

## 568 A Reference LPF methods on AlpacaFarm

569 Having defined and validated the pairwise feedback simulator and evaluations in AlpacaFarm, we  
570 now turn our attention to studying methods that learn from pairwise feedback on AlpacaFarm.  
571 Unfortunately, the lack of existing benchmarks for learning from pairwise feedback for instruction  
572 following means that there has not been any open study of these methods in the instruction-following  
573 setting. To address this, we implement a suite of reference methods and benchmark them on  
574 AlpacaFarm.

575 In the remainder of this section, we will introduce our reference methods, which fall into two  
576 categories based on whether they fit a surrogate reward model as part of the learning process.

### 577 A.1 Methods that directly learn from pairwise feedback

578 **Binary FeedME.** FeedME is a method proposed by OpenAI [45] that incorporates human feedback  
579 with supervised fine-tuning on model generations that are rated 7/7 by human labelers. We adapt  
580 this approach to the pairwise feedback setting and call this baseline binary FeedME. This approach  
581 fine-tunes the SFT model on the chosen response in each preference pair with supervised learning.

582 **Binary Reward Conditioning.** Motivated by controllable generation through conditioning [27, 34,  
583 29, 21], we propose binary reward conditioning, a baseline method that fine-tunes the SFT model  
584 with the feedback data  $\mathcal{D}_{\text{pairwise}}$  by conditioning instances with either a positive or negative control  
585 token. Specifically, for each instance  $(x, y_0, y_1, z) \in \mathcal{D}_{\text{pairwise}}$ , the string concatenation of instruction  
586  $x$  and response  $y_z$  denoted as  $[x, y_z]$  is prepended with the positive token and used in supervised  
587 fine-tuning (similarly  $[x, y_{1-z}]$  is prepended with the negative token). This process creates a modified  
588 demonstration dataset that is double the size of  $\mathcal{D}_{\text{pairwise}}$ . At test time, we draw samples from the  
589 fine-tuned model conditioned on the positive token.

### 590 A.2 Methods that optimize a surrogate reward function

591 We now describe methods that incorporate feedback by first building a surrogate reward model with  
592 pairwise feedback data. To start, we describe the step of training the surrogate reward model.

593 To train a parameterized surrogate  $\hat{R}_\phi$ , one can maximize the log-likelihood of the preferences  $z$   
594 under the Bradley-Terry model [10]

$$\text{maximize}_\phi \sum_j \log P(z^{(j)} | x^{(j)}, y_0^{(j)}, y_1^{(j)}) = \sum_j \log \frac{\exp(\hat{R}_\phi(x^{(j)}, y_z^{(j)}))}{\exp(\hat{R}_\phi(x^{(j)}, y_0^{(j)})) + \exp(\hat{R}_\phi(x^{(j)}, y_1^{(j)}))}. \quad (1)$$

595 Once the surrogate reward model is trained, both training and inference algorithms can optimize  
596 against the reward model rather than query pairwise feedback. While this can be a powerful approach,  
597 we will see that it can also lead to *over-optimization* [19] where models learn to exploit the reward  
598 model rather than achieve high true reward. We now describe 4 methods that leverage the surrogate  
599 reward model.

600 **Best-of- $n$  Sampling.** Best-of- $n$  sampling (or re-ranking) [64, 5, 22, 8] is a common inference-time  
601 method that aims to improve the generation quality. Given an input  $x$ , the method returns the response  
602 with the highest surrogate reward value among  $n$  i.i.d. responses drawn from the SFT model. While  
603 simple to implement and useful as a baseline, this approach incurs high inference costs.

604 **Expert Iteration.** Expert iteration [2, 61, 71] is a technique that has recently been used to train  
605 language models. We adapt this approach in AlpacaFarm as a two-step method. In the first step, we  
606 perform best-of- $n$  sampling and store the generated samples. In the second step, we fine-tune  $p_{\text{SFT}}$   
607 on these samples with supervised learning. While prior work applies expert iteration for multiple rounds  
608 by performing best-of- $n$  sampling again for intermediate models, we focus on performing a single  
609 round. In Appendix F, we include our preliminary study of multi-round expert iteration.

