음성인식을 이용한 항공교통관제(ATC) 보조시스템

Air Traffic Control Assistance System Using Speech Recognition

(성명) 김 동 균
(지도교수) 홍 정 표
(E-MAIL) kim.dk.bob@gmail.com

요약 (Abstract)

항공교통관제(Air Traffic Control, ATC)는 공중과 지상에서 항공기의 안전운항을 보장하고 원활한 항공교통흐름을 유지하게 하는 업무를 의미한다. 이러한 항공교통관제 업무 중 발생하는 인적오류(Human Error)는 관련 문헌에 따르면 항공교통사고의 60% 이상을 차지한다. 이러한 인적오류의 배경에는 관제사와 조종사의 피로 및 장시간 업무가 존재한다. 따라서 저하되는 집중력을 보완해주는 시스템의 필요성이 제기된다.

이 시스템은 항공교통관제 업무가 주로 음성통신을 통해 이뤄진다는 특성에서 음성인식을 활용한 보조시스템을 구성했다. Python 언어를 중심으로 개발을 진행했고, 각각의 서버에서 음성인식 및 웹서버가 따로 구동되도록 했다. GUI 프로그램과 웹페이지를 통해 녹음파일을 업로드 받으면, 별도의 쓰레드에서 음성인식이 작동하도록 했다. 음성인식이 완료되면 시간, 호출부호, 음성인식 결과, 음성파일 경로 순으로 DB에 Query된다. 이후 웹서버를 통해 해당 내용을 열람할 수 있고, 이에 기반한 통계데이터도 제공된다.

음성인식 모델은 68만 시간에 달하는 대량의 데이터로 학습된 범용 음성인식(STT) 모델인 OpenAI社의 Whisper 모델을 이용했다. 해당 모델을 항공교통관제 음성인식 데이터셋을 통해 미세조정(Fine-Tuning) 하였다. 그 결과, Original Whisper 모델의 WER(Word Error Rate)가 112.9%에 달하는 반면, 미세조정 된 모델은 9.9%에 그쳤다.

본 시스템은 항공교통관제에서 인적오류를 감소시켜 안전한 항공운항 환경을 조성하는데 기여할 수 있다. 또한, 항공교통사고의 조사 및 항공영어 구술 평가 등에도 활용될 수 있다. 본 연구를 통해 범용 모델을 이용한 특정 전문 분야의 맞춤 음성인식 모델 개발 및 활용 가능성이 제시될 수 있고, 유사한 환경에서의 시스템 설계에 기여할 것으로 기대된다.

Ⅰ. 서론

항공교통관제(Air Traffic Control, ATC)는 공중과 지상에서 항공기의 안전운항을 보장하고 원활한 항공교통흐름을 유지하게 하는 업무를 의미한다[1]. 항공교통관제 서비스는 HF(대양 비행 등), VHF(항로관제, 공항관제, ATIS 등 주요 사용), UHF(군 항공기용)의 대역에서 음성통신을 통해 제공한다. 이러한 무선 음성통신을 통해 이루어지는 항공교통관제에는 여러 장애요소가 존재한다. 먼저, 자연환경이나 전파의 특성상 발생하는 환경적 요소가 존재한다. 인접한 채널에서 발생하는 간섭, 비와 안개, 번개와 같은 기상 문제 등이 이에 해당한다. 이러한 환경적 요소는 안테나를 개선하거나, 신호 처리 회로를 개선하는 등 하드웨어의 개선으로 해결이 가능하다. 다른 장애요소로는 항공교통관제사나 항공기 조종사의 신체적, 심리적 요소로 인해 발생하는 인적요소가 존재한다. 인적요소는 과도한 업무로 인한 피로, 장시간 업무로 인한 집중력 저하, 업무 상의 상하관계 등의 요소를 포함한다. 이러한 문제는 업무, 교육 매뉴얼로 개선을 시도하지만, 환경적 요소에 비해 시간과 자원이 많이 소요되는 복잡한 문제에 해당한다. 박수애의 연구에 따르면 1994년 미국연방교통안전위원회(NTSB)가 항공 교통 사고 발생 원인의 60% 이상이 인적오류라는 결과를 내놓았다[2].

항공교통관제사는 대부분 피로와 긴장이 가중되기 쉬운 환경에서 근무하고 있다.

국제공항의 경우 24시간 근무체제로, 일일 8시간의 주·야간 순환근무체제가 반복되고 있다[2]. 항공안전법 시행규칙에 따르면, 항공기 조종사도 일일 8시간, 최대 13시간까지 승무가 가능하다. 28일 내에는 100시간까지, 1년에 1,000시간까지 비행이 가능하다[3]. 이러한 환경에서 관제사와 조종사의 피로는 누적되고 집중력 또한 저하될 수 밖에 없다. 따라서, 항공교통관제 업무에서의 보조시스템이 필요성이 제기된다.

