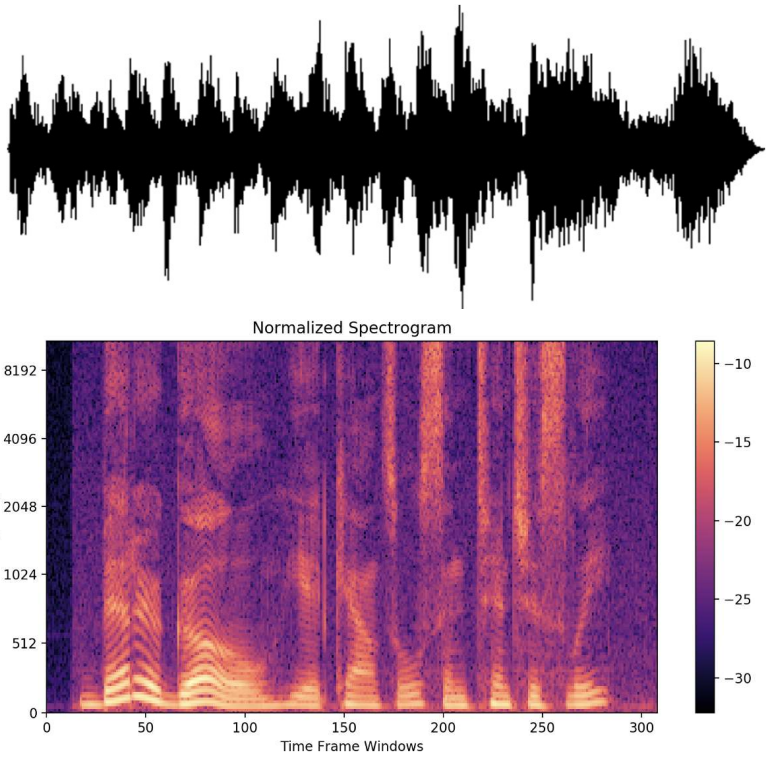
NPEX 2022 Deep learning for Speech

Day 1 Practice:
Speech Feature extraction and Metric

2022-06-22

Human Speech as Waveform



I LOVE SPEECH

1. Waveform (음성 파일)
2. Utterance (발화 텍스트)
3. Alignment (위치 정렬)

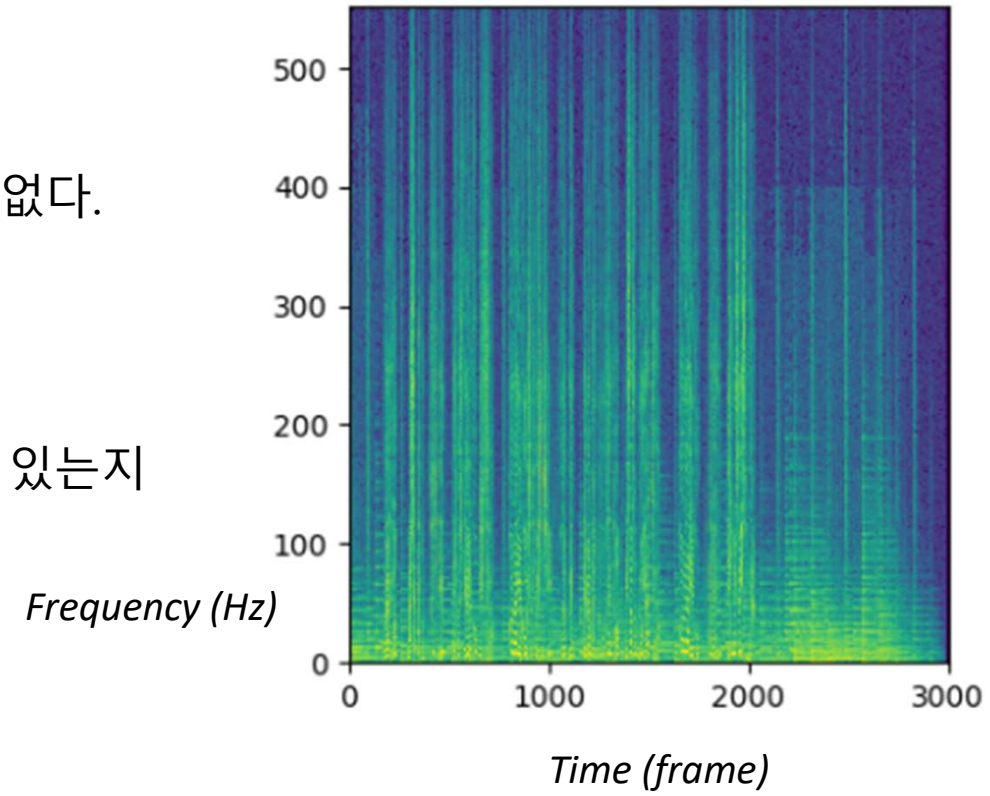
- **Styles**: reading, conversational, spontaneous, multi-speaker, command & control, keyword
- **Variances**: disfluency, stuttering, mic quality, channels, far field, reverb, echo, accents, Lombard effect, gender, age, locale, ...
- **Noises**: outdoor, room, school, subway, car, ...

