104 – Fake Voice

Team Information

Team Name: kimbabasaksaksak

Team Member: Jaeheon Kim, Donghyun Kim, Soyoung Yoo, Minhee Lee

Email Address: uaaoong@gmail.com

Instructions

Description Here is an audio file that supposedly added a fake voice intentionally. As a digital forensic investigator, solve the following questions.

Target	Hash (MD5)
original.wav	e3b2f872e7a93f1e39d831b07b88f3ae
modified.wav	5f90258daf2fa7dbf758854f3e6f5bd4

Questions

1) At what time does the fake voice play? (10 points)
2) Provide evidence to support the answer. (90 points)

Teams must:

- Develop and document the step-by-step approach used to solve this problem to allow another examiner to replicate team actions and results.
- Specify all tools used in deriving the conclusion(s).

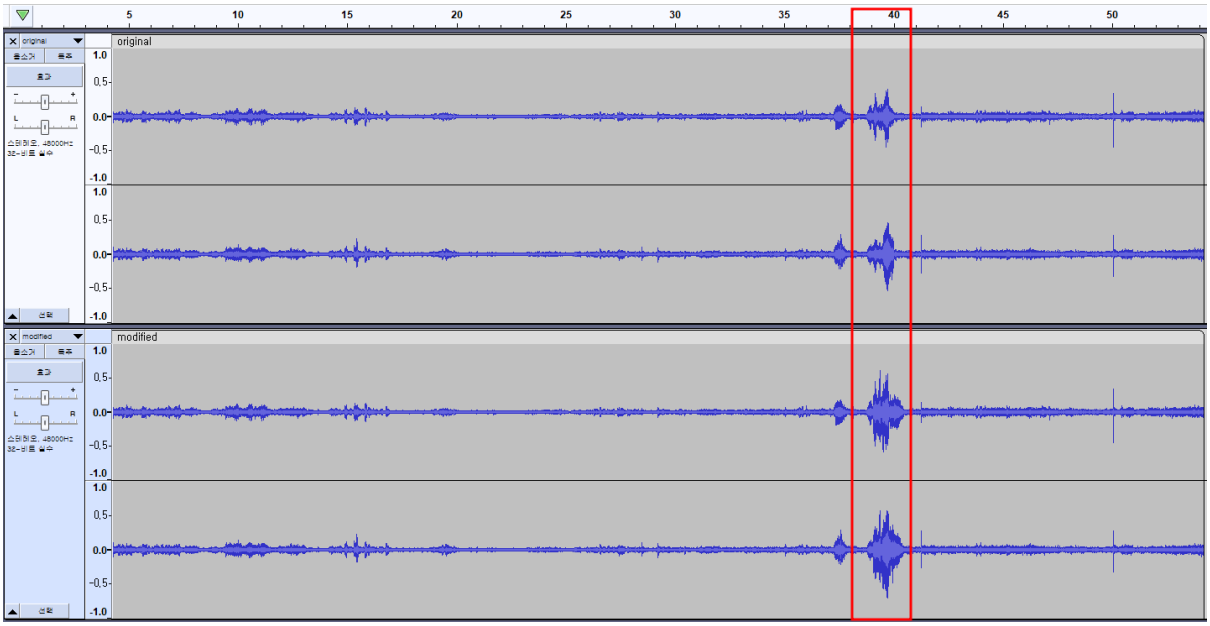
Tools used:

Name:	Audacity	Publisher:	Audacity Team
Version:	3.3.3		
URL:	https://www.audacityteam.org		

Step-by-step methodology:

Q1. At what time does the fake voice play? (10 points)

Audacity를 사용하여 음성 파일들을 확인하면, 'original.wav' 파일과 'modified.wav' 파일의 파형이 다른 부분이 존재하는 것을 확인할 수 있다. 즉, 변조된 음성은 'modified.wav' 파일의 약 38초부터 41초 사이에 존재한다.