항공교통관제업무가 무선 음성통신을 통해 주로 행해지는 특성에서 이를 보조할 수 있는 시스템은 음성인식(Speech Recognition) 시스템이 적절하다고 판단된다. 이번 연구에서는 음성인식을 이용해 항공교통관제를 문자화 해줄 수 있는 시스템을 제안하고자 한다.

항공교통관제(ATC)을 음성인식을 이용해 시스템화 시키려는 시도는 일찍부터 있어왔다. 기계학습 모델인 HMM(Hidden Markov Model), GMM(Gaussian Mixture Mode)-HMM과 하이브리드 모델인 DNN(Deep Neural Network)-HMM, 딥러닝 모델인 TDNN-F(Time-Delay Neural Network – Factorized) 순으로 발전해왔다. 최근 들어서는 BiLSTM(Bidirectional LSTM), CNN(Convolutional Neural Network) + TDNN-F와 같은 E2E(End to end) 기반 모델들이 음성인식에 사용하고 있다[4].

본 연구에서는 Transformer 기반의 음성인식 모델을 통해, ATC 음성인식 모델을 만들어보고자 한다. Transformer 모델은 2017년 구글 브레인에서 개발한 신경망 아키텍쳐로, 자연어처리(NLP)에서 높은 성능을 보여 최근 주목받는 모델이다. Transformer 모델은 시퀀스 데이터에서 단어들의 관계를 병렬처리를 통해 빠르게 처리할 수 있어 학습속도가 RNN, LSTM에 비해 빠르다. 또한 Attention Mechanism을 통해 중요한 단어에 더 많은 가중치를 주어 필요한 정보에 집중할 수 있게 해준다. 따라서 음성인식에 사용하기 적합한 모델이라 볼 수 있다[5].

음성인식 모델 학습에 사용된 데이터가 많을수록, 해당 모델의 정확도(Accuracy)가 올라간다. 하지만 항공교통관제 모델에 맞는 관련 데이터의 수는 제한적이다. 따라서 Top-down 접근 방식을 고려하여 다양한 학습데이터를 이용한 범용 음성인식모델에 항공교통관제 학습 데이터를 미세조정(Fine-Tuning)하는 방향으로 나아가기로 했다. 이러한 방식을 이용하면 소수의 학습데이터로도 높은 인식률을 달성할 수 있고, 비표준용어를 사용하더라도 문제없이 인식할 수 있다. 따라서 Transformer 모델이자, 학습에 사용한 데이터가 많은 범용 음성인식모델인 OpenAI社의 Whisper를 이용하기로 했다. Whisper는 Transformer 기반의 음성인식(Speech to Text, STT) 모델로 96개국의 68만 시간의 음성데이터로 학습시켰다[6].

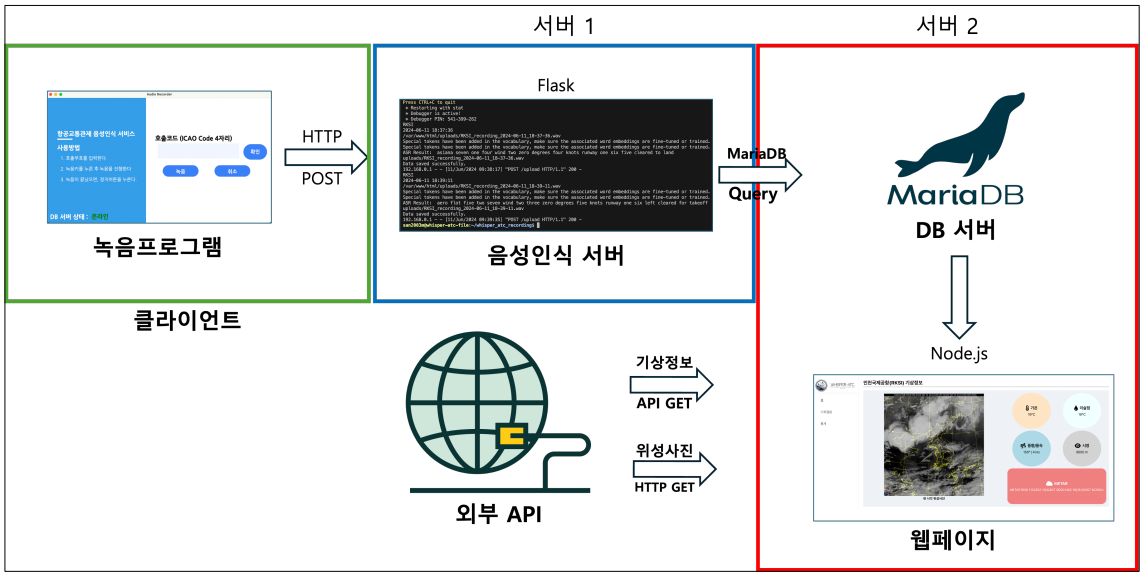
전체적인 흐름은 Original Whisper 모델을 미세조정하고, Fine-Tuned 모델을 실험할 수 있도록, 전체적인 시스템을 설계하고 Demo를 만들어 실험할 것이다. 또한, Whisper 오리지널 모델과의 성능 비교 순으로 이루어질 것이다.

