

2024년 전기

STT 기술을 이용한 다중 화자 탐지 구현



부산대학교 전기컴퓨터공학부 정보컴퓨터 공학 전공

지도교수 : 권준호

팀 명 : *Untouchable*

201824492 변상윤

201724433 문성필

<목차>

1. 과제 목표
2. 요구사항 / 제약 사항 분석 수정 사항
3. 설계 상세화
4. 갱신된 과제 추진 계획
5. 구성원별 진척도
6. 현재 과제 수행 내용 및 결과

1. 과제 목표

최근 인공지능 기술의 발전으로 음성인식 기술인 STT (Speech-To-Text) 모델이 다양하게 개발되었다. STT란 사람이 말하는 음성 언어를 컴퓨터가 해석하여 텍스트로 변환하는 처리를 의미한다. 이는 회의록 작성, 유튜브 자막 생성, 상담 기록, 음성 명령어 처리, 청각 장애인들의 학습권 보장 등 다양한 분야에서 활용되고 있다.

우리는 이러한 최신 stt를 이용하여 다중화자 기능을 추가함으로써 위의 효과와 더불어 추가적인 효용을 기대할 수 있다. 먼저 다중화자 인식에 대한 알고리즘 학습과 그에 대한 심화를 생각 해 볼 수 있다. 또한 다중화자 구현 이전에 STT 코드 분석이 선행되어야 하므로 STT 자체에 대한 이해도를 증가시킬 수 있다.

