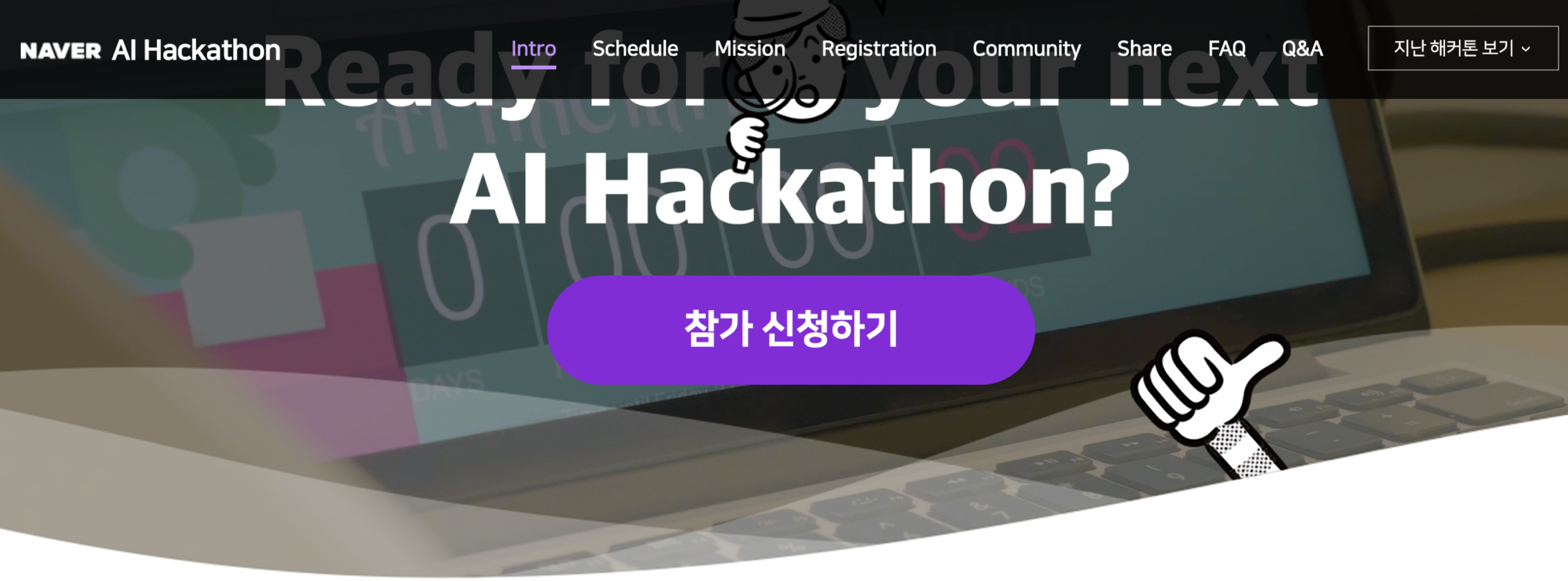
2019 음성인식 해커톤

Chapter 2. Deep Learning for Speech Recognition



https://campaign.naver.com/aihackathon_speech/

세번째 네이버 AI 해커톤, AI Speech

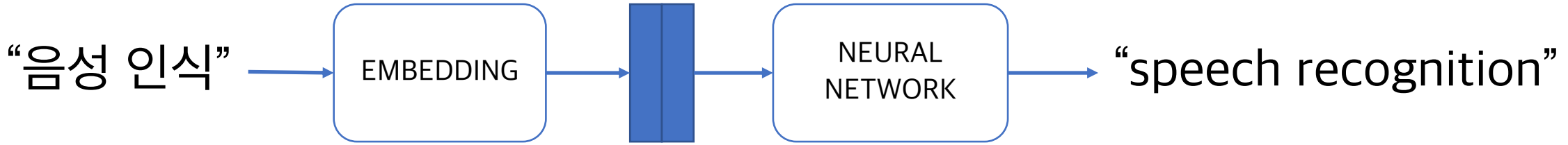
여러분들의 뜨거운 관심과 치열한 고민을 이어갈 수 있도록 이번에는 AI의 핵심 기술중의 하나인
음성인식 주제를 가지고 찾아왔습니다.

