CMU 11-785 Introduction to Deep Learning, Fall 2020

Lecture 17

Connectionist Temporal Classification

TAVE Research DL001

Changdae Oh

2021.04.11

Topics

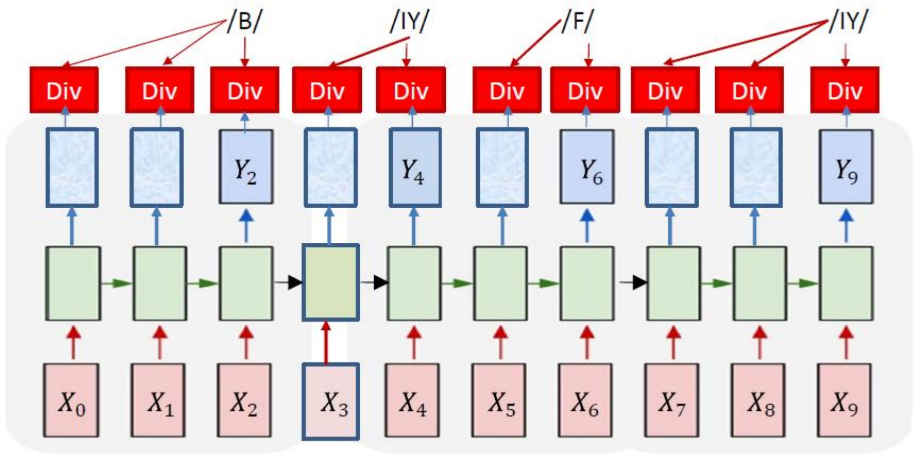
- Training the Seq2Seq without alignment
- Error over all possible alignments
- Using explicit blank symbol
- Decoding graph for the tree

Topics

- Training the Seq2Seq without alignment
- Error over all possible alignments
- Using explicit blank symbol
- Decoding graph for the tree

Training seq2seq model

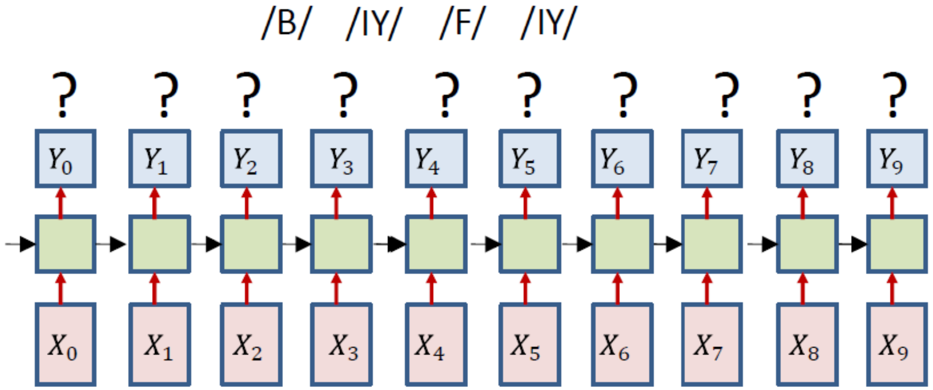
- If an alignment is given



$$DIV = \sum_t KL(Y_t, symbol_t) = -\sum_t \log Y(t, symbol_t)$$

$$\nabla_{Y_t} DIV = \begin{bmatrix} 0 & 0 & \dots & \dfrac{-1}{Y(t, symbol_t)} & 0 & \dots & 0 \end{bmatrix}$$

