SEQUENCE-TO-SEQUENCE ASR OPTIMIZATION VIA REINFORCEMENT LEARNING

Andros Tjandra[1], Sakriani Sakti[1,2], Satoshi Nakamura[1,2]

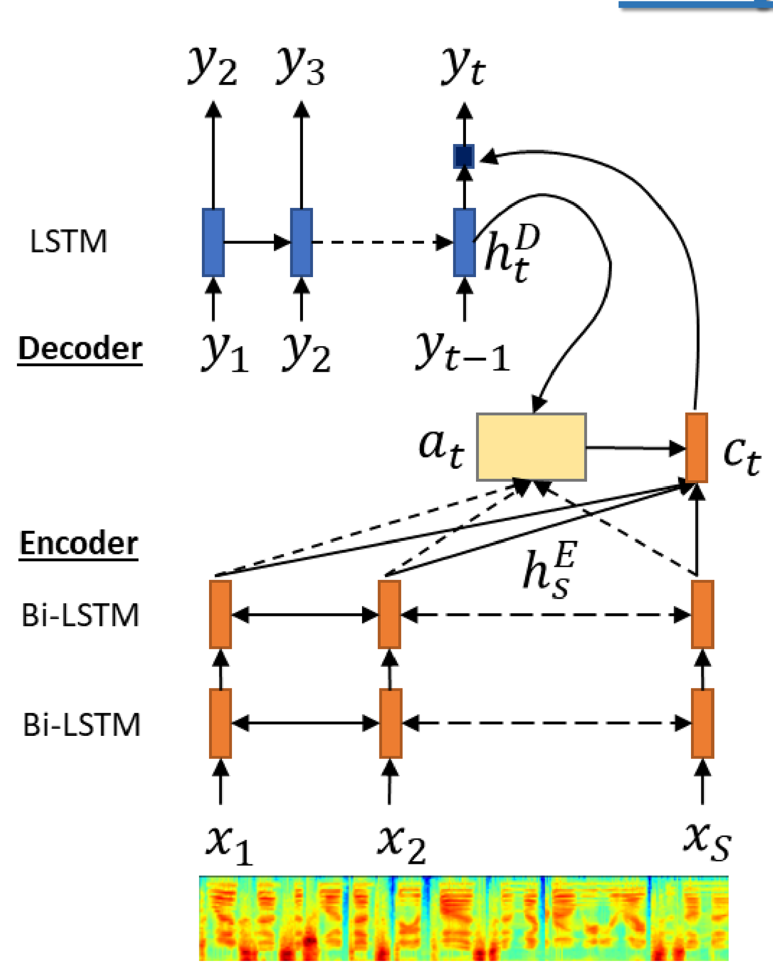
1) Nara Institute of Science and Technology, Japan 2) RIKEN, Center for Advanced Intelligence Project AIP, Japan

{andros.tjandra.ai6, ssakti, s-nakamura}@is.naist.jp

1. Introduction

- Sequence-to-sequence ASR is constructed by an encoder + an (autoregressive) decoder
- The autoregressive decoder is trained with **teacher-forcing** method
- Problem:
  - ✓ Generation: training (groundtruth) != inference (own model prediction)
  - ✓ Objective: maximum likelihood estimation (MLE) != task-specific metric (WER, CER)
- **Proposed an alternative optimization for seq2seq ASR via policy gradient**