Overview

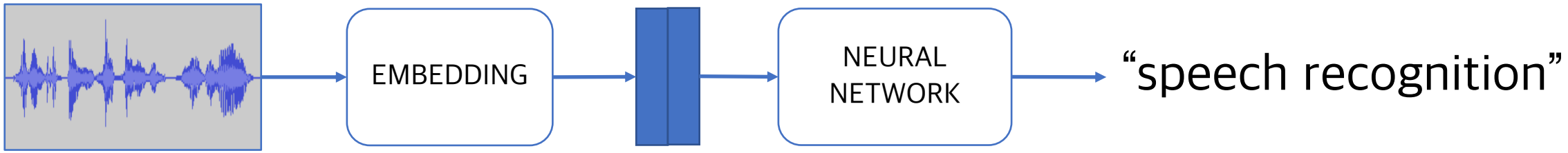
End-to-end Speech Recognition

Deep learning for 음성인식

- **기계 번역**



- **음성 인식**

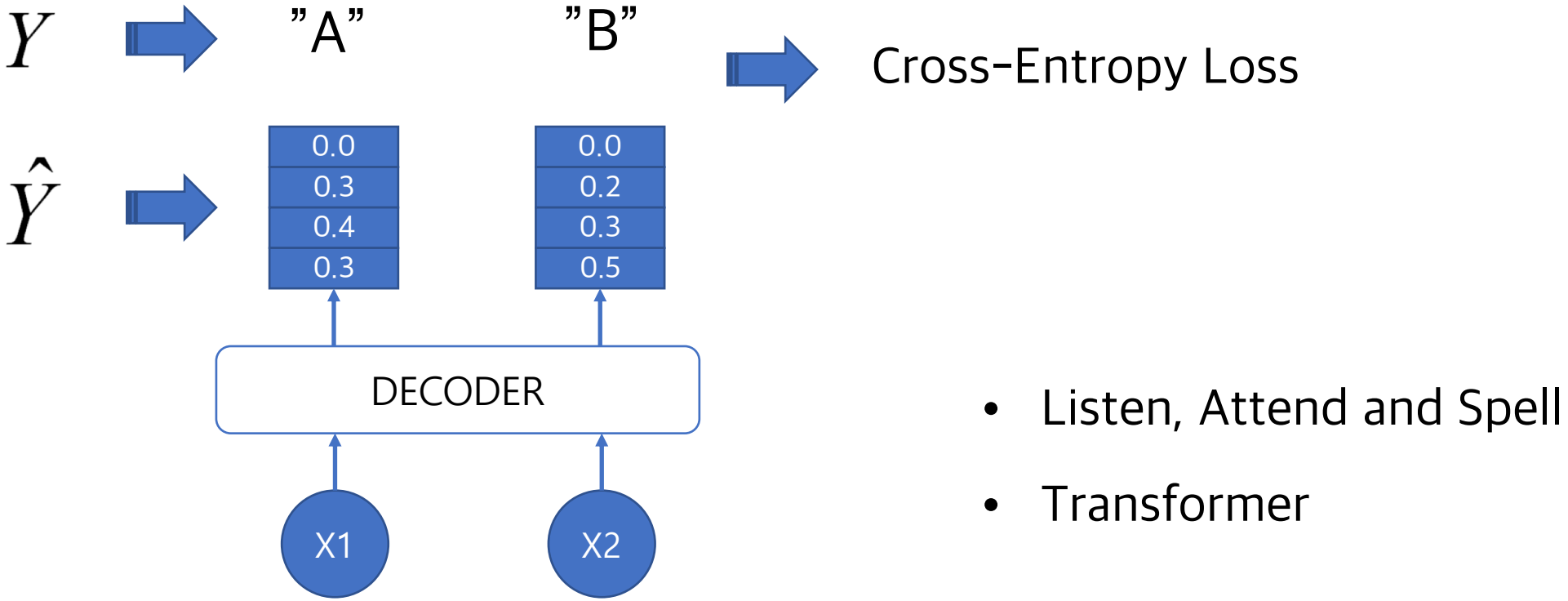


음성인식의 특징

- Long input sequence, short output sequence
  - 48,000 (3초 X 16,000 samples) => 20 (글자)
- Input and output sequences in same order
- Loss Function
  - CrossEntropy, MSE, …
  - CTCLoss
- Performance
  - Levenshtein Distance (Edit Distance)

