Concept to Code: Semi-Supervised End-To-End Approaches For Speech Recognition

Omprakash Sonie
omprakash.s@flipkart.com

Venkateshan Kannan
ventangled@gmail.com

Goal:

- Provide foundation and relevant self & semi-supervised learning methods for ASR
- Provide learning from executing ‘Supervised to Semi-Supervised’ Case Study

Part-II

Tutorial Unsupervised ASR

Venkateshan Kannan

Introduction: Nearly unsupervised ASR

- MostASR systems are constructed by training deep neural networks on large-scale labeled data using supervised learning
- Can we train ASR models using little to no parallel training data?
- Annotating audio data is expensive;unannotated audio data is relatively easy to collect.
- Especially important in low or zero resource setting (transcribed speech data is not available for the vast majority of the nearly 7,000 languages of the world)
- Build on the success of unsupervised approaches in machine translation.

Introduction: Nearly unsupervised ASR

APPROACHES:

- Use iterative training and fine-tuning.
- Use self-training with voice data that learns to identify context (type of unsupervised learning)
- Learn targeted representations
- Understand token/phoneme distribution properties

Papers covered in the tutorial

Baevski et al, *Unsupervised Speech Recognition* (2021), arXiv:2105.11084

Ren et al, *Almost Unsupervised Text to Speech and Automatic Speech Recognition,* 2019 (ICML) [arXiv:1905.06791]

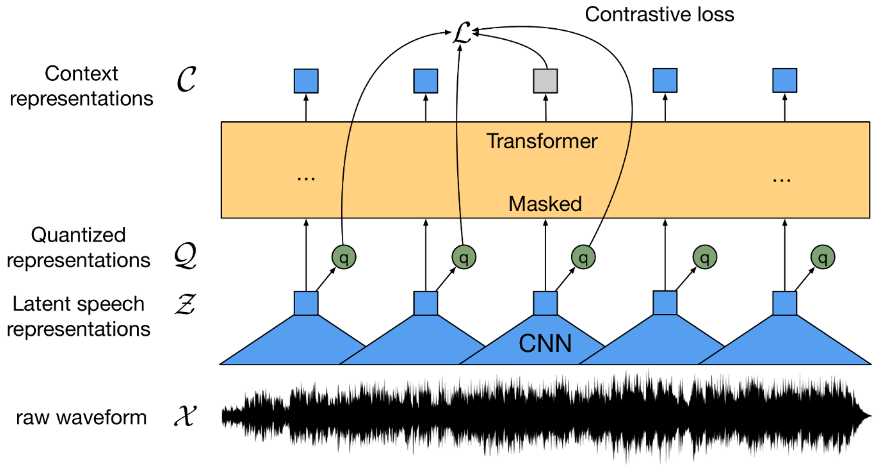
Chih-Kuan Yeh et al, *Unsupervised Speech Recognition via Segmental Empirical Output Distribution Matching (ICLR 2019)* arXiv:1812.09323

PAPER 1: Unsupervised Speech Recognition (2021)

- Introduce a model called **Wav-2-Vec Unsupervised**
- Builds on top of Wav-2-Vec 2.0 framework (also by the same group as Facebook)
- Use the context representation learned from Wav-2-Vec
- Wav-2-Vec itself was very successful at performing well despite using only a limited amount of parallel training data

Overview of Wav-2-Vec 2.0

- Using representation of audio from wav-2-vec



Contrastive loss
$\mathcal{L}$
Context representations $\mathcal{C}$
Transformer
...
Masked
...
Quantized representations $\mathcal{Q}$
q
Latent speech representations $\mathcal{Z}$
CNN
raw waveform $\mathcal{X}$

Baevski et al, wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations

Wav2Vec 2.0 Architecture

Consists of

1. Feature encoder : $f : \mathcal{X} \longrightarrow \mathcal{Z}$

   Raw speech audio frames are provided as input and that is mapped to **latent representation** $z_1, \cdots z_T \in \mathcal{Z}$

   Several blocks containing a temporal convolution followed by a GELU (Gaussian-error linear unit) activation function.

1. Context Network: $g : \mathcal{Z} \longrightarrow \mathcal{C}$

   output of the feature encoder is fed to a context network which follows the Transformer architecture giving context representation

1. Quantization Module: **discretize** the output of the feature encoder to a finite set of speech representations (called **codebook)** via product quantization. A set of G codebooks that are concatenated together. (Gumbel softmax is used to select the codebook from logits mapped from the latent representation)

Wav2Vec 2.0 Architecture

Quantization Module: **discretize** the output of the feature encoder to a finite set of speech representations (called **codebook)** via product quantization, i.e choosing quantized representations from multiple codebooks and concatenating them.

Given G codebooks,with V entries choose one entry from each and concatenate t
$e_1, \ldots, e_G$ and apply a linear transformation.
The Gumbel softmax enables choosing discrete codebook entries in a fully differentiable way.

The feature encoder output is mapped to $\mathbf{l} \in \mathbb{R}^{G \times V}$ logits and the probabilities for choosing the v-th codebook entry for group g are

$$p_{g,v} = \frac{\exp(l_{g,v} + n_v)/\tau}{\sum_{k=1}^{V} \exp(l_{g,k} + n_k)/\tau}$$

Wav2Vec Training

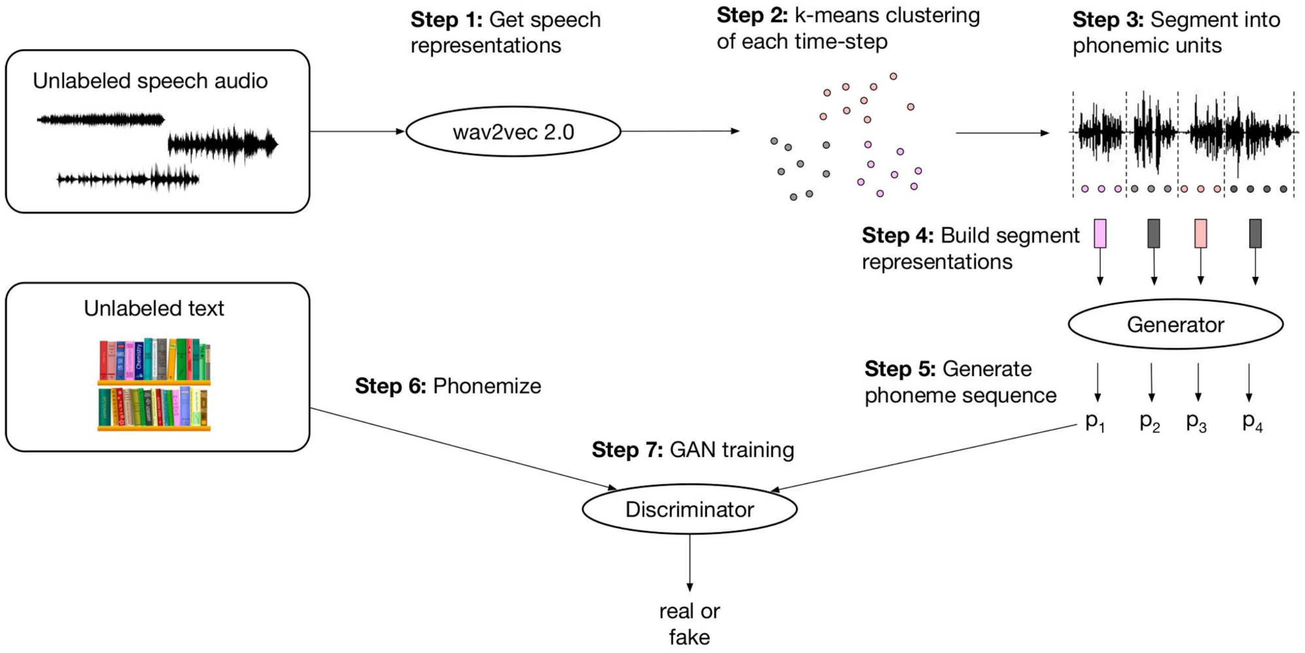
- Pre-trained through masking (self-learning) similar to BERT.
- Masking is applied to the output of the encoder at certain time-steps before input to the context representation layer. Instead a (trainable) feature vector is passed across all these masked time-steps.
- Note that the masking does **not** extend to the input to the quantization step.
- Contrastive loss is used to identify the true quantized speech representation q_t given the context vector c_t over a masked time step.

$$-\log \frac{\exp(sim(\mathbf{c}_t, \tilde{\mathbf{q}})/\kappa)}{\sum_{\tilde{\mathbf{q}} \sim \mathbf{Q}_t} \exp(sim(\mathbf{c}_t, \tilde{\mathbf{q}})/\kappa)}$$

Wav2Vec Training

- There is a **diversity loss** added for **uniform use of codebook representations**.