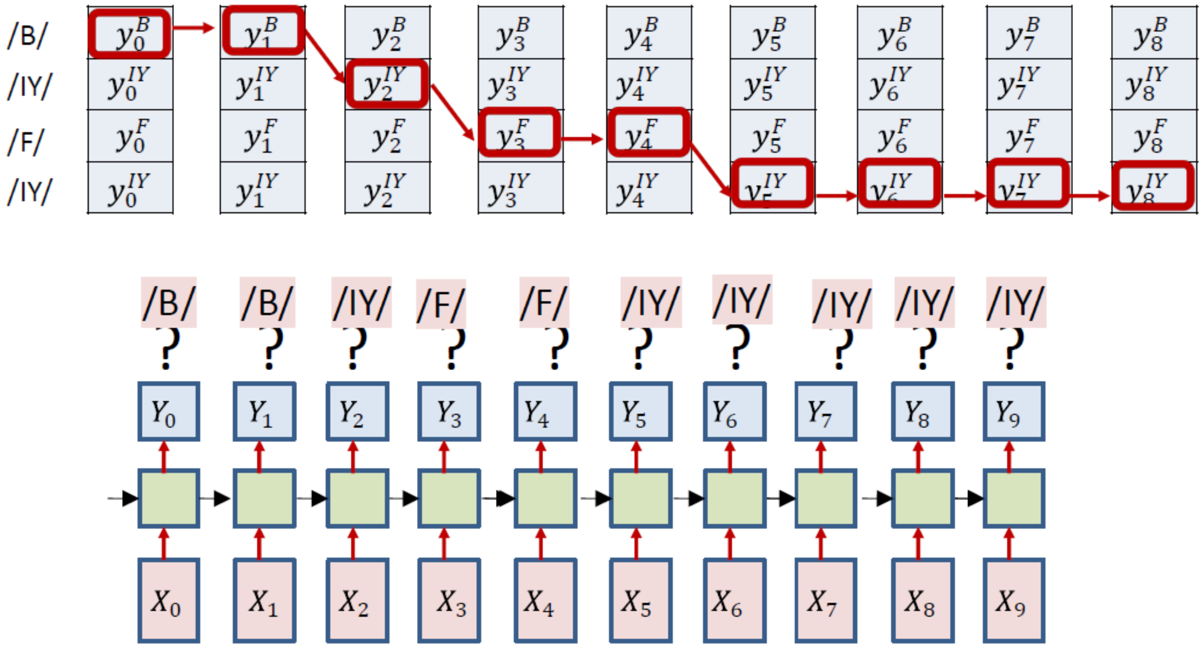
- However, it is not usually provided.



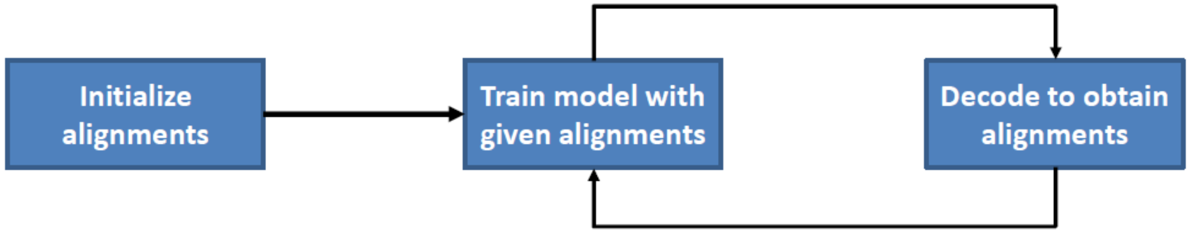
- Solution
  - ➢ Guess the alignment
  - ➢ Consider all possible alignments

Training seq2seq model

- Guess the alignment with 'Viterbi algorithm'



Iterative Estimate & Training



problems

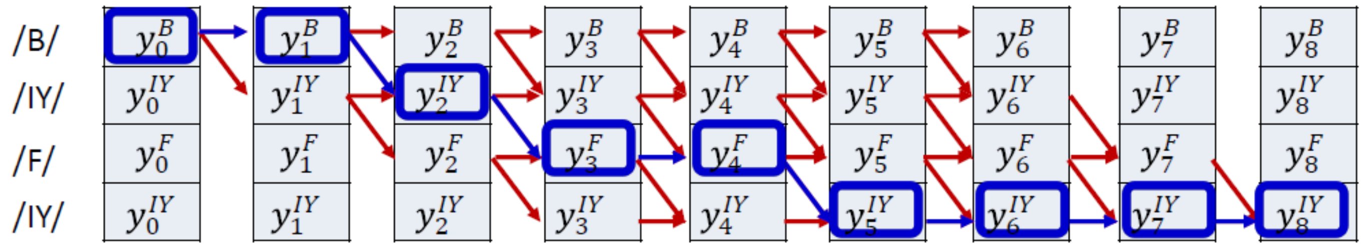
- Heavily dependent on initial alignment
- Poor local optima

