NPEX 2022 Deep learning for Speech

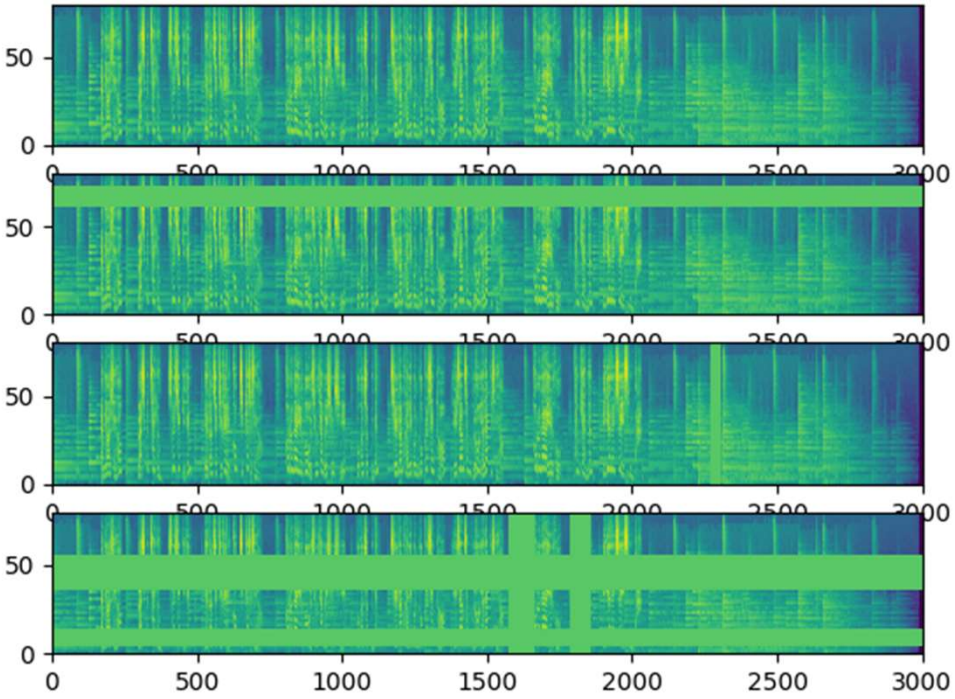
Day 3 Practice:
SpecAugment,
CTC Decoding – Greedy & Beamsearch

2022-06-24

SpecAugment

Input Augmentation

- Augmentation 은 모델이 일반화(generalization)를 더 잘 하게 하고 더 강인하게(robust) 도와줌
  - 가장 강력한 augmentation 중 하나는 입력을 변경하는 것
- 음성 모델의 입력 = **Spectrogram**



Original Mel-spectrogram

Frequency masking

Time masking

Frequency + Time masking

Audio augmentation methods

1) Noise Injection
2) Shifting
3) Speed Perturbation

Spectrogram augmentation

1) Time Warping
2) Frequency Masking
3) Time Masking

Conditional Independence in CTC

Goal of decoding

$$\text{maximize } P(Y|X)$$

- 음성 입력이 주어졌을 때 가장 확률이 높은 텍스트를 찾아내야 함.
- 문장 W가 나올 확률은 다음과 같이 Bayes rule에 의해 쪼개짐.

$$P(W) = P(w_1, w_2, \dots w_N) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2}) \dots P(w_N|w_{1:N-1}) = \prod_{i=1}^{N} P(w_i|w_{1:i-1})$$

- CTC는 conditional independence 가정을 하기 때문에, 다음 식으로 근사함.
  - 즉, 다음 단어가 나올 확률이 이전 단어가 뭐가 나왔는지에 따라 바뀌지 않음.
  - 성능에는 안 좋지만, 동시에 병렬처리 되는데 최적 (non-autoregressive)

$$P(W) = P(w_1, w_2, \dots w_N) = P(w_1)P(w_2)P(w_3) \dots P(w_N) = \prod_{i=1}^{N} P(w_i)$$