LOSS Function

Sequence to sequence model



- Listen, Attend and Spell
- Transformer

LOSS Function

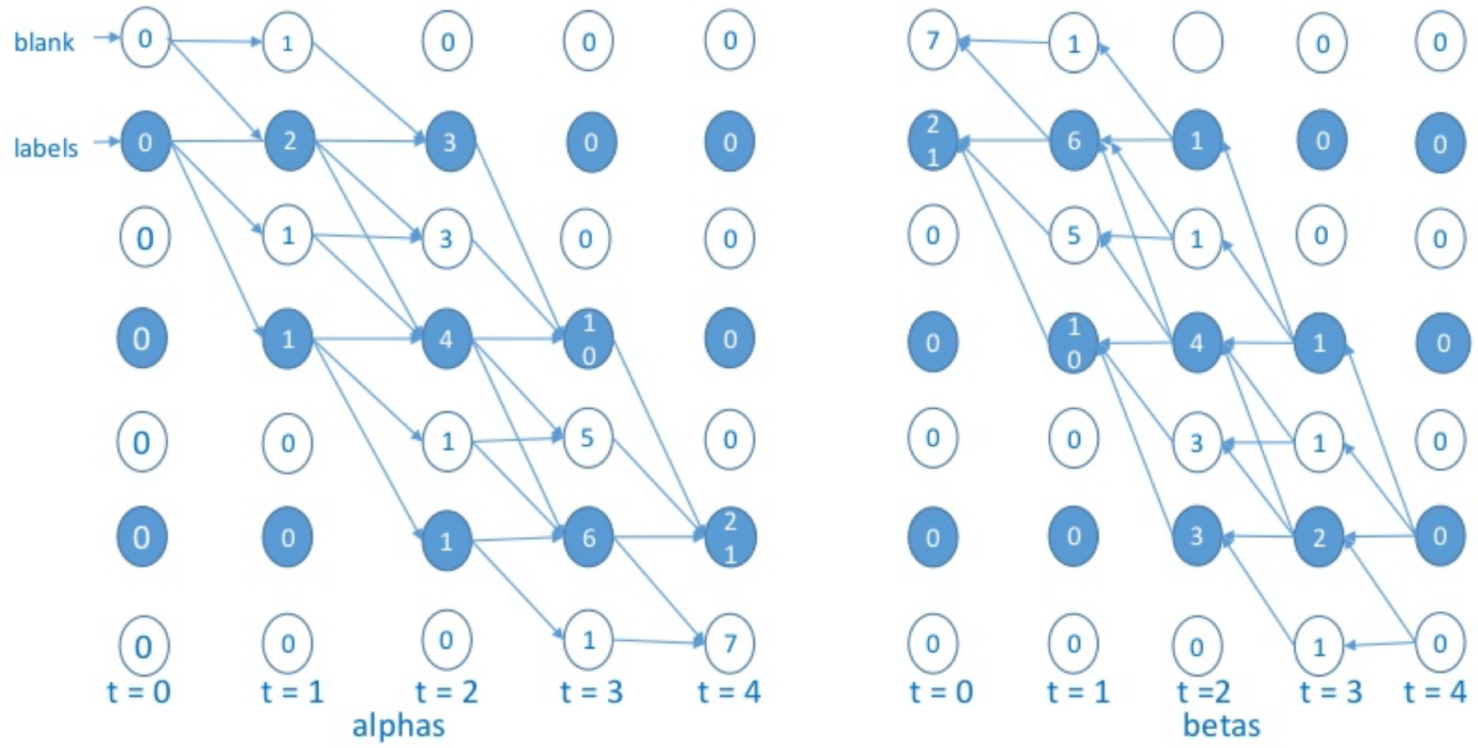
Sequence to sequence model

11 lines (7 sloc) | 306 Bytes

```
criterion = nn.CrossEntropyLoss(reduction='sum', ignore_index=PAD_token)

def train()
    # ...
    y_hat = model(features, features_lengths, scripts, teacher_forcing_ratio)
    y_hat = torch.stack(y_hat, dim=1)
    y = scripts[:, 1:]

    loss = criterion(y_hat.view(-1, y_hat.size(-1)), y.view(-1))