Consider all possible alignments

Topics

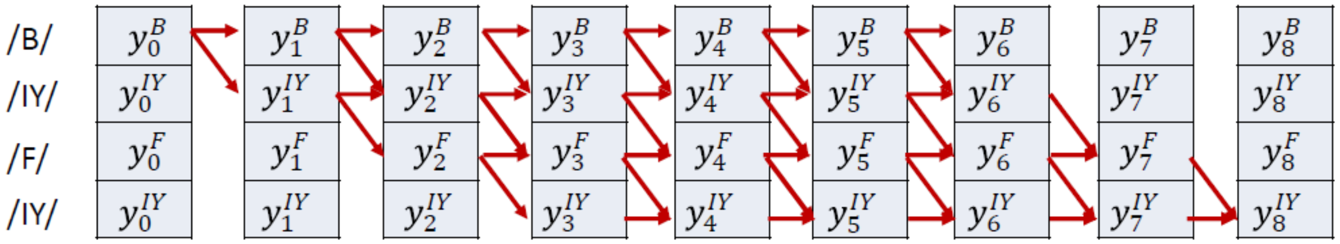
- ❖ Training the Seq2Seq without alignment
- ❖ Error over all possible alignments
- ❖ Using explicit blank symbol
- ❖ Decoding graph for the tree

Expectation over all possible alignments



$$DIV = -\sum_t \log Y\left(t, symbol_t^{bestpath}\right)$$

Each path have its own probability



$$DIV = E\left[-\sum_t \log Y(t, s_t)\right]$$

Averaging over all alignments

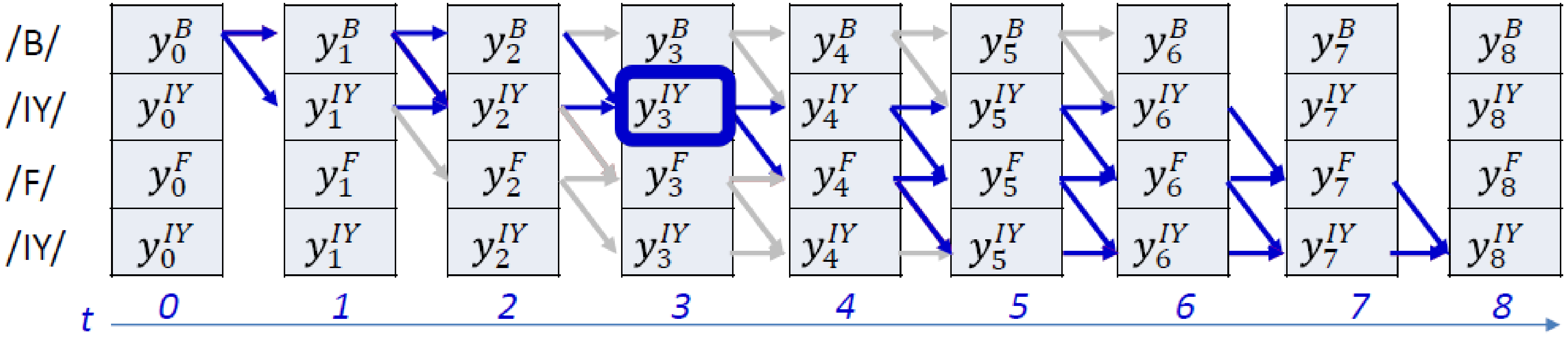
$$DIV = -\sum_t \sum_{S \in S_1 \dots S_K} P(s_t = S|\mathbf{S}, \mathbf{X}) \log Y(t, s_t = S)$$

How to compute?

Expectation over all possible alignments

$$DIV = -\sum_t \sum_{S \in S_1 \dots S_K} P(s_t = S|\mathbf{S},\mathbf{X}) \log Y(t, s_t = S)$$

How to compute?