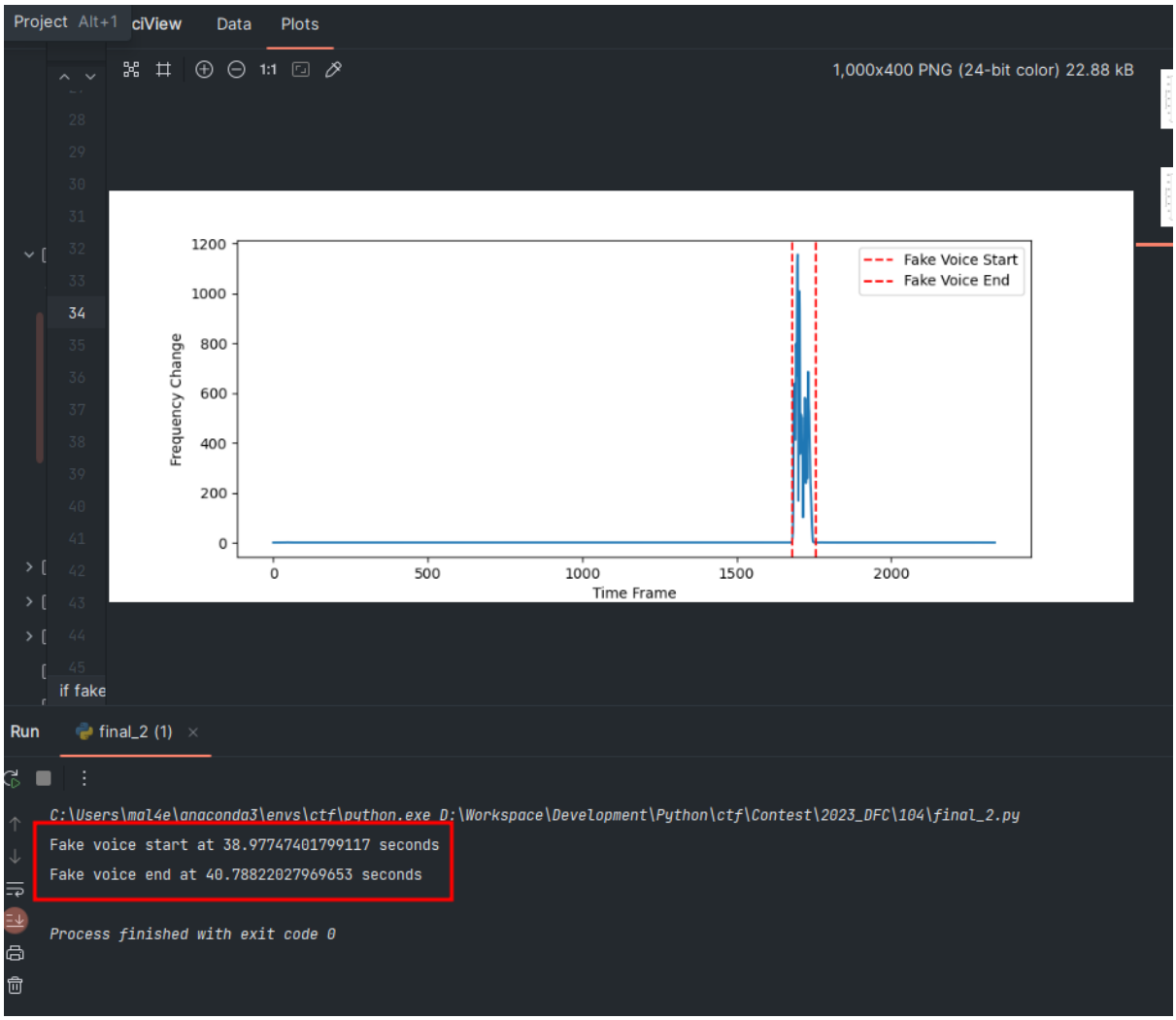


음성이 변조된 정확한 시간을 확인하기 위하여 Short-time Fourier Transform (STFT) 기법을 사용하고자 한다. STFT 기법을 사용하는 이유는 주파수와 시간 정보를 동시에 다룰 수 있기 때문이다. 음성 변조나 수정된 부분은 주로 시간과 주파수 도메인에서의 특징적인 변화를 가지고 있을 수 있다. STFT 기법은 이러한 변화를 분석하여 음성 신호에서의 주파수 성분 변화 및 시간적인 패턴을 잡아내는 데 유용하다.

STFT 기법을 사용하여 분석한 결과, 'modified.wav' 파일에서 변조된 음성이 존재하는 부분은 다음과 같다. 상세한 분석 과정과 코드는 2번 문제 풀이에 서술하였다.

- Fake voice start: 38.97747401799117 seconds

- Fake voice end: 40.78822027969653 seconds



Q2. Provide evidence to support the answer. (90 points)

STFT 기법을 사용하여 변조된 음성이 존재하는 부분을 찾으려면, 우선 적절한 임계값을 찾아야 한다. 아래 코드는 적절한 임계값을 찾아내기 위한 코드이다.

```
import numpy as np
import librosa

# 오디오 파일 불러오기
original, sr_original = librosa.load('original.wav')
fake_mix, sr_fake_mix = librosa.load('modified.wav')

# 스펙트럼 분석 (Short-time Fourier Transform)
D_original = np.abs(librosa.stft(original))
D_fake_mix = np.abs(librosa.stft(fake_mix))

# 주파수 변화 감지
threshold = 1 # 적절한 임계값 설정
time_frame_diff = np.sum(np.abs(D_original - D_fake_mix), axis=0)
indices_where_exceeds_threshold = np.where(time_frame_diff >
threshold)[0]
