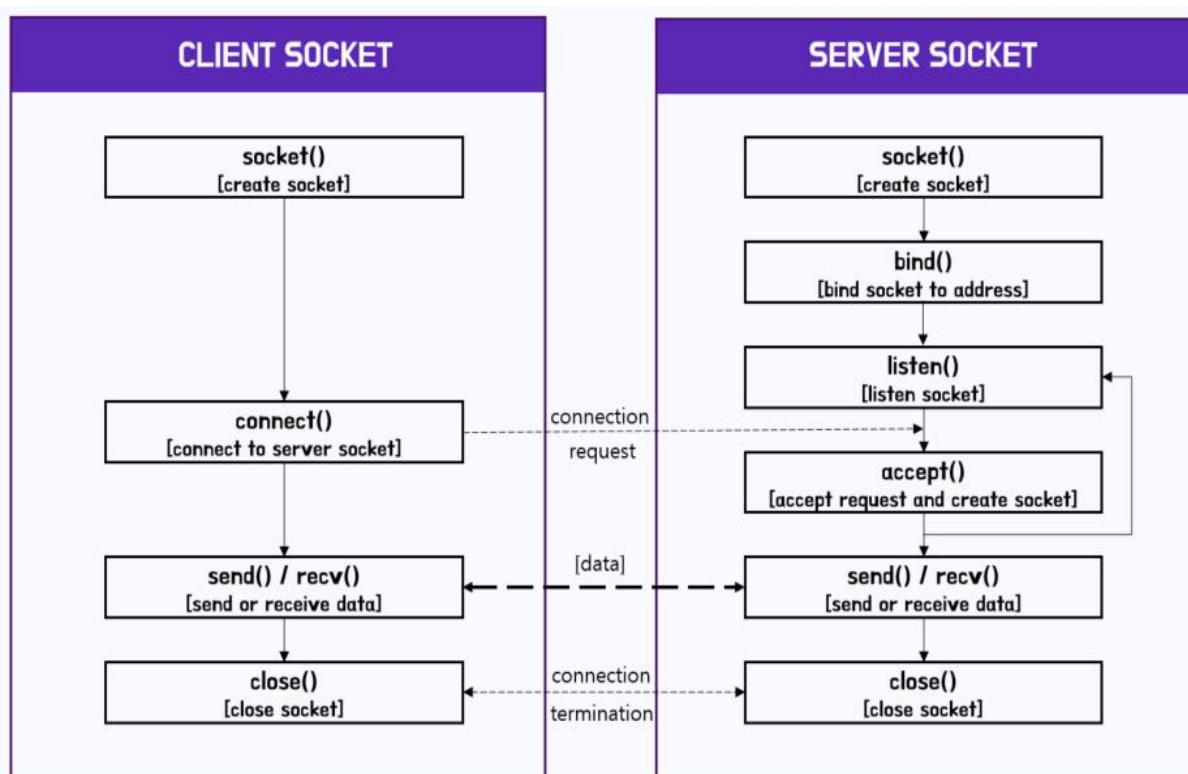
<https://www.youtube.com/watch?v=8fT478uopco>

<https://art-coding3.tistory.com/32>

3-2. Socket 통신

3-2-1. 소켓통신 개념 이해

소켓통신의 전체 흐름도는 [그림10]과 같다. 본 프로젝트는 하드웨어를 사용하지 않는 소프트웨어 프로젝트이기 때문에 특별한 통신이 필요하지 않을 것으로 판단하여 소켓통신을 사용하였다.



[그림10] 소켓통신 흐름도

Server 소켓은 Client 소켓의 신호를 대기(listen)하고 있다가 동일한 IP와 포트번호를 가진 Client 소켓이 신호를 보내면 Connection이 형성된다. 이때, 주의할 점은 Server 소켓 자체가 Client 소켓과 통신하는 것이 아니라, Connection이 형성됐을 때 Client 소켓과 통신을 주고받을 소켓이 Server에서 새롭게 생성된다. 즉, Server 소켓은 오직 Connection 형성을 위해 존재한다고 이해할 수 있다.


본 프로젝트에서 구현한 흐름도에서는 3개 영역간(Unity ↔ STT ↔ Chatbot ↔ TTS ↔ Unity) 소켓 통신이 필요했다. 이때, Python 정보는 Server소켓을 Receiver로, Unity 정보는 Client소켓을 Sender로 설정했다. 왜냐하면 Receiver가 Sender의 신호를 대기하고 있어야 원활한 통신이 이루어질 것이기 때문이다.

소켓통신을 구현하기 위해서는 C#으로 내장코드를 구성하여야 하기 때문에 C# 문법을 새롭게 배워야 한다는 점에서 큰 어려움을 겪었고 최종적으로 Chatbot과 Unity를 연결시키는데 까지는 나아가지 못하였다.