본 연구는 기존 연구의 한계점을 극복하고, 높은 정확도와 저부하 환경에서도 효율적으로 동작할 수 있는 시스템을 구축했다. 본 시스템을 통해서, 2042년을 목표로 하는 인터넷 프로토콜 기반의 CRV (Common Regional Virtual Private Network)을 구축 및 배치[7]까지의 공백기간 동안 항공교통관제에 있어서 조종사나 관제사의 인적오류(Human Error)를 줄여 안전한 항공기 운항환경 조성에 기여할 수 있을 것으로 기대된다.

Ⅱ. 시스템 설계

본 연구에서는 미세조정(Fine-Tuning)된 음성인식 모델을 이용해서 녹음, 음성인식, 결과열람 등의 기능을 구현한 시스템을 설계하였다. 전체적인 구조도는 [그림 1]와 같다. 본 시스템은 클라이언트에서 Local로 음성인식을 수행하는 것이 아닌, 중앙

의 서버가 파일을 업로드 받아 작업을 수행한다. 이러한 이유는 항공기의 공간적 제약에서 비롯된다. 음성인식 모델으로 작업을 수행할 정도의 연산능력을 갖추고, WAV 파일 형태로 저장되는 음성파일을 저장하고 보관하는 장비를 두는 것은 공간적, 비용적 문제가 있다고 판단했기 때문이다. 따라서, 육상에 서버를 두고 항공기의 저궤도(LEO) 위성을 이용한 광대역 인터넷으로 서버에 음성파일을 업로드할 것이라는 환경을 가정하고 시스템을 설계했다. 전체적으로 웹페이지나 GUI 프로그램에서 녹음한 WAV 파일을 관제탑 또는 항공기에서 파일을 업로드 받아, 음성인식서버에서 DB 및 웹서버에 시간, 호출부호, 인식내용, 파일경로를 Query하고, 이를 통해 웹페이지로 인식결과를 열람할 수 있도록 구성되었다.



[그림 1] 항공교통관제(ATC) 보조시스템 전체 구조도

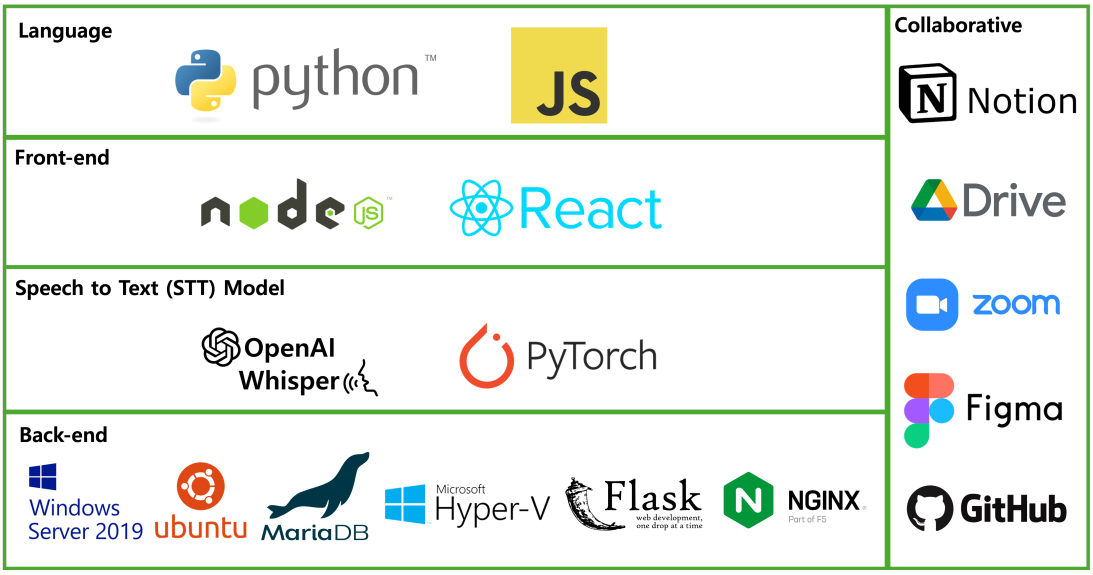
본 저자는 이 시스템에서 GUI 형식의 녹음프로그램 제작, 음성인식 모델 학습 및 인프라·백엔드 파트의 개발을 맡았다. Windows Server 상에서 Hyper-V을 이용해 2대의 Ubuntu 가상서버를 구축했다. 서버 1에는 음성인식용 서버, 서버 2에는 MariaDB 및 Node.js 기반 웹서버를 가동할 수 있도록 구축하였다. 이렇게 개발한 항공교통관제 보조시스템의 명칭을 'Whisper-ATC'로 명명하고 [그림 2]의 로고를 사용하여 GUI 프로그램이나 웹페이지에서 시스템의 특징이 잘 드러나도록 했다.

