# Reward Model Aggregation

Zihao Wang[†1,3], Chirag Nagpal[2], Alexander D'Amour[1], Victor Veitch [‡1,3], and Sanmi Koyejo [‡1,4]

[1]*Google DeepMind*
[2]*Google Research*
[3]*University of Chicago*
[4]*Stanford University*

## Abstract

Aligning language models requires guiding outputs towards desired properties using reward models. This paper tackles the challenge of combining multiple reward models for diverse objectives. We introduce methods for aggregating these rewards using logical operations. Experiments confirm our methods beat traditional aggregation techniques and underscore the significance of proper reference values.

## 1 Introduction

It is common practice to finetune large language models to bias their outputs towards having desirable properties—e.g., to be helpful, harmless, factual, or creative [Ouy+22; Sti+20; Zie+19]. This is typically done by defining a reward model that measures the degree to which an output has the desired property, and then running a finetuning procedure that biases towards outputs with high reward. In this note, we're concerned with the case where we have multiple distinct goals—and a reward model for each goal—and we want to finetune a language model to be 'good' on multiple accounts.

In practice, the reward models are usually by training a classifier on a dataset of prompts and responses labelled with the desired property. This can be done either pointwise—each example labelled good or bad—or pairwise—each prompt has a set of responses, labelled by which is preferred on the desired property. The idea in this note is simply that by interpreting the reward models probabilistically, we are led to natural procedures for combining multiple rewards.

## 2 Pointwise rewards

Suppose we have data $(X, Y, G_A)$ where $X$ is a prompt, $Y$ is a response, and $G_A$ is a binary label indicating whether the response is good on some property $A$. If we learn a reward model $R_A(x, y)$ by minimizing CrossEntropy risk of predicting $G_A$ from $(x, y)$—the standard for pointwise rewards—then, ignoring finite sample issues, we can interpret the learned reward model as:

$$R_A(x, y) = \operatorname{logit} P(G = 1|x, y) \tag{1}$$

(The logit scale is chosen for compatibility with the pairwise case below.)

Now suppose we have two reward models $R_A$ and $R_B$ for two different properties $A$ and $B$. The following result gives a natural way to combine them.

**Theorem 2.1.** *Define*

$$R_{G_A \wedge G_B} := \operatorname{logit}(\sigma(R_A)\sigma(R_B)) \tag{2}$$

$$R_{G_A \vee G_B} := \operatorname{logit}(\sigma(R_A) + \sigma(R_B) - \sigma(R_A)\sigma(R_B)). \tag{3}$$

[†]Work done while interning at Google DeepMind
[‡]Equal Contributions

Workshop on Instruction Tuning and Instruction Following at NeurIPS 2023.

*Suppose that if $R_A$ and $R_B$ correspond to logit probabilities as in eq. (1), and that $G_A$ and $G_B$ are independent conditional on $X, Y$. Then $R_{G_A \wedge G_B}$ and $R_{G_A \vee G_B}$ are the reward models we would have learned by predicting $G_A \wedge G_B$ and $G_A \vee G_B$ respectively.*

(The proof is straightforward)

Theorem 2.1 gives us a way to combine reward models trained from pointwise data. However, in practice, pairwise data are more abundant than pointwise data.

## 3 Pairwise rewards

The pairwise case is subtler. Here, we have data $(X, Y_0, Y_1, \tilde{G}_A)$ where $X$ is a prompt, $Y_0$ and $Y_1$ are responses, and $\tilde{G}_A$ is a binary label indicating whether $Y_1$ is preferred to $Y_0$ on some property $A$. Learning a reward model $R_A(x, y)$ via Bradley-Terry ([BT52]) (the standard approach) yields:

$$P(\tilde{G}_A = 1 \mid x, y_0, y_1) = \sigma(R_A(x, y_1) - R_A(x, y_0)) \tag{4}$$

The challenge is that now $R_A$ cannot be naturally interpreted as a logit probability. The issue is that the reward model is only identified up to an additive function of $x$; i.e., if we change

$$R_A(x, y) \to R_A(x, y) + c(x) \tag{5}$$

then eq. (4) is unchanged. However, logit probabilities do not have this property.