같은 데이터, 세 가지 다른 표현방식

Waveform – Spectrogram – Text

Spectrogram

- **Sampling rate**
  - 몇 개의 프레임이 1초에 들어 있는가?
  - Ex) 10초 길이 음성, 16kHz sampling
  - Nyquist Theorem:
    최고 주파수의 2배 이상으로 샘플링해야 손실이 없다.
- **STFT (Short-Time Fourier Transform)**
  - 짧은 구간에 대해 FFT를 적용
  - 주파수(frequency) 성분이 각각 얼마나 포함되어 있는지
  - Ex) 25ms window, 10ms stride
  - 1개 window = 1개 frame
  - 가청 주파수: ~20kHz
  - 음성인식에 보통 사용되는 주파수: ~ 8kHz

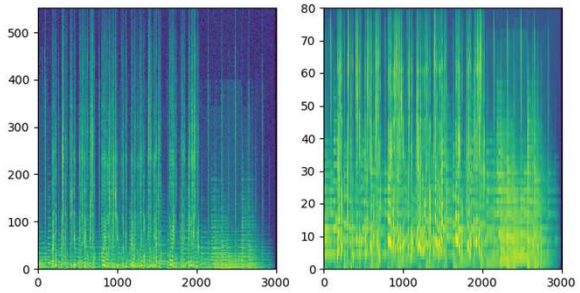


Audio Pre-processing

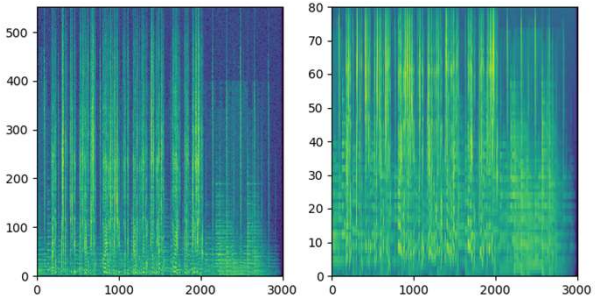
- **Pre-emphasis**: 연속적인 저주파 성분을 제거해 더 특징을 선명하게 잡아냄