[그림 2] Whisper-ATC 프로젝트 로고

Ⅲ. 시스템 구현

3.1 개발 환경

아래의 [그림 3]는 개발에 사용한 언어와 프로그램, UI를 정리한 기술스택이다. 음성인식 모델의 학습과 백엔드 구축에 Python를 중심으로 개발하였고 사용된 라이브러리는 [표 1]과 같다. 음성인식 모델의 미세조정에는 Pytorch를 이용해 [표 2]와 같은 환경하에서 학습이 진행되었다.



[그림 3] 개발에 사용된 기술 스택

라이브러리	버전
flask	2.2.5
jinja2	3.1.3
numpy	1.26.4
PyAudio	0.2.14
pytorch	2.2.2
urllib3	1.26.18
sounddevice	0.4.7
matplotlib	3.9.0
mysql-connector-python	8.4.0

[표 1] 사용된 라이브러리

CPU	Ryzen 7 2700
RAM	48GB
GPU	RTX3060TI
S/W	Python 3.11 Pytorch 2.2.2 CUDA 12.4

[표 2] 모델 학습환경

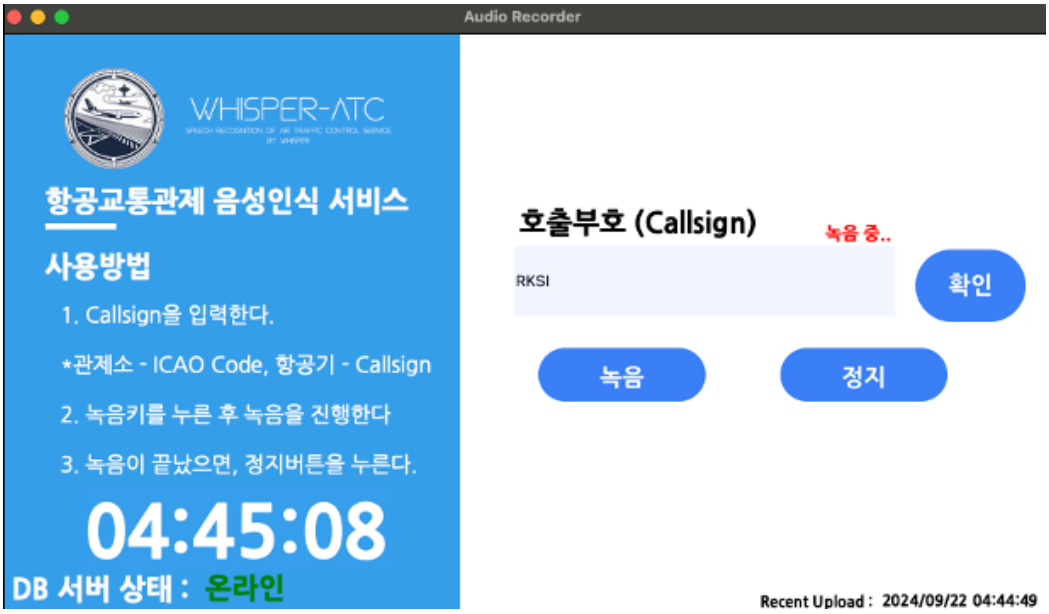
음성인식과 데이터 보관 및 웹서버의 사양은 [표 3]와 같다.

	물리 서버	VM 서버 1	VM 서버 2
CPU(Cores)	AMD Ryzen 5 2600X (6 Cores, 12 Threads)	6 Cores	1 Cores
RAM	16GB	8GB	2GB
GPU	GTX 1050	-	-
OS	Windows Server 2022 Datacenter	Ubuntu 24.04	Ubuntu 24.04

[표 3] 물리서버 및 VM 사양

3.2 구현 결과

먼저, [그림 4]에 보이는 **GUI 프로그램**의 경우, Python의 GUI 라이브러리인 Tkinter를 기반으로 작성했다. UI의 경우 디자인툴인 Figma[8]를 활용하여 디자인하였다. 호출부호를 입력하고 확인을 누르면, 녹음 버튼을 눌러 녹음을 시작할 수 있다. 녹음과 동시에 WAV파일이 생성되고 기록이 시작된다. 그리고, 녹음이 끝나면 정지버튼을 누른다. 정지버튼을 누르면 자동으로 녹음된 WAV파일이 음성인식 서버(서버 1)로 업로드되면, 서버 측에서 자동으로 음성인식을 시작한다. 업로드가 완료되어, 서버로부터 업로드 성공 응답을 받게되면 자동으로 우측하단의 시간이 현재시간으로 업데이트된다. 그리고 1초에 한번씩 DB서버에 ping을 보내 서버상태가 정상인지를 체크한다. 만약, 서버가 오프라인이거나 클라이언트가 인터넷에 제대로 연결이 되지 않은 경우, 빨간색 글자로 '오프라인'이라고 출력된다.



