Deep Speech: Scaling Up End to End Speech Recognition

Awni Hannun* , Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, Andrew Y. Ng

발표자: 진승정 github.com/jin-sj

#워크맨 #workman #장성규

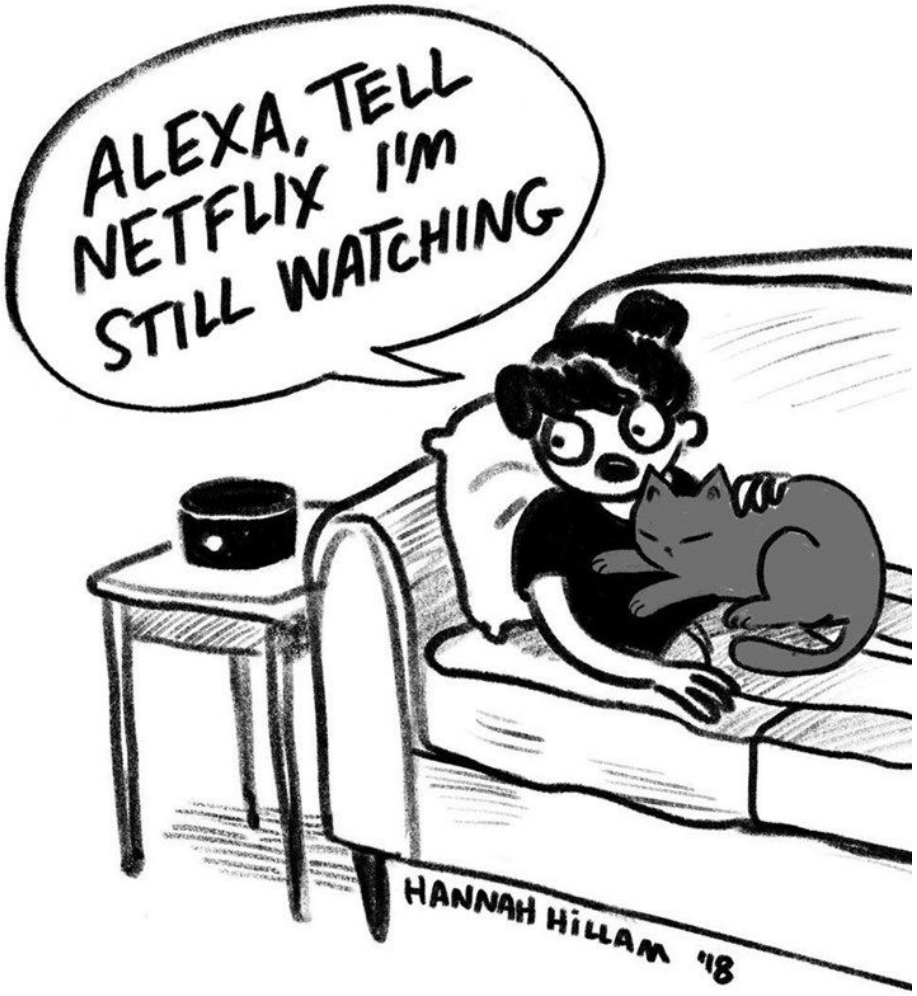
[ENG] 잡것들아 이제 만족하니???😬 😬 앞으로도 계속 함께하자 늘 함께하고 싶으니까💘(어금니 꽉) 감성 터진 장졸리 | 에뛰드 알바 리뷰 | 워크맨 ep.80

조회수 2,019,185회 • 2020. 12. 11.

4만 1.2천 공유 저장 ...