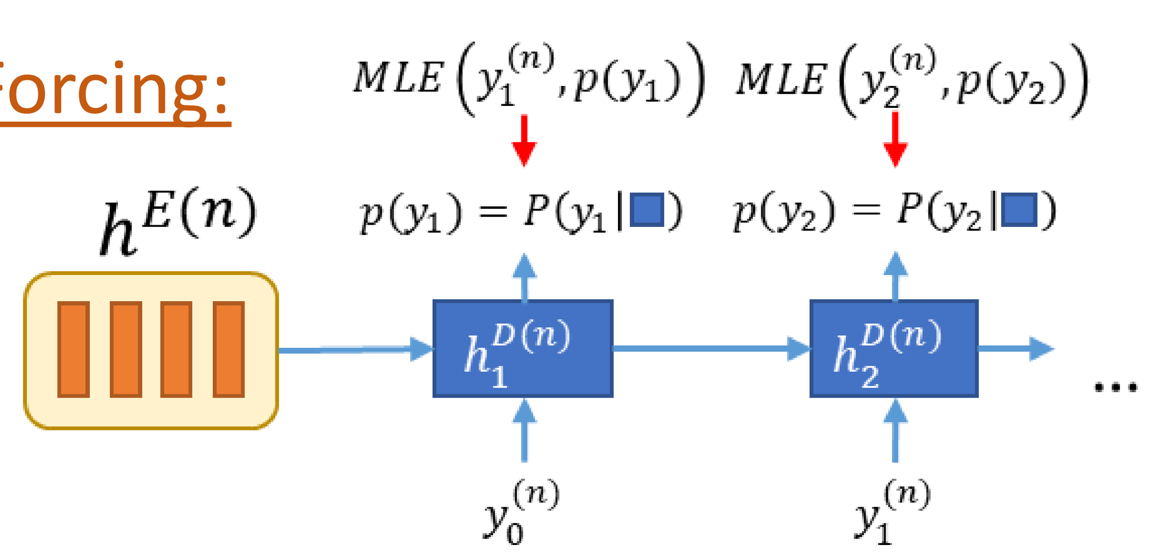
2. Seq2Seq ASR



$\boldsymbol{x} = [x_1, \dots, x_S]$ (speech feature)
$\boldsymbol{y} = [y_1, \dots, y_T]$ (text)
$h^e_{\{s,t\}}$ = encoder states
$h^d_{\{s,t\}}$ = decoder states
$a_{\{s,t\}}$ = attention probability
$c_{\{s,t\}}$ = weighted context vector

Encoder: 1 FC + 3 Bi-LSTM 256x2
Decoder: 1 LSTM 512
Attention: MLP scorer

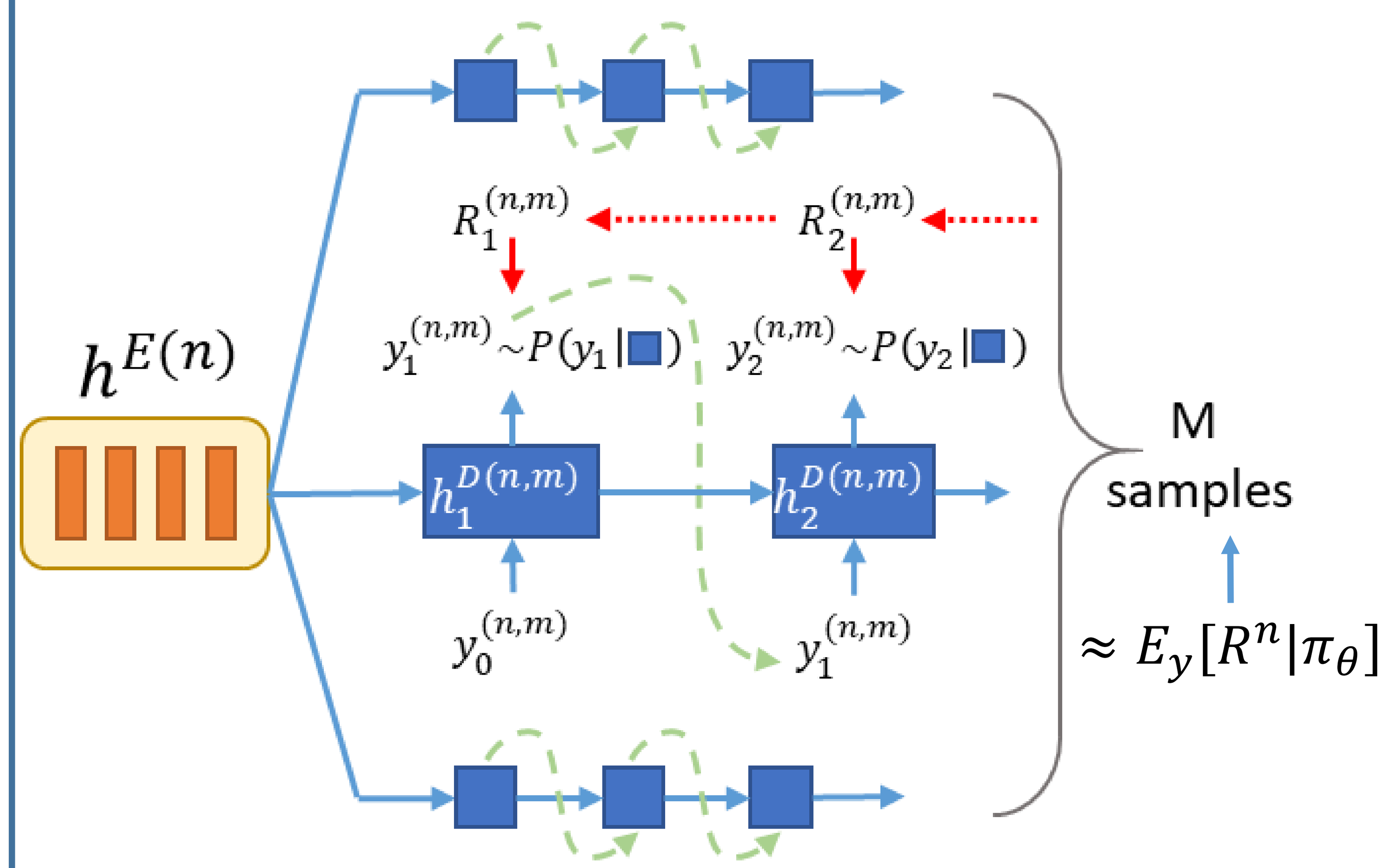
- Teacher Forcing:



$$\nabla_\theta MLE\left(y_t^n, p(y_t)\right) = \sum 1\{y_t^n = c\} * \nabla_\theta \log p(y_t = c)$$

3. Optimization with RL

- Policy gradient: RL algorithm for optimizing the expected rewards w.r.t a policy π_θ
- Given a pair speech-text $(x^{(n)}, y^{(n)})$, $R^{(n)}$ is the reward based on edit-distance (ED) between predicted y and ground-truth $y^{(n)}$



1. **Final reward (f-reward):**

$$R^{(n)} = -\mathrm{ED}\left(y, y^{(n)}\right)$$

$$\nabla_\theta E_y[R^n|\pi_\theta] = E_y\left[\nabla_\theta \log P\left(y|x^{(n)};\theta\right) R^{(n)}\right]$$

2. **Time distributed reward (t-reward):**

$$r_t^n = \begin{cases} -\left(ED\left(y_{1:t}, y^{(n)}\right) - ED\left(y_{1:t-1}, y^{(n)}\right)\right) & \text{if } t > 1 \\ -\left(ED\left(y_{1:t}, y^{(n)}\right) - \left|y^{(n)}\right|\right) & \text{if } t = 1 \end{cases}$$

$$R_t^{(n)} = \sum_{i=t}^{T} \gamma^{i-t} r_i(n)$$

$$\nabla_\theta E_y\left[\sum_{t=1}^T r_t^n \,|\pi_\theta\right] = E_y\left[\sum_{t=1}^{T} R_t^n \nabla_\theta \log P(y_t|y_{<t}, x^n;\theta)\right]$$

4. Experimental Setup & Results

- Features: log Mel-spec (40-dim + Δ + Δ^2)
- Text: 26 letters (A-Z)+(',-) + <noise> + <eos>
- Dataset: Wall Street Journal
  - ✓ Train: train_si84 (WSJ0) & train_si284 (WSJ1)
  - ✓ Dev: dev93 & Test: eval92
- RL hyperparameters: discount factor $\gamma = \{0, 0.5, 0.95\}$

Models (train_si284)	Result (CER%)
MLE	
CTC	8.97
Att Enc-Dec	7.69
MLE+RL	
Att Enc-Dec + RL (f-reward R)	7.26
Att Enc-Dec + RL (t-reward $R_t, \gamma = 0$)	6.64
Att Enc-Dec + RL (t-reward $R_t, \gamma = 0.5$)	6.37
Att Enc-Dec + RL (t-reward $R_t, \gamma = 0.95$)	6.10

5. Conclusion

- By treating our decoder as a policy network, we can
  (1) sample the whole transcription based on model's prediction in the training process
  (2) directly optimize the model with negative Levenshtein distance as the reward
- Our experiment shows by combining RL + MLE, we significantly improve the performance
- Best combination: MLE+RL time-distributed reward