3-3. Speech To Text(STT)

Google Speech To Text

Speech To Text



- ✓ 오디오 파일의 텍스트를 변환
- ✓ 음성 인식 모델에 특정 오디오 유형 및 소스를 학습
- ✓ 음성파일 크기 및 커스텀 단어, 감탄사를 제외해주는 기능들이 있고 성능이 우수

STT Test

```
(env) (base) C:\Users\USER\speech>python test1.py
안녕하세요 아카데미 15기 stt 테스트 중입니다
```

- ✓ STT 전송 test 및 성능 확인

[그림11] STT 설명 및 Test

Google Cloud Platform에서 제공하는 Speech To Text API를 사용하였다. Google Speech To Text API는 125개 이상의 언어 및 방언으로 된 음성을 텍스트로 정확하게 변환해준다. 한국어는 자연어 중에서도 어려운 난이도에 속하지만 형태소 분리 기능도 제공하는 등 높은 정확도를 가지고 있다. 이 기술을 통해 음성 파일을 텍스트로 바꾼 결과를 출력한다.

3-3-1. 사용방법

참고 링크 : <https://youtu.be/Ds-7D8d-FwA>

Google Cloud Platform에 가입한 뒤 해당 API를 사용한다. 본 프로젝트에 stt를 적용하기 위해 참고링크의 영상을 참고했다.

1. Google Cloud Platform에 가입한다.
2. 새 프로젝트를 생성한다.
3. 프로젝트 내에서 Cloud Speech to text API를 검색한 후, 사용 설정을 불러온다.
4. 사용자 인증 정보 만들기 서비스 계정을 클릭한다.
5. 계정 이름을 입력 후, 계정 권한을 소유자로 변경한다.
6. key 만들기를 클릭한 후, Json 형태로 만들어 컴퓨터에 저장한다.

3-3-2. 사용시 주의점

1. Google Cloud Platform 최초 가입 시 카드 정보를 입력하라는 명령이 있지만 무료로 사용하기를 선택했다면 유료 전환이 되지 않는다. 하지만 용량을 너무 많이 사용할 경우 유료 전환이 될 수 있다는 점은 주의해야 한다.
2. STT API는 1분 미만의 동기 인식, 1분 이상의 비동기 인식 기능을 제공하는데 본 프로젝트에서는 1분 미만의 음성이 들어올 것을 가정하여 동기 인식만 사용했다. 1분 이상의 비동기 기능을 사용하려면 Google Bucket(스토리지 서비스)를 거쳐야 한다.
3. 한국어는 상대적으로 잘 인식하는 반면에 영어의 경우 발음 문제로 인해 인식을 잘하지 못하는 문제가 있다.
4. Google API를 통해 STT 구현 및 프로젝트 진행 시간을 단축할 수 있다.

3-3-3. 구성 코드

```

class MicrophoneStream(object):
    """Opens a recording stream as a generator yielding the audio chunks."""
    def __init__(self, rate, chunk):
        self._rate = rate
        self._chunk = chunk

        # Create a thread-safe buffer of audio data
        self._buff = queue.Queue()
        self.closed = True

    def __enter__(self):
        self._audio_interface = pyaudio.PyAudio()
        self._audio_stream = self._audio_interface.open(
            format=pyaudio.paInt16,
            # The API currently only supports 1-channel (mono) audio
            # https://goo.gl/z757pE
            channels=1, rate=self._rate,
            input=True, frames_per_buffer=self._chunk,
            # Run the audio stream asynchronously to fill the buffer object.
            # This is necessary so that the input device's buffer doesn't
            # overflow while the calling thread makes network requests, etc.
            stream_callback=self._fill_buffer,
        )

        self.closed = False

        return self

    def __exit__(self, type, value, traceback):
        self._audio_stream.stop_stream()
        self._audio_stream.close()
        self.closed = True
        # Signal the generator to terminate so that the client's
        # streaming_recognize method will not block the process termination.
        self._buff.put(None)
        self._audio_interface.terminate()

    def _fill_buffer(self, in_data, frame_count, time_info, status_flags):
        """Continuously collect data from the audio stream, into the buffer."""
        self._buff.put(in_data)
        return None, pyaudio.paContinue

```

```

def main():
    # See http://g.co/cloud/speech/docs/languages
    # for a list of supported languages.
    language_code = 'ko-KR' # a BCP-47 language tag

    client = speech.SpeechClient()
    config = speech.RecognitionConfig(
        encoding=speech.RecognitionConfig.AudioEncoding.LINEAR16,
        sample_rate_hertz=RATE,
        language_code=language_code)
    streaming_config = speech.StreamingRecognitionConfig(
        config=config,
        interim_results=True)

    with MicrophoneStream(RATE, CHUNK) as stream:
        audio_generator = stream.generator()
        requests = (speech.StreamingRecognizeRequest(audio_content=content)
                    for content in audio_generator)

        responses = client.streaming_recognize(streaming_config, requests)

        # Now, put the transcription responses to use.
        listen_print_loop(responses)

