소음 환경에 강한 내성을 가진 음성인식 모델 개발

분과: D (지능형융합보안)

Team: 포네틱 코드

부산대학교 정보의생명공학대학 정보컴퓨터공학부

College of Information and Biomedical Engineering

Computer Engineering Major

Pusan National University

2021년 5월 14일

지도교수: 백윤주 (인)

목 차

1. 과제의 배경 및 목표

1.1 과제 배경

음성은 사람 간의 가장 자연스러운 의사소통 방식이다. 음성인식 기술은 이미 스마트폰, 자동차, 콜센터 등 현재 우리 생활의 많은 부분에 녹아들어서 서비스화 되고 있다. 최근 몇 년 동안 보편화 되는 클라우드 서버 및 고성능 GPU와 같은 하드웨어의 눈부신 발전에 힘입은 딥러닝 기술에 의해, 혁신적인 성능을 보이는 음성인식 기술이 등장하고 있다.

하지만, 음성인식 기술의 비약적 발전에도 불구하고 여전히 음성인식은 사람들이 웅성거리는 식당, 회의실, 버스나 지하철 등과 같은 소음이 발생하는 환경에서는 매우 낮은 정확도를 보이는 등 개선할 점이 많다. 이에, 우리는 딥러닝 기술 등을 활용하여, 소음이 존재하는 환경에서의 원활한 음성인식을 제공하는 기술 및 서비스를 개발하고자 한다.

아래는 우리가 직접 녹음 및 소음 합성한 파일을 Kakao STT REST API 서비스를 이용하여 인식시켜 본 결과이다. 인식 음성은 적절한 길이를 갖춘 문장으로 택하였다. 구체적인 문장 내용은 '국가는 재해를 예방하고 그 위험으로부터 국민을 보호하기 위하여 노력하여야 한다.' 이다.

비고	출력 내용
원본	국가는 재해를 예방하고 그 위험으로부터 국민을 보호하기 위하여 노력하여야 한다.
대화 도중 양치질하는 소음	국가는 재해를 예방하고 그 위험으로부터 국민을 보호하기 위하여 노력하여야 한다.
샤워장의 물이 떨어지는 소음	북한의 대회로 예방하고 그 위험으로부터 국민을 보험 위하여 노력하여야 한다.
스포츠경기장의 환호 소음	인식 불능
대중들이 야유하는 소음	국가는 재해를 예방하고 위험도 있고 민의 고향이에요 그려야 한다.

[표 1] Kakao STT REST API 소음 환경에서의 테스트 결과
(node.js / request module 이용)
원본과 다른 출력 부분은 빨강으로 표시했다.

이렇듯 원본 음성은 완벽하게 잘 인식하였지만, 소음이 있는 환경에서는 음성인식에 오인식 및 미인식 등의 상당한 오류가 발생한다. 이에 소음 환경에서의 원활한 음성인식을 제공하는 기술 및 서비스를 개발하고자 한다.

1.2 과제 목적

본 졸업 과제는 소음이 학습된 음성인식 모델을 사용해 다양한 소음 환경에서도 정상적으로 작동하는 음성인식 프로그램을 만드는데 목표를 둔다.

- 소음이 있는 다양한 음성 파일을 제작
  - ▶ 음성 파일과 소음 파일을 합성시킨다.
  - ▶ 소음 파일은 실생활에서 얻을 수 있는 소음이거나 인위적으로 만든 소음으로 한다.

- 다양한 소음 환경 중 어떤 소음이 음성인식을 가장 어렵게 하는지 판단
  - ▶ Google STT나 Kakao STT 등 음성 파일을 텍스트로 변환시켜주는 API 및 서비스를 이용해 합성된 음성 파일을 얼마나 잘 인식하는지 확인한다.
  - ▶ 결과를 확인하고 인식에 가장 취약한 소음을 찾는다.

- 취약한 소음 환경에 내성을 가진 모델을 제작
  - ▶ Pytorch 및 TensorFlow를 이용해 다양한 합성 데이터 파일로 학습을 시키고 모델을 제작한다.