워크맨-Workman
구독자 376만명

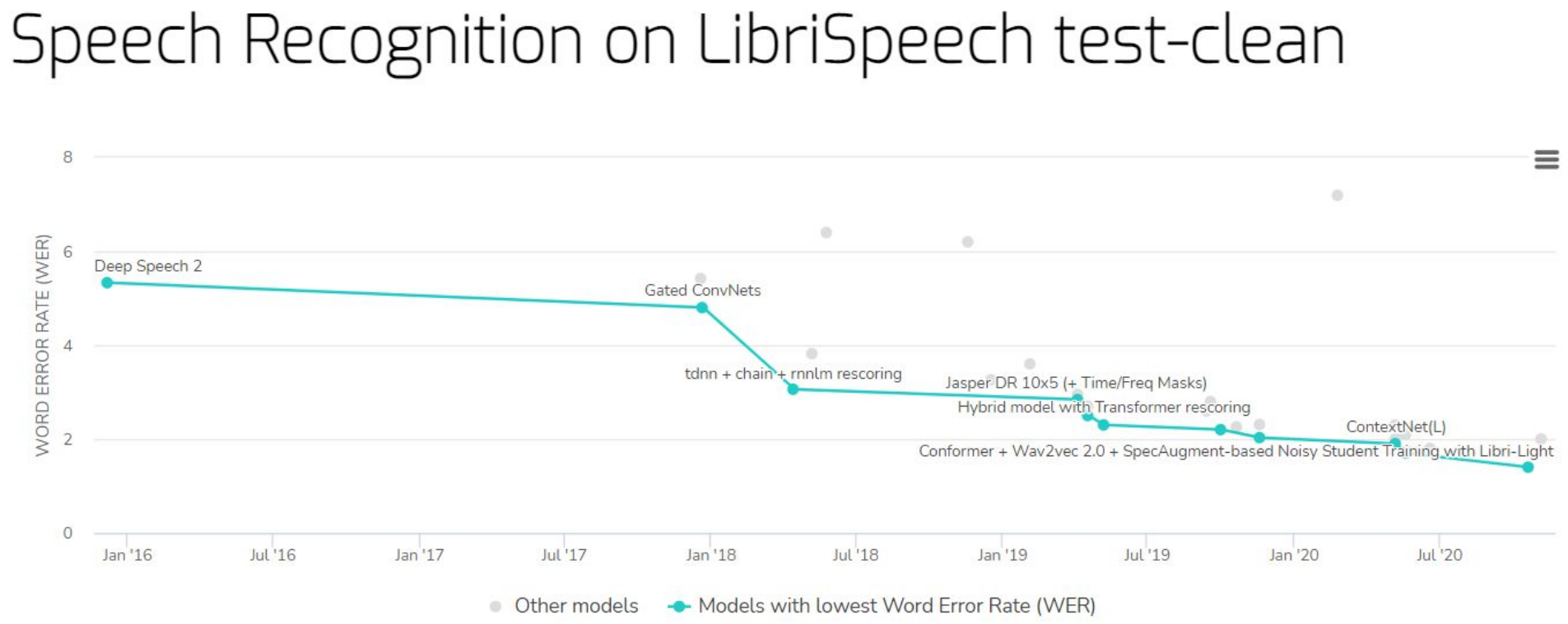
구독중


ALEXA, TELL NETFLIX I'M STILL WATCHING
HANNAH HILLAM '18

Before Deep Speech

- 음성 인식 파이프라인의 모든 스테이지에 feature engineering = 각 domain 마다 전문가가 필요함
  - specialized input features (MFCC)
  - acoustic models
  - Hidden Markov Models (HMMs)