```

[그림12] STT 코드 구성

3-4. NLP(Natural Language Processing)

3-4-1. NLP

자연어를 이해(NLU)하고, 생성(NLG)함으로써 기계와 인간 사이에서 자연어로 interaction을 할 수 있도록 처리해주는 기술이다

(1) NLU(Natural Language Understanding)

단순히 단어나 문장의 형태만 기계가 인식하도록 하는 것이 아니라, '의미를 인식하도록 하는 것'을 의미한다. NLU 기능의 예로는 문장의 의도 분류, 서로 다른 언어 간 번역 문장 생성, 자연어 질문에 대한 답변 추출 등이 있다.

(2) NLG(Natural Language Generation)

자연어 생성은 자연어 처리의 또 다른 하위 집합입니다. 자연어 이해는 컴퓨터 읽기 이해에 초점을 맞추는 반면, 자연어 생성은 컴퓨터가 쓸 수 있도록 합니다. NLG는 일부 데이터 입력을 기반으로 인간 언어 텍스트 응답을 생성하는 프로세스입니다. 이 텍스트는 TTS(텍스트 음성 변환) 서비스를 통해 음성 형식으로 변환할 수도 있습니다.

※참고 : intent Classification(의도 분류)

Intent Classification(의도 분류)는 사용자와 상호 소통하는 온라인 챗봇에 주로 사용되는 분류 방법이다. Dialogflow에서 데이터를 학습하여 채팅 의도 분류를 하거나 유연하게 적용할 수 있다. 챗봇에 원하는 문장을 입력했을 때, 가장 적절한 답변을 제공하고, 이러한 패턴을 자동 학습하여 지속적인 챗봇 성능 개선을 통해 처음 보는 문장이 들어오더라도 가장 가능성이 높은 답변을 제공할 수 있게 된다.

3-5. Model 생성

3-5-1. Train data set 구성


```
{
  "label": "greeting",
  "questions": ["Hi", "Hello", "How are you?", "Is anyone there?", "What's up?", "Good morning.", "Good evening.", "Good afternoon.", "Morning!", "How do you do?", "Hey!", "Yo!", "Sup?", "How have you been?", "How are you doing?", "How's it going?", "It's been a while.", "Long-time no see.", "Nice to see you.", "It's great to see you.", "Good to see you.", "Good day!", "What's good?"],
  "response": "Hey! How can I help you today?"
},
{
  "label": "information",
  "questions": ["How can you help me?", "What can you do?", "What do you do?", "What is your name?", "Name?", "Who are you?", "What should I call you?"],
  "response": "My name is Foodie. I can assist you in placing an order."
},
{
  "label": "order",
  "questions": ["Can I place an order?", "I want to order.", "I would like to buy something."],
  "response": "Sure! What would you like to have?"
},
{
  "label": "menu",
  "questions": ["Menu?", "What's on the menu?", "Show me the menu.", "What can I order?", "Suggest me something.", "What should I eat?", "What would be your recommendation?"],
  "response": "You can order anything from the menu: Chicken Sandwich, Spicy Chicken Sandwich, Fries, Cheesy Fries, Chicken Nuggets, Strawberry milkshake, Chocolate milkshake, Chocolate Chip Cookie and Coke."
},
{
  "label": "add",
  "questions": ["Can I get a Chicken Sandwich?", "Could I get a Chicken Sandwich?", "Can I have a Chicken Sandwich?", "I would like to have a Chicken Sandwich.", "Can you add a Chicken Sandwich to the order?", "Give me a Chicken Sandwich.", "Can I get a Spicy Chicken Sandwich?", "Could I get a Spicy Chicken Sandwich?", "Can I have a Spicy Chicken Sandwich?", "I would like to have a Spicy Chicken Sandwich.", "Can you add a Spicy Chicken Sandwich to the order?", "Give me a Spicy Chicken Sandwich.", "Can I get a Fries?", "Could I get a Fries?", "Can I have a Large Fries?", "I would like to have a small Fries?", "Can you add Fries to the order?", "Give me a Fries?", "Can I get a Cheesy Fries?", "Could I get a Cheesy Fries?", "Can I have a Cheesy Fries?", "I would like to have a Cheesy Fries.", "Can you add Cheesy Fries?", "Give me a Cheesy Fries?", "Can I get Chicken Nuggets?", "Could I get Chicken Nuggets?", "Can I have Chicken Nuggets.", "I would like to have Chicken Nuggets?", "Can you add Chicken Nuggets?", "Give me Chicken Nuggets?", "Can I get a Strawberry milkshake?", "Could I get a Strawberry milkshake?", "Can I have a Strawberry milkshake?", "I would like to have a Strawberry milkshake.", "Can you add Strawberry milkshake to the order?", "Give me a Strawberry milkshake?", "Can I get a Chocolate milkshake?", "Could I get a Chocolate milkshake?", "Can I have a Chocolate milkshake?", "I would like to have a Chocolate milkshake.", "Can you add Chocolate milkshake?", "Give me a Chocolate milkshake?", "Can I get a Chocolate Chip Cookie?", "Could I get a Chocolate Chip Cookie?", "Can I have a Chocolate Chip Cookie?", "I would like to have a Chocolate Chip Cookie?", "Can you add Chocolate Chip Cookie?", "Give me a Chocolate Chip Cookie?", "Can I get a Coke?", "Could I get a Coke?", "Can I have a Coke?", "I would like to have a Coke.", "Can you add Coke to the order?", "Give me a Coke?"],
  "response": "Added to your order"
},
{
  "label": "delete",
  "questions": ["Can you remove a Chicken Sandwich?", "Can you delete a Chicken Sandwich?", "Do not give me Chicken Sandwich.", "Can you remove a Spicy Chicken Sandwich?", "Can you delete a Spicy Chicken Sandwich?", "Do not give me Spicy Chicken Sandwich.", "Can you remove a Fries?", "Can you delete a Fries?", "Do not give me Fries.", "Can you remove a Cheesy Fries?", "Can you delete Cheesy Fries?", "Do not give me Cheesy Fries.", "Can you remove Chicken Nuggets?", "Can you delete Chicken Nuggets?", "Do not give me Chicken Nuggets.", "Can you remove a Strawberry milkshake?", "Can you delete a Strawberry milkshake?", "Do not give me Strawberry milkshake.", "Can you remove a Chocolate milkshake?", "Can you delete a Chocolate milkshake?", "Do not give me Chocolate milkshake.", "Can you remove a Chocolate Chip Cookie?", "Can you delete a Chocolate Chip Cookie?", "Do not give me Chocolate Chip Cookie.", "Can you remove a Coke?", "Can you delete a Coke.", "Do not give me Coke."],
  "response": "Removed from your order"
}
```