```

LOSS Function

Connectionist Temporal Classification

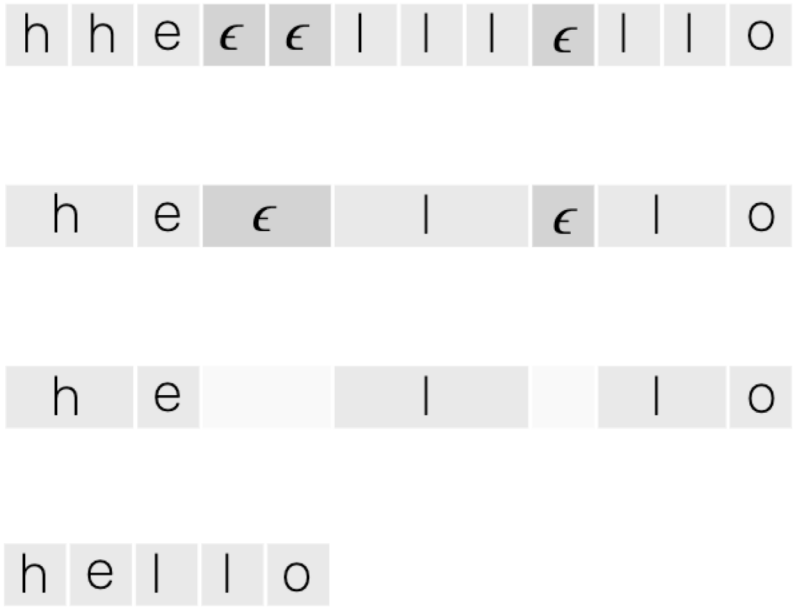


$$p(\mathbf{l}|\mathbf{x}) = \sum_{t=1}^{T} \sum_{s=1}^{|\mathbf{l}|} \frac{\alpha_t(s)\beta_t(s)}{y_{l_s}^t}.$$

- Suhas Pillai, “Intelligent Handwriting Recognition_MIL_presentation_v3_final”, Slideshare
- Alex Graves et al. “Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks”, ICML, 2006

LOSS Function

Decoding Connectionist Temporal Classification



First, merge repeat characters.

Then, remove any ϵ tokens.

The remaining characters are the output.

- Deep Speech 2

- Awni Hannun, “Sequence Modeling With CTC”, 10.23915/distill.00008

Performance

Edit Distance

substitution = 2

deletion = 1

insertion = 1

$$ED = \frac{2 + 1 + 1}{\text{length of Ref.}} = \frac{4}{8} = 0.5$$

https://en.wikipedia.org/wiki/Levenshtein_distance

Performance

Edit Distance

4 lines (2 sloc) | 54 Bytes

```
1  import Levenshtein as Lev
2
3  CER = Lev.ratio(ref, hyp)
```