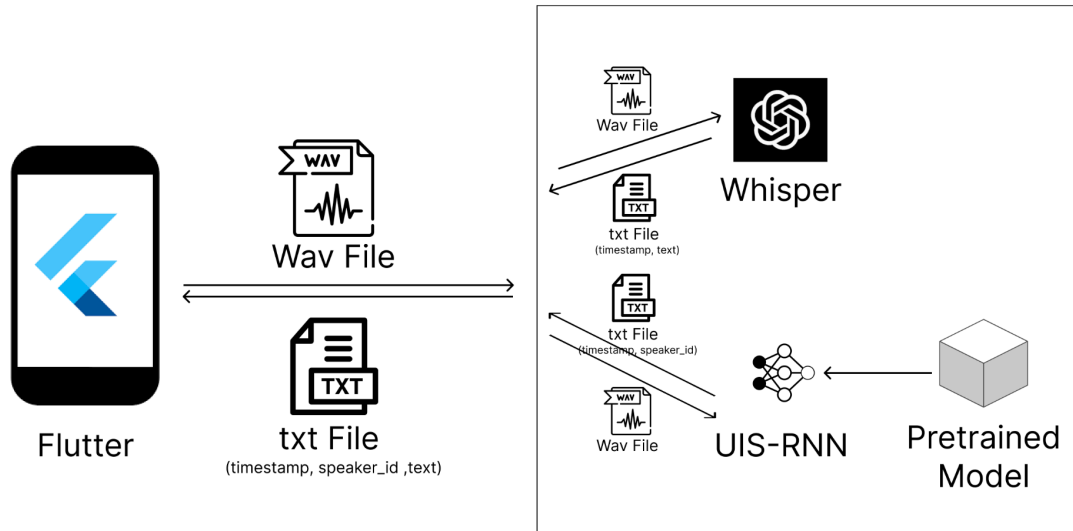
2. 요구사항 / 제약 사항 분석 수정 사항

2.1. 요구 조건 분석

2.1.1. STT 구현

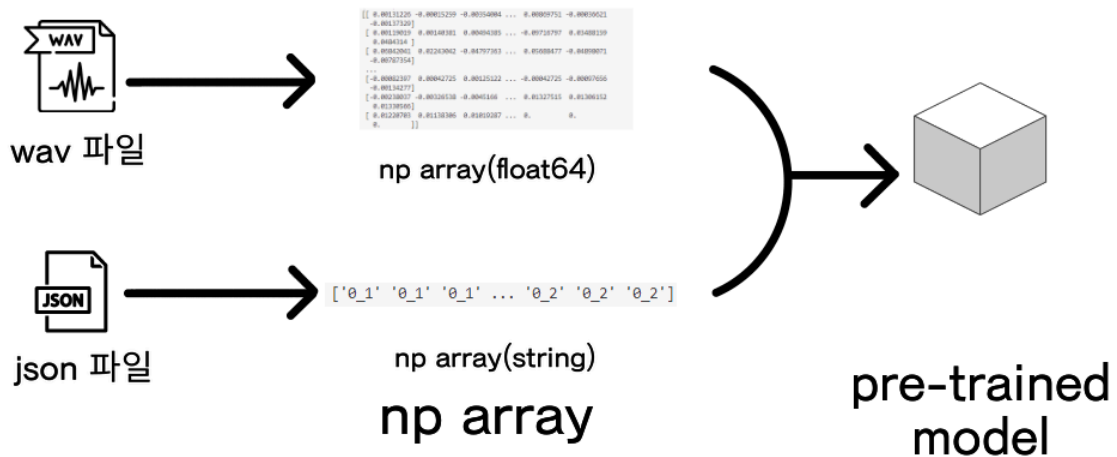
STT를 자체 구현하는 방향에서 기존 연구된 API를 사용하는 방향으로 변경하였다. 비용이 너무 많이 들어서 다중 화자 구현이 더 우선순위가 높다고 생각을 했다. 그래서 우선 다중 화자 구현을 한 뒤, 여유가 되면 STT 까지 구현을 해보기로 했다.

3. 설계 상세화



flutter를 이용하여 사용자 인터페이스를 구성하고, 자세한 기능 구현은 whisper와 UIS-RNN을 이용한다.

UIS-RNN에서는 우리가 만든 pre-trained 모델을 이용하여 speaker diarization을 수행하도록 한다. 그 후 출력 결과로 time stamp와 speaker id가 주어진 txt를 return값으로 하여 whisper의 출력물인 utterance, time stamp와 적절히 결합하여 하나의 txt 파일로 출력하도록 한다.



aihub에서 받은 wav파일과 labeling이 된 json 파일을 UIS-RNN에서 사용하는 np array 형식으로 변조 한 후 이를 list로 묶어 pre-trained model(.npz)을 만든다.

4. 갱신된 과제 추진 계획

4.1. speaker diarization : UIS-RNN을 이용한 구현

4.1.1. UIS-RNN을 이용한 구현으로 방향성을 설정했다.

4.1.1.1. UIS-RNN을 사용하여 test npz를

실행해보았고 aihub에서 제공하는 data로 train npz를 구현하여 실행하기로 하였다.

4.2. windows로 변경

4.2.1. windows와 Linux를 비교하여 볼 때 Windows의 환경에서도 목표로 하는 바를 충분히 해결할 수 있을 것으로 보아 조금 더 친숙한 플랫폼으로 변경하게 되었다.

5. 구성원별 진척도

5.1. 문성필

5.1.1. Whisper에 대한 조사

5.1.1.1. Whisper는 OpenAI에서 개발한 음성 인식 모델이다. Whisper와 우리가 다중 화자 분리를 구현한 부분을 결합해서 프로그램을 제작할 것이다.

5.1.2. 다중화자 구현에 대한 방법 탐구

5.1.2.1. 다중 화자 분리 구현은 여러 가지 방법으로 구현이 가능하다. 여러 자료들을 찾아서 다양한 방법으로 구현을 해보았다. 현재 과제 수행 내용 및 결과에 자세하게 후술하였다.