[그림13] Data set 구성

초기 프로젝트를 진행하면서 챗봇 만드는 과정에 비지도 학습으로 bert를 이용하려 했으나 어려운 부분들이 있었다. 비즈니스 상황에서는 상호 대화에서 어느정도 룰이 있어야 하기 때문에 비지도 학습으로 챗봇을 구성하는 데에 큰 어려움이 있다고 판단했다. 따라서 Question에 해당하는 response를 설정하여 지도학습이 가능하도록 Json 형식의 Data set을 구축하였다. 발화 목적에 따라서 label을 설정하여 영어 회화 Script를 구성하였다.

3-5-2. Train data set 학습

```
def preprocess(text):
    tokens = []
    words = nltk.word_tokenize(text)

    for word in words:
        tokens.append(lemmatizer.lemmatize(word.lower()))

    return tokens

tokens = set()
labels = set()

document_x = []
document_y = []

for intent in data['intents']:
    for question in intent['questions']:
        words = preprocess(question)
        tokens.update(words)
        document_x.append(words)
        document_y.append(intent["label"])
        labels.add(intent['label'])

X_train = []
y_train = []

tokens = list(tokens)
labels = list(labels)

class_zeros = [0 for _ in range(len(labels))]

for i, document in enumerate(document_x):
    bag = []
    class_row = class_zeros[:]

    for token in tokens:
        if token in document:
            bag.append(1)
        else:
            bag.append(0)

    class_row[labels.index(document_y[i])] = 1
    X_train.append(bag)
    y_train.append(class_row)
```