- 음성인식 프로그램을 만들어 실시간으로 사용할 수 있게끔 구현
  - ▶ 학습된 모델을 서버를 구축 및 연동하여 실시간 음성인식이 가능하도록 구현한다.
  - ▶ 실생활에서 인식이 잘 되는지 확인한다.

2. 요구조건 분석

2.1 실시간 음성인식

서버를 이용해 음성인식이 실시간으로 작동하게끔 제작한다.
소음이 발생하는 환경에서도 기존 서비스 대비 향상된 음성인식이 이루어져야 한다.

2.2 다양한 소음 데이터 합성

실생활에서 들을 수 있는 소음뿐만 아니라 인위적으로 음성 파일을 편집해 노이즈를 추가할 수 있다. 추가한 소음 파일을 이용해 과제를 진행한다. [1]STT API를 통해 취약한 소음 종류 등을 알아낸다.

2.2.1 자연적 소음

무료로 제공되는 [2]샘플 데이터를 이용한다. STT API를 사용해 녹음한 음성 파일과 소음 샘플 파일을 합성한 결과를 확인한다.

2.2.2 인위적 소음

개발자가 인위적으로 sin 파형과 같은 소음 파형을 만든 뒤, 녹음한 파일과 합성한다. 확인 방법은 2.2.1의 방법과 같은 방법으로 진행한다.

2.3 취약한 소음에서 테스트

STT API를 통해 알아낸 취약한 소음에 대해 얼마나 개선이 되었는지 실험해본다.

2.4 실제 환경에서 테스트

소음 저항이 학습된 모델이 실제 소음이 있는 환경에서 작동이 얼마나 잘 되는지 실험해본다. 소음이 강하다고 생각되는 장소를 지정해 실험을 진행할 예정이다.

3. 현실적 제약 사항 및 대책

- 심한 소음 환경에서 인식의 어려움
  - ▶ 사람이 이해할 수 있는 정도의 소음으로 학습 진행할 예정
  - ▶ 작은 소음이지만 STT API가 올바르게 인식하지 못하는 소음에 대해 학습을 진행할 예정

- 사용되는 언어 제한
  - ▶ 언어는 한국어로 진행할 예정
  - ▶ 향후 영어도 확장할 계획

- 긴 문장에 대해 인식의 어려움
  - ▶ 짧은 명령어 위주로 인식할 예정
  - ▶ 문장을 뭉텅이(chunk)로 분리하여 인식 시도

- 신규 또는 전문 어휘를 실시간으로 반영하지 못함
  - ▶ 미리 정해둔 단어들로 프로젝트를 진행할 예정
  - ▶ 인식하고 싶은 단어가 있다면 따로 자료를 모은 뒤 반영할 예정