5.2. 변상윤

5.2.1. speaker diarization 예제 코드 실행

5.2.2. UIS-RNN(Unbounded Interleaved-State Recurrent Neural Network)에 대한 조사

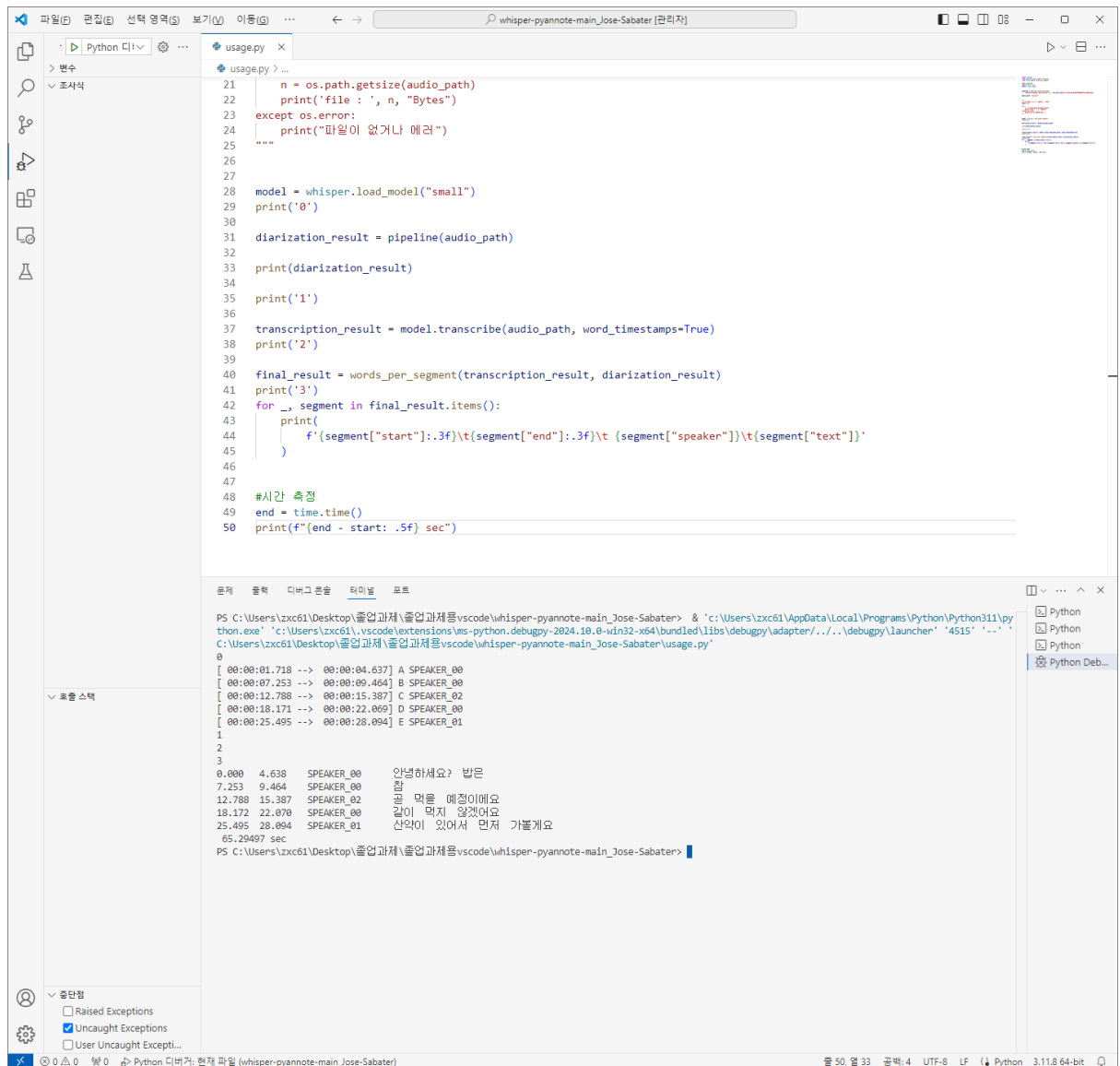
5.2.2.1. UIS-RNN은 fully supervised speaker diarization 방법을 제안한다.

5.2.2.2. 기존의 clustering 방식은 unsupervised기 때문에 supervised 된 speaker label을 유용하게 쓸 수 없었다. 우리는 speaker label이 있는 데이터를 가지고 있으므로 uis rnn을 사용하여 더 정확도 높은 결과를 기대할 수 있다.