610 **Proximal Policy Optimization.** Proximal Policy Optimization [PPO; 25, 59] is a popular RL  
611 algorithm that has been recently used to develop InstructGPT [47] and ChatGPT [44]. When applied

612 to fine-tune LMs with RLHF, PPO maximizes the following KL-regularized objective w.r.t. model  
613 parameters  $\theta$

$$\mathbb{E}_{x \sim p(x), y \sim p_\theta(y|x)} \left[ \hat{R}_\phi(x, y) - \beta \log \frac{p_\theta(y|x)}{p_{\text{SFT}}(y|x)} \right], \quad (2)$$

614 where  $p(x)$  is an unlabeled instruction distribution,  $p_\theta(y|x)$  is fine-tuned from the  $p_{\text{SFT}}$  model, and  
615  $\beta \in \mathbb{R}$  is a regularization coefficient. Each step of PPO alternates between drawing samples from  
616 the current policy and performing gradient updates based on the pool of samples with importance  
617 sampling and clipping. We follow existing PPO implementations and document the modifications we  
618 made in Appendix F.

619 **Quark.** Quark is inspired by reward conditioning and has been shown to be effective for controllable  
620 generation tasks. Like binary reward conditioning, Quark on train sequences with prepended control  
621 tokens. Unlike binary reward conditioning, Quark bins model samples into multiple groups based on  
622 the reward value, adds KL and entropy regularization, and repeats the entire process across multiple  
623 rounds.

624 In our preliminary analysis, we find the top-quantile variant reported in [38], i.e. only training on the  
625 best reward group, to perform the better than the all-quantiles variant which trains on all groups. We  
626 document our tuning process and implementation modifications in Appendix F.

## 627 B Tradeoffs in simulator design

628 We have now seen that insights on methods development in the simulator successfully transfer to  
629 real human data. To do so, pairwise feedback in the Farm relies on two important design decisions:  
630 the use of multiple API models with different prompts, and injecting label noise into the training  
631 preferences. Though these components were designed to better simulate humans, who exhibit a  
632 high level of inter-annotator disagreement, it is reasonable to assume that each of these components  
633 degrade the quality of the resulting training data. Specifically, both Davinci003 and ChatGPT are  
634 worse pairwise evaluators than GPT-4 (Figure 5), and added label noise further degrades the training  
635 signal. One way to analyze whether these design decisions are necessary is by comparing to a simple  
636 yet high-quality baseline: a single-prompt GPT-4 (red in Figure 5). Specifically, in the remainder of  
637 this section, we answer the two following questions:

- 638 • Does modeling variability in preferences result in a better simulator over the simple GPT-4  
639 alternative?
- 640 • How much does the variability impact the performance of the trained model outside Alpaca-  
641 Farm?

### 642 B.1 Comparing simulators to reality

643 To answer the question on simulator design, we first analyze learning dynamics on real human  
644 preference data, as shown in Figure 3 left. We look at the three most effective methods in our suite,  
645 all of which optimize against a fixed reward model trained on human preferences.

646 Figure 3 left displays the tell-tale sign of *reward model over-optimization*, the phenomenon where  
647 optimizing against a fixed reward model only increases model performance up to a point. Noted in  
648 prior work [19], this effect is the result of the human reward proxy model  $\hat{R}_\phi$  being an imperfect  
649 estimate of the (unknown) human reward  $R$ . This effect is most evident for PPO (shown with steps  
650 40, 80, 200) and Best-of- $n$  ( $n = 4, 16, 128$ ). Expert Iteration (shown with  $n = 4, 16, 128$ ) plateaus  
651 and does not over-optimize yet, which may be tied to its inability to optimize the reward model  
652 reward effectively.

653 We find exactly this result with the simulator. Figure 3 middle shows the analogous plot for each of  
654 the methods trained in the AlpacaFarm. We see a similar story of over-optimization: PPO (steps 20  
655 to 200) and Best-of- $n$  ( $n = 1$  to 1024) show a clear increase then decrease in win rate, with Expert  
656 Iteration ( $n = 4$  to 128) plateauing in performance.