[그림 4] GUI 프로그램의 UI

본 시스템에서 **음성인식 모델**은 항공교통관제 음성인식을 연구하는 기존 연구들에서 사용했던 학습데이터를 모두 합친 데이터셋을 사용하였다[9]. 해당 데이터는 16kHz의 WAV파일로 구성된 데이터로, 18,900여 개의 훈련(Train)데이터와 4,720여 개의 검증(Test) 데이터로 구성되어 있다. 해당 데이터셋은 [표 4]과 같이 구성 되어있다.

라벨	데이터
id	고유번호
audio	음성파일(문제)
text	스크립트(정답)
segment_start_time	음성파일 묶음에서 시작지점
segment_end_time	음성파일 묶음에서 종료지점
duration	음성파일의 길이

[표 4] 학습 데이터셋의 구성

음성인식 모델을 미세조정하기 위해 Orginal이 되는 모델은 Hugging Face에 업로드되어 있는 Whisper-small 모델을 이용하였다. 이 데이터 셋을 [표 5]에 있는 훈련 파라미터로 미세조정(Fine-Tuning) 하였다.

learning_rate: 1e-05
train_batch_size: 8
eval_batch_size: 8
seed: 42
gradient_accumulation_steps: 2
total_train_batch_size: 16
optimizer: Adam with betas=(0.9,0.999) and epsilon=1e-08
lr_scheduler_type: linear
lr_scheduler_warmup_steps: 500
training_steps: 5000

[표 5] 미세조정 훈련 파라미터

본 저자는 Hyper-V를 이용한 VM에 우분투 서버를 2대를 구축하여, 각각 음성인식 서버와 DB 및 웹서버로 이분화하였다. 먼저 음성인식 모델을 작동시키기 위한 서버 1에는 Python을 중심으로 구성하였다. [코드 1]는 GUI 프로그램이나 웹페이지에서 Flask를 통해 파일을 업로드부터 음성인식 시작까지의 코드이다.

```
def upload_file():
                                        ...
        # 고유한 파일명을 생성 (UUID 추가) - 같은 시간에 똑같은 파일을 업로드 받았을 경우 예외사항을 방지하기 위함.
        original_filename = file.filename
        unique_filename = f"{uuid.uuid4()}_{original_filename}"
        filepath = os.path.join(UPLOAD_FOLDER, unique_filename)

        # 파일을 저장
        filepath = "/var/www/html/"+filepath # 웹서버의 루트 디렉토리에 저장
        file.save(filepath)
        print(f"File saved to: {filepath}")

        # 별도의 스레드에서 파일 처리 시작
        threading.Thread(target=process_file, args=(filepath, original_filename)).start()

    # 파일 업로드 이후 클라이언트에 즉시 응답 반환
    return jsonify({'message': 'File uploaded successfully. Processing will continue in background.'}), 200
```

[코드 1] upload_file 함수 (업로드~음성인식 시작까지)

GUI 프로그램이나 웹페이지를 통해 WAV파일이 업로드된다면, 클라이언트(녹음자) 측에 정상으로 업로드 받았음을 반환해준다. 그리고 파일의 이름에 UUID를 먼저 추가해 중복파일 관련 예외처리를 한다. 그리고 음성파일을 저장 후에 별도의 쓰레드로 음성인식 모델에 투입시킨다. 이후 DB에 시간, 호출부호, 변환한 스크립트, 파일위치 등을 Query한다.

서버 2에서는 MariaDB를 기반으로 데이터를 저장하고, 불러오는 백엔드 환경을 구성했다. 먼저 [코드 2]는 이 시스템의 데이터베이스의 구조를 보여주고 있다.

```
MariaDB [whisper]> show tables;
+------------------+
| Tables_in_whisper |
+------------------+
| ATC              |
| Weather          |
+------------------+
2 rows in set (0.000 sec)
```

[코드 2] 시스템 DB의 구조

ATC 테이블은 음성인식 모델이 변환한 데이터를 저장하고, weather 테이블은 웹페이지의 홈에서 제공하는 날씨 정보를 저장하고 있다. [코드 3]에서는 ATC 테이블의 구조를 보여주고 있다.

```
MariaDB [whisper]> DESC ATC;
+------------+-------------+------+-----+---------+----------------
+
| Field      | Type        | Null | Key | Default | Extra          |
+------------+-------------+------+-----+---------+----------------
+
| id         | int(20)     | NO   | PRI | NULL    | auto_increment |
| time       | datetime    | NO   |     | NULL    |                |
| radio_code | varchar(50) | NO   |     | NULL    |                |
| script     | text        | NO   |     | NULL    |                |
| path       | text        | NO   |     | NULL    |                |
+------------+-------------+------+-----+---------+----------------
+
5 rows in set (0.040 sec)
```

[코드 3] ATC 테이블의 구조

ATC 테이블은 고유번호(id), 시간(time), 호출부호(radio_code), 변환결과(script), 음성파일 경로(path)로 구성되어 있다. 고유번호(id)는 입력하지 않으면 자동으로 순서에 맞춰 입력된다. 따라서, 값이 중복될 이유가 없기에 Primary Key로 설정했다.