ASR Timeline

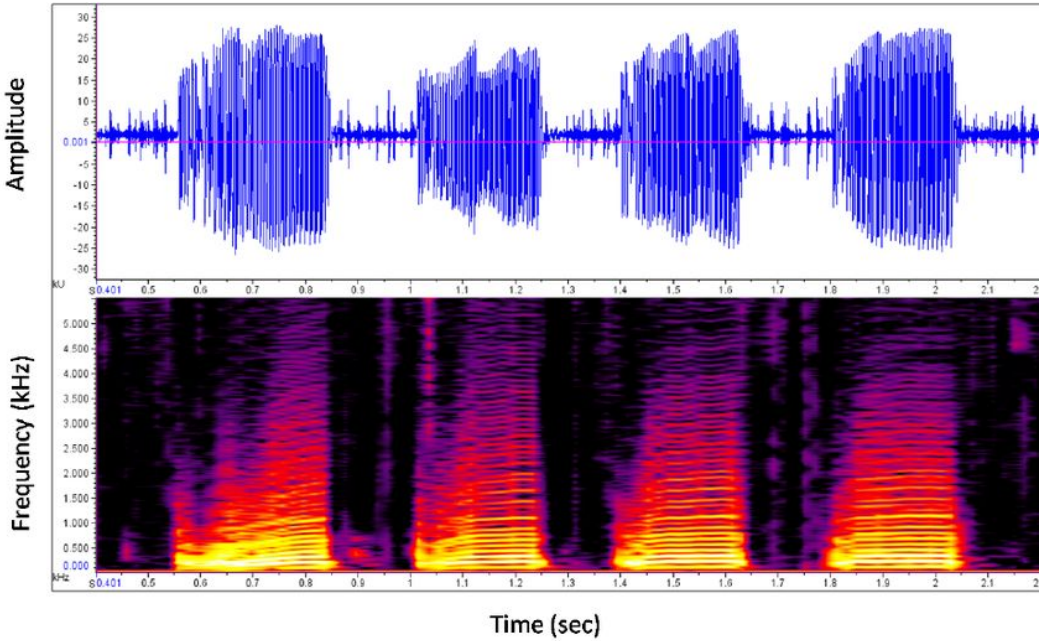


Deep Speech

- 자동 음성인식 (ASR)의 딥러닝화
- Data, data, data
  - Newly collected data: 5000 raw hours
  - Data synthesis: Raw + noise ~= 100,000 hours
  - CTC Loss: No alignment needed
- Data/GPU parallelization
  - Handle bottleneck from Bidirectional RNN layer
- 모든 부분이 딥러닝? NO, but getting there!
  - Spectrogram
  - Decode with: Beam search + n-gram language model

Input Data

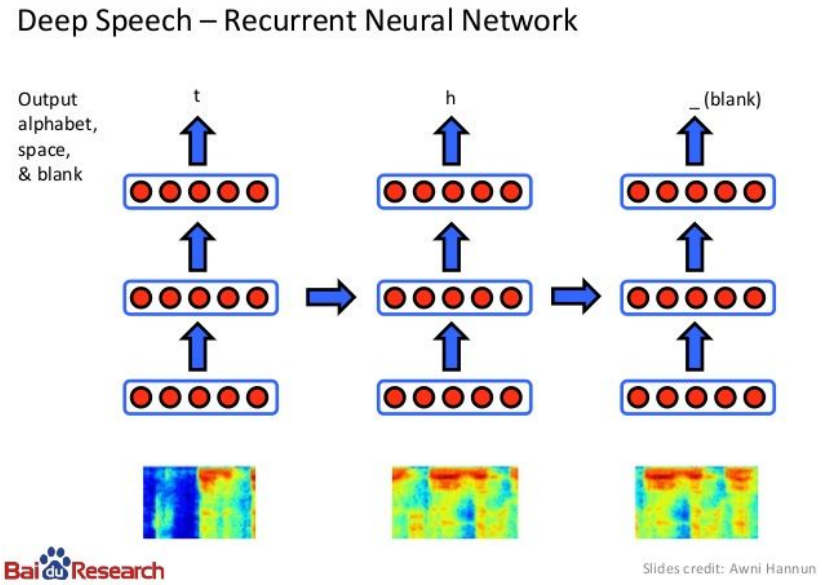
- Input:
  - Time series of Length T(i): 보통 25ms로 나누어짐
  - Spectrogram: power of the p'th frequency bin in the audio frame at time t