Baseline system

End-to-end Speech Recognition

Baseline system

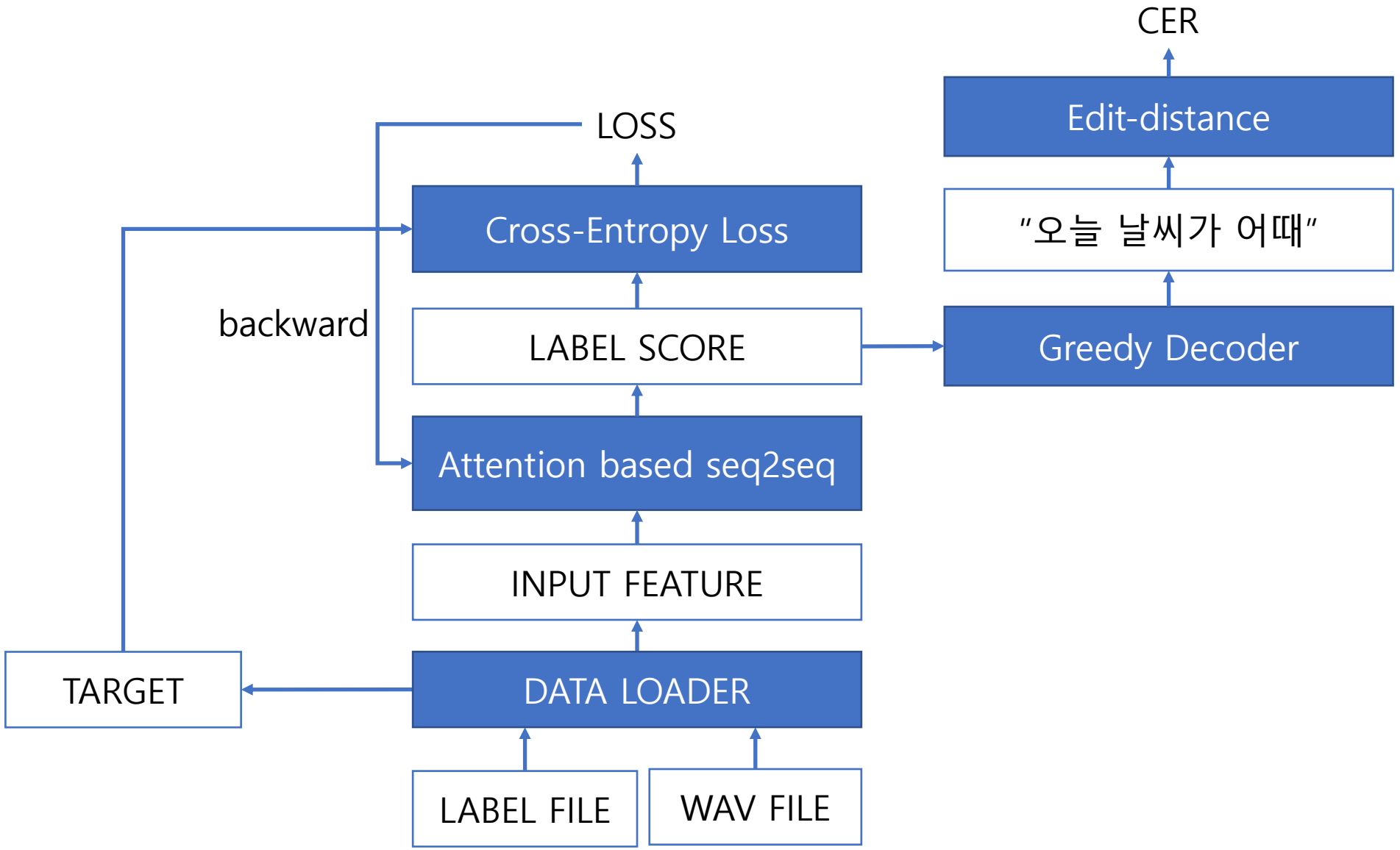
Dataset

약 3천 명의 사람들이 전화망을 통해 녹음한 3만개의 음성 파일과 정답 스크립트

FILE	CONTENTS
20190508192212_01074147469_378_0.wav	RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono 16000 Hz
20190508192212_01074147469_378_0.script	크리스마스날 영업하나요?
20190508192212_01074147469_378_0.label	533 94 384 623 384 706 662 211 532 690 62 661 123 662

META-FILE	CONTENTS
train/train_data/data_list.csv	3만개의 파일 목록
script.labels	Character와 label 간의 매핑 테이블 정보

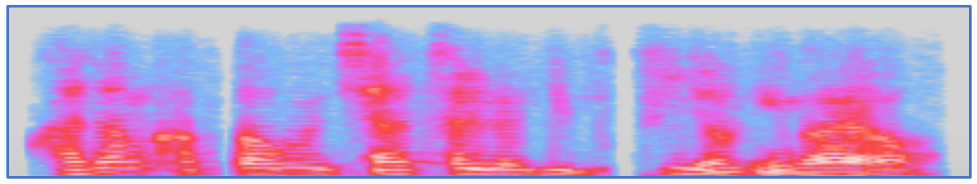
Baseline system

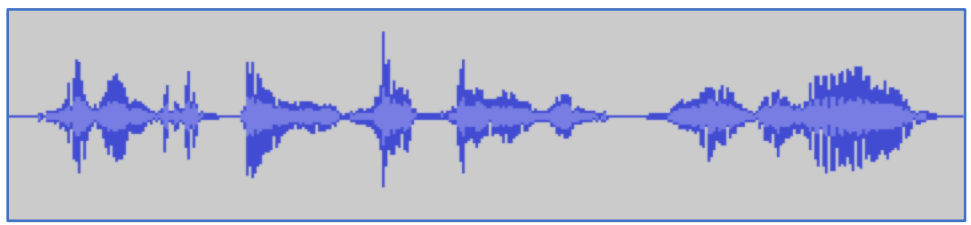


Baseline system

Data Loader

INPUT ([512, 257])
([4.7639, 2.7057, 8.1335, 3.4812, 2.5112
1.1522, 7.9136, 1.3469, 6.5892, 4.4722
3.6681, 1.7279, 5.4906, 7.6762, 2.5324 ...





TARGET ([21])
([192 755 662 192 678 476 662 408 690 2
125 610 662 220 640 125 662 179 192 661
123])

script.labels

가장 가까운 지하철역이 무슨 역 인가요?

Baseline system

Feature extractor

20 lines (16 sloc) | 681 Bytes

```
def get_feature(filepath, feature_size):
    (rate, width, sig) = wavio.readwav(filepath)

    sig = sig.ravel()

    stft = torch.stft(torch.FloatTensor(sig),
                      512,
                      hop_length=int(0.01*16000),
                      win_length=int(0.030*16000),
                      window=torch.hamming_window(int(0.030*16000)),
                      center=False,
                      normalized=False,
                      onesided=True)
    stft = (stft[:,:,0].pow(2) + stft[:,:,1].pow(2)).pow(0.5);
    spect = stft.numpy();
    spect = torch.FloatTensor(spect)

    spect = torch.FloatTensor(spect).transpose(0, 1)
    return spect
```