Wav2Vec Unsupervised


Unlabeled speech audio
Step 1: Get speech representations
wav2vec 2.0
Step 2: k-means clustering of each time-step
Step 3: Segment into phonemic units
Step 4: Build segment representations
Generator
Step 5: Generate phoneme sequence
p_1 p_2 p_3 p_4
Unlabeled text
Step 6: Phonemize
Step 7: GAN training
Discriminator
real or fake

Baevski et al, Unsupervised Speech Recognition (2021)

Wav2-Vec Unsupervised

- Speech representation obtained from Wav2Vec 2.0.
- Speech is **segmented** so that each segment can be mapped to a phoneme.
- Goal is to map audio representations to phonemes under no supervision.
- Uppermost block of context transformer is unsuitable: features are trained to predict masked representations spanning 25ms, considerably shorter than the typical duration of a phoneme.

Wav2-Vec Unsupervised

Train phoneme classifiers on top of the frozen representations of each of the 24 blocks of the English wav2vec 2.0

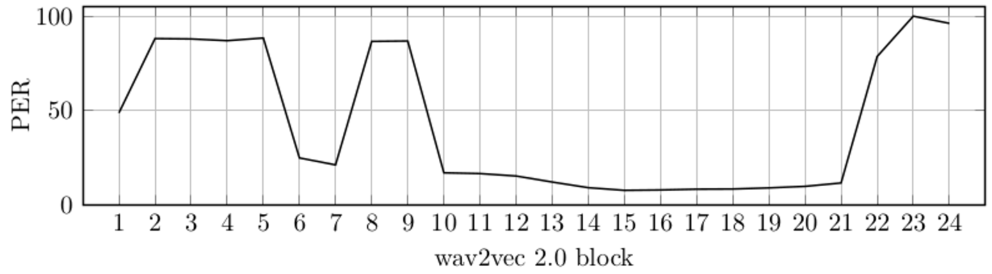


Figure 2: Supervised phoneme classification using representations from different wav2vec 2.0 blocks on dev-other of English Librispeech. Low and high blocks do not provide good features, while as blocks 14-19 do. Block 15 performs best.

Wav2-Vec Unsupervised

- Next, the complete speech representations of the training (unpaired) audio data are clustered using k-means with k=128.
- Each speech representation c_t is labeled by the cluster ID it belongs to.
- 512-dimensional PCA is performed on all speech representation and for a given segment (i.e, cluster id), the PCA components are mean-pooled.
- Segment representation

$$s_1, s_2 \cdots s_T \in \mathcal{S}$$

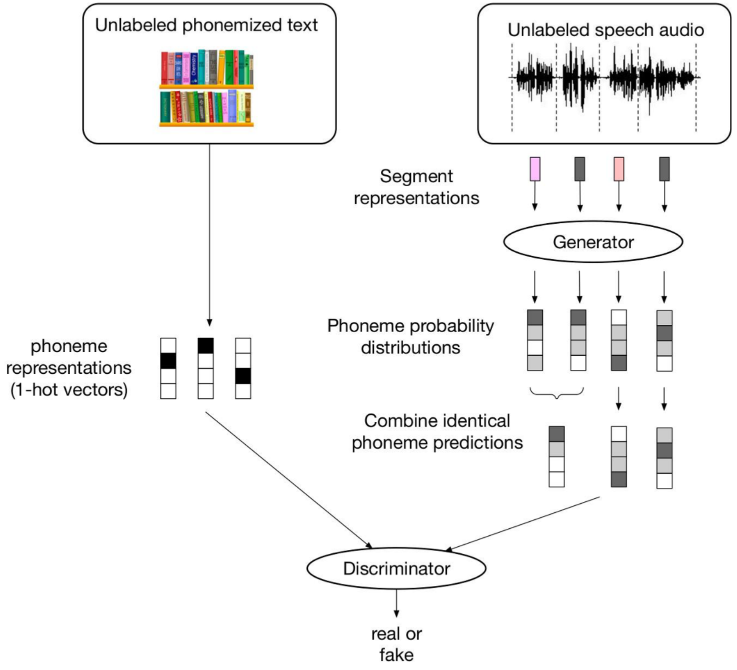
Wav2Vec-U (Training)

- Text data is also phonemized: the sequence of words is mapped to a sequence of phonemes
- The actual training happens through a generative adversarial model (GAN)
- Generator produces sequence of phonemes from the segments that are given as input
- Discriminator determines whether the sequence of phonemes is generated from actual textual data (or NOT).

Generative Adversarial networks

- In the maximization stage, the discriminator D is trained to recognize legitimate phonemized text (output close to 1) and simultaneously reject the generated sequence (output close to 0).
- In the minimization stage, the generator G is trained to "fool" the discriminator to maximize the scores for the sequences generated by G(z).
- The broader idea is to train the generator to produce phoneme sequence that resembles true sequences.

$$\min_G \max_D \left[\mathbb{E}_{x\in P} D(x) + \mathbb{E}_{s\in \mathcal{S}}(1 - D(G(s)))\right]$$


Unlabeled phonemized text
Unlabeled speech audio
Segment representations
Generator
Phoneme probability distributions
phoneme representations (1-hot vectors)
Combine identical phoneme predictions
Discriminator
real or fake

Baevski et al, Unsupervised Speech Recognition (2021)

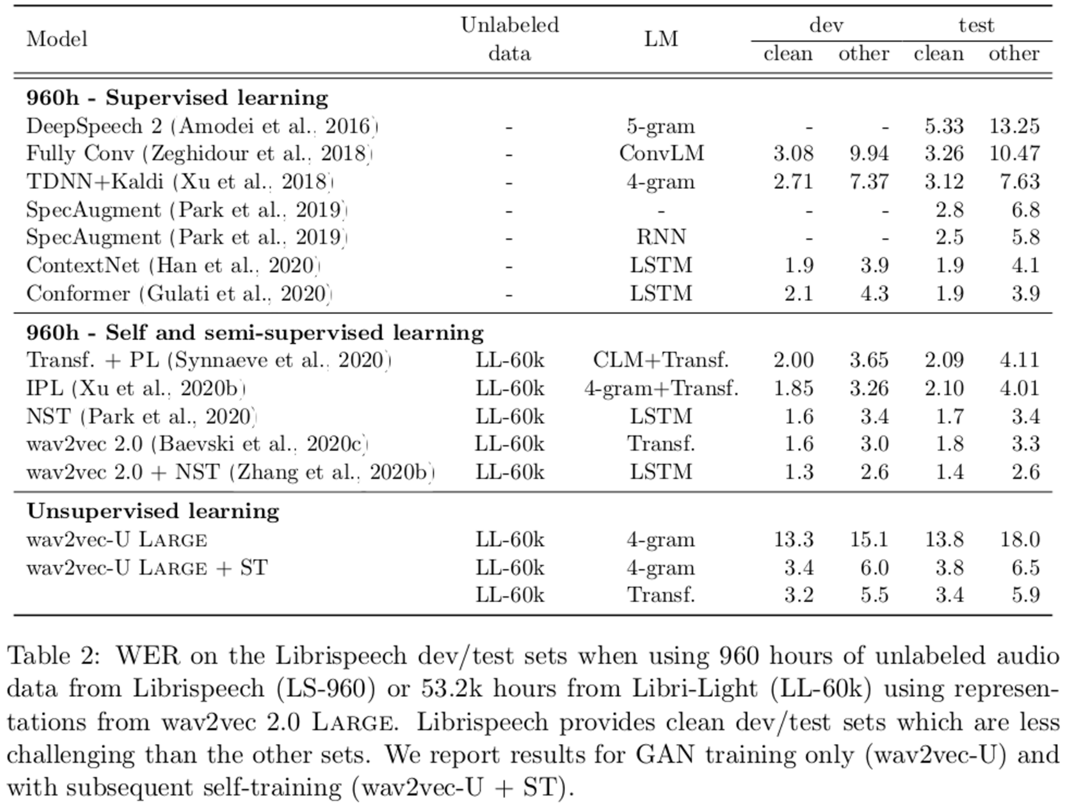
Wav2Vev-U (Training)