모델의 Goal

- Input sequence x -> sequence of character probabilities for the transcription y

$$\hat{y}_t = \mathbb{P}(c_t|x), \text{where } c_t \in \{\text{a,b,c,} \ldots \text{, z}, space, apostrophe, blank\}.$$



Model Architecture

- 5 layers
  - 1: CNN with c frames on each size for context (c = {5, 7, 9})
  - 2-3: non-recurrent layers (CNN/Feed Forward)
  - 4: bidirectional RNN
  - 5: FC of RNN
  - Output: Softmax function
    - Predicted character probabilities for each time slice *t* and character *k* in the alphabet
- Rectified ReLU:

$$g(z) = \min\{\max\{0, z\}, 20\}$$

- Optimizer: Nesterov Accelerated Gradient
  - Similar to Momentum

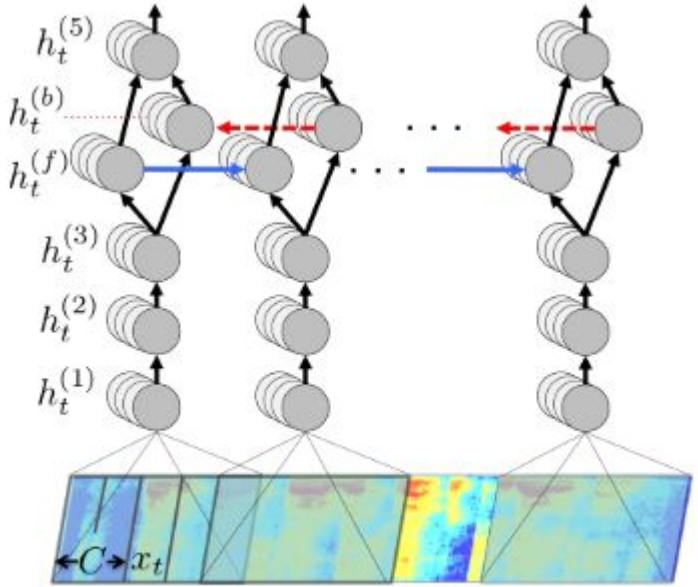
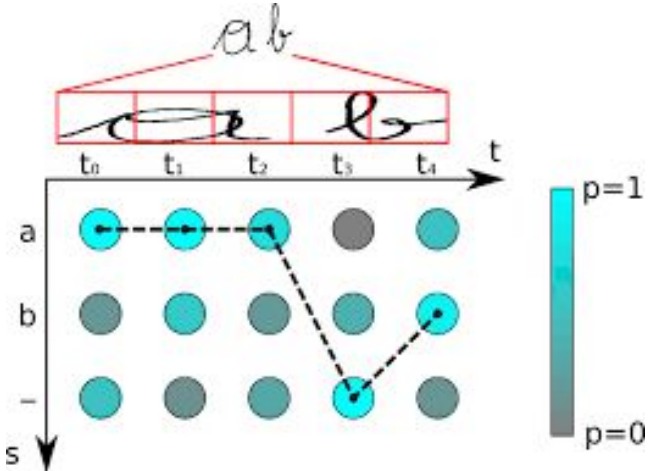
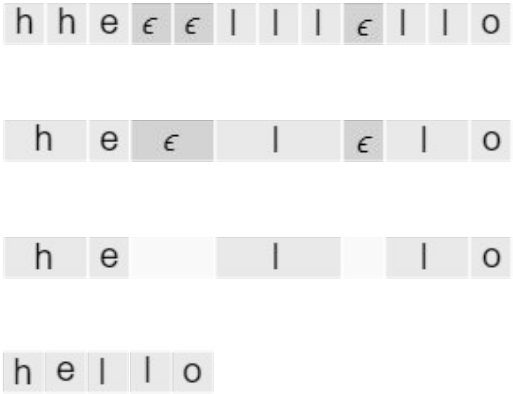


Figure 1: Structure of our RNN model and notation.