fake_voice_start = indices_where_exceeds_threshold[0] if
indices_where_exceeds_threshold.size > 0 else None

# 최대값 및 평균값을 분석하여 적절한 임계값을 threshold 변수에 지정
print(f'diff value of fake_voice_start index:
{time_frame_diff[fake_voice_start]}')
print(f'max value of diff until fake_voice_start index:
{np.max(time_frame_diff[:fake_voice_start])}')
print(f'average value of diff until fake_voice_start index:
{np.average(time_frame_diff[:fake_voice_start])}')
```

위 코드의 설명은 다음과 같다.

1. librosa 라이브러리: librosa는 오디오 및 음악 분석을 위한 파이썬 라이브러리이다. 여기서는 librosa.load 함수를 사용하여 오디오 파일을 불러오고, librosa.stft 함수를 사용하여 짧은 시간 동안의 주파수 변화를 스펙트럼 분석을 통해 계산한다.

2. 스펙트럼 분석 (Short-time Fourier Transform): 스펙트럼 분석은 시간 도메인의 신호를 주파수 도메인으로 변환하는 기술이다. 코드에서는 librosa.stft를 사용하여 오디오 파일의 스펙트럼을 분석하고, np.abs 함수를 사용하여 절대값을 취한 복소수 스펙트로그램을 계산한다.

3. 주파수 변화 감지: 두 개의 스펙트로그램을 비교하여 주파수 변화를 감지한다. np.sum(np.abs(D_original - D_fake_mix), axis=0) 코드는 두 스펙트로그램 간의 차이를 계산하고, 각 시간 프레임에서의 차이의 합을 계산한다. 이 값은 주파수 변화 정도를 나타낸다.

4. 임계값 설정: threshold 값을 사용하여 주파수 변화가 실제 음성 변화인지 아니면 노이즈인지를 구분한다. 주파수 변화가 임계값을 초과하면, 해당 시간 프레임에서 가짜 음성이 시작될 가능성이 높다.