CTC Greedy Decoding

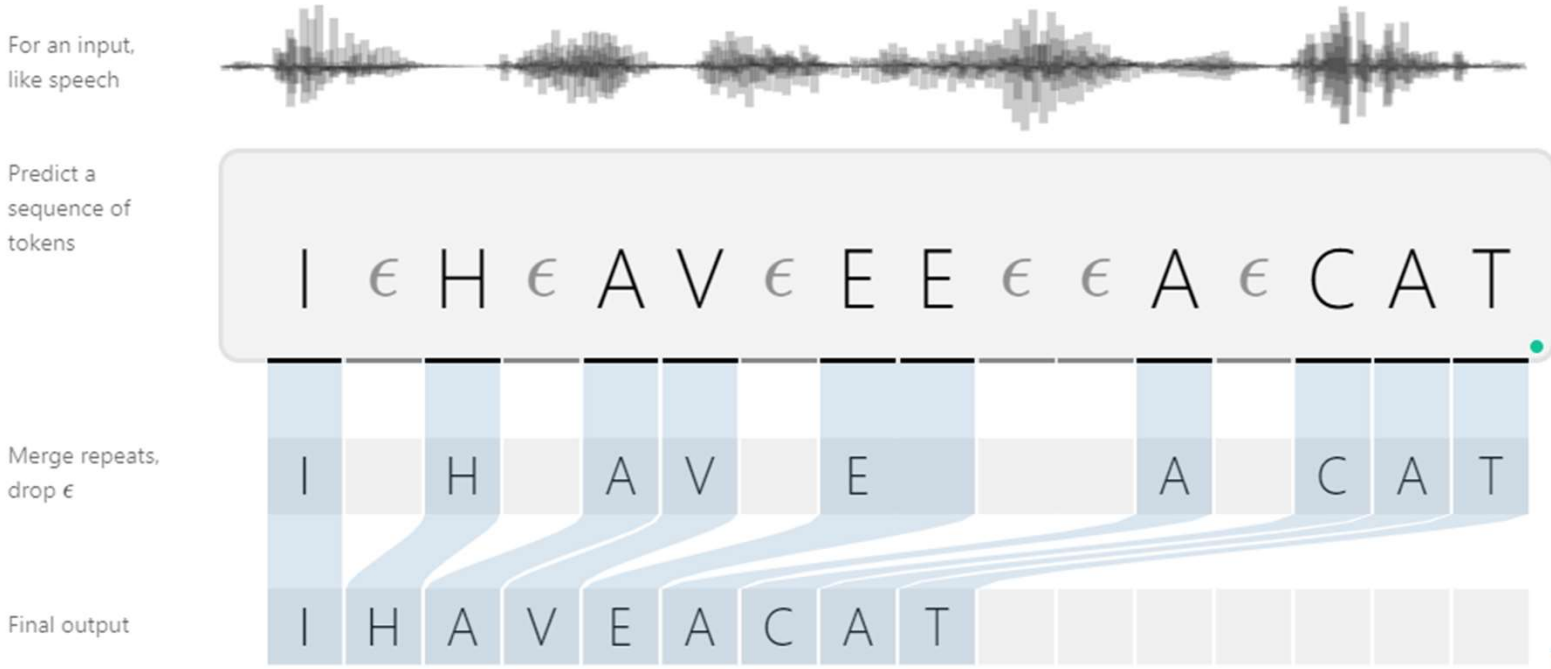
Greedy Decoding = Argmax of each frame

1. 매 프레임에서 가장 확률이 높은 단어를 고름.