- There are also other losses: gradient norm of discriminator (for stability), phoneme segment sequence smoothness and phoneme diversity.
- The generator is a single convolution with kernel size 4.
- The discriminator is composed of three convolution blocks with a hidden size of 384 and a kernel size of 6, resulting in a receptive field size of 16 segments.

Wav2Vev-U (Self-training)

- Semi-supervised learning is used to progressively refine the quality of transcriptions.
- Two iterations:
  - First, pseudo-label the training data with the unsupervised GAN model and train an HMM on the pseudo-labels
  - Second, relabel the training data with the HMM (in inference mode) and then fine-tune the original wav2vec 2.0 model using the HMM pseudo-labels with a CTC loss.
- Note that HMM models use phonemes as output, while wav2vec 2.0 use letter. Both are decoded into words.

Results



Model	Unlabeled data	LM	dev clean	dev other	test clean	test other
960h - Supervised learning						
DeepSpeech 2 (Amodei et al., 2016)	-	5-gram	-	-	5.33	13.25
Fully Conv (Zeghidour et al., 2018)	-	ConvLM	3.08	9.94	3.26	10.47
TDNN+Kaldi (Xu et al., 2018)	-	4-gram	2.71	7.37	3.12	7.63
SpecAugment (Park et al., 2019)	-	-	-	-	2.8	6.8
SpecAugment (Park et al., 2019)	-	RNN	-	-	2.5	5.8
ContextNet (Han et al., 2020)	-	LSTM	1.9	3.9	1.9	4.1
Conformer (Gulati et al., 2020)	-	LSTM	2.1	4.3	1.9	3.9
960h - Self and semi-supervised learning						
Transf. + PL (Synnaeve et al., 2020)	LL-60k	CLM+Transf.	2.00	3.65	2.09	4.11
IPL (Xu et al., 2020b)	LL-60k	4-gram+Transf.	1.85	3.26	2.10	4.01
NST (Park et al., 2020)	LL-60k	LSTM	1.6	3.4	1.7	3.4
wav2vec 2.0 (Baevski et al., 2020c)	LL-60k	Transf.	1.6	3.0	1.8	3.3
wav2vec 2.0 + NST (Zhang et al., 2020b)	LL-60k	LSTM	1.3	2.6	1.4	2.6
Unsupervised learning						
wav2vec-U Large	LL-60k	4-gram	13.3	15.1	13.8	18.0
wav2vec-U Large + ST	LL-60k	4-gram	3.4	6.0	3.8	6.5
	LL-60k	Transf.	3.2	5.5	3.4	5.9

Table 2: WER on the Librispeech dev/test sets when using 960 hours of unlabeled audio data from Librispeech (LS-960) or 53.2k hours from Libri-Light (LL-60k) using representations from wav2vec 2.0 Large. Librispeech provides clean dev/test sets which are less challenging than the other sets. We report results for GAN training only (wav2vec-U) and with subsequent self-training (wav2vec-U + ST).

Baevski et al, Unsupervised Speech Recognition (2021)

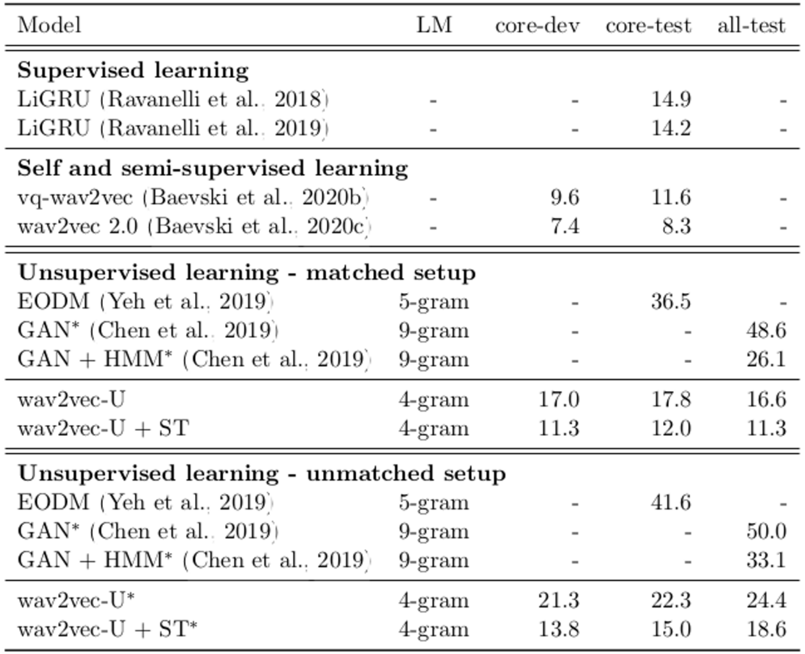
Benchmark data

Librispeech: 960h of transcribed audio. Here only the audio is used and not the transcriptions. Libri-Light has about 54h of audio data.

TIMIT: This dataset contains about five hours of audio recordings with time-aligned phonetic transcripts. To compare to prior work, two setups are considered:

- the matched setting uses text and speech from the same set of utterances to train the model (note that this is still unsupervised)
- unmatched setting ensures that the unlabeled text data does not contain the transcriptions of the audio data.

Results



Model	LM	core-dev	core-test	all-test
Supervised learning				
LiGRU (Ravanelli et al., 2018)	-	-	14.9	-
LiGRU (Ravanelli et al., 2019)	-	-	14.2	-
Self and semi-supervised learning				
vq-wav2vec (Baevski et al., 2020b)	-	9.6	11.6	-
wav2vec 2.0 (Baevski et al., 2020c)	-	7.4	8.3	-
Unsupervised learning - matched setup				
EODM (Yeh et al., 2019)	5-gram	-	36.5	-
GAN* (Chen et al., 2019)	9-gram	-	-	48.6
GAN + HMM* (Chen et al., 2019)	9-gram	-	-	26.1
wav2vec-U	4-gram	17.0	17.8	16.6
wav2vec-U + ST	4-gram	11.3	12.0	11.3
Unsupervised learning - unmatched setup				
EODM (Yeh et al., 2019)	5-gram	-	41.6	-
GAN* (Chen et al., 2019)	9-gram	-	-	50.0
GAN + HMM* (Chen et al., 2019)	9-gram	-	-	33.1
wav2vec-U*	4-gram	21.3	22.3	24.4
wav2vec-U + ST*	4-gram	13.8	15.0	18.6

Table 3: TIMIT Phoneme Error Rate (PER) in comparison to previous work for the matched and unmatched training data setups (§ 5.1). PER is measured on the standard Kaldi dev and test sets (core-dev/core-test) as well as a slightly larger version of the test set (all-test) as used by some of the prior work. (*) indicates experiments that do not use the standard split excluding SA utterances.