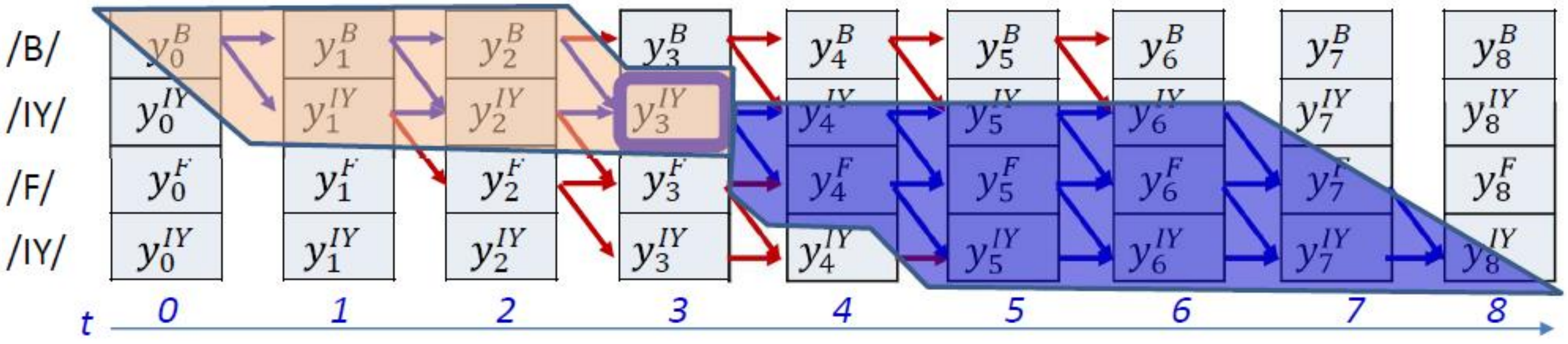
Bayes rule

$$P(s_t = S_r|\mathbf{S},\mathbf{X}) \propto \underline{P(s_t = S_r, \mathbf{S}|\mathbf{X})}$$

$$= P(S_0, \dots, S_{K-1}, s_t = S_r|\mathbf{X})$$

- Joint probability of obtaining target sequence and aligning a symbol to time t

Calculating a posteriori symbol probability



$$P(s_t = S_r, \mathbf{S}|\mathbf{X})$$

$$= \underbrace{P(S_0 \ldots S_r, s_t = S_r\,|\mathbf{X})}_{\alpha(t,r)} \underbrace{P(s_{t+1} \in succ(S_r), succ(S_r), \ldots, S_{K-1}\,|\mathbf{X})}_{\beta(t,r)}$$

Forward algorithm

$$\alpha(t,r) = \sum_{q:S_q \in pred(S_r)} P(subgraph\ ending\ at\ (t-1,q)) Y_t^{S(r)}$$

$$\alpha(t,r) = \sum_{q:S_q \in pred(S_r)} \alpha(t-1,q)\, Y_t^{S(r)}$$

- Initialization:

  $\hat{\alpha}(0,0) = 1, \quad \hat{\alpha}(0,r) = 0, \quad r > 0$

  $\alpha(0,r) = \hat{\alpha}(0,r) y_0^{S(r)}, \quad 0 \le r \le K-1$

- for $t = 1 \ldots T-1$

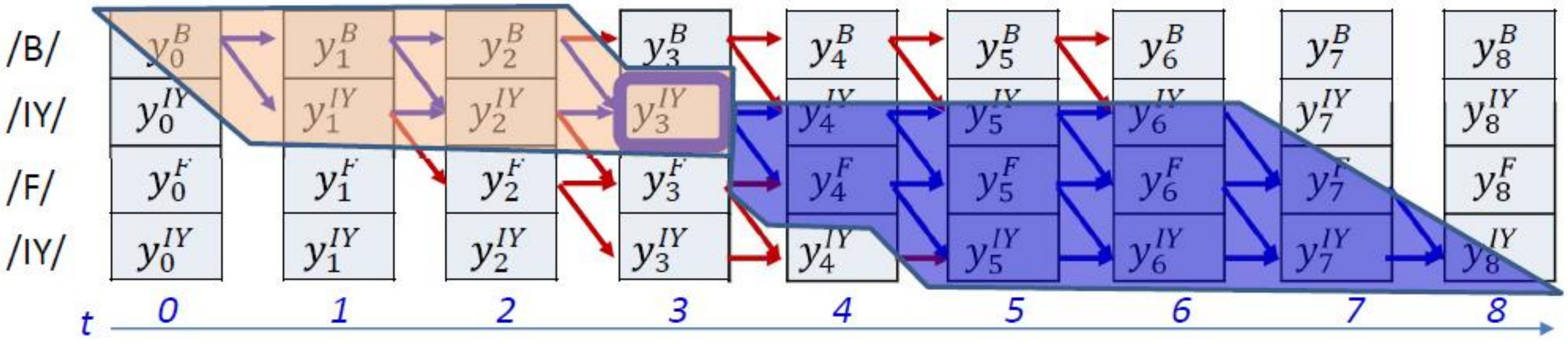
  $\hat{\alpha}(t,0) = \alpha(t-1,0)$

  for $l = 1 \ldots K-1$

  - $\hat{\alpha}(t,l) = \alpha(t-1,l) + \alpha(t-1,l-1)$

  $\alpha(t,r) = \hat{\alpha}(t,r) y_t^{S(r)}, \quad 0 \le r \le K-1$

Calculating a posteriori symbol probability



$$P(s_t = S_r, \mathbf{S}|\mathbf{X})$$

$$= \underbrace{P(S_0 \ldots S_r, s_t = S_r\, |\mathbf{X})}_{\alpha(t,r)} \underbrace{P(s_{t+1} \in succ(S_r), succ(S_r), \ldots, S_{K-1}\, |\mathbf{X})}_{\beta(t,r)}$$

backward algorithm

- Initialization:

  $\beta(T-1, K-1) = 1,\ \beta(T-1, r) = 0,\ r < K-1$

- for $t\ = T-2\ downto\ 0$

  $\beta(t, K)\ = \beta(t+1, K)y_{t+1}^{S(K)}$

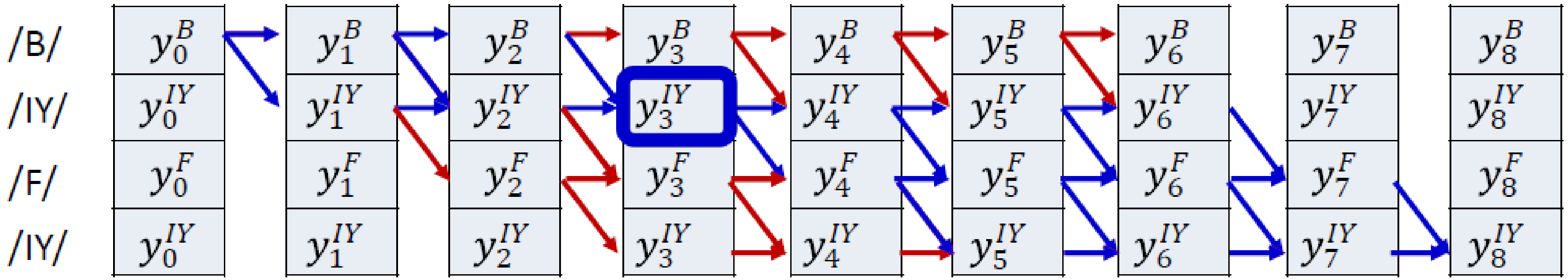
  for $r\ = K-2 \ldots 0$

  - $\beta(t,r) = y_{t+1}^{S(l)}\beta(t+1,r)\ +\ y_{t+1}^{S(r+1)}\beta(t+1,r+1)$

$$\hat{\beta}(t,r) = y_t^{S(r)}(\hat{\beta}(t+1,r)\ + \hat{\beta}(t+1,r+1))$$

$$\beta(t,r) = \sum_{q:S_q \in succ(S_r)} \beta(t+1,q) y_{t+1}^{S_q}$$

Calculating a posteriori symbol probability



$$P(s_t = S_r|\mathbf{S},\mathbf{X}) = \frac{P(s_t = S_r, \mathbf{S}|\mathbf{X})}{\sum_{S_r'} P(s_t = S_r', \mathbf{S}|\mathbf{X})} = \frac{\alpha(t,r)\beta(t,r)}{\sum_{r'} \alpha(t,r')\beta(t,r')} = \gamma(t,r)$$

- Our objective

$$DIV = -\sum_t \sum_{s \in S_0 \ldots S_{K-1}} P(s_t = s|\mathbf{S},\mathbf{X}) \log Y(t, s_t = s)$$

$$DIV = -\sum_t \sum_r \gamma(t,r) \log y_t^{S(r)}$$

$$\nabla_{Y_t} DIV = \begin{bmatrix} \frac{dDIV}{dy_t^{S_0}} & \frac{dDIV}{dy_t^{S_1}} & \cdots & \frac{dDIV}{dy_t^{S_{L-1}}} \end{bmatrix}$$

$$\frac{dDIV}{dy_t^l} = -\frac{1}{y_t^l} \sum_{r : S(r) = l} \gamma(t,r)$$

(approximation)

Overall training procedure for seq2seq

1. Setup the network.
2. Initialize all parameters.
3. Pass the training instance through the network and obtain all symbol probabilities at each time.
4. Construct the graph representing the specific symbol sequence in the instance.
5. Perform the forward-backward algorithm to compute gamma(t, r) for each row of nodes at each time.
6. Compute derivative of DIV for each time output.
7. Backprop derivative of DIV

Note,

- Training can be performed by iteratively estimating the alignment by Viterbi-decoding.
- Alternately, it can be performed by optimizing the expected error over all possible alignments.
- Inference is another matter.