$$A^* = \text{argmax}x_A \prod_{t=1}^{T} p_t(a_t|X)$$

2. 만들어진 문장에서 collapse & remove
   - Collapse: 동일한 단어가 반복되면 하나로 합침 / Remove: <B> 를 없앰

https://distill.pub/2017/ctc/



CTC Greedy Decoding

Conditional independence + Greedy decoding

- 그때그때 가장 확률이 높은 단어를 선택
- 사전에 없는 단어, 고유명사, 발음이 비슷한 조합, 문법적으로 이상한 문장, …

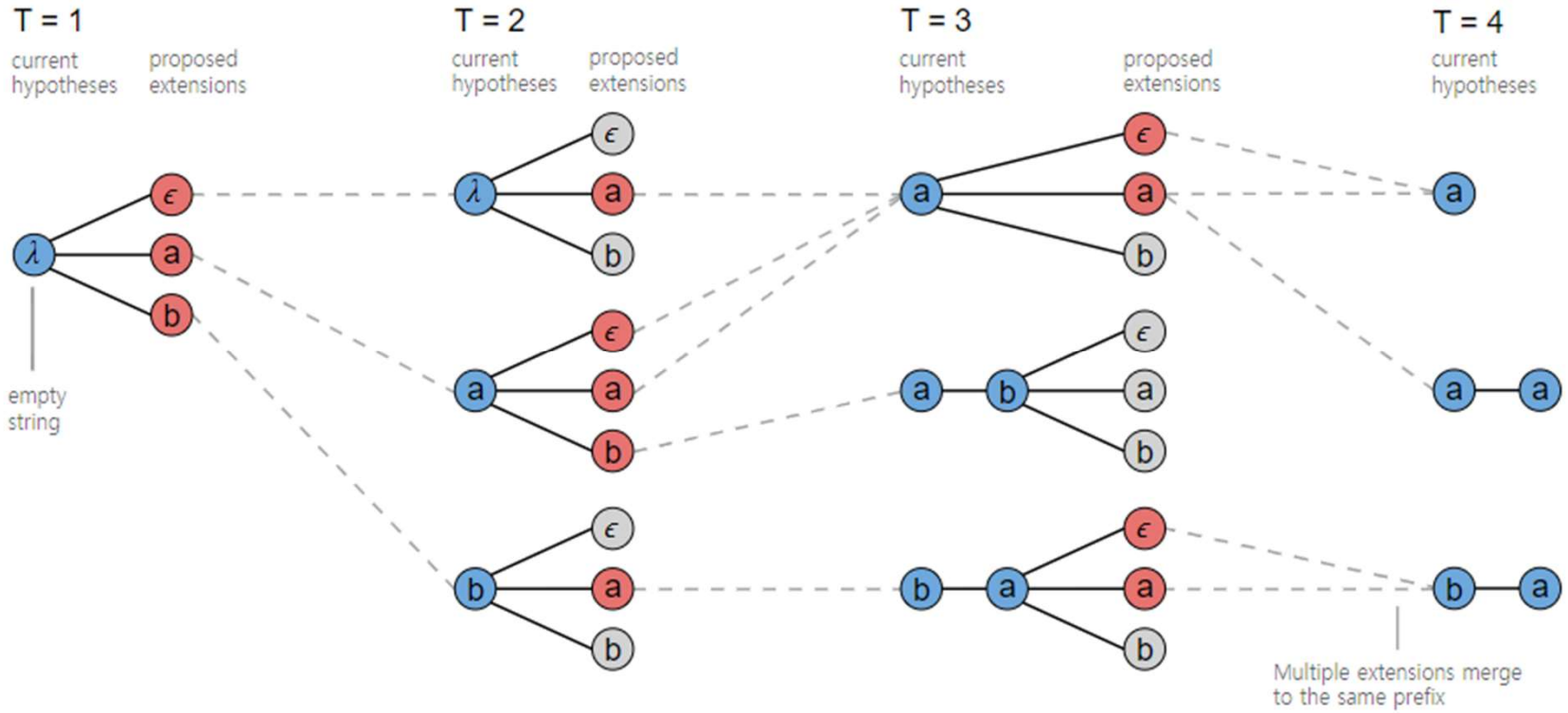
정답: this last remark he made at a venture for he had naturally not devoted any supposition
예측: this last remark he made that adventure for he had naturally not devoted in his supposition

정답: a headlong pardon on the eve of a bye election with threats of a heavy voting defection
예측: i had long pardon on the eve of a violection with threats of a heavy voting affection

CTC Beamsearch Decoding

Beamsearch = Keep track of highest probability prefix

- 한 프레임씩 봐 가면서 가장 확률 높은 Top-B개 prefix 후보군을 유지
- B = Beam width (beam size)
- 확률 값은 이미 다 계산되어 있지만, collapse 때문에 beam search 가 성립.