위 코드에서 threshold(임계값)를 1로 설정하여 실행한 결과는 다음과 같다.

- diff value of fake_voice_start index: 2.230407238006592

- max value of diff until fake_voice_start index: 0.6315811276435852

- average value of diff until fake_voice_start index: 0.13858655095100403

위 결과를 해석하면 다음과 같다.

- 2.230407238006592: 임계값 1을 넘는 첫 번째 값

- 0.6315811276435852: 2.230407238006592 값 전까지의 최대값

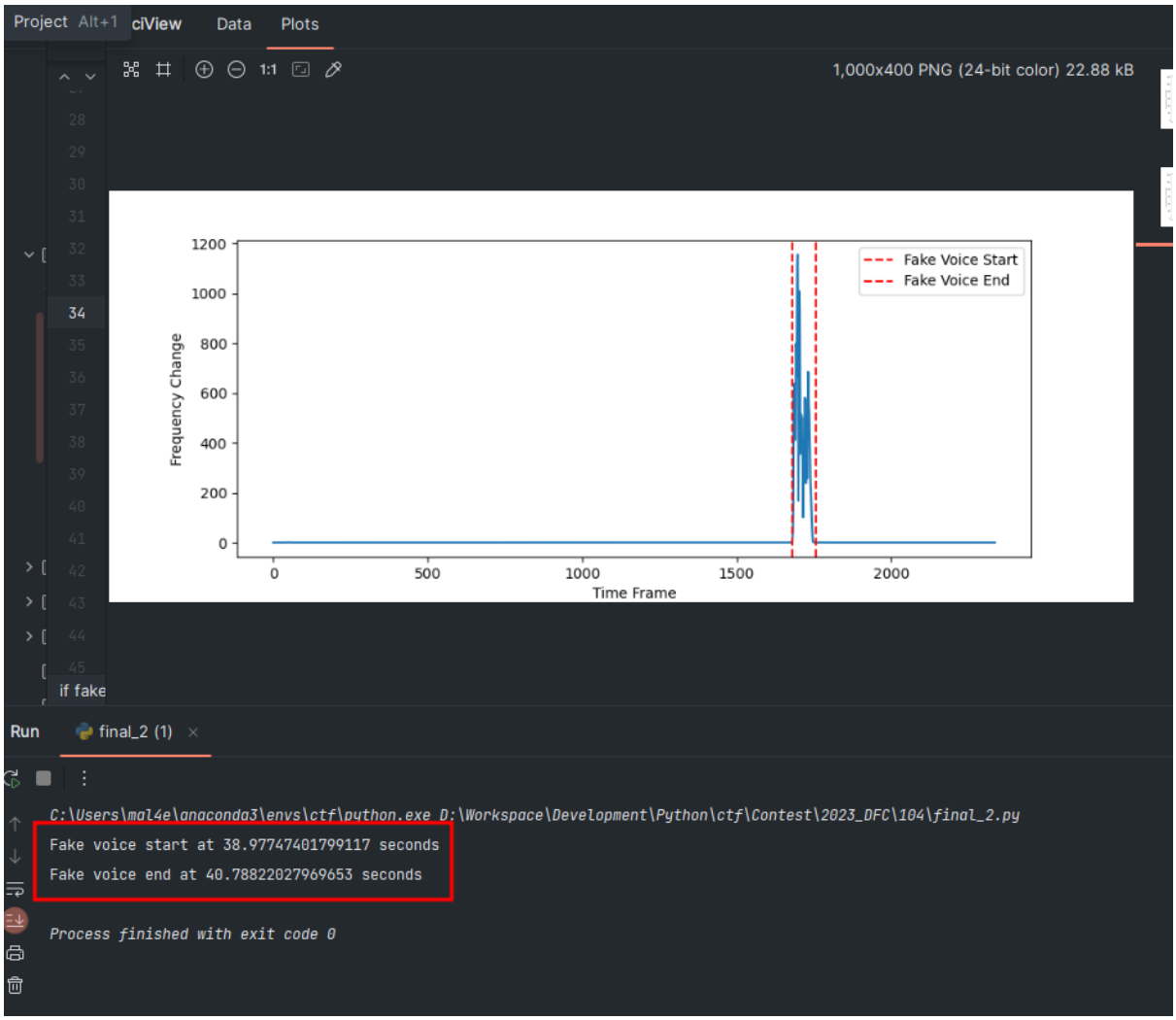
- 0.13858655095100403: 2.230407238006592 값 전까지의 평균값