Topics

- ❖ Training the Seq2Seq without alignment
- ❖ Error over all possible alignments
- ❖ Using explicit blank symbol
- ❖ Decoding graph for the tree

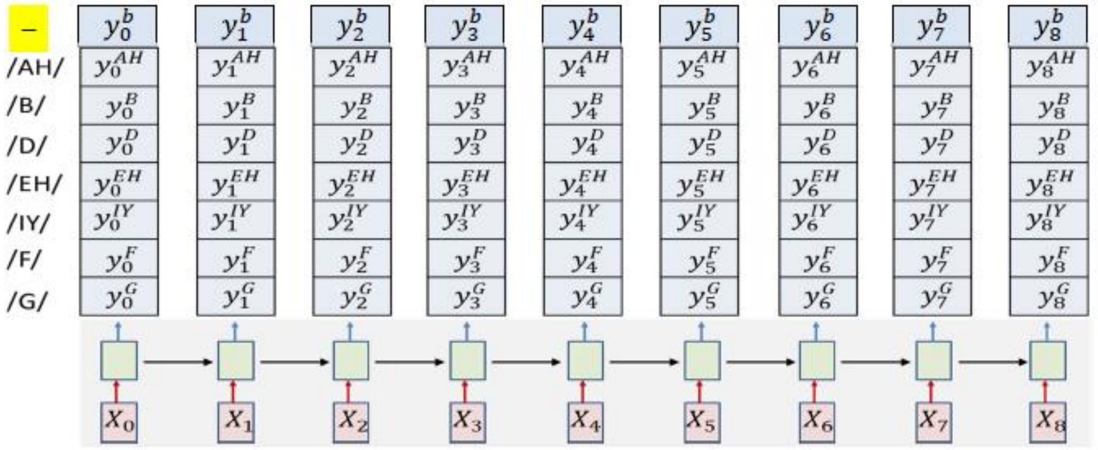
Decode with extra 'blank' symbol

- Decode : R R R E E E E D
- RED? REED?

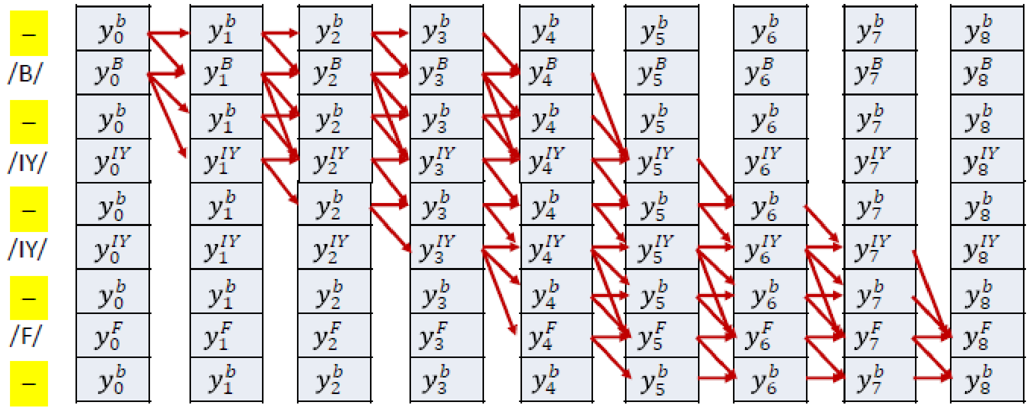
Solution

- Introduce an explicit extra symbol 'blank' which serves to separate discrete versions of a symbol

At inference



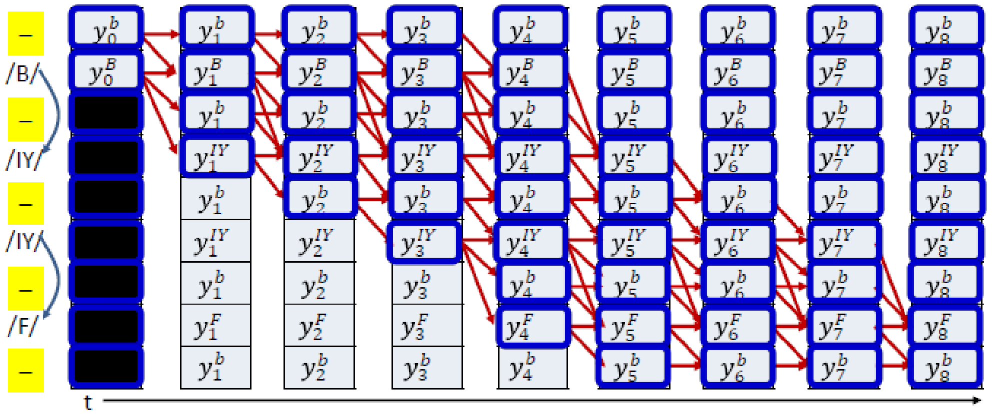
At training



- Skips are permitted across a blank, but only if the symbols on either side are different.