[그림14] Data set 학습

입력 받은 문장을 Tokenization하여 리스트에 저장 후, 중복을 제거한 각각의 단어에 index를 부여한다. 부여한 index에 따라 Embedding Vector가 생성된다. 새로운 문장이 들어왔을 때, Vector들의 조합을 통해 해당 문장이 어떤 label에 포함되어있는지 구분할 수 있게 된다. 그러면 label에 적절한 response를 출력할 수 있게 된다.

```

X_train = numpy.array(X_train)
y_train = numpy.array(y_train)

ops.reset_default_graph()

net = tflearn.layers.core.input_data(shape=[None, len(X_train[0])])
net = tflearn.layers.core.fully_connected(net, 8)
net = tflearn.layers.core.fully_connected(net, 8)
net = tflearn.layers.core.fully_connected(net, len(y_train[0]), activation="softmax")
net = tflearn.layers.estimator.regression(net)

model = tflearn.DNN(net)
model.fit(X_train, y_train, n_epoch=1000, batch_size=8, show_metric=True)

model_information = {
    "tokens": tokens,
    "labels": labels,
    "X_train": X_train,
    "y_train": y_train
}

save(model_information, "model_information")
model.save("model.tflearn")

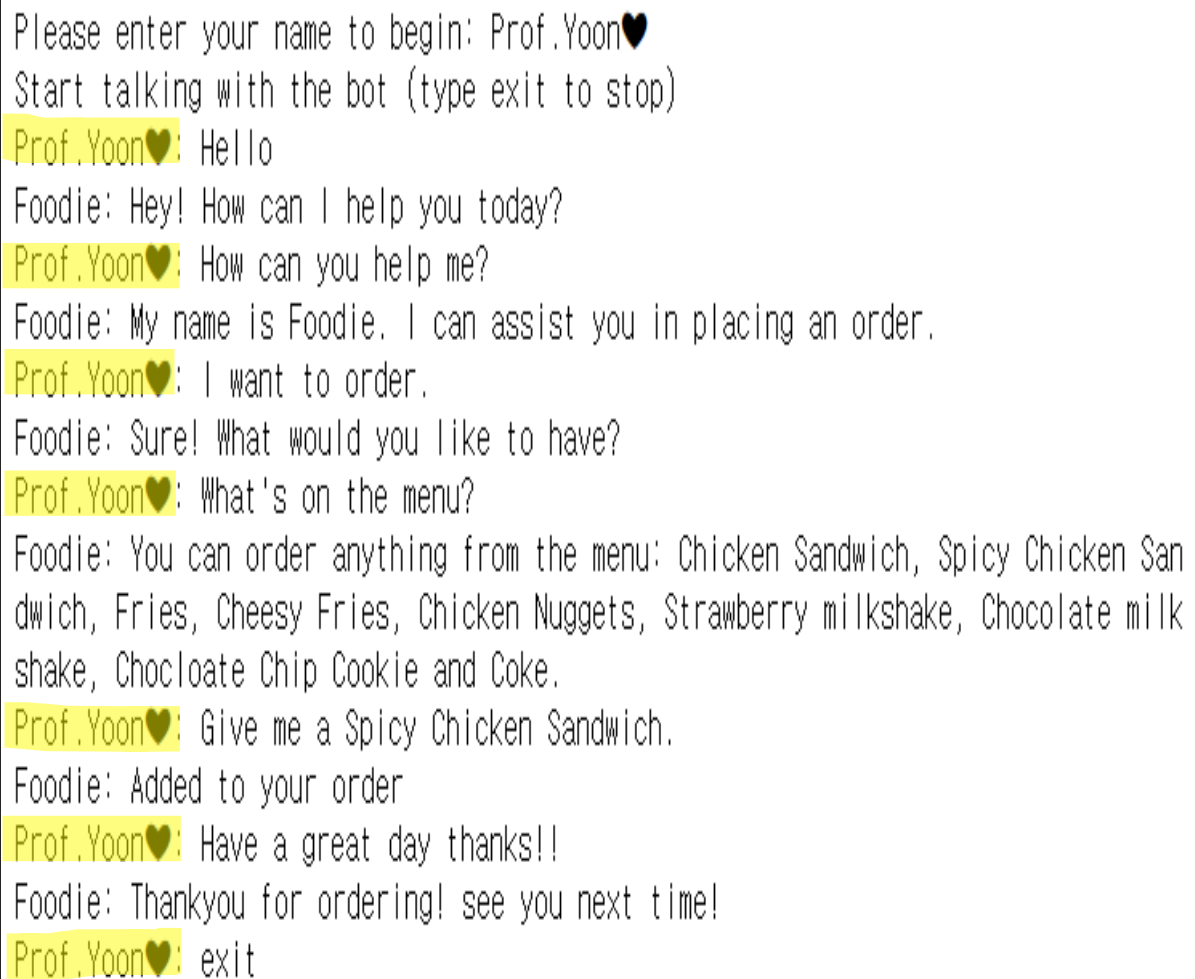
```

[그림15] Model 생성

Train Data를 DNN(Deep Neural Network)을 통해 학습시켜 model을 생성했다. 여기서 DNN이란, 입력층(input layer)과 출력층(output layer) 사이에 여러 개의 은닉층(hidden layer)들로 이루어진 인공신경망으로 분류 및 수치 예측을 하는데 유용하게 쓰인다. 특히 이미지 트레이닝이나 문자인식과 같은 분야에서 상용화되어 있다. 대표적인 DNN 응용 알고리즘에는 CNN, RNN, LSTM, GRU 등이 있다.

본 프로젝트에서 사용한 model의 정확도는 80.7692%이다.