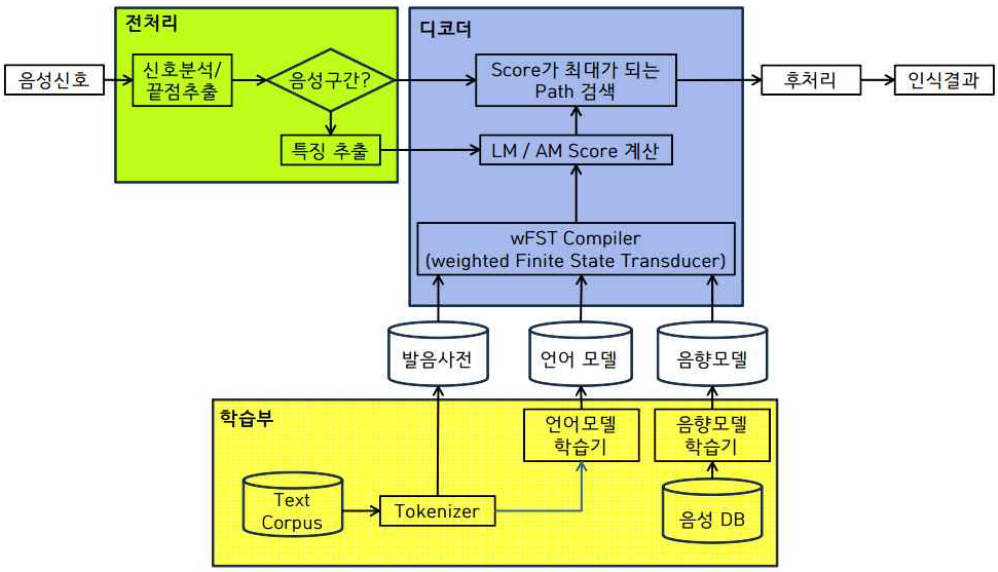
4. 설계 문서

4.1 개발 환경

- 실행 환경 : Raspberry Pi OS, Windows NT
- 개발 언어 : Python, TypeScript, JavaScript, C++ 등
- 개발 도구 : Cloud STT API, Pytorch, TensorFlow, librosa 등
- 데이터 샘플 : PNL 100 Nonspeech Sound, Audioset, FSD50K, 직접 Sampling 등

4.2 사용 기술

4.2.1 Cloud STT(Speech to Text) API



[그림 1] STT 음성인식 엔진 구조

STT는 사람이 말하는 음성을 컴퓨터가 해석해 그 내용을 문자 데이터로 전환하는 기술을 말하며, 쉽게 말해 음성인식을 뜻한다. 로봇, 텔레메틱스, 스마트 디바이스 등 음성으로 기기 제어, 정보 검색 등이 필요한 경우에 응용된다.

음성인식에 사용되는 대표적인 알고리즘 HMM(Hidden Markov Model)의 원리는 모델링 하는 시스템이 미지의 파라미터를 가진 Markov process일 것이라고 가정하여, 그 가정에 기초해서 측정된 파라미터로부터 숨겨진 파라미터를 결정하려는 하나의 통계모델이다. 즉, 음성신호가 Markov Model에 의해 발생했다고 가정하고 학습 단계에서 모델의 파라미터를 추정한 후, 인식기에서는 추정된 파라미

터를 이용하여 미지의 입력 음성에 적합한 모델을 찾아내는 방식이다.

현재 개발된 STT API는 이러한 기능들을 제공하고 있다. 본 과제에서 음성인식 및 출력 기능 구현, 소음 환경 테스트에 이 API를 사용할 계획이다.

4.2.2 librosa

librosa는 음악과 오디오 분석을 위한 Python 패키지다. Clean 음성 파일과 다양한 소음 파일을 합성하기 위해 사용한다. 음원 데이터를 불러오는 것뿐만 아니라 STFT(Short Time Fourier Transform), Mel Spectrogram 생성, MFCC(Mel Frequency Cepstral Coefficient) 생성 등 다양한 기능을 제공하고 있다.

4.2.3 TensorFlow

TensorFlow는 Google 브레인 팀에서 개발한 다양한 작업에 대해 데이터 흐름 프로그래밍을 위한 End-to-End 오픈소스 소프트웨어 플랫폼이다. 심볼릭 수학 라이브러리이자, 딥러닝, 머신러닝 등의 기계학습 응용 프로그램에도 사용되는 TensorFlow는 Python, C++, Java 등 다양한 언어에 대해 API를 지원하기 때문에 사용 언어에 크게 제약을 받지 않아서 전 세계적으로 활발하게 쓰이고 있다.

즉각적인 '모델 반복' 및 쉬운 디버깅 기능을 가능하게 하는, 즉시 실행 기능이 포함된 Keras와 같은 API를 사용하여 ML 모델을 쉽게 빌드하고 학습시킬 수 있다. 데이터 플로우 그래프를 사용하여 시각화하기 편하고 다양한 추상화 라이브러리와 혼용해서 사용 가능하다는 장점이 있다. 또한, 유연한 아키텍처로 구성되어 있어 코드의 수정 없이 데스크탑, 서버 혹은 모바일 디바이스에서 CPU나 GPU를 사용하여 연산을 구동할 수 있다.

