GOP-CTC-SF-Norm

January 2026

1 Introduction

This document illustrate how do we compute the proposed method GOP-CTC-SF-Norm, as defined below:

$$\text{GOP-SF-Norm}(l_i) = \frac{\log\left(p(l_i|O_1^T, L_{\mathrm{L}}, L_{\mathrm{R}})\right)}{\mathbb{E}[t_2 - t_1|O_1^T, L_{\mathrm{L}}, L_{\mathrm{R}}]}. \tag{1}$$

where the numerator is unnormalized version that can be calculated as:

$$\text{GOP-SF}(l_i) = \mathcal{L}_{CTC}(L_{\mathrm{SDI}}, O_1^T) - \mathcal{L}_{CTC}(L_{\mathrm{C}}, O_1^T). \tag{2}$$

Next, we show the pseudo-code for computing the terms needed for Eq.1, specifically for the version "GOP-SF-SD-numerical".

2 Details of the dynamic programming

We illustrate the methods using pseudo code and note that:

- For simplified illustration, this pseudo-code does not cover any edge or special condition, e.g., the target position $i \neq 1 \wedge i \neq L$ is assumed, or when two subsequent identical tokens are in L. Please refer to the github-repo for the full python implementation.
- Some special operations are needed to prune duplications or to ensure deterministic paths. This is required for the normalized version. Please refer to the github-repo for more details of implementing the normalized version.
- For the time being, this normalized version only supports GOP-SF-S/-SD, not -SDI), the following discussion is targeting on only GOP-SF-SD-Norm. Check the githut-repo for other variants which contains the versions using un-normalized forward variables.

2.1 Computing $\mathcal{L}_{CTC}(L_{\mathbf{SDI}}, O_1^T)$

Algorithm 1 CTC Posterior Probability Computation with Normalization

Require: Input features $\mathbf{O} = (o_1, \ldots, o_T)$
Require: Token sequence $\mathbf{L} = (l_1, \ldots, l_L)$
Require: Network outputs $y(t, v)$, per-frame probability $p(v|o_t)$ for $v \in V$
Create extended token sequence $\mathbf{L}' = (\phi, l_1, \phi, l_2, \ldots, l_L, \phi)$ of length $|\mathbf{L}'| = 2L + 1$
// **Initialize forward variables**
$\alpha \leftarrow \mathbf{0}_{T\times(2L+1)}$ ▷ Initialize to zeros
$\ln p(\mathbf{L}|\mathbf{O}) \leftarrow 0$ ▷ Initialize log probability
// **Set initial forward variables at** $t = 1$
$\alpha(1, 1) \leftarrow y(1, \text{blank})$ ▷ Start with blank
$\alpha(1, 2) \leftarrow y(1, l_1)$ ▷ Start with first token
$C_1 \leftarrow \sum_{s=1}^{|\mathbf{l}'|} \alpha(1, s)$
$\ln p(\mathbf{L}|\mathbf{O}) \leftarrow \ln p(\mathbf{L}|\mathbf{O}) + \ln(C_1)$
$\alpha(1, :) \leftarrow \frac{\alpha(1,:)}{C_1}$ ▷ Normalize: $\hat{\alpha}(1, s)$ for all s
// **Dynamic programming**
for $t = 2$ to T **do**
  start $\leftarrow \max(|L'| - 2 * (T - t), 0)$ ▷ Crucial for CTC with normalization
  for $s =$ start to $|\mathbf{L}'|$ **do**
    if $l'_s =$ blank OR $l'_s = l'_{s-2}$ **then**
      $\alpha(t, s) \leftarrow (\alpha(t-1, s) + \alpha(t-1, s-1)) \cdot y(t, l'_s)$
    else
      $\alpha(t, s) \leftarrow (\alpha(t-1, s) + \alpha(t-1, s-1) + \alpha(t-1, s-2)) \cdot y(t, l'_s)$
    end if
  end for
  $C_t \leftarrow \sum_{s=1}^{|\mathbf{l}'|} \alpha(t, s)$
  $\ln p(\mathbf{L}|\mathbf{O}) \leftarrow \ln p(\mathbf{L}|\mathbf{O}) + \ln(C_t)$
  $\alpha(t, :) \leftarrow \frac{\alpha(t,:)}{C_t}$ ▷ Normalize: $\hat{\alpha}(t, s)$ for all s
end for
return $\ln p(\mathbf{L}|\mathbf{O})$

2.2 Computing $\mathcal{L}_{CTC}(L_{\mathbf{C}}, O_1^T)$ and $\mathbb{E}[t_2 - t_1 | O_1^T, L_{\mathbf{L}}, L_{\mathbf{R}}]$