The idea here is to transform the pairwise reward model into one that can be interpreted as a logit probability. Then, theorem 2.1 can be used to combine rewards. For a fixed language model $\pi(Y \mid x)$ and reward function $R_A$ we define $r_A^q(x)$ to be the $q$-th quantile of the distribution of rewards for $x$, when $Y$ is sampled from the language model $\pi(Y \mid x)$. Then,

**Theorem 3.1.** *Suppose that $y_q(x)$ is a response such that $R(y_q, x) = r_A^q(x)$. Assuming Bradley-Terry, we have*

$$R_A(x, y) - r_A^q(x) = \operatorname{logit} P(Y \text{ preferred to } y_q \text{ on property } A \mid x) \tag{6}$$

That is: if we subtract off a quantile of the distribution of rewards, the modified reward function can be naturally interpreted as a logit probability of a binary random variable. Then we can use theorem 2.1 for reward aggregations.

## 4 Experiment

### 4.1 Experiment Setup

**Training the Reward Models:** We train models for evaluating helpfulness and harmlessness using the anthropics hh-rlhf preference dataset [Bai+22]. This training involves fine-tuning a T5 base model ([Raf+20]) utilizing the Bradley-Terry model.

**Sampling Approach:** We use the best-of-k approach, a method that chooses the answer with the highest (combined) reward from the top $k$ potential responses generated by the model. This method helps refine the sampling distribution of the selected Language Model (LM), specifically the PaLM 2 XS ([Ani+23]), which has been fine-tuned on FLAN instructions ([Wei+21]) and the helpfulness data.

**Evaluations:** Evaluation involves comparing the new and original sampling distributions. We randomly select a sample, $y_0$, from the original LM and another sample, $y_1$, using the best-of-k approach. The win rate is computed by comparing these two samples, scored using the same reward models used for aggregations. The win in "AND" for $y_1$ means it has better rewards than $y_0$ in both properties A and B. The win in "OR" for $y_1$ means it outperforms $y_0$ in at least one property. This method offers insight into the effectiveness of the new sampling distribution. Refer to fig. 1 for visual representation.

**Combining Rewards:** In the process of combining rewards, two significant decisions are made: the choice of combination methods, and the selection of reference reward ($r^{\text{ref}}$).

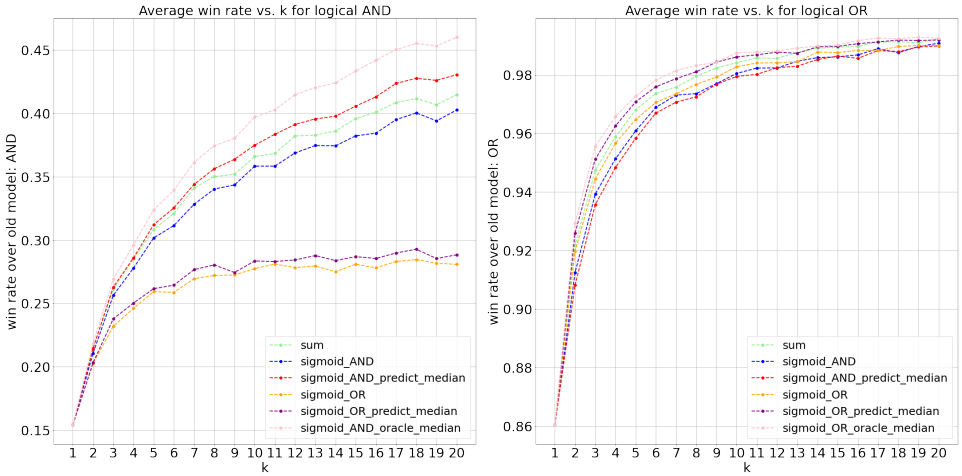
1. **Combination Method:** Rewards are combined using distinct methods (after subtracting off reference $r^{\text{ref}}$):

- **Derived Method:** Applying theorem 2.1 gives $\sigma_{\text{AND}}$ and $\sigma_{\text{OR}}$ functions, defined as:

$$R_{\sigma_{\text{AND}}}(x,y) := \text{logit}(\sigma(R_A(x,y) - r_A^{\text{ref}}(x))\sigma(R_B(x,y) - r_B^{\text{ref}}(x)))$$
$$R_{\sigma_{\text{OR}}}(x,y) := \text{logit}(\sigma(R_A(x,y) - r_A^{\text{ref}}(x)) + \sigma(R_B(x,y) - r_B^{\text{ref}}(x)) - \sigma(R_A(x,y) - r_A^{\text{ref}}(x))\sigma(R_B(x,y) - r_B^{\text{ref}}(x)))$$

- **Simple Summation Method:** Employs straightforward summation of the rewards.