$$y(t) = x(t) - \alpha x(t-1), \qquad \alpha = 0.95 \text{ or } 0.97$$

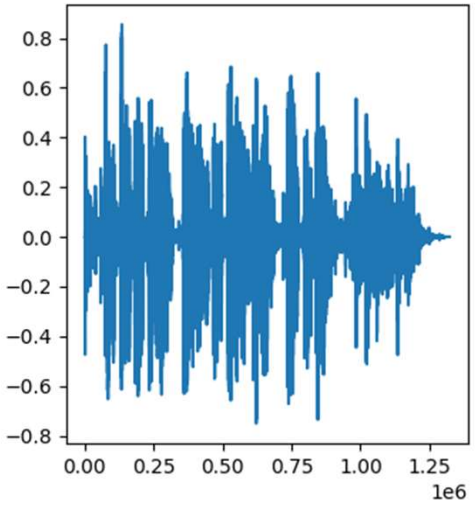
Original



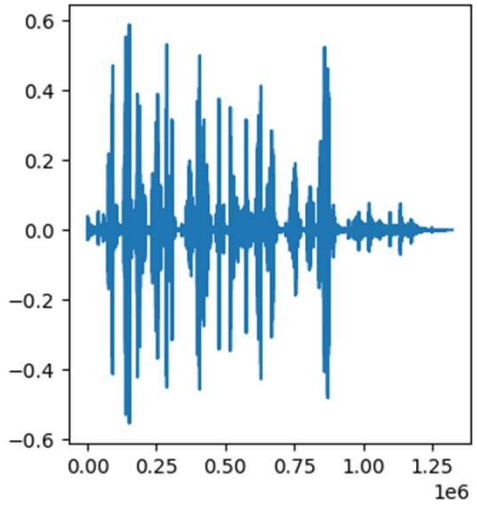
After Pre-emphasis



Original



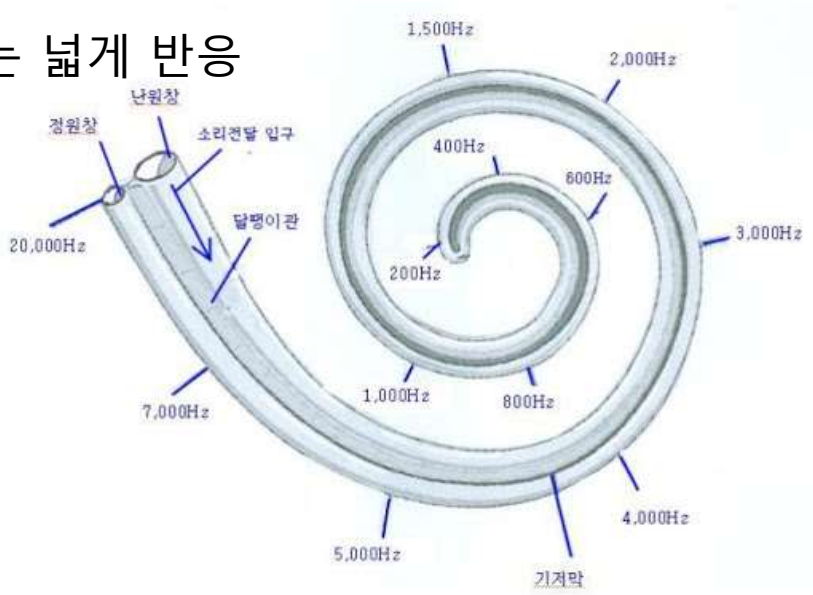
After Pre-emphasis



- Silence removal: 중간에 나올 수 있는 묵음(소리가 없는 부분)을 제거
- Dithering: 16-bit/32-bit sampling 과정에서 생길 수 있는 양자화 노이즈를 보정

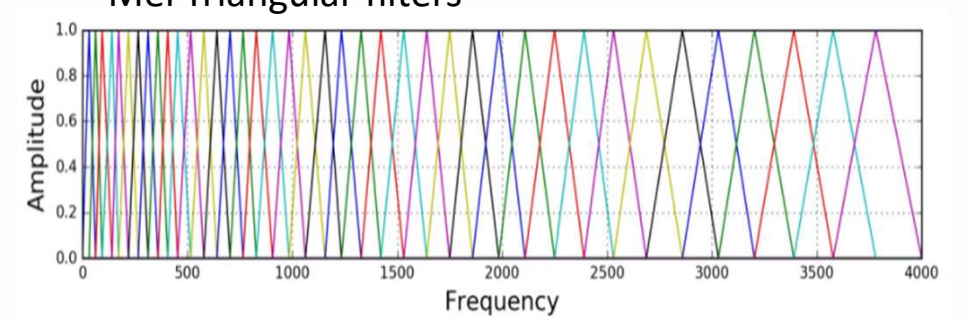
Mel-Spectrogram

- **Mel Spectrogram**
  - 달팽이관: 저주파수에는 촘촘하게(민감하게), 고주파수에는 넓게 반응
  - Mel Filter Bank를 이용해 Spectrogram에 적용
  - $\mathrm{M(mel)} = 2595 \log_{10}\left(1 + \frac{\mathrm{F(Hz)}}{700}\right)$
  - $F(Hz) = 700\left(e^{\frac{m}{1127}} - 1\right)$