6. 현재 과제 수행 내용 및 결과

6.1. 현재 과제 수행 내용

6.1.1. whisper-pyannote 예제 실행



```
21 n = os.path.getsize(audio_path)
22 print('file : ', n, "Bytes")
23 except os.error:
24     print("파일이 없거나 에러")
25
26
27
28 model = whisper.load_model("small")
29 print('0')
30
31 diarization_result = pipeline(audio_path)
32
33 print(diarization_result)
34
35 print('1')
36
37 transcription_result = model.transcribe(audio_path, word_timestamps=True)
38 print('2')
39
40 final_result = words_per_segment(transcription_result, diarization_result)
41 print('3')
42 for _, segment in final_result.items():
43     print(
44         f'{segment["start"]:.3f}\t{segment["end"]:.3f}\t {segment["speaker"]}\t{segment["text"]}'
45     )
46
47
48 #시간 측정
49 end = time.time()
50 print(f"end - start: .5f sec")
```

```
PS C:\Users\zxc61\Desktop\졸업과제\졸업과제음\vscode\whisper-pyannote-main_Jose-Sabater> & 'c:\Users\zxc61\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\zxc61\vscode\extensions\ms-python.debugpy-2024.10.0-win32-x64\bundle\libs\debugpy\adapter\..\..\debugpy\launcher' '4515' '-...' 'c:\Users\zxc61\Desktop\졸업과제\졸업과제음\vscode\whisper-pyannote-main_Jose-Sabater\usage.py'
0
[ 00:00:01.718 --> 00:00:04.637] A SPEAKER_00
[ 00:00:07.253 --> 00:00:09.464] B SPEAKER_00
[ 00:00:12.788 --> 00:00:15.387] C SPEAKER_02
[ 00:00:18.171 --> 00:00:22.069] D SPEAKER_00
[ 00:00:25.495 --> 00:00:28.094] E SPEAKER_01
1
2
3
0.000 4.638 SPEAKER_00 안녕하세요? 밥은
7.253 9.464 SPEAKER_00 참
12.788 15.387 SPEAKER_02 곧 먹을 예정이에요
18.172 22.070 SPEAKER_00 같이 먹지 않겠어요?
25.495 28.094 SPEAKER_01 산약이 있어서 먼저 가볼게요
65.29497 sec
PS C:\Users\zxc61\Desktop\졸업과제\졸업과제음\vscode\whisper-pyannote-main_Jose-Sabater>
```

¹whisper-pyannote 예제 실행

대사:

A: 안녕하세요 밥은 먹었나요

A: 날씨가 참 좋네요

B: 아니오, 곧 먹을 예정이에요

A: 그렇다면 같이 먹지 않겠어요?