Almost Unsupervised Text to Speech and Automatic Speech Recognition

Ren et al, 2019 (ICML) [arXiv:1905.06791]

Four components:

- ❖ **Denoising auto-encoder**, which reconstructs speech and text sequences respectively
- ❖ **Dual transformation**, where the TTS model transforms the text y into speech $\hat{x}$ and the ASR model leverages the transformed pair $(\hat{x}, y)$

  for training

- ❖ **Bidirectional sequence modeling**, which addresses error propagation especially in long speech and text sequence when training with few paired data
- ❖ **Unified model structure,** which combines all the above components for TTS and ASR based on Transformer model.

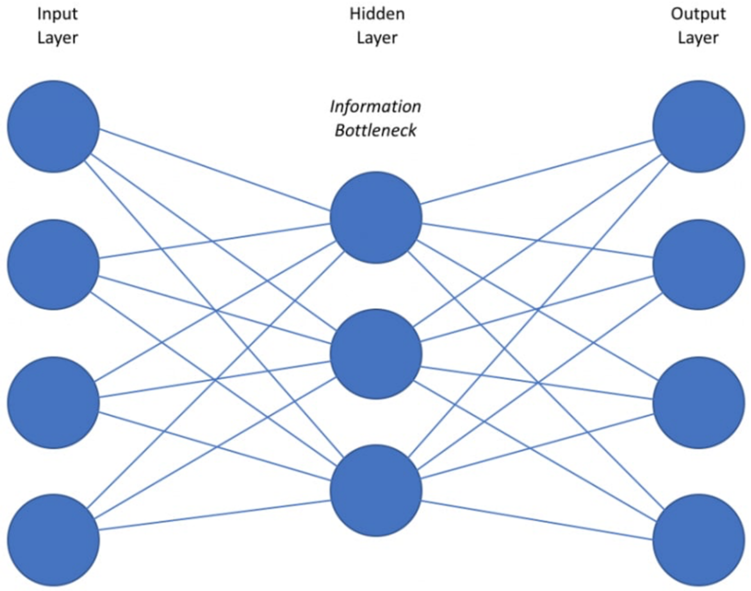
Denoising auto-encoders

$$\mathcal{L}^{dae} = \mathcal{L}_{\mathcal{S}}(x|C(x); \theta^{\mathcal{S}}_{enc}, \theta^{\mathcal{S}}_{dec}) + \mathcal{L}_{\mathcal{T}}(y|C(y); \theta^{\mathcal{T}}_{enc}, \theta^{\mathcal{T}}_{dec}),$$

$\theta^{\mathcal{S}}_{enc}, \theta^{\mathcal{S}}_{dec}$:Speech encoder & decoder parameters

$\theta^{\mathcal{T}}_{enc}, \theta^{\mathcal{T}}_{dec}$: Text encoder & decoder parameters.

$C(x), C(y)$:corruption of the speech & text sequences respectively



Corruption operation

$$C(x), C(y)$$

Corruption:

- randomly masks some elements with zero vectors, or
- swaps the elements in a certain window of the speech and text sequences

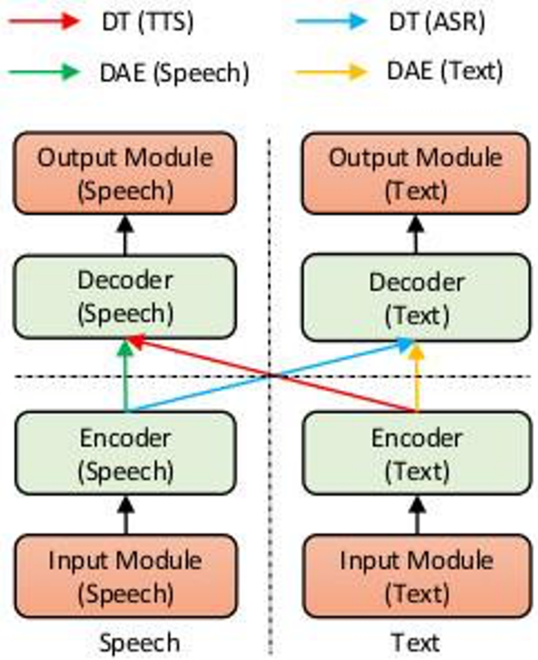
Loss Functions

$$\mathcal{L}_{\mathcal{S}}(y|x;\theta_{enc},\theta_{dec}) = \mathrm{MSE}(y, f(x;\theta_{enc},\theta_{dec})),$$

$$\mathcal{L}_{\mathcal{T}}(y|x;\theta_{enc},\theta_{dec}) = -\log P(y|x;\theta_{enc},\theta_{dec}))$$

X would be the logarithm of Mel filterbank energies of the Short Term Fourier Transform (STFT) of speech.

Overall Structure



Dual Transformation

Transform the speech sequence x into text sequence [illegible] using the ASR model an train TTS on $(\hat{y}, x)$

Likewise, train the ASR model on the transformed pair $(\hat{x}, y)$ generated by the TTS model.

$$\mathcal{L}^{dt} = \mathcal{L}_{\mathcal{S}}(x|\hat{y}; \theta_{enc}^{\mathcal{T}}, \theta_{dec}^{\mathcal{S}}) + \mathcal{L}_{\mathcal{T}}(y|\hat{x}; \theta_{enc}^{\mathcal{S}}, \theta_{dec}^{\mathcal{T}}),$$

where $\hat{y} = \arg\max P(y|x; \theta_{enc}^{\mathcal{S}}, \theta_{dec}^{\mathcal{T}})$ and $\hat{x} = f(y; \theta_{enc}^{\mathcal{T}}, \theta_{dec}^{\mathcal{S}})$ denote the text and speech

sequence transformed from speech x and text y respectively.

Bidirectional sequence modeling

- Error propagation in sequence to sequence learning during inference. Error in the **earlier part will be propagated to future tokens**.
- Solve this by generating sequences in both directions.
- Reformulate denoising auto-encoder and dual transformation

$$\mathcal{L}^{\overrightarrow{dae}} = \mathcal{L}_{\mathcal{S}}(\overrightarrow{x}|C(\overrightarrow{x}); \theta^{\mathcal{S}}_{enc}, \theta^{\mathcal{S}}_{dec})$$

$$+ \mathcal{L}_{\mathcal{T}}(\overrightarrow{y}|C(\overrightarrow{y}); \theta^{\mathcal{T}}_{enc}, \theta^{\mathcal{T}}_{dec}),$$

$$\mathcal{L}^{\overleftarrow{dae}} = \mathcal{L}_{\mathcal{S}}(\overleftarrow{x}|C(\overleftarrow{x}); \theta^{\mathcal{S}}_{enc}, \theta^{\mathcal{S}}_{dec})$$

$$+ \mathcal{L}_{\mathcal{T}}(\overleftarrow{y}|C(\overleftarrow{y}); \theta^{\mathcal{T}}_{enc}, \theta^{\mathcal{T}}_{dec})$$