Baseline system

DataLoader를 python thread로 구현해서 속도 향상

23 lines (18 sloc) | 674 Bytes

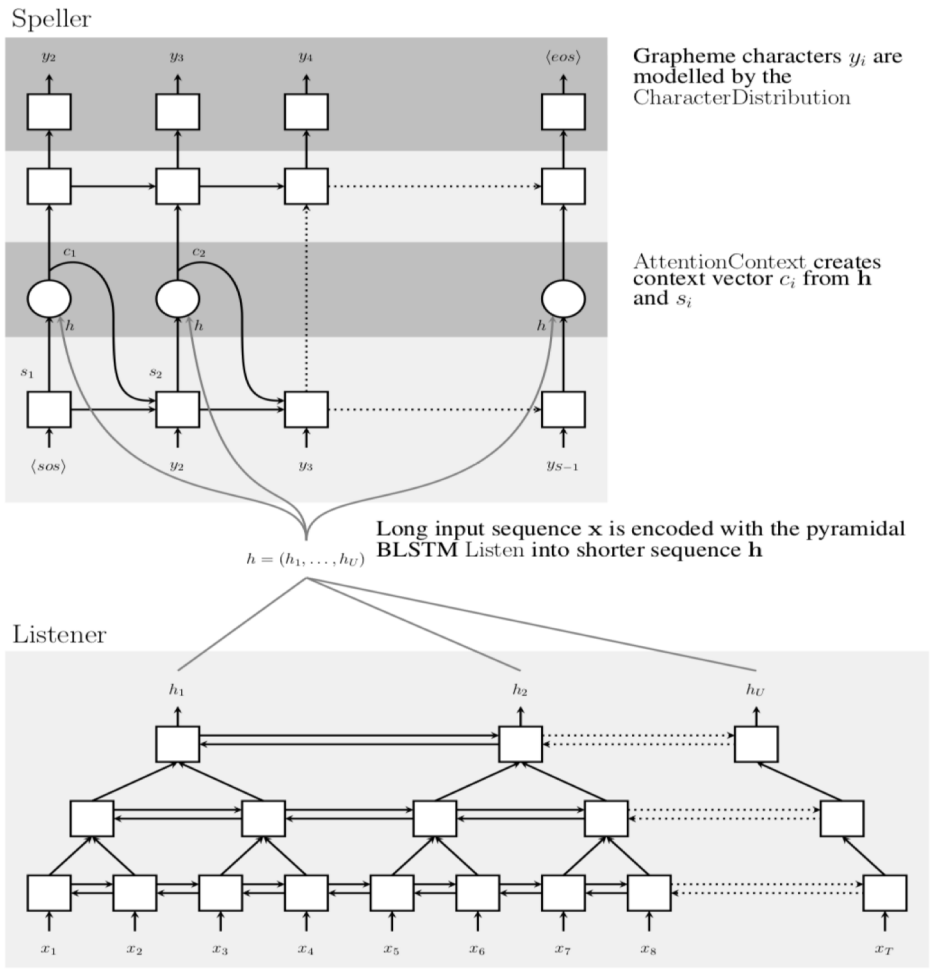
```python
class BaseDataLoader(threading.Thread):
    def run(self):
        while True:
            items = self.read_batches()
            if len(items) == 0:
                break

            random.shuffle(items)
            batch = self.collate_fn(items)
            self.queue.put(batch)

class MultiLoader():
    def __init__(self, dataset_list, queue, batch_size, worker_size):
        self.worker_size = worker_size
        self.loader = list()
        for i in range(self.worker_size):
            self.loader.append(BaseDataLoader(dataset_list[i], queue, batch_size, i))

    def start(self):
        for i in range(self.worker_size):
            self.loader[i].start()
```