Algorithm 2 GOP-SF-SD denominator for the target position

Require: The target position **pos** for extension
Require: Input features $\mathbf{O} = (o_1, \ldots, o_T)$
Require: Token sequence $\mathbf{L} = (l_1, \ldots, l_L)$
Require: Network outputs $y(t, v)$, represents per-frame probability $p(v|o_t)$ for $v \in V$

Create extended token sequence $\mathbf{L}' = (\phi, l_1, \ldots, \phi, *, \phi..., l_L, \phi)$.
// **Initialize forward variables, the last dimension is only used for the target token of** $*$
$\alpha \leftarrow \mathbf{0}_{T\times(2L+1)\times V}$ ▷ Initialize to zeros
$\ln p(\mathbf{L}|\mathbf{O}) \leftarrow 0$ ▷ Initialize log probability
// **Set initial forward variables at** $t = 1$
$\alpha(1, 1, 1) \leftarrow y(1, \text{blank})$ ▷ Start with blank
$\alpha(1, 2, 1) \leftarrow y(1, l_1)$ ▷ Start with first token
$C_1 \leftarrow \text{sum}(\alpha(1, :, :))$
$\ln p(\mathbf{L}|\mathbf{O}) \leftarrow \ln p(\mathbf{L}|\mathbf{O}) + \ln(C_1)$
$\alpha(1, :, :) \leftarrow \frac{\alpha(1,:,:)}{C_1}$ ▷ Normalize: $\hat{\alpha}(1, s)$ for all s
// **Dynamic programming**
$\text{mask} \leftarrow \mathbf{0}_{V\times V}$ ▷ Create the mask for GOP-SF-SDI calculation
$\text{diag}(M) \leftarrow \mathbf{1}_{|v|}$ ▷ Diagonal elements only, excluding insertions
for $t = 2$ to T **do**
  $\text{start} \leftarrow \max(|L'| - 2 * (T - t), 0)$ ▷ Crucial for CTC with normalization
  for $s = \text{start}$ to $|L'|$ **do**
    if $l'_s = \text{blank}$ **then**
      $\alpha(t, s, 1) \leftarrow (\alpha(t-1, s, 1) + \text{sum}(\alpha(t-1, s-1, :)) \cdot y(t, l'_s)$
    else if $l'_{s-2} = *$ **then**
      $\text{del} \leftarrow \alpha(t-1, s-3, 1) + \alpha(t-1, s-4, 1)$ ▷ Merge deletion paths
      $\text{subst} = \text{sum}(\alpha(t-1, s-2, :))$ ▷ Merge substitution paths
      $\alpha(t, s, 1) \leftarrow (\alpha(t-1, s, 1) + \alpha(t-1, s-1, 1) + \text{subst} + \text{del})) \cdot y(t, l'_s)$
    else if $l'_s = *$ **then**
      $\text{entry} \leftarrow (\alpha(t-1, s-2, 1) + \alpha(t-1, s-1, 1)) * y(t, :)$ ▷ Entering * state
      // **Define intermediate matrix** $M(|V| \times |V|)$ **and** $M'(|V|)$ **for the looping paths**
      $M(i, j) \leftarrow \alpha(t-1, s, i) \times y(t, j) \times \text{mask}(i, j)$ for all $i, j \in \{1, \ldots, V\}$
      $M'(v) \leftarrow \text{sum}(M(v, :))$ for all $v \in \{1, \ldots, |V|\}$
      $\alpha(s, t, :) \leftarrow M' + \text{entry}$ ▷ All the possible paths for substitutions are tracked
    else
      $\alpha(t, s, 1) = (\alpha(t-1, s, 1) + \alpha(t-1, s-1, 1) + \alpha(t-1, s-2, 1)) \cdot y(t, l'_s)$
    end if
  end for
  $C_t \leftarrow \text{sum}(\alpha(t, :, :))$
  $\ln p(\mathbf{L}|\mathbf{O}) \leftarrow \ln p(\mathbf{L}|\mathbf{O}) + \ln(C_t)$
  $\alpha(t, :, :) \leftarrow \frac{\alpha(t,:,:)}{C_t}$ ▷ Normalize: $\hat{\alpha}(t, s)$ for all s
end for
$\text{occ} \leftarrow \text{sum}(\alpha(:, 2 * \mathbf{pos}, :))$ ▷ Estimating the expected activation length
return $\{\ln p(\mathbf{L}|\mathbf{O}), \text{occ}\}$