2. **Selection of Reference Reward ($r^{\text{ref}}$):** Based on the combination method used, the reference reward is chosen using two approaches:
   - **Oracle-Median Method:** Sample 60 additional responses from the original LM and use the sample median as reference. This is not computationally feasible during deployment.
     - Referred as "sigmoid_AND_oracle_median" when used with $\sigma_{\text{AND}}$.
     - Referred as "sigmoid_OR_oracle_median" when used with $\sigma_{\text{OR}}$.
   - **Predict-Median Method:** Uses a model to estimate the median rewards from the original LM.
     - Referred as "sigmoid_AND_predict_median" when used with $\sigma_{\text{AND}}$.
     - Referred as "sigmoid_OR_predict_median" when used with $\sigma_{\text{OR}}$.
   - **No Subtraction Method:** No subtraction is performed, keeping the original reward values intact.
     - Referred as "sigmoid_AND" when used with $\sigma_{\text{AND}}$.
     - Referred as "sigmoid_OR" when used with $\sigma_{\text{OR}}$.




**(a)** win rate for AND **(b)** win rate for OR

**Figure 1:** Reference centering and choosing the aggregation function matched to the logical goal gives good performance.

**Summary of Methods:**

- **sigmoid_AND_oracle_median:** Uses sample median rewards from additional responses as $r^{\text{ref}}$, combined with $\sigma_{\text{AND}}$.
- **sigmoid_AND_predict_median:** Uses a model to estimate the median rewards as $r^{\text{ref}}$, combined with $\sigma_{\text{AND}}$.
- **sigmoid_AND:** Uses 0 as $r^{\text{ref}}$, combined with $\sigma_{\text{AND}}$.
- **sigmoid_OR_oracle_median:** Uses sample median rewards from additional responses as $r^{\text{ref}}$, combined with $\sigma_{\text{OR}}$.

- **sigmoid_OR_predict_median:** Uses a model to estimate the median rewards as $r^{\text{ref}}$, combined with $\sigma_{\text{OR}}$.
- **sigmoid_OR:** Uses 0 as $r^{\text{ref}}$, combined with $\sigma_{\text{OR}}$.
- **sum:** Employs simple summation for combining rewards.

### 4.2 Observations

We have two main claims:

1. We can implement logical OR or logical AND with a suitable choice of combination function, and can beat summation in their corresponding metrics.
2. The choice of reference reward $r^{\text{ref}}$ used to convert pairwise rewards to pointwise rewards has a substantive effect.

**Combining functions correspond to logical AND, OR** In fig. 1, we can see these our derived approaches correspond to logical AND, OR respectively, and are not compatible. Using the corresponding approach beats using simple summation in AND, OR respectively.

**The choice of reference rewards $r^{\text{ref}}$ has a substantive effect** It's a natural choice to use median rewards for reference values, on both property A and B. In fig. 1, we see that using a good estimate of it improves results compared to subtracting off 0.

## References


[Ani+23] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen, et al. "Palm 2 technical report". *arXiv preprint arXiv:2305.10403* (2023) (cit. on p. 2).

[Bai+22] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan, et al. "Training a helpful and harmless assistant with reinforcement learning from human feedback". *arXiv preprint arXiv:2204.05862* (2022) (cit. on p. 2).

[BT52] R. A. Bradley and M. E. Terry. "Rank analysis of incomplete block designs: i. the method of paired comparisons". *Biometrika* 3/4 (1952) (cit. on p. 2).

[Ouy+22] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. "Training language models to follow instructions with human feedback". *Advances in Neural Information Processing Systems* (2022) (cit. on p. 1).

[Raf+20] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. "Exploring the limits of transfer learning with a unified text-to-text transformer". *The Journal of Machine Learning Research* 1 (2020) (cit. on p. 2).

[Sti+20] N. Stiennon, L. Ouyang, J. Wu, D. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. F. Christiano. "Learning to summarize with human feedback". *Advances in Neural Information Processing Systems* (2020) (cit. on p. 1).

[Wei+21] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le. "Finetuned language models are zero-shot learners". *arXiv preprint arXiv:2109.01652* (2021) (cit. on p. 2).

[Zie+19] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving. "Fine-tuning language models from human preferences". *arXiv preprint arXiv:1909.08593* (2019) (cit. on p. 1).