End-to-end models

End-to-end Speech Recognition

Listen, Attend and Spell

Sequence to sequence with Attention



cross-entropy loss

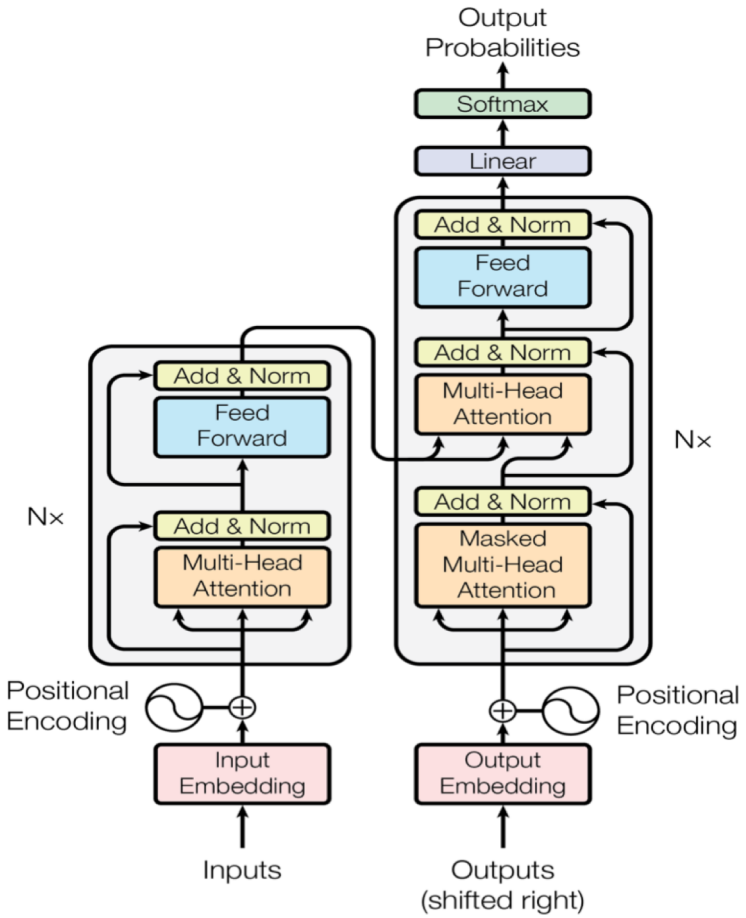
input context에 대해서
global attention 사용

pyramidal LSTM을 사용해서
input sequence를 ¼로 축소

William Chan et al, "Listen, Attend and Spell", ICASSP, 2016

Transformer

Sequence to sequence with Attention

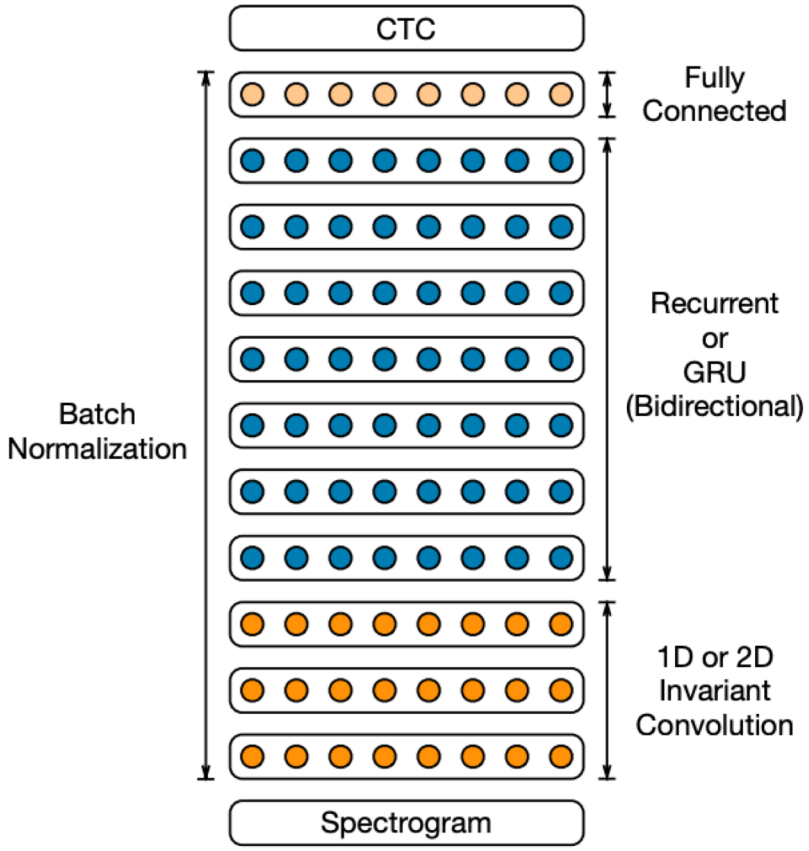


RNN 대신에, Positional Encoding과 Feed-Forward layer를 이용해서 Parallel Encoding & Decoding, 학습 속도 향상