Bidirectional DT

$$
\begin{aligned}
\mathcal{L}^{\overrightarrow{dt}} = \; & \mathcal{L}_{\mathcal{S}}(\overrightarrow{x} | \hat{\overrightarrow{y}}; \theta^{\mathcal{T}}_{enc}, \theta^{\mathcal{S}}_{dec}) \\
& + \mathcal{L}_{\mathcal{S}}(\overrightarrow{x} | R(\hat{\overleftarrow{y}}); \theta^{\mathcal{T}}_{enc}, \theta^{\mathcal{S}}_{dec}) \\
& + \mathcal{L}_{\mathcal{T}}(\overrightarrow{y} | \hat{\overrightarrow{x}}; \theta^{\mathcal{S}}_{enc}, \theta^{\mathcal{T}}_{dec}) \\
& + \mathcal{L}_{\mathcal{T}}(\overrightarrow{y} | R(\hat{\overleftarrow{x}}); \theta^{\mathcal{S}}_{enc}, \theta^{\mathcal{T}}_{dec}), \\
\mathcal{L}^{\overleftarrow{dt}} = \; & \mathcal{L}_{\mathcal{S}}(\overleftarrow{x} | \hat{\overleftarrow{y}}; \theta^{\mathcal{T}}_{enc}, \theta^{\mathcal{S}}_{dec}) \\
& + \mathcal{L}_{\mathcal{S}}(\overleftarrow{x} | R(\hat{\overrightarrow{y}}); \theta^{\mathcal{T}}_{enc}, \theta^{\mathcal{S}}_{dec}) \\
& + \mathcal{L}_{\mathcal{T}}(\overleftarrow{y} | \hat{\overleftarrow{x}}; \theta^{\mathcal{S}}_{enc}, \theta^{\mathcal{T}}_{dec}) \\
& + \mathcal{L}_{\mathcal{T}}(\overleftarrow{y} | R(\hat{\overrightarrow{x}}); \theta^{\mathcal{S}}_{enc}, \theta^{\mathcal{T}}_{dec}),
\end{aligned}
$$

$$
\begin{aligned}
\text{where } \hat{\overrightarrow{y}} &= \arg\max P(\overrightarrow{y} | \overrightarrow{x}; \theta^{\mathcal{S}}_{enc}, \theta^{\mathcal{T}}_{dec}), \\
\hat{\overleftarrow{y}} &= \arg\max P(\overleftarrow{y} | \overleftarrow{x}; \theta^{\mathcal{S}}_{enc}, \theta^{\mathcal{T}}_{dec}), \\
\hat{\overrightarrow{x}} &= f(\overrightarrow{y}; \theta^{\mathcal{T}}_{enc}, \theta^{\mathcal{S}}_{dec}) \\
\hat{\overleftarrow{x}} &= f(\overleftarrow{y}; \theta^{\mathcal{T}}_{enc}, \theta^{\mathcal{S}}_{dec})
\end{aligned}
$$

Bidirectional DT

Bidirectional sequence modeling based on denoising auto-encoder and dual transformation shares the models between left-to-right and right-to-left generations, i.e., they use the same set of parameter.s

This reduces the model parameter.

Bidirectional DT

TO give sense of which direction the sequence will be generated:

- two learnable embedding vectors as passed as two start elements for the decoder, representing the training and inference directions, one from left to right and the other from right to left.

- There are therefore four start embeddings in total, two for speech generation and the other two for text generation

Supervised Training

$$\mathcal{L}^{\overrightarrow{sup}} = \mathcal{L}_{\mathcal{S}}(\overrightarrow{x} \mid \overrightarrow{y}; \theta_{enc}^{\mathcal{T}}, \theta_{dec}^{\mathcal{S}}, (x, y) \in (\mathcal{S}, \mathcal{T}))$$

$$\mathcal{L}_{\mathcal{T}}(\overrightarrow{y} \mid \overrightarrow{x}; \theta_{enc}^{\mathcal{S}}, \theta_{dec}^{\mathcal{T}}, (x, y) \in (\mathcal{S}, \mathcal{T})),$$

$$\mathcal{L}^{\overleftarrow{sup}} = \mathcal{L}_{\mathcal{S}}(\overleftarrow{x} \mid \overleftarrow{y}; \theta_{enc}^{\mathcal{T}}, \theta_{dec}^{\mathcal{S}}, (x, y) \in (\mathcal{S}, \mathcal{T}))$$

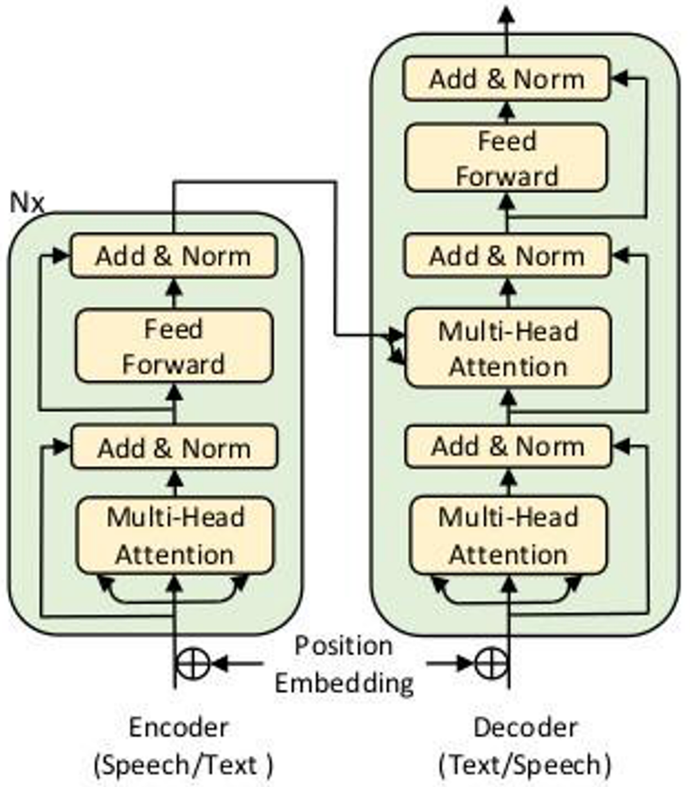
$$\mathcal{L}_{\mathcal{T}}(\overleftarrow{y} \mid \overleftarrow{x}; \theta_{enc}^{\mathcal{S}}, \theta_{dec}^{\mathcal{T}}, (x, y) \in (\mathcal{S}, \mathcal{T})),$$

Total Loss

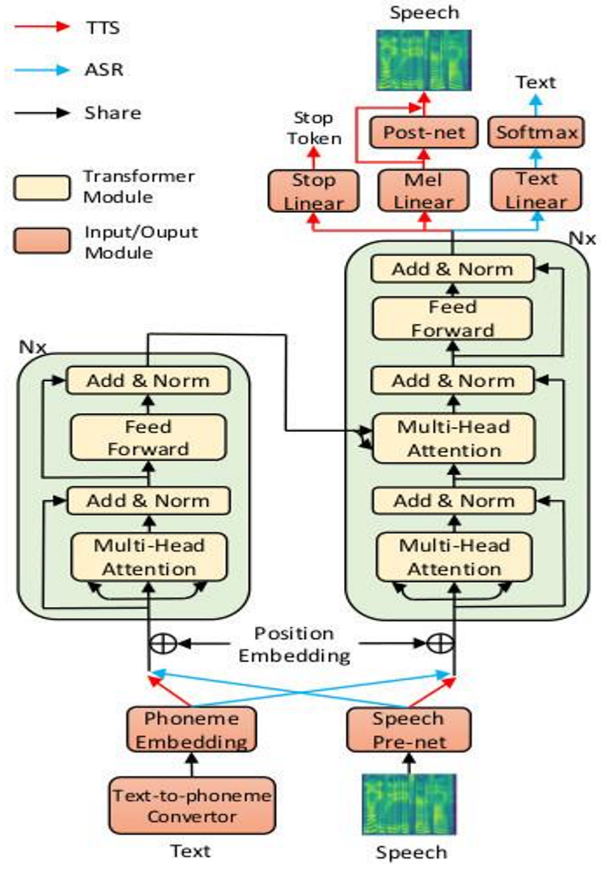
$$\mathcal{L} = \mathcal{L}^{\overrightarrow{dae}} + \mathcal{L}^{\overleftarrow{dae}} + \mathcal{L}^{\overrightarrow{dt}} + \mathcal{L}^{\overleftarrow{dt}} + \mathcal{L}^{\overrightarrow{sup}} + \mathcal{L}^{\overleftarrow{sup}}$$

Model Structure

Encoder and decoder structure are both transformers.