4.2.4 PyTorch

PyTorch는 Python을 위한 오픈소스 머신 러닝 라이브러리이다. Torch를 기반으로 하며, 자연어 처리와 같은 애플리케이션을 위해 사용된다. GPU 사용이 가능하므로 속도가 상당히 빠르다. 최근 TensorFlow에 대항해서 사용자가 증가하는 추세이다. Facebook의 인공지능팀이 개발했으며, Uber의 "Pyro" 소프트웨어가 Pytorch를 기반으로 한다.

Pytorch는 두 개의 높은 수준의 파이선 패키지 형태로 제공한다. 첫째로 강력한 GPU 가속화를 통한 Tensor 계산과 두 번째로 테이프 기반 자동 삭제 시스템을 기반으로 구축된 심층 신경망이다. 장점으로는 설치가 간편하고 이해와 디버깅이 쉽고 직관적이고 간결한 코드로 구성된다. "Define by Run"[1] 방식을 기반으로 한 실시간 결과 값을 시각화한다. 이는 TensorFlow의 "Define and Run"[2] 방식과 대비된다.

5. 개발 일정 및 역할 분담

5.1 개발 일정

5월			6월					7월				8월				9월				
2주	3주	4주	1주	2주	3주	4주	5주	1주	2주	3주	4주	1주	2주	3주	4주	1주	2주	3주	4주	5주
관련 기술 공부 및 Case Study																				
	음성 파일과 소음 파일 합성 및 테스트																			
					소음 내성을 가진 학습 모델 제작															
									중간보고서 준비											
											서버 환경 구축									
												설계 문서 수정								
													정확도 평가							
													테스트 및 디버깅							
																최종 발표 및 보고서 준비				

[표 2] 개발 일정 표

5.2 구성원 역할 분담

이름	역할 분담
안준수	- 음성 파일 녹음 및 합성 - 라즈베리파이 코드 작성
강동민	- 서버 개발 - Tensorflow, Pytorch 관련 모델 제작
공통	- 전반적인 지식 이해 - 시스템 테스트 - 성능 평가 - 보고서 작성 - 발표 및 시연 준비 - Git을 이용한 버전 관리

[표 3] 구성원 역할 분담 표

[Reference 및 출처]

<Cloud STT API>
Google Cloud STT: https://cloud.google.com/speech-to-text/pricing?hl=ko
KAKAO NEWTON STT: https://speech-api.kakao.com/
Microsoft Azure STT: https://docs.microsoft.com/ko-kr/azure/cognitive-services/speech-service/rest-speech-to-text

<Data Samples>
PNL 100 Nonspeech Sounds: http://web.cse.ohio-state.edu/pnl/corpus/HuNonspeech/HuCorpus.html
AudioSet: https://research.google.com/audioset/
FSD50K: https://zenodo.org/record/4060432#.YJTrJrUzaUI

<Python Library>
librosa: https://librosa.org/

[그림 1] 출처 : https://blog.skcc.com/3764

1) “Define by Run" 방식은 일반적인 Python 코딩이랑 크게 다르지 않다. 선언과 동시에 데이터를 집어넣고, 세션 같은 것도 필요 없이 그냥 돌리면 된다. 덕분에 코드가 간결하고, 난이도가 높지 않다.
2) "Define and Run" 방식은 코드를 직접 돌리는 환경인 세션을 만들고, placeholder를 선언하고 이것을 계산 그래프로 만들고(Define), 코드를 실행하는 시점에 데이터를 넣어 실행하는(Run) 방식이다. 이는 계산 그래프를 명확히 보여주면서 실행 시점에 데이터만 바꿔줘도 되는 유연함을 장점으로 갖지만, 그 자체로 비관적이다. 그래서 딥러닝 프레임워크 중 난이도가 가장 높은 편이다.