Ashish Vaswani, “Attention Is All You Need”, NIPS, 2017

Encoder with Forward-backward algorithm

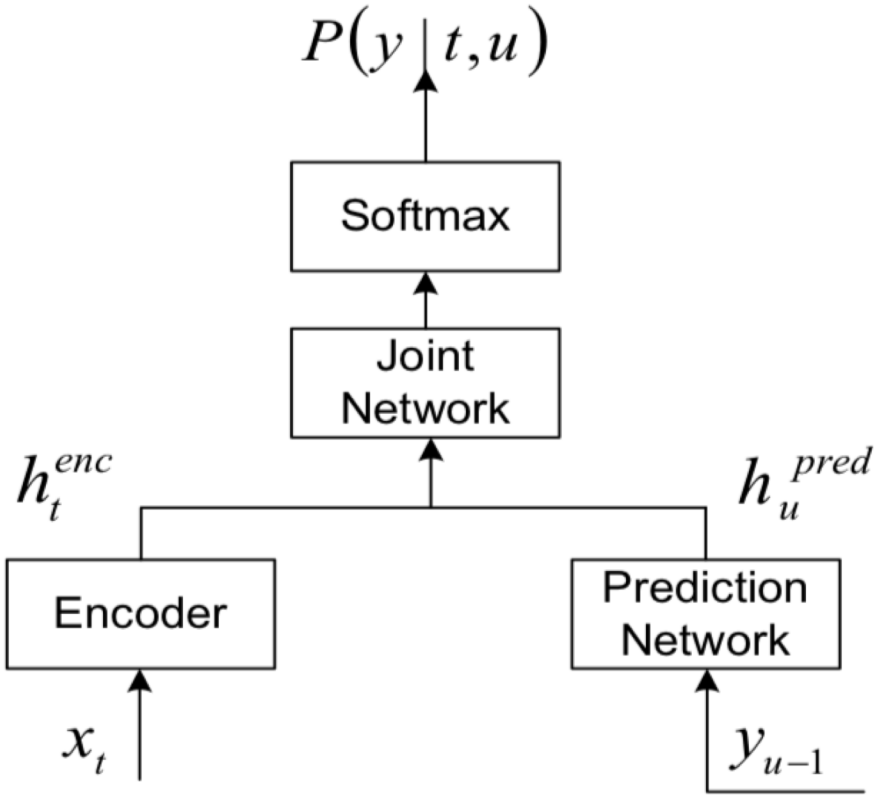
Deep Speech2

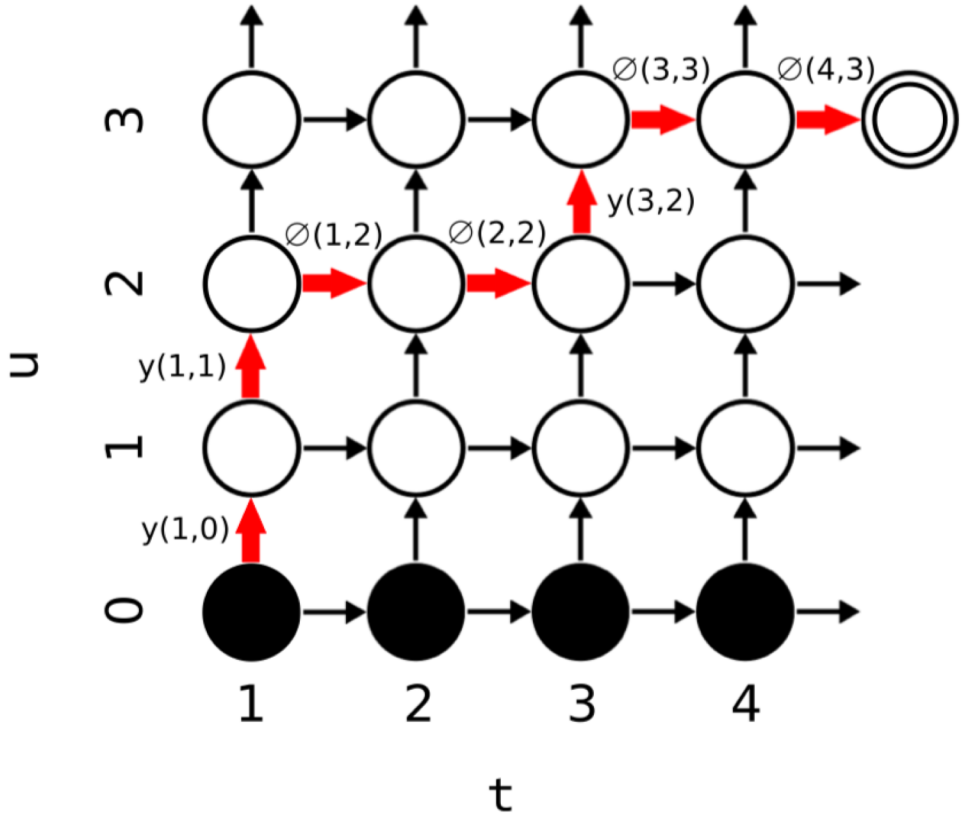


Dario Amodei et al, “Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin”, jmlr.org, 2016

Encoder with Forward-backward algorithm

RNN Transducer





- Alex Graves, “Sequence Transduction with Recurrent Neural Networks”, ICML, 2012
- Senmao Wang et al, “Exploring RNN-Transducer for Chinese Speech Recognition”, arXiv:1811.05097

End of Document