3-5-3. Chatbot 실행

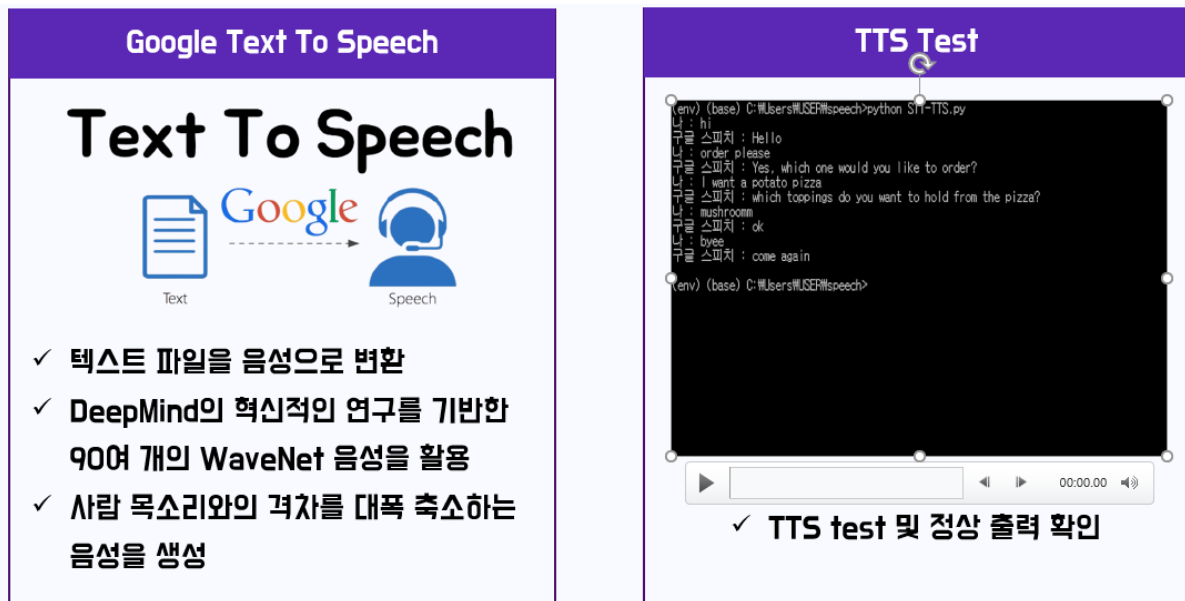


```
Please enter your name to begin: Prof.Yoon♥
Start talking with the bot (type exit to stop)
Prof.Yoon♥: Hello
Foodie: Hey! How can I help you today?
Prof.Yoon♥: How can you help me?
Foodie: My name is Foodie. I can assist you in placing an order.
Prof.Yoon♥: I want to order.
Foodie: Sure! What would you like to have?
Prof.Yoon♥: What's on the menu?
Foodie: You can order anything from the menu: Chicken Sandwich, Spicy Chicken Sandwich, Fries, Cheesy Fries, Chicken Nuggets, Strawberry milkshake, Chocolate milkshake, Chocloate Chip Cookie and Coke.
Prof.Yoon♥: Give me a Spicy Chicken Sandwich.
Foodie: Added to your order
Prof.Yoon♥: Have a great day thanks!!
Foodie: Thankyou for ordering! see you next time!
Prof.Yoon♥: exit
```

[그림16] Chatbot 실행 및 성능 확인

Train Data set을 구성하고, DNN을 통해 학습시켜 생성한 model을 이용하여 Chatbot을 실행하였다. [그림16]을 보면 사용자 입력에 따라 적절한 답변을 제공해주는 것을 확인할 수 있다. 적절한 답변을 TTS를 통해 음성으로 출력하는 것이 다음 과정이다.

3-6. Text to Speech(TTS)



[그림17] TTS 설명 및 Test

3-6-1. TTS 개요

본 프로젝트에서 Chatbot이 제공한 답변(Text file)을 음성으로 변환하기 위해 TTS(음성합성시스템)을 사용한다.

TTS란, 컴퓨터의 프로그램을 통해 사람의 목소리를 구현해내는 것이다. google Text to Speech API는 북경어, 힌디어, 스페인어, 아랍어, 러시아어 등 40개 이상의 언어 및 방언을 지원하는 220여 개의 음성 조합 중에서 선택할 수 있고, DeepMind의 혁신적인 연구를 기반으로 빌드된 90여 개의 WaveNet 음성을 활용하여 사람 목소리와의 격차를 대폭 축소한 음성을 생성할 수 있다는 장점이 있다.