Decode with extra ‘blank’ symbol

Modified forward algorithm



- Iteration:

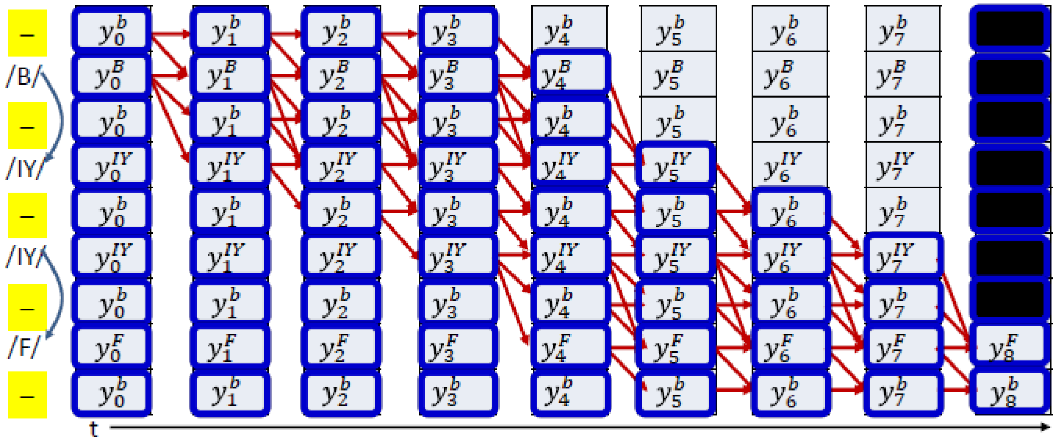
  $\alpha(t,r) = \left(\alpha(t-1,r) + \alpha(t-1,r-1)\right)y_t^{S(r)}$

  - If $S(r) = " - "$ or $S(r) = S(r-2)$

  $\alpha(t,r) = \left(\alpha(t-1,r) + \alpha(t-1,r-1) + \alpha(t-1,r-2)\right)y_t^{S(r)}$

  - Otherwise

Modified backward algorithm



$\beta(t,r) = \sum_{q:S_q \in succ(S_r)} \beta(t+1,q)y_{t+1}^{S_q}$

- Iteration:

  $\beta(t,r) = \beta(t+1,r)y_{t+1}^{S(r)} + \beta(t+1,r+1)y_{t+1}^{S(r+1)}$

  - If $S(r) = " - "$ or $S(r) = S(r+2)$

  $\beta(t,r) = \beta(t+1,r)y_{t+1}^{S(r)} + \beta(t+1,r+1)y_{t+1}^{S(r+1)} + \beta(t+1,r+2)y_{t+1}^{S(r+2)}$

  - Otherwise

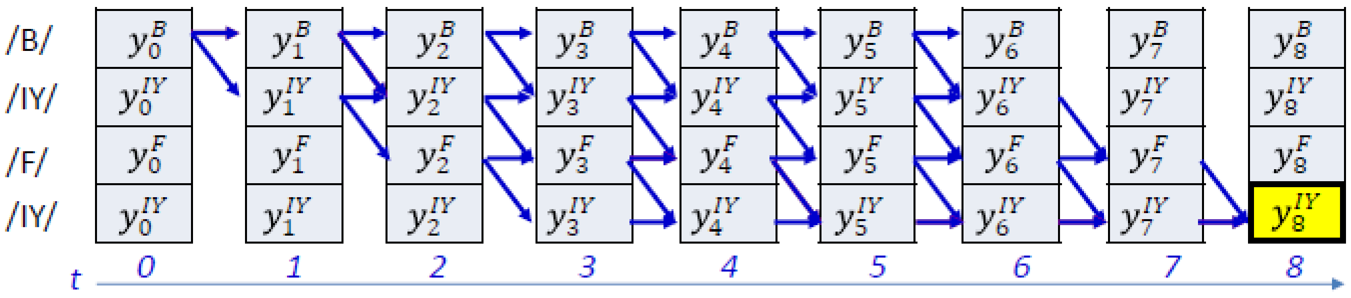
Topics

- ❖ Training the Seq2Seq without alignment
- ❖ Error over all possible alignments
- ❖ Using explicit blank symbol
- ❖ Decoding graph for the tree

Greedy decodes are suboptimal

- R R – E E D (RED, 0.7)
- R R – – E D (RED, 0.68)
- R R E E E D (RED, 0.69)
- T T E E E D (TED, 0.71) Greedy decoding pool solution ...
- T T – E E D (TED, 0.3)
- T T – – E D (TED, 0.29)