Ⅳ. 동작 및 성능 시험

4.1 동작 시험

앞에서 설명한 Python을 기반으로 개발한, GUI 프로그램의 경우 [로그 1]에서 보이는 바와 같이 서버와 연동하여 실시간으로 음성인식 서버로 파일을 정상적으로 업로드하는 모습을 볼 수 있다.

```
/Users/dk/IdeaProjects/whisper_atc_recording/venv/bin/Python
/Users/dk/IdeaProjects/whisper_atc_recording/GUI/test.py
RKSI
start Upload... Filename : RKSI_recording_2024-12-08_18-48-24.mp3
{
  "message": "File uploaded successfully. Processing will continue in background."
}
filename : RKSI_recording_2024-12-08_18-48-24.mp3
```

[로그 1] GUI 프로그램의 로그

또한, [로그 2]에서 https를 통해 파일이 정상적으로 업로드 되며, 음성인식 모델으로 투입 또한 정상적으로 이루어지는 것을 알 수 있다.

```
File saved to:
/var/www/html/uploads/bddaac03-ef5f-4186-bbe5-fc121746f8d3_RKSI_recording_2024-12-08_18-48-24.mp3
Callsign: RKSI192.168.0.1 - - [08/Dec/2024 09:48:36] "POST /upload HTTP/1.1" 200 -
Extracted Time: 2024-12-08 18:48:24
Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.
Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.
ASR Result:  delta one two three cleared for takeoff runway one two right
DB Filepath:
uploads/bddaac03-ef5f-4186-bbe5-fc121746f8d3_RKSI_recording_2024-12-08_18-48-24.mp3
Data saved successfully.
```

[로그 2] 음성인식 서버(Flask)의 로그

출력된 결과가 정상적으로 DB서버에 Query 되었는지에 관해서는 [코드 4]를 통해서 확인할 수 있다.

```
MariaDB [whisper]> select * from ATC;
...
+-----+---------------------+------------+------------------------
| id | time | radio_code | script | path |
+-----+---------------------+------------+------------------------
...
| 226 | 2024-12-08 18:48:24 | RKSI |  delta one two three cleared for tak
eoff runway one two right | uploads/bddaac03-ef5f-4186-bbe5-fc121746f8
d3_RKSI_recording_2024-12-08_18-48-24.mp3 |
...
122 rows in set (0.001 sec)
```

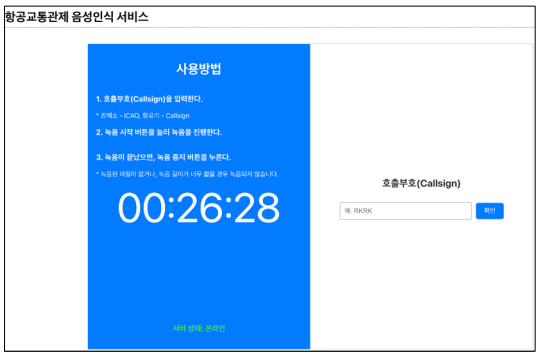
[코드 4] Database 상에 데이터가 입력된 모습

전체적인 시스템의 동작 모습은 참고문헌의 영상에서 6분 20초에서 8분 5초를 볼 수 있다[10]. 음성인식 모델의 전반적인 정확도에 관련된 자세한 성능은 4.3 성능평가에서 후술할 예정이다.

4.2 상호 연동 시험

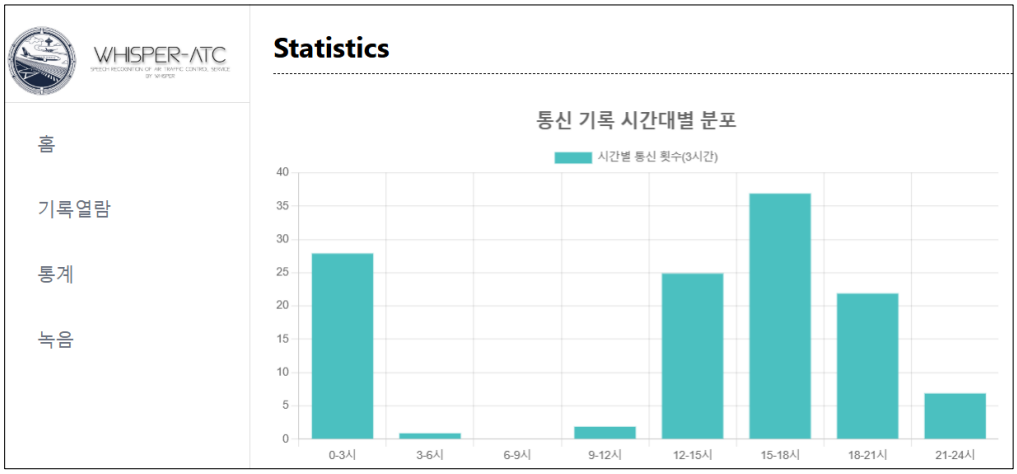
위와 같이 훈련된 음성인식 모델과 Python 기반 GUI 프로그램, 2대의 VM서버를 Node.js와 React를 이용해 전체적인 항공교통관제 보조시스템을 구축하였다. 사용자가 [그림 5]의 GUI프로그램 또는 [그림 6]의 웹페이지를 통해, 항공교통관제 내용을 녹음시, 16kHz WAV형식 파일로 저장된다.