3-6-2 코드 구성

본 프로젝트에서는 최종적으로 STT - 챗봇 - TTS를 연동하여 실제 회화하는 것과 비슷한 상황을 구현하였다.

```
class MicrophoneStream(object):
    """Opens a recording stream as a generator yielding the audio chunks."""
    def __init__(self, rate, chunk):
        self._rate = rate
        self._chunk = chunk

        # Create a thread-safe buffer of audio data
        self._buff = queue.Queue()
        self.closed = True

    def __enter__(self):
        self._audio_interface = pyaudio.PyAudio()
        self._audio_stream = self._audio_interface.open(
            format=pyaudio.paInt16,
            # The API currently only supports 1-channel (mono) audio
            # https://goo.gl/z757pE
            channels=1, rate=self._rate,
            input=True, frames_per_buffer=self._chunk,
            # Run the audio stream asynchronously to fill the buffer object.
            # This is necessary so that the input device's buffer doesn't
            # overflow while the calling thread makes network requests, etc.
            stream_callback=self._fill_buffer,
        )

        self.closed = False

        return self

    def __exit__(self, type, value, traceback):
        self._audio_stream.stop_stream()
        self._audio_stream.close()
        self.closed = True
        # Signal the generator to terminate so that the client's
        # streaming_recognize method will not block the process termination.
        self._buff.put(None)
        self._audio_interface.terminate()

    def _fill_buffer(self, in_data, frame_count, time_info, status_flags):
        """Continuously collect data from the audio stream, into the buffer."""
        self._buff.put(in_data)
        return None, pyaudio.paContinue

    def generator(self):
        while not self.closed:
```

```

        chunk = self._buff.get()
        if chunk is None:
            return
        data = [chunk]

        # Now consume whatever other data's still buffered.
        while True:
            try:
                chunk = self._buff.get(block=False)
                if chunk is None:
                    return
                data.append(chunk)
            except queue.Empty:
                break

        yield b''.join(data)
# [END audio_stream]

cmdlists = [
    #명령어      대답      종료 리턴값
    [u'goodbye',      'Thankyou for ordering! see you next time!',      0],
    [u'hi',           'Hey! How can I help you today?',      1],
    [u"What's on the menu?",      'You can order anything from the menu: Chi',
    [u'Can I get a Fries?', 'Added to your order',      1],
    [u'I want to order.',      'Sure! What would you like to have?', 1],
    [u'add cheese string',      'yes we will make it delicious',      1]]

```

```

def main():
    # See http://g.co/cloud/speech/docs/languages
    # for a list of supported languages.
    #language_code = 'en-US' # a BCP-47 language tag
    language_code = 'en-US' # 영어로 변경

    client = speech.SpeechClient()
    config = speech.RecognitionConfig(
        encoding=speech.RecognitionConfig.AudioEncoding.LINEAR16,
        sample_rate_hertz=RATE,
        language_code=language_code)
    streaming_config = speech.StreamingRecognitionConfig(
        config=config,
        interim_results=True)

    with MicrophoneStream(RATE, CHUNK) as stream:
        audio_generator = stream.generator()
        requests = (speech.StreamingRecognizeRequest(audio_content=content)
                    for content in audio_generator)

        responses = client.streaming_recognize(streaming_config, requests)

        # Now, put the transcription responses to use.
        listen_print_loop(responses)

```

[그림17] STT - TTS 코드 구성

4. 결론

4-1 구현 결과

시연 영상 ; 링크

4-2 기대효과 및 발전방향

본 AI 프로젝트는 대한민국의 문법/읽기 위주의 영어교육의 단점을 보완하고, Digital Learning 플랫폼으로써 실제 상황과 유사한 가상세계를 구현하여 학생들의 영어회화 자신감 함양 및 경험 제공에 기반이 되는 것에 의의를 두고 있다. 가상세계에서 영어 회화를 진행하기 위해서 AI 챗봇을 통해 답변을 주고받을 수 있도록 하였다. 가상세계 영어 회화 교육을 통해 교육자들의 말하기 자신감 상승과 더불어 영어 교육에 대한 흥미를 유지할 수 있을 것으로 예상된다. 특히, OPIc/TOEIC Speaking과 같은 회화 학습에도 큰 도움이 될 것으로 예상된다. 또한 교육 플랫폼 활용으로 높은 사교육 비용으로 인해 영어 회화 교육에 접근이 어려운 교육자들의 학습율을 높여주어 교육 격차를 해소할 수 있을 것이다.

코로나19로 삶에 여러 방면에 변화가 생기면서 다양한 산업계, 특히 교육계에 큰 변화가 생겼다. 초·중·고등학생은 온라인 개학을 시작했으며, 대학과 사업체 역시 원격수업, 재택근무를 전면 도입하여 상반기를 채우고 있다. 외부 활동이 자유롭지 못하는 상황 속에 '언택트 시대'가 시작되면서 일부 전문적인 분야에만 적용되어 오던 인공지능(AI) 기술이 교육 서비스 분야까지 그 영역을 확장하고 있다. 언택트 시대에서 교육의 질이 하락하고 있는 상황에서 VR환경을 이용한 교육을 진행한다면 학생들의 참여도 상승 효과 예상 및 집중도 하락 문제를 해결할 수 있을 것으로 생각된다.