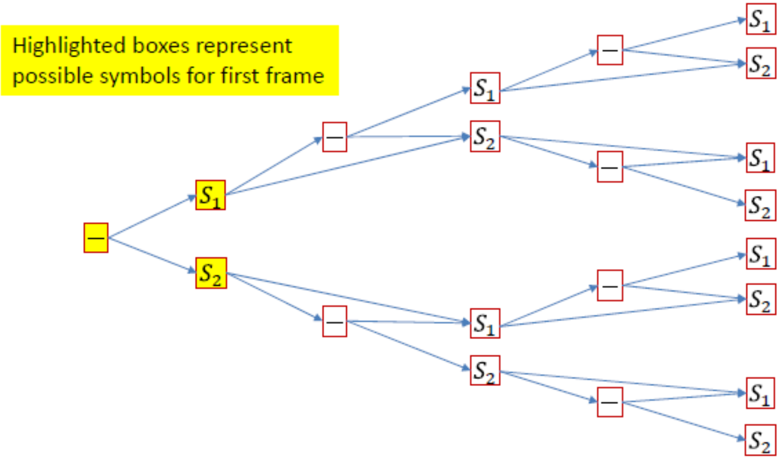
We hope to find : $\hat{\mathbf{S}} = \underset{\mathbf{S}}{\operatorname{argmax}}\, \alpha_{\mathbf{S}}(S_{K-1}, T-1)$

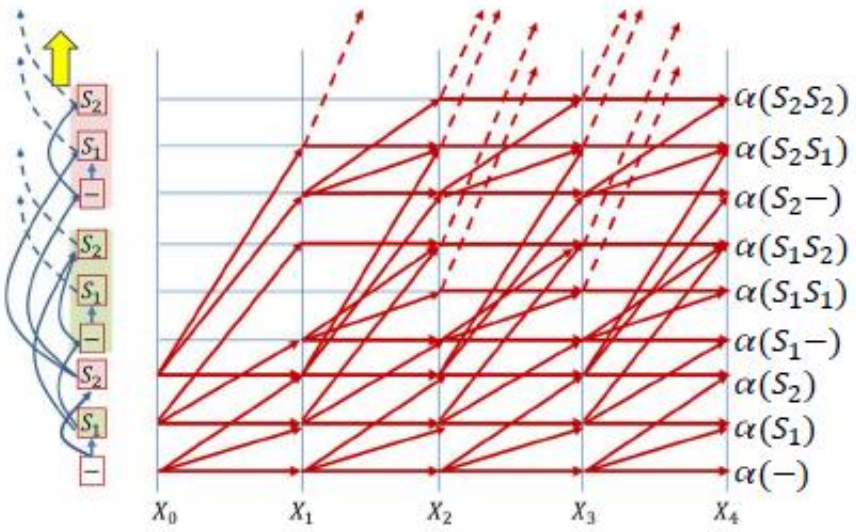


Exponential computation is required ...

$$\alpha_{S_0..S_{K-1}}(T-1, K-1) = P(S_0..S_{K-1}|\mathbf{X})$$

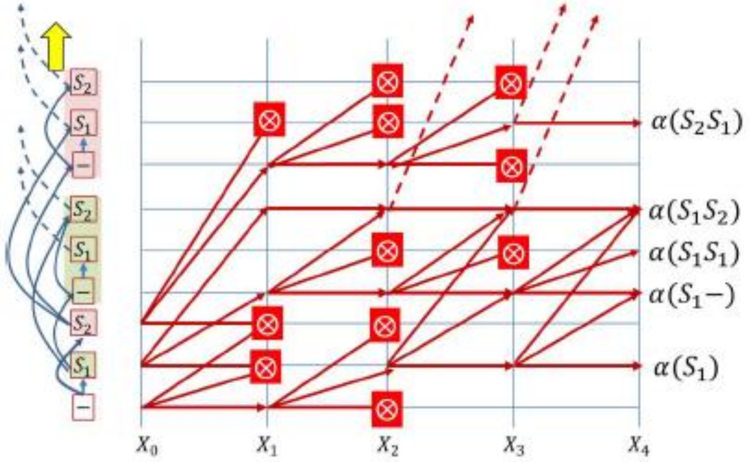
Tree structure





- The forward score alpha(r, T) at the final time represents the full forward score for a unique symbol sequence.
- Select the symbol sequence with the largest alpha at the final time.

But this also still requires a lot of computation...



- Use Beam Search !