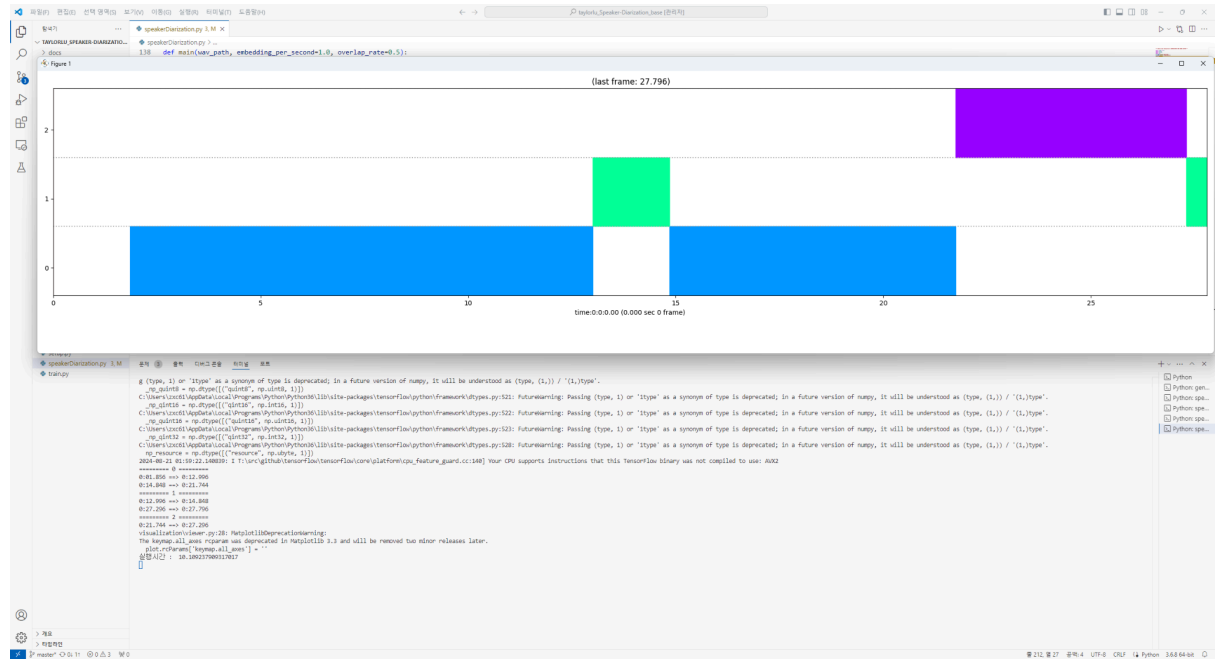
B: 저는 산약이 있어서 먼저 가볼게요

직접 녹음 하여 진행 하였다. 화자가 3명으로 나오고 한글 인식도 조금 저조하였다.

whisper(basic)만 따로 구동하였을 때는 인식후 text변환이 훨씬 매끄럽게 되었다.

whisper model load 가 small이라 인식률이 낮은 듯하다

6.1.2. vgg-speaker-recognition,UIS-RNN 혼합 예제 실행



²vgg-speaker-recognition 과 uisrnn이 적용되고 시각적 분석을 포함하는 예제

공백 시간때에도 화자가 생기거나 화자 개수 이상의 사람이 인식되었다. 발화 시간 외에 발화시간으로 인식되는 경우가 많았다. 공백 자체가 허용되지 않는 듯하다.

이 방법은 코드를 뜯어보니 speaker diarization부분은 uis-rnn만 사용하는 것 같아 이전의 예제와 본질적으로 같은 것 같아 보류. 또한 구버전이라 환경 설정에 애로사항이 많았다.

6.1.3. UIS-RNN 예제 실행

```
1.000000
1.000000
0.989362
1.000000
0.989796
1.000000
1.000000
0.991870
1.000000
0.990654
1.000000
1.000000
1.000000
1.000000
1.000000
1.000000
0.992000
1.000000
1.000000
=====
25
실행 시간 : 165.42289423942566
```

```
Performance:
averaged accuracy: 0.996880
accuracy numbers for all testing sequences:
1.000000
1.000000
0.989362
0.989362
1.000000
1.000000
0.989583
1.000000
1.000000
0.989362
1.000000
0.989796
1.000000
1.000000
0.991870
1.000000
0.990654
1.000000
1.000000
1.000000
1.000000
0.992000
1.000000
1.000000
=====
실행 시간 : 670.804033279419
```

³uisrnn 예제. 왼쪽은 iteration 없이 실행한 결과. 오른쪽은 iteration을 포함해 실행한 결과.

매 실행마다 iteration을 수행 할 수는 없어 처음 iteration을 수행한 뒤 train된 model을 npy로 저장하여 읽어오도록 하였다. 첫 실행에서는 670초 가량이 걸렸지만 첫 실행 이후 model로 npy파일을 만들고 save한 후 다음부터 load하는 코드를 추가해주었다.