[그림 5] GUI 프로그램의 모습



[그림 6] 녹음 페이지의 모습

녹음된 파일은 https 프로토콜을 통해 Flask 웹서버로 전달된다. 해당 서버에서 파일이름의 가공과 저장이 이루어진 후, 별도의 쓰레드에서 음성인식이 시작된다. 음성인식이 끝나면 결과가 출력되고 시간, 호출부호, 스크립트, 파일경로가 MariaDB 서버로 Query 되어 저장된다. 해당 정보는 [그림 7]의 웹 페이지에서 시간 및 호출부호별로 청취가 가능하다. 또한 관제 데이터를 집계해 시간대별로 통계를 내어 관제 데이터를 별도의 데이터 형태로 가공할 수 있다.



[그림 7] 통계 페이지의 모습

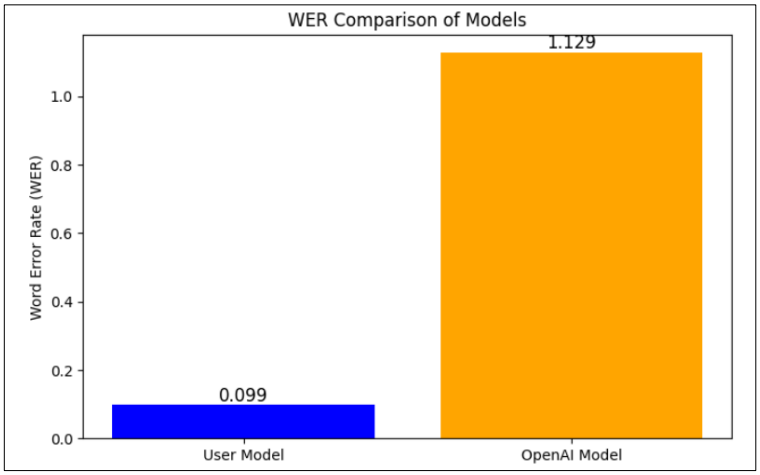
4.3 성능 평가

음성인식 모델의 성능을 나타내는 척도로는 SNR, PESQ 등 다양한 척도가 존재한다. 본 연구에서는 발화한 문장에서 정확하게 인식한 단어의 비율을 나타내는 단어 오인식률(Word Error Rate, WER)을 사용하고자 한다. WER는 음성인식시스템의 성능을 평가하는 표준적인 방법이다[11]. WER의 계산식은 [식 1]과 같다. WER은 오인식률을 의미하므로, 낮을수록 음성인식이 정확하다는 의미이다.

$$Word\,Error\,Rate\,(WER) = 100 \times \frac{\text{잘못 대체된 단어 수}(S) + \text{잘못 삭제된 단어 수}(D) + \text{잘못 추가된 단어 수}(I)}{\text{정답 텍스트의 총 단어수}(N)}$$

[식 1] Word Error Rate(WER) 공식 [12]

이 단계에서는 Original Whisper 모델과의 비교를 통해 기존 모델에서 얼마나 항공 교통관제에 적합하게 개선되었는지를 알아볼 것이다. 검증(Valid) 데이터셋은 학습에 사용했던 데이터셋의 별도로 마련된 Valid 데이터셋을 이용할 것이다. 해당 데이터는 학습에 일절 사용하지 않았다. 검증 환경은 위의 모델 학습과정에서 사용한 환경과 상동하다.



[그림 8] 모델의 WER(Word Error Rate) 측정 결과

[그림 8]은 검증(Valid) 데이터셋 4,723개 샘플의 WER를 평균화한 값이다.

Original Model은 1.129(112.9%), Whisper-ATC는 0.099(9.9%)의 WER를 출력하였다. Original Model의 경우, 100%를 초과하였다. WER은 정답 텍스트의 단어 수 대비 대체 및 삭제, 삽입된 단어의 수를 나타낸다. 따라서, 다음과 같은 이유가 있을 수 있다. 첫 번째로 정답 텍스트의 단어 수의 길이가 짧거나 0에 가까운 경우(N이 작을 때), 두 번째로 모델이 예측한 텍스트가 쓸모없는 단어를 과도하게 추가 및 삭제한 경우(S, I, D가 큰 경우)가 있다. WER가 1 이상(100%)의 경우, 대체하거나 추가한 단어들이 많음을 의미한다[13]. 따라서 Fine-Tuning 된 Whisper-ATC의 경우가 항공교통관제를 인식하는데 더 적합하다고 볼 수 있다.