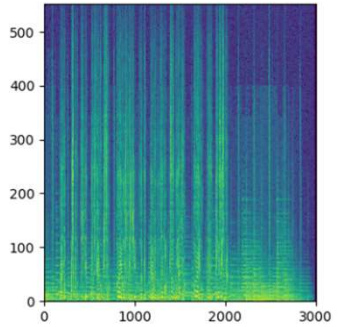


- **Log-Mel-spectrogram**
  - 사람과 유사하게 데시벨 (dB) 단위로 변환
  - $dB = 10 \log_{10} A$

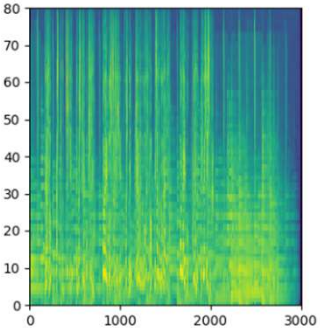
Mel Triangular filters



Spectrogram



Mel-Spectrogram



Tokenizer

- **Tokenizer**
  컴퓨터가 이해할 수 있는 형태(=숫자)로 자연어 문장을 변환
  1. 최소단위로 쪼개고
  2. 각 단위에 맞는 숫자(index)열로 변환

- Word tokenizer = 단어가 최소단위
- Grapheme tokenizer (=English character tokenizer): 글자가 최소단위

$$\{"\_", A, B, C, D, \ldots, Z, '\} \ (28)$$

  - Ex) I HAVE A CAT → [9, 0, 8, 1, 22, 5, 0, 1, 0, 3, 1, 20]

- Sub-word (or, word-piece) tokenizer: 부분단어가 최소단위
  - ae, the, ish, …
  - 실제로 가장 많이 사용되는 tokenization

Edit Distance

- **Edit distance (= Levenshtein distance)**

두 문자열 A, B가 있을 때, A에 최소 몇 번의 연산을 거쳐야 B와 동일하게 만들 수 있는가?

- Edit 연산의 종류
  1. Insertion ex) to -> too (o 삽입)
  2. Deletion ex) two -> to (w 삭제)
  3. Substitution ex) sea -> see (a -> e로 대체)

  - Edit distance를 구하는 방법 중 가장 쉬운 방법 : Wagner-Fischer Algorithm
  - 왜 음성인식은 Edit distance 를 지표로 사용하는가? → 예측한 답과 정답의 길이가 다르기 때문.

$$\text{Error Rate} = \frac{|\text{Insertion}| + |\text{Deletion}| + |\text{Substitution}|}{|\text{Reference}|}$$

Character Error Rate (CER), Word Error Rate (WER)

- **Character Error Rate** (띄어쓰기 무시)

	I	A	M	A	R	M	Y
I							
M							
R							
M							
Y							

CER = 2/7

- **Word Error Rate**

	WE	ARE	GOOD	AT	KOREAN	AND	ENGLISH
WER							
GOOD							
AT							
KOREN							
ENGLISH							

WER = 4/7

Character Error Rate (CER), Word Error Rate (WER)

- **Character Error Rate** (띄어쓰기 무시)

	I	A	M	A	R	M	Y
I	**0**	**1**	2	3	4	5	6
M	1	2	**1**	**2**	3	4	5
R	2	2	2	2	**2**	3	4
M	3	3	2	3	3	**2**	3
Y	4	4	3	3	4	3	**2**

- 비교하는 두 문자가 같은 경우
  : 왼쪽 위에 있는 숫자 그대로
- 비교하는 두 문자가 서로 다른 경우
  : 왼쪽, 왼쪽 위, 위에 있는 숫자 중 가장 작은 숫자 + 1

1. 왼쪽 숫자 + 1: Insertion
2. 왼쪽 위 숫자 + 1: Substitution
3. 위쪽 숫자 + 1: Deletion

CER = 2/7

- **Word Error Rate**

	WE	ARE	GOOD	AT	KOREAN	AND	ENGLISH
WER	**1**	**2**	3	4	5	6	7
GOOD	2	2	**2**	3	4	5	6
AT	3	3	3	**2**	**3**	4	5
KOREN	4	4	4	3	3	**4**	5
ENGLISH	5	5	5	4	4	4	**4**

WER = 4/7

Thank You!

NPEX 2022 Deep learning for Speech