657 This over-optimization behavior is critically linked to our choice of modeling annotator variability in  
658 the simulator. As an alternative, we re-trained all the models using a single high-accuracy prompt for

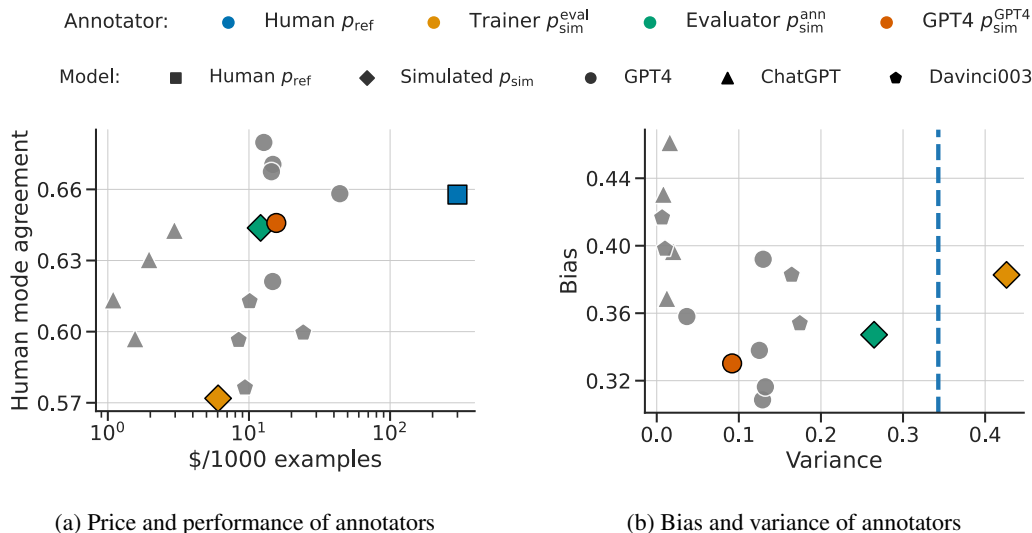


Figure 5: Our simulated annotators are cheap and match well with human annotators. Grey points are all the annotators in our simulated pool, the green point shows the resulting pool of annotators (which we use for evaluation), the orange point shows the same simulated pool with additional noise (which we use for training), the blue point shows the pool of human annotators, and the red point shows a single low variance GPT-4 annotator which we analyze in appendix B. (a) shows the price (x-axis) and the performance (y-axis) of annotators as measured by their agreement with the mode of human annotations. (b) shows the estimated bias and variance of simulated annotators compared to humans, whose bias is by definition 0, and variance is shown with a blue line.

659 GPT-4,  $p_{sim}^{GPT-4}$ . The prompt was chosen as it has a high correlation with human annotators (red in  
 660 Figure 5) while still being deterministic (preferences were sampled with greedy decoding,  $\text{temp}=0$ ).

661 The results are shown in Figure 3 right, which displays a clear departure from the over-optimization  
 662 result in the other two figures. The win rate of all three methods – Best-of- $n$ , PPO, and Expert  
 663 Iteration – continues to increase with compute and shows little signs of plateauing. While this lack  
 664 of over-optimization may appear to be a benefit, it poses a major challenge for reliable methods  
 665 development. In a simulator without over-optimization, picking the best method would simply  
 666 be decided by which method was dedicated more compute. For example, Figure 3 right shows  
 667 Best-of-1024 to be much better than PPO, which is a reversal of the trend observed on human data.

668 Returning back to the first question asked in this section, these results make it clear that the design of  
 669 pairwise preferences in AlpacaFarm is necessary to create a simulator that closely matches reality.  
 670 In Appendix E, we ablate the design more finely and find that the added label noise provides the  
 671 majority of the benefit for inducing over-optimization.