Ⅴ. 결론

음성인식을 이용한 항공교통관제 보조시스템을 개발하게 된 배경은 인적오류(Human Error)로 발생하는 항공사고를 줄이기 위해서였다. 이 시스템의 특징은 음성인식을 통해 항공교통관제 내용을 텍스트로 바꾸고, 그 데이터를 체계적으로 관리할 수 있다는 특징을 가지고 있다. 본 연구에서 개선한 음성인식모델은 Original Whisper 모델에 비해 항공교통관제에 적합하도록 학습되어 오류율 9.9%까지 개선하여 높은 정확도를 달성할 수 있었다. 또한 물리적 서버가 제한되어 있는 상황에서도 VM을 활용해 고부하 작업과 저부하 작업을 분산시켜 수행하였다. 따라서, 서비스의 안정적인 운영이 가능한 환경을 조성하였다. 또한 웹서비스와 연동하여 데이터를 열람할 수 있게 하고, 통계 데이터로도 활용할 수 있도록 하였다.

본 연구에 사용된 데이터셋이 유럽 및 북미권의 관제 데이터를 중심으로 구성되었기 때문에, 발화자의 억양에 영향을 받는 경우가 있다. 이 부분은 추후 연구를 진행할 때, 아시아권 발화자의 데이터를 더욱 수집해 아시아권 발화자의 인식률을 향상해야 할 것이다. 또한 녹음 시, 호출부호를 입력하지 않고 음성인식을 통해 호출부호를 음성인식을 통해 자동으로 변환할 수 있는 기능을 강화할 필요가 있다.

본 시스템은 항공교통관제 내용을 문자적으로 볼 수 있으므로, 교통의 인적오류(Human Error)를 줄일 수 있는데 기여할 수 있다. 또한, 항공교통사고의 조사, 조종사의 항공영어 구술 평가 등과 같은 방향으로도 사용될 수 있다. 거시적인 시각에서 보면, 대량의 데이터를 이용해 학습시켜 만들어진 Whisper 음성인식(Speech Recognition) 모델의 활용 방안을 제시했다. 항공교통관제 등 특정 전문 분야에서 데이터셋을 이용해 해당 산업에 맞는 모델을 제작하고 활용할 수 있다는 의의가 있다. 따라서 여러 산업 분야에서 범용 모델을 이용한 다양한 딥러닝 모델이 만들어질 것으로 기대된다.

■ 참 고 문 헌 ■

[1] 국토교통부, "항공교통업무," 국토교통부 항공정책실. https://www.molit.go.kr/sroa/USR/WPGE0201/m_35744/DTL.jsp (접속일: 2024년 12월 1일).

[2] 문우춘, 최연철, 양한모, “항공교통관제사와 조종사 인적오류의 연계성”, 한국항공운항학회지, vol 16, 호 4, pp 35-40, 2008.

[3] 항공안전법 시행규칙, 별표 19 "객실승무원의 비행근무시간 및 휴식시간기준", 제128조제2항 관련.

[4] Z. Wang et al., “Enhancing Air Traffic Control Communication Systems with Integrated Automatic Speech Recognition: Models, Applications and Performance Evaluation,” Sensors, vol. 24, no. 14, Art. no. 14, Jan. 2024, doi: 10.3390/s24144715.

[5] A. Vaswani et al., “Attention Is All You Need,” Aug. 02, 2023, arXiv: arXiv:1706.03762. doi: 10.48550/arXiv.1706.03762.

[6] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, “Robust Speech Recognition via Large-Scale Weak Supervision,” Dec. 06, 2022, arXiv: arXiv:2212.04356. doi: 10.48550/arXiv.2212.04356.

[7] 국토교통부, "국가항행계획 - NARAE, National ATM Reformation And Enhancement," 대한민국 국토교통부, 서울, 대한민국, 2024.

[8] https://www.figma.com/ (접속일: 2024년 12월 4일)

[9] Shiry, "ATC_combined," Hugging Face. https://huggingface.co/datasets/Shiry/ATC_combined (접속일: 2024년 12월 3일).

[10] YouTube, "음성인식을 이용한 항공교통관제(ATC) 보조시스템," YouTube, 2024. [온라인]. Available: https://youtu.be/tM9iTJJBPTE. (접속일: 2024년 12월 4일).

[11] 이여진, “인공지능 자동음성인식기들의 한국인의 영어 발음 산출 훈련 적합성 평가”, Thesis, 서울대학교 대학원, 2022. 접근된: 2024년 12월 8일. [Online]. Available at: https://s-space.snu.ac.kr/handle/10371/188458

[12] D. Jurafsky, J. H. Martin, S. Russell, and P. Norvig, Speech and language processing. Upper Saddle River, N.J.: Pearson/Prentice Hall, 2009. [Online]. Available: https://www.riss.kr/link?id=M11355120

[13] 지승은과 김우일, “음성 인식용 데이터베이스 검증시스템을 위한 새로운 음성 인식 성능 지표”, 한국정보통신학회논문지 = Journal of the Korea Institute of Information and Communication Engineering, vol 20, 호 3, pp 464-470, 2016, doi: 10.6109/jkiice.2016.20.3.464.