The following is a derivation of the optimal policy for outputting a certain word as a target $P_t$ given a belief about the the word being picked $P_p$, about the targets that have been outputted so far being picked $P_a$, the reward function and the KL penalty coefficient $\beta$. The reward function is split into two functions of the number of targets $n$ – an accept reward $R_a$ for when the targets are accepted by the overseer, and a reject reward $R_r$ for when the targets are rejected. We start with the expected reward $R_e$, the KL divergence $KL$ as a function of the reference policy's probability of outputting a target $P_b$ and the objective function $J$. We then take the derivative of the objective function with respect to $P_t$ and set it to zero to find the optimal policy.

$$
\begin{aligned}
R_e &= P_t \left[P_a P_p R_a(n) + (1 - P_a P_p) R_r(n)\right] \\
&\quad + (1 - P_t) \left[P_a R_a(n-1) + (1 - P_a) R_r(n-1)\right] \\
KL &= P_t \log \frac{P_t}{P_b} + (1 - P_t) \log \frac{1 - P_t}{1 - P_b} \\
J &= R - \beta KL \\
\frac{dR_e}{dP_t} &= P_a P_p R_a(n) + (1 - P_a P_p) R_r(n) \\
&\quad - P_a R_a(n-1) - (1 - P_a) R_r(n-1) \\
\frac{dJ}{dP_t} &= \frac{dR_e}{dP_t} - \beta \log \frac{P_t(P_b - 1)}{P_b(1 - P_t)} = 0 \\
\frac{P_t}{1 - P_t} &= \frac{P_b}{1 - P_b} e^{\frac{1}{\beta} \frac{dR_e}{dP_t}} \\
P_t &= 1 - \frac{1}{1 + \frac{P_b}{1 - P_b} e^{\frac{1}{\beta} \frac{dR_e}{dP_t}}}
\end{aligned}
$$

In order to design a reasonable reward function, we impose the following heuristics:

- The policy for outputting a specific target should be as independent as possible of the number of targets already outputted. In practice, this means $R_a$ needs to scale exponentially in the number of targets to reflect the geometric nature of the risk of the targets being rejected. We operationalize this by having one calibrated probability $P_c$ for which $P_p = P_c => P_t = P_p$, across all $n$ and $P_a$.
- On average, the reward for a calibrated policy should be similar to the true game score. We approximate this with $R_a(0) = 0$ and $R_a(1) = 1$.
- It is valid for $P_c$ to change during training to reflect the current ability of the model to give good clues. To implement this, we take the average number of targets across all outputs in the current batch, and divide by the maximum allowed number of targets $m$. This should ensure that on average the model is being penalized for outputting both too many targets and too few targets.

To find $R_r(n)$, we can set $P_a$ to 0:

$$\frac{P_c}{1-P_c} = \frac{P_b}{1-P_b} e^{\frac{1}{\beta}(R_r(n)-R_r(n-1))}$$
$$R_r(n) = R_r(n-1) + \beta \log \frac{P_c(1-P_b)}{P_b(1-P_c)}$$
$$R_r(n) = R_r(1) + (n-1)\beta \log \frac{P_c(1-P_b)}{P_b(1-P_c)}$$

Now set $P_a = 1$:

$$\frac{P_c}{1-P_c} = \frac{P_b}{1-P_b} e^{\frac{1}{\beta}(P_c R_a(n)+(1-P_c)R_r(n)-R_a(n-1))}$$
$$R_a(n) = \frac{R_a(n-1) - (1-P_c)R_r(n) + \beta \log \frac{P_c(1-P_b)}{P_b(1-P_c)}}{P_c}$$

With $n = 1$, we get:

$$R_a(1) = \frac{R_a(0) - (1-P_c)R_r(1) + \beta \log \frac{P_c(1-P_b)}{P_b(1-P_c)}}{P_c}$$
$$P_c = -(1-P_c)R_r(1) + \beta \log \frac{P_c(1-P_b)}{P_b(1-P_c)}$$
$$R_r(1) = \frac{\beta \log \frac{P_c(1-P_b)}{P_b(1-P_c)} - P_c}{1-P_c}$$