## 672 B.2 Transfer properties of AlpacaFarm trained models

673 Finally, we have shown that replicating annotator variability is important to capture over-optimization  
 674 behaviors in AlpacaFarm. However, we highlight that the single GPT-4 prompt annotator can be  
 675 useful when the goal is to directly transfer models from AlpacaFarm to the real world (rather than  
 676 re-training with human preferences).

677 To illustrate this point, we compare a PPO model (step 40) trained in the Farm (on  $p_{sim}^{ann}$ ) with a PPO  
 678 model (step 30) trained on the single-prompt GPT-4 (on  $p_{sim}^{GPT-4}$ ). We then measure their win rate  
 679 according to human preference evaluation,  $p_{human}$ .

680 We find that PPO $_{sim}^{ann}$  trained in the Farm achieves a win rate of 43%, while PPO $_{sim}^{GPT-4}$  trained on  
 681 single-prompt GPT-4 data achieves a win rate of 50%. To contextualize these results, the initial SFT  
 682 model has a win rate of 44%, PPO $_{human}$  has a win rate of 55%, and the best non-PPO human method  
 683 has a win rate of 51% (Best-of-16) (full table in Appendix E).

684 These results demonstrate a strong tradeoff in simulator design: more faithful simulators which  
685 display greater over-optimization train objectively worse models. The standard AlpacaFarm pairwise  
686 evaluators are suitable and ideal for developing new methods and comparing to others, as Figure 2  
687 demonstrates that these comparisons transfer when the methods are re-trained on human data.  
688 However, for training new models in the simulator and deploying directly thereafter, a single consistent  
689 annotator such as  $p_{\text{sim}}^{\text{GPT-4}}$  can provide significant gains on real-world evaluation.

## 690 C Related work

691 **Instruction following.** Many works on instruction following focused on cross-task generalization,  
692 where models were trained on a pool of NLP tasks, and evaluated on other held-out tasks based  
693 on simple metrics [42, 75, 54, 4, 74]. Increasingly, instruction following methods have focused on  
694 real world human interaction patterns [47, 6], rather than collections of existing NLP benchmarks.  
695 For example, InstructGPT was developed on user instructions submitted to OpenAI API [47]. Our  
696 work builds upon these works by attempting to bridge the gap between the ease of development and  
697 evaluation of traditional academic benchmarks and the more complex algorithms and real-world  
698 settings of recent works on instruction following.

699 **Simulating human feedback.** Constitutional AI [7] simulates human feedback with AI feedback  
700 for model development to improve harmlessness and helpfulness. AlpacaFarm, on the other hand,  
701 simulates human feedback with API LLMs so that simulated experiments reflect the outcomes of  
702 experiments performed with real human feedback. Due to the difference in goals, the construction  
703 and usage of the feedback simulator are different in the two settings. For example, AlpacaFarm’s  
704 simulator perturbs LLM preferences with bit-flip label noise to mimic the noisiness of human  
705 annotation, whereas Constitutional AI’s simulator doesn’t inject extra noise.

706 The evaluation aspects of our work are related to a growing line of work on simulating human  
707 annotation for evaluation [16, 52, 15, 51, 35]. Our core evaluation and feedback mechanism makes  
708 use of the same underlying ideas, but our work is distinguished by a focus on using pairwise feedback  
709 for training, as well as careful validation beyond per-example agreement metrics. AlpacaFarm shows  
710 that LLM feedback can capture method-level correlations as well as qualitatively important features  
711 of human annotation, such as stylistic preferences and variability.

712 Our goal of emulating human annotators also connects to work on simulating humans with LMs  
713 based on personas [49, 48, 1, 3], as well as works that simulate human behavior in the context of  
714 cognitive science, social science, and economics [70, 26]. Our work complements these works by  
715 showing that simulated LLM annotators can replicate many of the qualitative features of training on  
716 pairwise human feedback.

717 More broadly, building a simulator environment to enable low-cost experimentation is common in the  
718 field of reinforcement learning and robotics [11, 68, 67, 65, 17, 24, 18]. Our work shares the same  
719 underlying motivations, but instead of simulating physical systems, AlpacaFarm simulates human  
720 preference feedback.