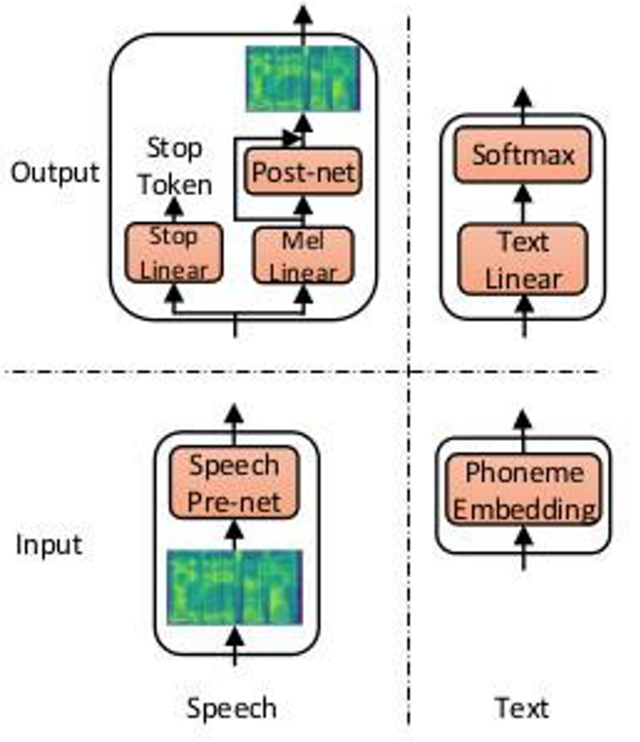
Model Structure



Input/Output Modules

- ❖ Speech Input module : pre-net 2-layer dense-connected, hidden size of 256, and the output dimension equals to the hidden size of Transformer.

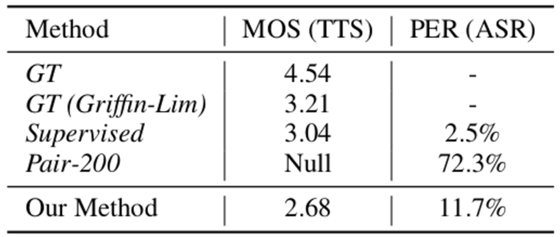
- ❖ Speech Output Module: Two components:
  - ➢ stop linear layer - predict if decoding should stop
  - ➢ mel linear layer +post-net to generate the mel-spectrogram with 80dim
- ❖ Text Input module: phoneme embedding.
- ❖ Text Output Module: share phoneme embedding with text linear layer.



Training

- LJSpeech: Paired 13,100 English audio clips and transcripts (24 hours).
- Unpaired data:
  - 12500 samples (training)
  - 300 samples (validation) set
  - 300 (test)
- 200 samples paired.
- Convert text into phonemes before feeding to model
- Upsample paired data in training
- Corruption operation is through masking.

Results



Method	MOS (TTS)	PER (ASR)
GT	4.54	-
GT (Griffin-Lim)	3.21	-
Supervised	3.04	2.5%
Pair-200	Null	72.3%
Our Method	2.68	11.7%

Table 1. The comparison between our method and other systems on the performance of TTS and ASR.

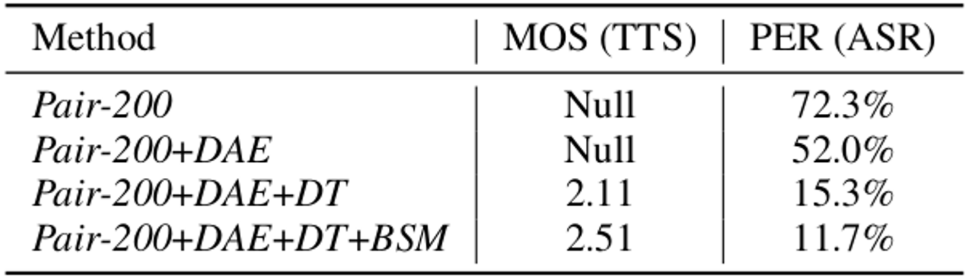
PER: Phoneme Error Rate
MOS: Mean Opinion Score

Pair-200: Using only paired data of the 200 samples

Supervised: Using all data with transcripts

GT(Griffin-Lim): Convert GT audio to mel-spectrogram and transform mel-spectrogram back to audio using the Griffin-Lin method. As all audio in this experiment uses GL, this **represents the upper bound.**

Results (Ablation)



Method	MOS (TTS)	PER (ASR)
Pair-200	Null	72.3%
Pair-200+DAE	Null	52.0%
Pair-200+DAE+DT	2.11	15.3%
Pair-200+DAE+DT+BSM	2.51	11.7%

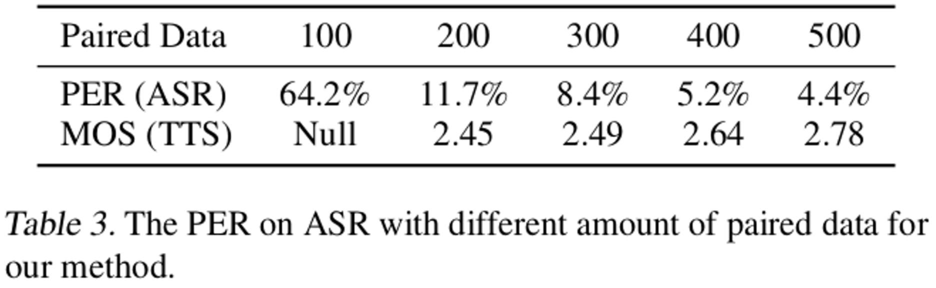
Table 2. Ablation studies on the components of our method.

PER: Phoneme Error Rate
MOS: Mean Opinion Score

Adding one component after another. Dual transformation brings about the greatest change but there is a significant improvement with the bidirectional sequence training.

Results (Ablation)

Increasing the amount of paired data.



Paired Data	100	200	300	400	500
PER (ASR)	64.2%	11.7%	8.4%	5.2%	4.4%
MOS (TTS)	Null	2.45	2.49	2.64	2.78

Table 3. The PER on ASR with different amount of paired data for our method.

Note how dramatically the PER reduces as we increase paired data from 100 to 200.

Paper 3: Segmental Empirical Output Distribution Matching

Chih-Kuan Yeh et al, 2019

- A segment of consecutive input samples (frames) that are associated

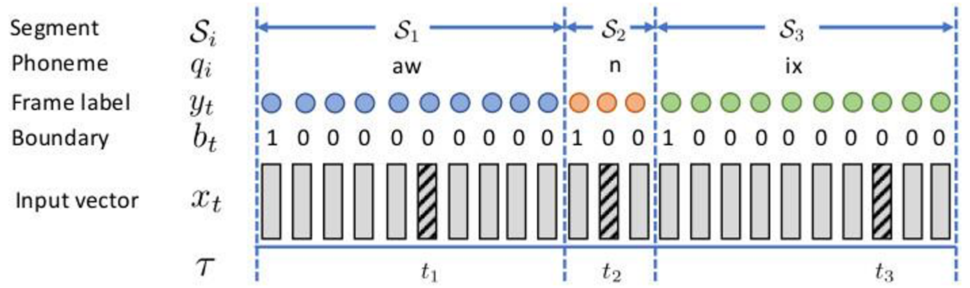
to the same phoneme label.

- Lengths and the boundaries of these segments are usually unknown a priori.

Key idea:

- Distribution of the predicted outputs across consecutive segments shall match the phoneme language model and
- Predicted outputs within each segment should be equal to each other as they belong to the same phoneme.