메타버스 영어회화는 최근 대중화된 VR(가상환경) 플랫폼에 적용함으로써 AI 챗봇과의 대화뿐만 아니라 여러 사람들과 다양한 환경에서 대화할 수 있는 기회를 제공할 수 있다. 본 프로젝트에서 구현한 식당, 택시, 길거리뿐만 아니라 공항, 커피숍, 호텔, 편의점, 은행 등 여러 상황에 적용하여 회화를 경험할 수 있을 것이다. 나아가 영어뿐만 아니라 Google API에서 제공해주는 각국의 언어를 적용하여 일본어, 중국어, 스페인어, 힌디어, 아랍어 등 다양한 언어를 활용한 회화를 기대해볼 수 있다.

4-3 한계점 및 개선방안

첫째로, 실시간 소켓통신을 구현하려 했으나 실패하였다. Unity는 C#을 이용하고 Chatbot은 Python을 이용하는데 둘을 결합하기 위해 소켓통신이 이용되지만 C#을 새로 배우고 적용시키기에는 다소 무리가 있었다. 조금 더 시간을 가지고 소켓 통신과 C#에 관해 공부하면 충분히 보완할 수 있을 것 같다.

둘째로, 각 상황별로 10~50개 정도의 문장을 학습시켰는데, 데이터 양이 충분하지 않아 사용자가 다른 문장을 이용하려 해도 대화할 수 있는 주제의 폭이 적었다. 발생가능한 여러 상황에 대한 데이터를 수집해서 학습시킨다면 조금 더 다양하고 자연스러운 대화흐름을 만들 수 있을 것이라 생각된다.

셋째로, STT 과정에 있어 발화자의 발음이 부정확하거나 주변의 소음이 있을 경우 정확한 인식이 불가능한 문제가 발생하였다. 또한, 음성을 인식해서 텍스트로 바로 변환시키지 못하고 시간이 약간 지연되었다. Speech Recognition 학습 과정에 있어 더 많은 양질의 데이터가 있다면 이 문제를 해결할 수 있을 것이라고 생각한다.

넷째로, 문법 교정 및 발음 교정 기능을 구현하지 못했다. 원래의 프로젝트 목적은 메타버스 환경에서 멀티유저간의 대화가 가능하고 AI가 문법이나 발음을 교정해주는 것이었다. 하지만 앞선 문제점들로 인해 문법 및 발음 교정 구현까지 나아가지 못하여 본래의 목적을 이루지 못하였다. 만약 이 문제를 모두 해결한다면 문법 발음 교정을 넘어 사용자의 발화가 지체될 시 문장을 추천해주는 기능까지 추가적으로 구현할 수 있을 것이라 생각한다.

부록

Learned lessons

김민경

서형준	
손유정	
이영호	
이찬희	
임솔이	

서형준

김민경	
손유정	
이영호	
이찬희	
임솔이	

손유정

김민경	
서형준	
이영호	
이찬희	
임솔이	

이영호

김민경	
서형준	
손유정	
이찬희	
임솔이	

이찬희

김민경	
서형준	
손유정	
이영호	
임솔이	

임솔이

김민경	
서형준	
손유정	
이영호	
이찬희	

References

VR/AR 기반 언어교육용 지능형 AI 챗봇 개발 방안

김인석. "VR/AR 기반 언어교육용 지능형 AI 챗봇 개발 방안", 정보통신기획평가원, Weekly ICT Trend 1973호, 2020, pp.12-25.

AI 챗봇을 도입한 초등학교 영어 디지털 교과서 활용 수업의 정의적 효과

김소연(Kim So-Yeon), 김정렬(Kim Jeong-Ryeol). (2021). AI 챗봇을 도입한 초등학교 영어 디지털 교과서 활용 수업의 정의적 효과. 학습자중심교과교육연구, 21(10): 37-49

영어교육에서의 AI 활용 연구에 대한 체계적 문헌 고찰

전혜리, 이상민 and 박일이. (2021). 영어교육에서의 AI 활용 연구에 대한 체계적 문헌 고찰. 멀티미디어 언어교육, 24(1), 87-103.

가상현실 기반의 인공지능 영어회화 시스템

Cheon, E. (2019). 가상현실 기반의 인공지능 영어회화 시스템. 한국융합학회논문지, 10(11), 55-61.

원어민 교사 회화 수업에 나타난 대학생들의 말하기 불안 연구

유영미 and 표경현. (2009). 원어민 교사 회화 수업에 나타난 대학생들의 말하기 불안 연구. 외국어교육연구, 23(2), 59-84.

영어 학습자의 탈동기 요인 연구: S대학의 영어회화 수업 사례를 중심으로

강은영."영어 학습자의 탈동기 요인 연구: S대학의 영어회화 수업 사례를 중심으로."영어영문학연구45.2(2019):209-235.

원어민담당 영어회화 강의에 대한 대학생들의 인식 연구

Yoo, Beom & Gwag, Kyongseob. (2016). A study on Korean university student's perceptions of their English conversation classes taught by native English speakers. Foreign Language Education, 23(1), 243-266.