721 **Methods for learning from feedback.** To hold annotation cost constant across learning methods,  
722 we have focused only on methods that learn from pairwise feedback in this work. However, there  
723 exist methods in the literature other than those explored in Section A that can incorporate alternative  
724 sources of feedback such as natural language [76, 33, 23, 60, 55, 13, 56, 41], numeric ratings [45, 32],  
725 or execution traces [14]. We view extensions of AlpacaFarm to these settings as exciting future work.

726 We have included a set of RL algorithms in our study that optimize the surrogate reward, but this set  
727 is by no means comprehensive. RL research applied to NLP has a long history [77, 63, 28, 50, 43,  
728 31, 30, 53, 62], and we expect future work in this direction to benefit from the ideas and artifacts in  
729 AlpacaFarm.

## 730 D Limitations and future directions

731 **Difference between GPT4 and human feedback.** Our sandbox assumes that LLMs can simulate  
732 human feedback. Section 4 shows that this assumption holds in the sense that LLM annotations  
733 align with the mode of human preferences and replicate its many features. However, we also observe

|  |
|--|
| Discuss the causes of the Great Depression                                 |
| Make a list of desirable Skills for software engineers to add to LinkedIn. |
| Are there any free SAST tools out there?                                   |
| I'm trying to teach myself to have nicer handwriting. Can you help?        |
| What if Turing had not cracked the Enigma code during World War II?        |
| Take MLK speech "I had a dream" but turn it into a top 100 rap song        |
| What are some toys I can buy my kids for imaginative play?                 |
| Hi, I have a question about MFCC (mel frequency cepstral coefficients).    |
| Are they the same thing as a MEL-spectrogram, or is there a difference?    |

Table 2: Example instructions in AlpacaFarm’s evaluation data.



Figure 6: Root verb distribution of the eval instructions.

734 that no single LLM-based annotator captures the heterogeneity of human annotation, and substantial  
735 amounts of noise had to be injected in the simulated preference for rankings of methods trained in  
736 AlpacaFarm to match those trained with real human feedback.

737 In addition, we found that suitable hyperparameters for learning algorithms can be different for  
738 training with simulated feedback compared to human feedback. For example, due to changes in the  
739 scale of values of the surrogate reward model, the range of suitable KL regularization coefficients for  
740 RLHF is different.

741 Finally, we note that our preference simulator is validated with respect to our recruited pool of  
742 crowdworkers, and mimics their preference variability as a result. The best prompts for transferring  
743 insights to other, more consistent crowdworker pools may involve lower levels of noise or a smaller  
744 prompt ensemble.

745 **Limitation in current experiments.** Due to resource constraints, we have not conducted extensive  
746 hyperparameter tuning for the methods we studied in Section A. For each method, we have performed  
747 at least 3 tuning runs, and we detail our tuning process in Appendix F. It is possible that with more  
748 careful tuning, the comparison between PPO and alternative approaches like expert iteration and  
749 Quark might change.

750 Our study focuses on comparisons of methods that learn from pairwise feedback. We do not study  
751 other forms of feedback, such as numerical ratings or natural language. Neither do we study quality-  
752 quantity tradeoffs for learning from pairwise feedback. We leave this exposition to future work.

753 **Future directions.** We showed that AlpacaFarm substantially lowers the cost and iteration time of  
754 research on and development of methods for learning with pairwise feedback. AlpacaFarm provides  
755 a blueprint for constructing other useful simulators for AI research that requires human supervision,  
756 and we view it as an exciting opportunity to expand this simulation approach to support data from  
757 other domains as well as methods that learn from alternative forms of human feedback.

## 758 E Additional results

### 759 E.1 Analysis of instruction data

760 We show in Table 2 examples from the Alpaca evaluation data, showcasing their diversity. We plot  
761 in Figure 7 and Figure 8 the breakdowns of the Alpaca training instruction distribution and the  
762 AlpacaFarm evaluation instruction distribution respectively. In the inner wheel, we plot the root verb  
763 distribution of the instructions and in the outer wheel, we plot the direct subject distribution. We find  
764 that both the training distribution and the evaluation distribution cover a diverse range of instructions  
765 and the distributions match at a high level.