CTC Loss

- 문제: Alignment Problem
  - 오디오를 character-level로 label해야함: 사실상 불가능
  - Hello를 말할때
    - 길게 늘어뜨리기: Hhhhhhelloooooooo -> Hello
    - 짧게: Hello -> Hello
- 해결책: CTC Loss





First, merge repeat characters.

Then, remove any ϵ tokens.

The remaining characters are the output.

CTC Loss Calculation

- Input에 대한 ground truth의 모든 output probability를 계산
  - CTC Loss = -log(sum of probability of all paths to ground truth)

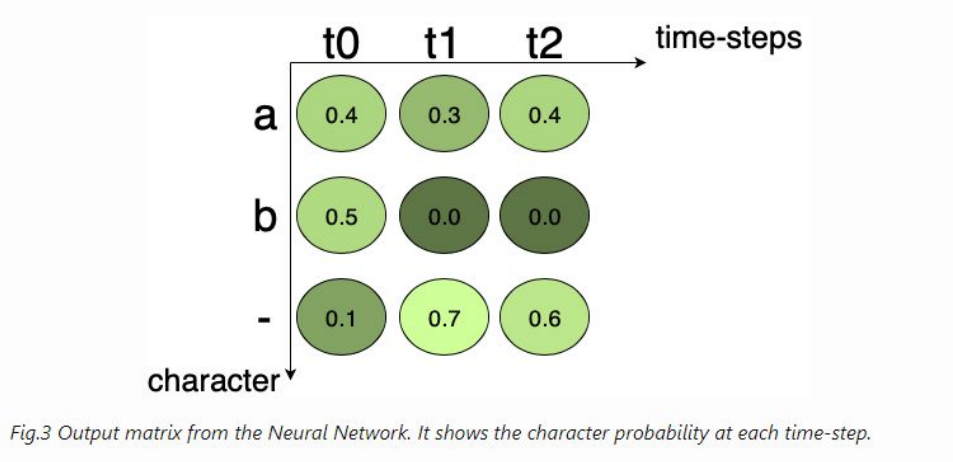


Fig.3 Output matrix from the Neural Network. It shows the character probability at each time-step.

- GT가 'a'일때 가능한 path:
  - "aaa", "a--", "-a-", "aa-", "-aa", "--a"
  - sum of prob = 0.048 + 0.168 + 0.018 + 0.072 + 0.012 + 0.028 = 0.346

Regularization

- RNN이 overfitting 하는 문제점이 있음
- 해결책:
  - Dropout: 5-10% on non-recurrent layers
  - Jittering (augmentation via transformation):
    i. Translate audio by +/- 5ms
    ii. Propagate values into the network
    iii. Average output probabilities
    iv. Test time: Ensemble of several RNNs

Language Model

- RNN이 '음성적'으로는 가능한 prediction을 함. 하지만 언어적으로 맞지 않을 수가 있음

RNN output	Decoded Transcription
what is the weather like in bostin right now	what is the weather like in boston right now
prime miniter nerenr modi	prime minister narendra modi
arther n tickets for the game	are there any tickets for the game

Table 1: Examples of transcriptions directly from the RNN (left) with errors that are fixed by addition of a language model (right).

Language Model

- RNN이 자주 틀리는 곳은 들어보지 못하거나 많이 들어보지 못한 단어들이다
- text data >> 음성 data
  - text: 220 million phrases, 495,000 vocab
  - 음성: 3 million utterances
- 5-gram model + beam search
  - Optimize Q(c); c = sequence of characters

$$Q(c) = \log(\mathbb{P}(c|x)) + \alpha \log(\mathbb{P}_{\text{lm}}(c)) + \beta \text{ word\_count}(c)$$

where α and β are tunable parameters (set by cross-validation) that control the trade-off between the RNN, the language model constraint and the length of the sentence. The term $\mathbb{P}_{\text{lm}}$ denotes the probability of the sequence c according to the N-gram model. We maximize this objective using a highly optimized beam search algorithm, with a typical beam size in the range 1000-8000—similar to the approach described by Hannun et al. [16].