Segmental Empirical Output Distribution Matching



$\mathcal{S}_1, \ldots, \mathcal{S}_K$

Input x_t (m-dimensional acoustic vector) and frame-wise output y_t (phoneme label) amd the final phoneme sequence $q_1, \ldots, q_U$ from $y_1, \ldots, y_T$

Speech Segments

Frame-wise phoneme classifier: $p_\theta(y_t \mid x_t)$

Unsupervised algorithm to learn the classification model with a given set of segmentation boundaries b_t

Assuming there are K segments in the training data, $\tau = (t_1, \ldots, t_K)$
sampled one per segment from $S_1, \ldots, S_K$

Segmental Empirical Output Distribution Matching

N-gram language model: $p_{LM}(z) \equiv p_{LM}(q_{i-N+1} = z_1, \ldots, q_i = z_N)$

Subsequence that ends in t_i : $\tau_i = (t_{i-N+1}, \ldots, t_i)$

Cost function:

$$J_{\mathrm{ODM}}(\theta) = -\sum_{\tau \in \mathcal{S}_1 \times \cdots \times \mathcal{S}_K} \sum_{z \in \mathcal{Y}^N} p_{LM}(z) \ln \overline{p}_\theta^\tau(z)$$

Where $\overline{p}_\theta^\tau(z) = \frac{1}{K} \sum_{i=1}^{K} p_\theta(y_{\tau_i} = z | x_{\tau_i})$ $\qquad p_\theta(y_{\tau_i} = z | x_{\tau_i}) \triangleq \prod_{j=i-N+1}^{i} p_\theta(y_{t_j} = z_j | x_{t_j})$

Cost function takes a cross entropy form which attains a minimum when the two distributions are equal, i.e $p_{LM}(z) = \overline{p}_\theta^\tau(z)$
Minimization would narrow the difference between the empirical output inter-segment and that of the N-gram phomene language model while making no distinction between the timesteps within a segment.

Segmental Empirical Output Distribution Matching

Intra-segment distribution matching cost:

$$J_{\mathrm{FS}}(\theta) = \sum_{i=1}^{K} \sum_{t,t+1 \in \mathcal{S}_i} \sum_{y \in \mathcal{Y}} \left(p_\theta(y_t = y | x_t) - p_\theta(y_{t+1} = y | x_{t+1}) \right)^2$$

for frame-wise smoothness.

$$\min_\theta \{ J(\theta) \triangleq J_{\mathrm{ODM}}(\theta) + \lambda J_{\mathrm{FS}}(\theta) \}$$

Overall cost (Segmental Empirical-ODM)

Improving segment Boundary

Recognize that $b_t = \mathbb{I}(y_t \neq y_{t-1})$

Given an output sequence $(y_1, \cdots y_T)$ we can determine b_t?

Having determined θ one can compute the MAP estimate of y

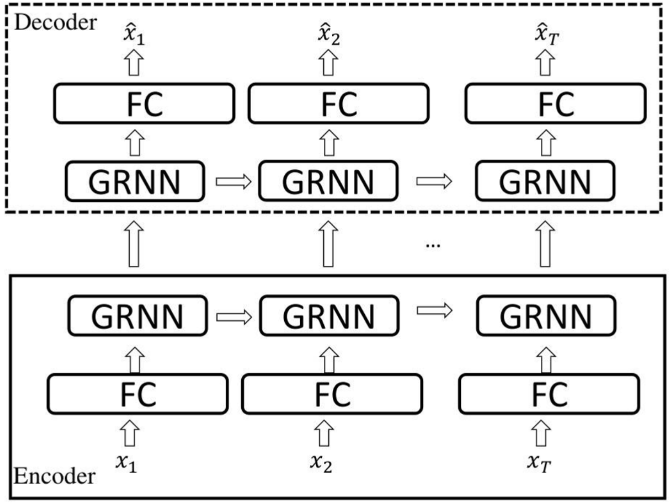
$$\arg\max_y p(y|x) = \arg\max_y \prod_{t=1}^{T} p(y_t|y_1, \ldots, y_{t-1}) \frac{p_\theta(y_t|x_t)}{p(y_t)}$$

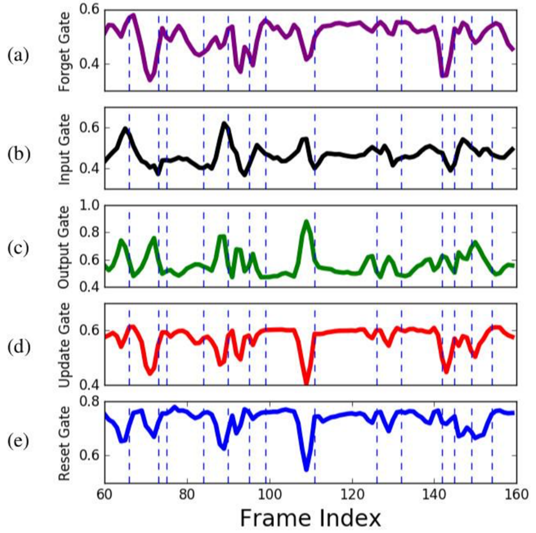
$$\begin{aligned} p(y_t|y_1,\ldots,y_{t-1}) &= \mathbb{I}(y_t = y_{t-1})p(y_t = y_{t-1}) + \mathbb{I}(y_t \neq y_{t-1})p(y_t \neq y_{t-1})p_{\mathrm{LM}}(q_i = y_t|q_{1:i-1}) \\ &= \mathbb{I}(y_t = y_{t-1})p(b_t = 0) + \mathbb{I}(y_t \neq y_{t-1})p(b_t = 1)p_{\mathrm{LM}}(q_i = y_t|q_{1:i-1}) \end{aligned}$$

Improving segment Boundary

Approximate $p(b_t = j) \sim p(b_t = j|x)$ for j=0,1

The conditional probability is obtained from a gated RNN auto-encoder described in Wang et al (arXiv:1703.07588, 2017)

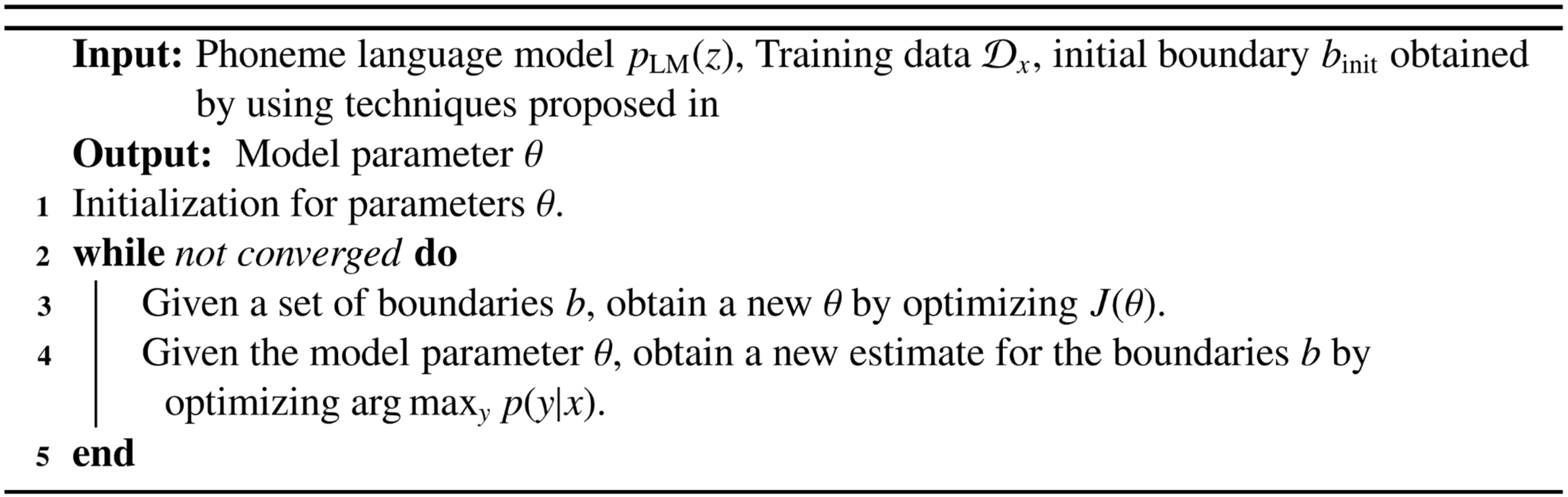




Improving Segment Boundary

Once the MAP estimate of y is obtained, they compute b_t.