CTC Beamsearch Decoding

Prefix should keep two probabilities: **[END_BLANK, END_NO_BLANK]**

- 지금까지의 prefix가 언제 확장되는가?

Ex) AB = (0.45, 0.25)

1. 다음 frame에서 beam candidate 확장 (순서대로 _, A, B)

AB_:	P(_) = 0.15	AB:	(0.45 x 0.15 + 0.25 x 0.15, 0)	= (0.105, 0)
ABA:	P(A) = 0.30	ABA:	(0, 0.45 x 0.30 + 0.25 x 0.30)	= (0, 0.21)
ABB:	P(B) = 0.55	ABB:	(0, 0.45 x 0.55)	= (0, 0.2475)
	+ update AB	AB:	(0.45 x 0.15 + 0.25 x 0.15, 0 + 0.25 x 0.55)	= (0.105, 0.1375)

2. Collapse

AB_	→	(0.105, 0.1375) = 0.2425
ABA	→	(0, 0.21) = 0.21
ABB	→	(0.105, 0) = 0.105

3. Keep Top-B(1)

   AB = (0.105, 0.1375)

N-gram Language Model

다음 단어가 나올 확률을 최대 **이전 (N-1)개까지 보고 결정**

- Bi-gram

$$P(W) = P(w_1, w_2, \dots w_N) = P(w_1)P(w_2|w_1)P(w_3|w_2) \dots P(w_N|w_{N-1})$$

- Tri-gram

$$P(W) = P(w_1, w_2, \dots w_N) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2}) \dots P(w_N|w_{N-2:N-1})$$

- 각각의 확률은 corpus 에서 사전에 미리 계산해 놓을 수 있음.

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_n)} \text{ (bi-gram)}$$

  - C = count
  - 더 많이 볼 수록 더 정확해지지만…
  - N-gram 조합이 더 희귀해질 것: 아주 많은 데이터가 있어야 정교해짐
  - 실제로는 못 본 조합을 위해 여유 확률을 남겨두고 (discounting), 특정 조합으로만 등장하는 것을 보정함 (ex: Kneser-Ney Smoothing)
- 가장 보편적으로 사용되는 LM library: *KenLM (https://github.com/kpu/kenlm)*

Beamsearch with Shallow Fusion

Use external LM to rescore the prefix

$$\text{maximize } P(Y|X)P(Y)^{\alpha}|Y|^{\beta}$$

- $P(Y|X)$: 지금까지의 음성에서 prefix 가 나올 확률 (AM)
- $P(Y)^{\alpha}$: 만들어진 prefix 가 실제 텍스트에서 나올 확률 (LM)
- $|Y|^{\beta}$: Prefix가 과하게 짧게 나오지 않는지에 대한 보정 (length penalty)
  (길이가 길면 확률 곱으로 확률이 낮아져 살아남지 못할 수 있음)

Language	Architecture	Dev no LM	Dev LM
English	5-layer, 1 RNN	27.79	14.39
English	9-layer, 7 RNN	14.93	9.52
Mandarin	5-layer, 1 RNN	9.80	7.13
Mandarin	9-layer, 7 RNN	7.55	5.81

DeepSpeech2, w/LM and w/o LM

- LM 으로는 N-gram LM 혹은 Neural-network LM 사용
- "Shallow": AM과 LM이 각각 훈련된 다음 decoding 때만 합쳐서 사용

Thank You!

NPEX 2022 Deep learning for Speech