Optimizations: Data Parallelism

- Concatenate many examples to a single matrix (batch)
- 오디오의 size가 다를 때는 어려움이 있음
  - 오디오를 length별로 sort한 후, 비슷한 길이의 오디오들을 이용
  - 필요시, silence로 padding

Optimizations: Model Parallelism

- Model ‘partitioning’
  - RNN의 forward & backward를 2개의 GPU를 나누면 5번째 layer계산 할 때 data transfer 때문에 시간 절감 효과가 극히 저감 됨

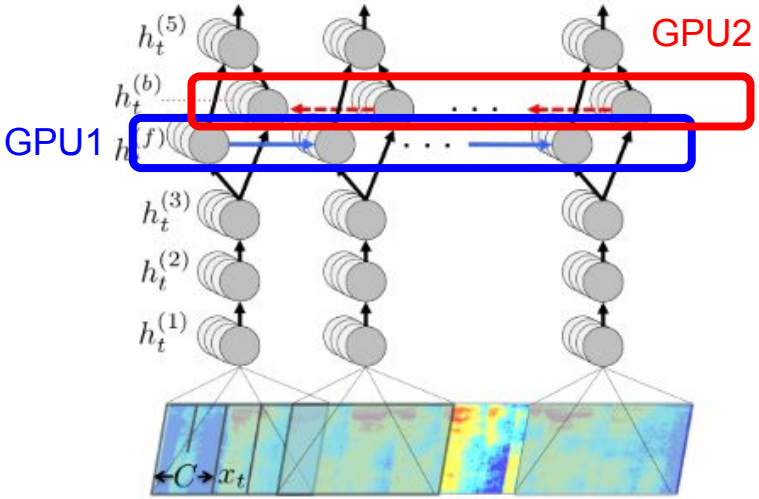


Figure 1: Structure of our RNN model and notation.

Optimizations: Model Parallelism

- Model ‘partitioning’
  - RNN의 forward & backward를 2개의 GPU를 나누면 5번째 layer계산 할 때 data transfer 때문에 시간 절감 효과가 극히 저감 됨
  - Model을 input의 중간 지점으로 partition함
    - gpu1: 1… t/2까지, forward calcation
    - gpu2: t/2+1부터 t까지, backward calculation
    - 중간(t/2)에서 intermediate activation교환
    - gpu1: 1… t/2까지, backward calculation
    - gpu1: 1… t/2까지, forward calculation

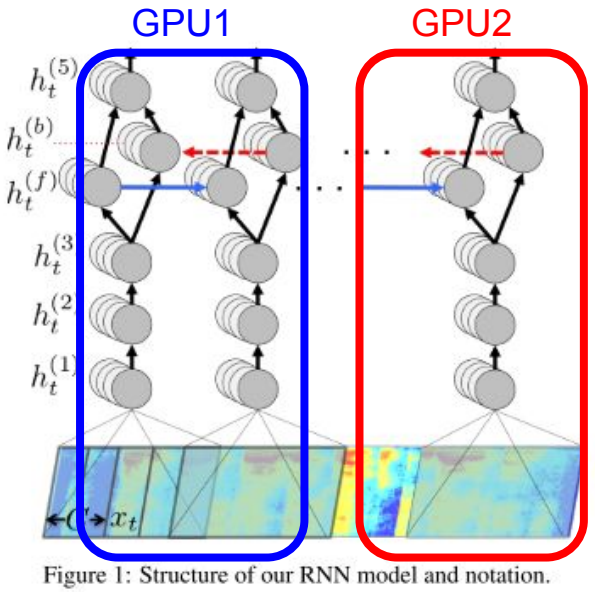


Figure 1: Structure of our RNN model and notation.