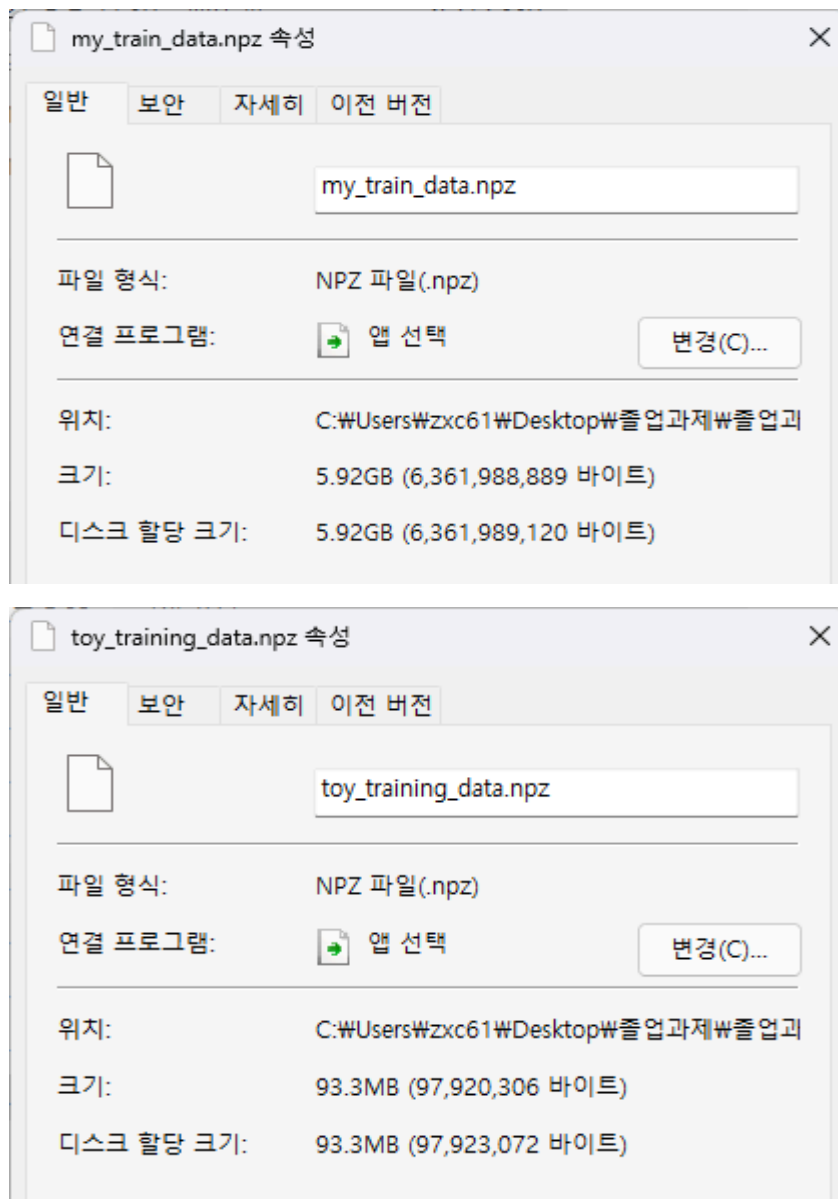
그렇게 하니 test case가 25개에 실행시간은 165초가 걸렸다. 정확도는 simple 한 train case와 test case 여서 99%가 넘게 나왔다.

임베딩 되어있는 train data를 학습을 위한 값으로 받는데 train data 내의 train sequence는 numpy float64였다. wav 파일의 utterance에 관한 정보로 추측된다.

cluster id 는 화자 번호가 1차원 배열로 나열되어있었는데 train sequence와 배열의 길이가 같다.

일정한 frame으로 자르고 해당하는 sequence에 대한 화자 번호인 듯하다.

위와 같은 이유로 UIS-RNN에 대해 중점적으로 연구하기로하였다.



