Let's say we have 2 segments, and $A_1 = \sum_{\sigma_1} \gamma^t \alpha log\pi_\theta(a_{t1}|s_{t1})$, $A_2 = \sum_{\sigma_2} \gamma^t \alpha log\pi_\theta(a_{t2}|s_{t2})$, then the likelihood in the CPL paper can be written as (please refer to formula 6 in the paper)

$$
\begin{aligned}
P[\sigma_1 \succ \sigma_2] &= \frac{\exp(A_1)}{\exp(A_1) + \exp(\lambda A_2)} \\
&= \frac{1}{1 + \exp(\lambda A_2 - A_1)} \\
&= sigmoid(A_1 - \lambda A_2)
\end{aligned}
\tag{1}
$$

When you want to maximize it in the codebase, you can simply minimize a Binary Cross Entropy loss, i.e.,

$$
\begin{aligned}
Loss &= -ylogP[\sigma_1 \succ \sigma_2] - (1-y)logP[\sigma_2 \succ \sigma_1] \\
&= -ylog\big(sigmoid(A_1 - \lambda A_2)\big) - (1-y)\big(sigmoid(A_2 - \lambda A_1)\big)
\end{aligned}
\tag{2}
$$

Here $y = 1$ if human operators annotate $\sigma_1 \succ \sigma_2$, otherwise $y = 0$.

If $\lambda = 1$ and you use CPL with behaviour cloning loss as the constraint (see Appendix B of the paper), then $sigmoid(A_1 - \lambda A_2) = 1 - sigmoid(A_2 - \lambda A_1)$, and Pytorch can achieve (2) with 3 lines of code,

```
>>> m = nn.Sigmoid()
>>> criterion = nn.BCELoss()
>>> loss = criterion(m(A1-lambda*A2))
```

But if you have $\lambda \in [0,1]$ as the constraint, $sigmoid(A_1 - \lambda A_2)$ may not be equal to $1 - sigmoid(A_2 - \lambda A_1)$. The nn.BCELoss does not support this situation. You must write your custom loss function. You can rewrite (1) as

$$
\begin{aligned}
P[\sigma_1 \succ \sigma_2] &= \frac{\exp(A_1)}{\exp(A_1) + \exp(\lambda A_2)} \\
&= \frac{\exp(A_1)/\exp(\lambda A_2)}{\exp(A_1)/\exp(\lambda A_2) + 1} \\
&= \frac{\exp(A_1 - \lambda A_2)}{\exp(A_1 - \lambda A_2) + 1}
\end{aligned}
\tag{3}
$$

A problem of (3) is that $\exp(A_1 - \lambda A_2)$ may be too large and lead to numerical instability. You can set:

$$
\begin{aligned}
b_1 &= A_1 - \lambda A_2, if\ A_1 - \lambda A_2 > 0 \\
b_1 &= 0, otherwise
\end{aligned}
\tag{4}
$$

And (3) becomes:

$$P[\sigma_1 \succ \sigma_2] = \frac{\exp(A_1 - \lambda A_2 - b_1)\exp(b_1)}{\exp(A_1 - \lambda A_2 - b_1)\exp(b_1) + 1}$$
$$= \frac{\exp(A_1 - \lambda A_2 - b_1)}{\exp(A_1 - \lambda A_2 - b_1) + \exp(-b_1)} \tag{5}$$

Take the logarithm of both sides, then:

$$logP[\sigma_1 \succ \sigma_2] = log\frac{\exp(A_1 - \lambda A_2 - b_1)}{\exp(A_1 - \lambda A_2 - b_1) + \exp(-b_1)}$$
$$= (A_1 - \lambda A_2 - b_1) - \log\left[\exp(A_1 - \lambda A_2 - b_1) + \exp(-b_1)\right] \tag{6}$$

Due to symmetry,

$$logP[\sigma_2 \succ \sigma_1] =$$
$$(A_2 - \lambda A_1 - b_2) - \log\left[\exp(A_2 - \lambda A_1 - b_2) + \exp(-b_2)\right] \tag{7}$$

You can substitute (6) and (7) back to (2) to get your custom BCE loss.
biased_bce_with_logits() in research/algs/cpl.py takes the same implementation.