Optimizations: Striding

- CNN에서의 stride랑 비슷한 개념
- Audio 데이터를 size 2 step을 적용하여 계산 양을 절반으로 줄임

Training Data

- 저자들이 9600명의 목소리가 담긴 5000 hours분량의 데이터를 수집
- Data synthesis: Improve performance on noisy environments
  - noisy audio를 여러개 수집 후, raw audio + 여러개의 noise audio를 결합하여 데이터를 synthesize
  - Frequency들의 average power가 실제 noisy environment data랑 다를 시에 reject됨
  - 5000 hours of clean audio -> 100,000 hours of noisy audio
- Lombard effect: 사람은 주변 소리에 반응하여 pitch (음조) 또는 inflection(음성의 조절)을 다르게 말을 하는 효과
  - 데이터 수집 과정에서 사람들에게 시끄러운 소리가 나오는 헤드셋을 착용시켰음

Results - Metrics

- WER: Word Error Rate

Word error rate can then be computed as:

$$WER = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C}$$

where

- S is the number of substitutions,
- D is the number of deletions,
- I is the number of insertions,
- C is the number of correct words,
- N is the number of words in the reference (N=S+D+C)

Results - Metrics

Let's say that a person speaks 29 total words in an original transcription file. Among those words spoken, the transcription included 11 substitutions, insertions, and deletions.

Correct text	Google output
We wanted people to know that we've got something brand new and essentially this product is uh what we call disruptive changes the way that people interact with technology.	We wanted people to know that ~~how~~ to me where i know and essentially this product is uh what we call scripted changes the way that people are rapid technology.

To get the WER for that transcription, you would divide 11 by 29 to get 0.379. That rounds up to .38, making the WER 38 percent.

Results

Model	SWB	CH	Full
Vesely et al. (GMM-HMM BMMI) [44]	18.6	33.0	25.8
Vesely et al. (DNN-HMM sMBR) [44]	12.6	24.1	18.4
Maas et al. (DNN-HMM SWB) [28]	14.6	26.3	20.5
Maas et al. (DNN-HMM FSH) [28]	16.0	23.7	19.9
Seide et al. (CD-DNN) [39]	16.1	n/a	n/a
Kingsbury et al. (DNN-HMM sMBR HF) [22]	13.3	n/a	n/a
Sainath et al. (CNN-HMM) [36]	11.5	n/a	n/a
Soltau et al. (MLP/CNN+I-Vector) [40]	**10.4**	n/a	n/a
Deep Speech SWB	20.0	31.8	25.9
Deep Speech SWB + FSH	12.6	**19.3**	**16.0**

Table 3: Published error rates (%WER) on Switchboard dataset splits. The columns labeled "SWB" and "CH" are respectively the easy and hard subsets of Hub5'00.

Hub500: SWB (easy) + CH (hard) test sets
SWB: 300 hours of train data
Fisher: 2000 hours of train data

Results

System	Clean (94)	Noisy (82)	Combined (176)
Apple Dictation	14.24	43.76	26.73
Bing Speech	11.73	36.12	22.05
Google API	6.64	30.47	16.72
wit.ai	7.94	35.06	19.41
Deep Speech	**6.56**	**19.06**	**11.85**

Table 4: Results (%WER) for 5 systems evaluated on the original audio. Scores are reported o
for utterances with predictions given by all systems. The number in parentheses next to each datas
e.g. Clean (94), is the number of utterances scored.

New raw (clean) data: 5000 hours
Noisy data: > 100,000 hours

질의응답

Resources

- https://sid2697.github.io/Blog_Sid/algorithm/2019/10/19/CTC-Loss.html
- https://towardsdatascience.com/intuitively-understanding-connectionist-temporal-classification-3797e43a86c
- https://towardsdatascience.com/intuitively-understanding-connectionist-temporal-classification-3797e43a86c
- https://wikidocs.net/21692
- https://www.youtube.com/watch?v=g-sndkf7mCs&t=3697s
- https://www.youtube.com/watch?v=P9GLDezYVX4
- https://www.rev.com/blog/resources/what-is-wer-what-does-word-error-rate-mean