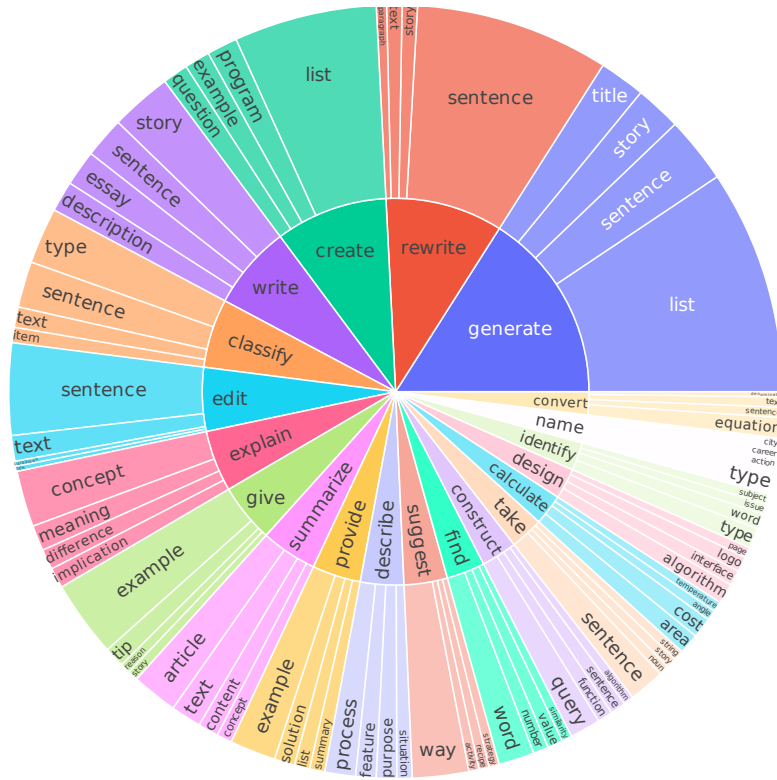


Figure 7: Breakdowns of the 52k Alpaca training instructions.

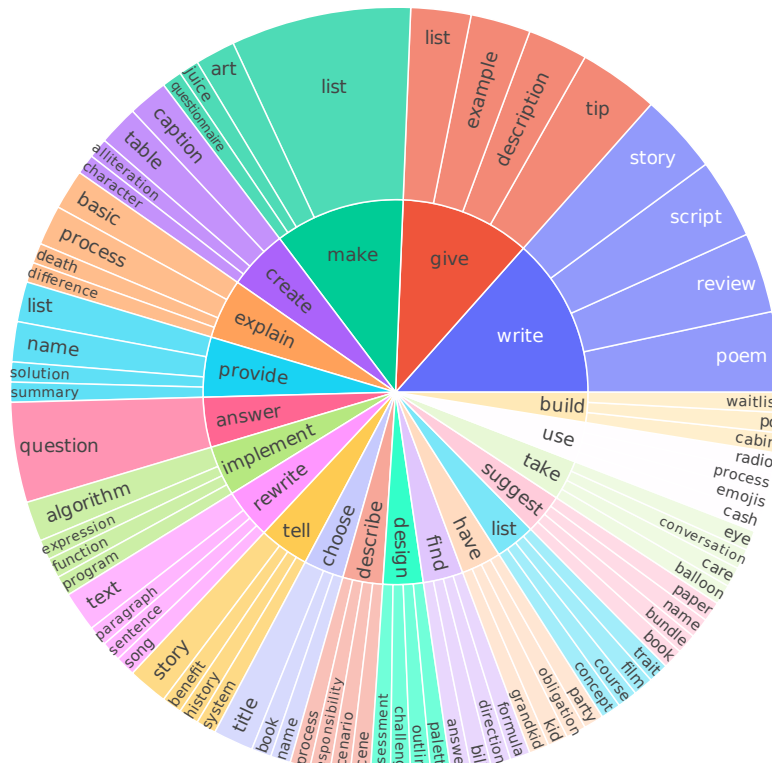


Figure 8: Breakdowns of 805 AlpacaFarm evaluation instructions.