The overall algorithm



```
Input: Phoneme language model $p_{\mathrm{LM}}(z)$, Training data $\mathcal{D}_x$, initial boundary $b_{\mathrm{init}}$ obtained
       by using techniques proposed in
Output: Model parameter $\theta$
1 Initialization for parameters $\theta$.
2 while not converged do
3     Given a set of boundaries $b$, obtain a new $\theta$ by optimizing $J(\theta)$.
4     Given the model parameter $\theta$, obtain a new estimate for the boundaries $b$ by
        optimizing $\arg\max_y p(y|x)$.
5 end
```

Using HMM in training

- To further improve performance, HMM model is used.
- However instead of using paired data (x,y), they use the unsupervised method to generate labels and then bootstrap the HMM-training with (x,\hat{y}).

Training Details

- 39 dim input feature vectors including 13 MFCC + double_difference obtained from 25 ms Hamming window at 10 ms interval hop.
- p θ (y t |x t) is modeled by a fully connected neural network with one hidden layer of 512 ReLU units.
- The input to the neural network is a concatenation of frames within a context window of size 11.

Evaluation Details

PER: Phoneme Error Rate, FER: Frame error rate

TIMIT data: training and validation sets of 3696 and 400 utterances

Two settings:

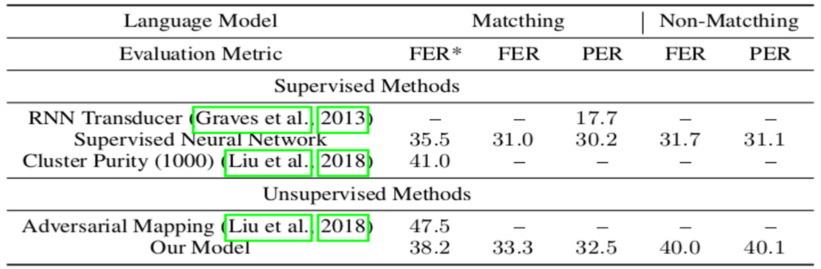
- matching language model: 3696 utterances to train our language modelp_{LM} (z)
- Non-matching language model: training (3000) and validation (1096) utterances. p_{LM} (z) is trained on 1096 while input the unsuperivsed model is 3000 set.

Further they consider:

(a) unsupervised with oracle phoneme boundary and

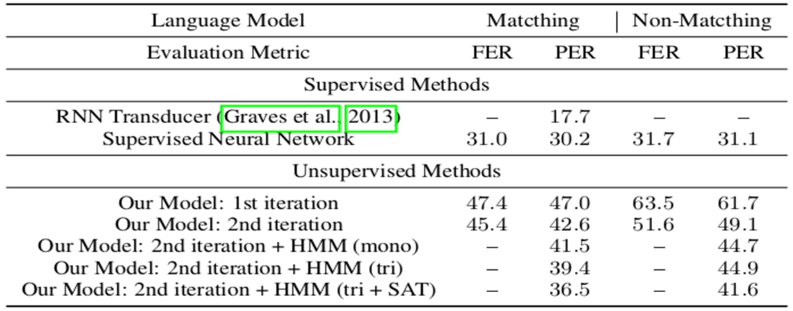
(b) fully unsupervised.

RESULTS



Language Model	Matcthing			Non-Matcthing	
Evaluation Metric	FER*	FER	PER	FER	PER
Supervised Methods					
RNN Transducer (Graves et al., 2013)	–	–	17.7	–	–
Supervised Neural Network	35.5	31.0	30.2	31.7	31.1
Cluster Purity (1000) (Liu et al., 2018)	41.0	–	–	–	–
Unsupervised Methods					
Adversarial Mapping (Liu et al., 2018)	47.5	–	–	–	–
Our Model	38.2	33.3	32.5	40.0	40.1

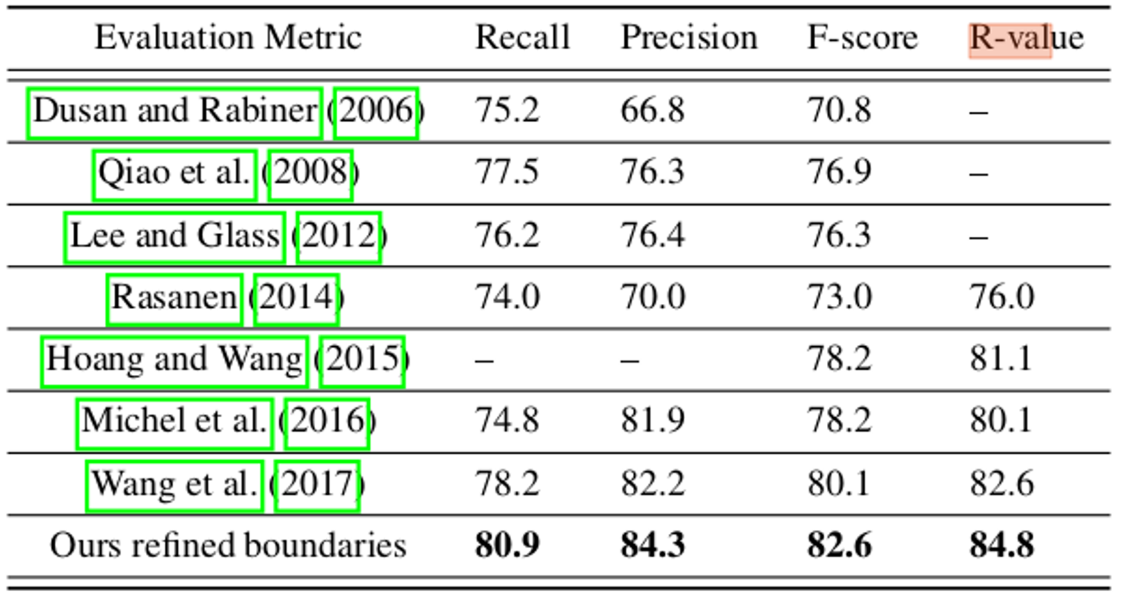
Table 1: Phoneme classification results when phoneme boundaries are given by a supervised oracle.



Language Model	Matcthing		Non-Matcthing	
Evaluation Metric	FER	PER	FER	PER
Supervised Methods				
RNN Transducer (Graves et al., 2013)	–	17.7	–	–
Supervised Neural Network	31.0	30.2	31.7	31.1
Unsupervised Methods				
Our Model: 1st iteration	47.4	47.0	63.5	61.7
Our Model: 2nd iteration	45.4	42.6	51.6	49.1
Our Model: 2nd iteration + HMM (mono)	–	41.5	–	44.7
Our Model: 2nd iteration + HMM (tri)	–	39.4	–	44.9
Our Model: 2nd iteration + HMM (tri + SAT)	–	36.5	–	41.6

Table 2: Results for fully unsupervised phoneme classification.

RESULTS (Unsupervised Phoneme Segmentation)



Evaluation Metric	Recall	Precision	F-score	R-value
Dusan and Rabiner (2006)	75.2	66.8	70.8	–
Qiao et al. (2008)	77.5	76.3	76.9	–
Lee and Glass (2012)	76.2	76.4	76.3	–
Rasanen (2014)	74.0	70.0	73.0	76.0
Hoang and Wang (2015)	–	–	78.2	81.1
Michel et al. (2016)	74.8	81.9	78.2	80.1
Wang et al. (2017)	78.2	82.2	80.1	82.6
Ours refined boundaries	**80.9**	**84.3**	**82.6**	**84.8**

Table 3: Results for unsupervised phoneme boundary segmentation.

Thank You

For more information visit

www.DeepThinking.AI or email omisonie@gmail.com