즉, 임계값 1은 평균값인 0.13858655095100403와 많은 차이가 나며, 최대값인 0.6315811276435852와도 적절한 차이가 나는 값이므로 변조된 음성을 찾기 위한 좋은 임계값이라고 할 수 있다. 아래 코드는 임계값을 1로 설정하여, 변조된 음성이 존재하는 시간대를 찾는 코드이다.

```
import numpy as np
import matplotlib.pyplot as plt
import librosa

# 오디오 파일 불러오기
original, sr_original = librosa.load('original.wav')
fake_mix, sr_fake_mix = librosa.load('modified.wav')

# 스펙트럼 분석 (Short-time Fourier Transform)
D_original = np.abs(librosa.stft(original))
D_fake_mix = np.abs(librosa.stft(fake_mix))

# 주파수 변화 감지
threshold = 1 # 적절한 임계값 설정
time_frame_diff = np.sum(np.abs(D_original - D_fake_mix), axis=0)
indices_where_exceeds_threshold = np.where(time_frame_diff >
threshold)[0]
```

```
fake_voice_start = indices_where_exceeds_threshold[0] if
indices_where_exceeds_threshold.size > 0 else None

# 시작 시각 계산 (fake_voice_start 가 None 이 아닐 때만 계산)
if fake_voice_start is not None:
    start_time_seconds = fake_voice_start * (len(original) / sr_original)
/ len(time_frame_diff)
    print(f'Fake voice start at {start_time_seconds} seconds')
else:
    print('Threshold not exceeded')

# 주파수 변화 감지
threshold = 1 # 적절한 임계값 설정
indices_where_exceeds_threshold =
np.where(time_frame_diff[fake_voice_start:] <= threshold)[0]
fake_voice_end = indices_where_exceeds_threshold[0] + fake_voice_start
if indices_where_exceeds_threshold.size > 0 else None

# 종료 시각 계산 (fake_voice_end 가 None 이 아닐 때만 계산)
if fake_voice_end is not None:
    start_time_seconds = fake_voice_end * (len(original) / sr_original) /
len(time_frame_diff)
    print(f'Fake voice end at {start_time_seconds} seconds')
else:
    print('Threshold not exceeded')

# 시각화
plt.figure(figsize=(10, 4))
plt.plot(time_frame_diff)
plt.axvline(fake_voice_start, color='r', linestyle='--', label='Fake
Voice Start')
plt.axvline(fake_voice_end, color='r', linestyle='--', label='Fake Voice
End')
plt.legend()
plt.xlabel('Time Frame')
plt.ylabel('Frequency Change')
plt.show()
```

위 코드는 감지된 주파수 변화가 임계값을 초과하는 지점을 기준으로 변조된 음성의 시작 및 종료 시각을 계산한다. 또한, 'matplotlib' 라이브러리를 사용하여 주파수 변화를 시각화한다. plt.plot 함수로 주파수 변화 데이터를 그래프로 표시하고, plt.axvline 함수로 변조된 음성 시작 및 종료 지점을 빨간 점선으로 표시한다. 실행결과는 다음과 같다.



즉, STFT 기법을 사용한 분석 결과에 따라, 다음과 같은 결론을 낼 수 있다.

- Fake voice start: 38.97747401799117 seconds

- Fake voice end: 40.78822027969653 seconds