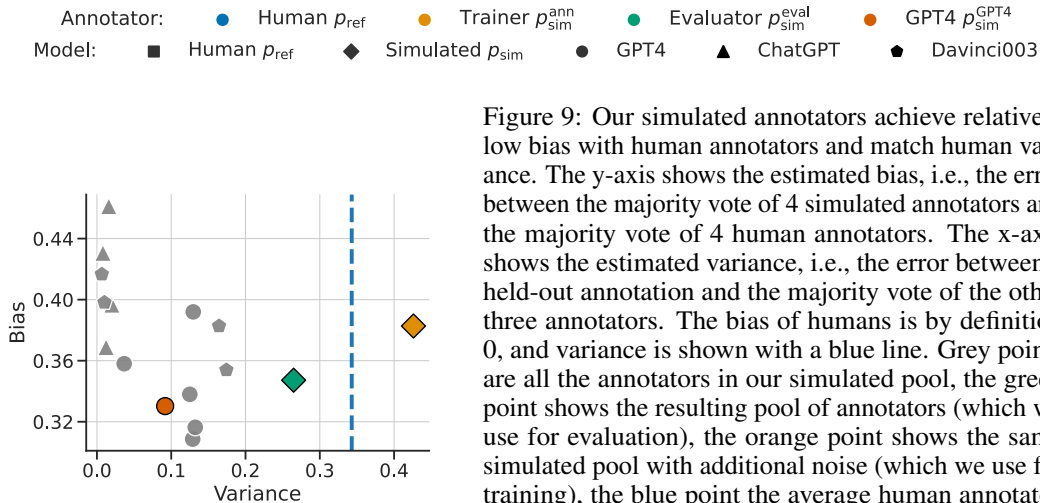


Figure 9: Our simulated annotators achieve relatively low bias with human annotators and match human variance. The y-axis shows the estimated bias, i.e., the error between the majority vote of 4 simulated annotators and the majority vote of 4 human annotators. The x-axis shows the estimated variance, i.e., the error between a held-out annotation and the majority vote of the other three annotators. The bias of humans is by definition 0, and variance is shown with a blue line. Grey points are all the annotators in our simulated pool, the green point shows the resulting pool of annotators (which we use for evaluation), the orange point shows the same simulated pool with additional noise (which we use for training), the blue point the average human annotator, and the red point shows a single low variance GPT-4 annotator we analyze.

766 **E.2 Understanding computational cost**

767 While we have focused primarily on the performance of the final model, the computational cost of  
 768 these methods is an important consideration. We provide time estimates for training on our specific  
 769 implementation and compute environment (a single  $8 \times A100$  machine). While these timings are  
 770 specific to our experiments, we believe these insights may be useful in understanding the cost of  
 771 learning from pairwise feedback.

772 To begin with, supervised fine-tuning and methods that directly adapt supervised fine-tuning like  
 773 Binary Reward Conditioning and Binary FeedME are generally fast, taking less than an hour for 10k  
 774 instructions. Best-of- $n$  sampling incurs no training cost but instead suffers a substantial inference  
 775 time cost. The optimal  $n$  for best-of- $n$  was around 16, which (in principle) translates into an increased  
 776 cost of 16x for batched inference.

777 In our experiments, Expert Iteration works best when training on the outputs of best-of-16 sampling,  
 778 which involves substantial amounts of compute to perform best-of-16 sampling on the unlabeled split.  
 779 Concretely, this sampling process takes around 4 hours on the 20k unlabeled split and the subsequent  
 780 fine-tuning takes less than an hour. We observe that PPO performance peaks on both human data  
 781 and in simulation after 40-80 steps, which translates to less than 2 hours of compute time on the full  
 782 machine.