UIS-RNN에서 기본적으로 제공하는 toy_training_data.npz는 simple한 데이터이기 때문에 직접 aihub에서 라벨링과 wav 파일을 적절히 변조하여 npz파일을 만들었다. 다만 포맷상의 문제인지 아니면 우리가 만든 데이터가 너무 컸는지 npz의 용량이 예측보다 크게 나왔다. 원천 데이터(wav, labeling json)의 폴더가 5.26GB임을 감안해도 적절한 변형인지는 확실치 않다.

```
155
>156 n_speak = int(float(dic['speaker_id']))
```

```
예외가 발생했습니다. ValueError ×
could not convert string to float: '?'
```

```
could not convert string to float: '?'
```

```
File "C:\Users\zxc61\Desktop\졸업과제\졸업과제용vscode\suan_diarization\tmpDiarization
n_speak = int(float(dic['speaker_id']))
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

```
ValueError: could not convert string to float: '?'
```

```

157 |         "end": 53
      |     },
      |     "term": null
      | },
      | {
      |     "id": "DGBBB21000004.1.1.55",
      |     "start": "271.768",
      |     "end": "281.606",
      |     "speaker_id": "?",
      |     "speaker_role": "?",
      |     "form": "전시 등에 유사시를 대비한 예비 병력의 훈련
도입됐고요. 공모를 마친 다음날부터 /(bgm)",
      |     "original_form": "전시 등에 유사시를 대비한 예비 병력
도입됐고요. 공모를 마친 다음날부터",
      |     "environment": "배경 음악",
      |     "isldiom": false,
      |     "hangeulToEnglish": null,
      |     "hangeulToNumber": [
      |         f
    
```

aihub에서 받은 Json 데이터중 labeling에 적절하지 않은 값이 있어 이를 예외처리 해주었다.

The image shows two screenshots of a Python debugger window. The top screenshot shows the execution of a script named 'tempDiarization.py'. The code includes a 'start' function and a list of numerical values. The bottom screenshot shows the same code with a warning message: 'Warning: transition_bias cannot be correctly estimated from a concatenated sequence; train_sequences will be treated as a single sequence. This can lead to inaccurate estimation of transition_bias. Please, consider estimating transition_bias before concatenating the sequences and passing it as argument.'

```
hon.debugpy-2024.10.0-win32-x64\bundle\libs\debugpy\adapter\...\debugpy\launcher '12917' '--' 'C:\Users\zxc61\Desktop\출업과제\출업과제용\vscode\suan_diarization\tempDiarization.py'
start
aaaaaaaaaaaaaaaa
(47350, 256)
(47350, 256)
(47350,)
[[ 0.0011389  0.03821772 -0.00648595 ... -0.02149146  0.00329872
 -0.10071674]
 [-0.12522156  0.12027011 -0.11167086 ... -0.00742709  0.09596947
 -0.00112058]
 [ 0.00149188  0.03890027 -0.00106493 ...  0.01663821 -0.02963326
  0.0673404 ]
 ...
 [-0.02248289  0.04978857  0.02713239 ...  0.13398466 -0.00084103
 -0.04162925]
 [ 0.01669886 -0.00630479 -0.08952245 ...  0.07478793  0.06175092
 -0.039444 ]
 [ 0.04728676  0.09279967  0.00040307 ...  0.08652191 -0.02060125
 -0.08981664]]
npz done : 0.10953998565673828
c:\Users\zxc61\AppData\Local\Programs\Python\Python311\Lib\site-packages\torch\storage.py:414: FutureWarning: You are using `torch.load` with `weights_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for more details). In a future release, the default value for `weights_only` will be flipped to `True`. It is recommended to explicitly set `weights_only=True` to silence this warning.
npz done : 0.25510215759277344
1
Warning: transition_bias cannot be correctly estimated from a concatenated sequence; train_sequences will be treated as a single sequence. This can lead to inaccurate estimation of transition_bias. Please, consider estimating transition_bias before concatenating the sequences and passing it as argument.
```

첫번째 사진은 pre-trained data이고, 두 번째 사진은 AI-hub에서 받은 데이터를 가지고 우리가 만든 트레이닝 데이터이다. 값의 크기가 많이 달라 문제를 겪고 있다. wav 파일을 ndarray로 변환하는데에 있어 다른 방법으로 변환한 듯 하다.

배열 크기가 너무 커서 생긴 문제인지 판별해 보았는데, toy_training_data보다 배열의 크기가 3배 크기에 불과한데 밤새 돌려도 model fit가 수행되지 않았다. 배열 값의 문제로 예상된다.