783 **E.3 Simulated annotators**

784 We now provide additional results for understanding our pairwise annotators.

785 **Our pool of annotators has low bias and matches human variance.** Figure 9 shows the estimated  
 786 bias (y-axis) and variance (x-axis) of simulated evaluators. We see that single evaluators have a  
 787 smaller variance (less than 0.2) than humans (blue line, 0.34). This lack of variability makes emulating  
 788 it with a proxy reward very easy and leads to unrealistic over-optimization properties in the simulator,  
 789 as seen in Figure 3. Using a pool of annotators (green point) for evaluation and additionally adding  
 790 noise (orange) during training gives an estimated variance significantly closer to humans (blue line  
 791 0.35). We hypothesize that this is necessary for the simulator to show a similar over-optimization  
 792 behavior as humans. Concerning the bias, we find that our simulated annotators for evaluation  $p_{sim}^{eval}$   
 793 and training  $p_{sim}^{train}$  both have low bias values (0.38 and 0.35) on par with one of our best GPT-4  
 794 annotators (0.33).

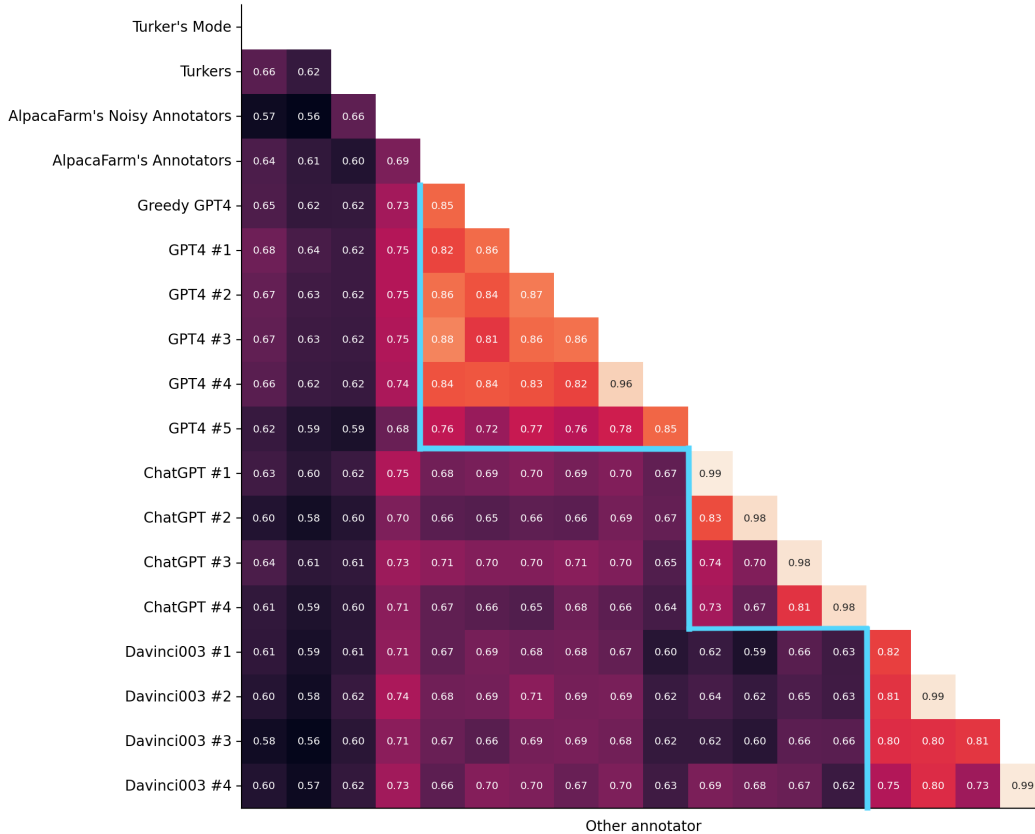


Figure 10: The largest source of variability between annotators comes from the underlying model. Every cell of the heatmap shows the agreement between two annotators (x- and y- axis).

795 **Variability in a pool of annotators mostly comes from the underlying model.** In Figure 10 we  
 796 show the pairwise agreement between all annotators in our pool and all other annotators including  
 797 the majority vote of humans (first column) and single humans (second column). The desired high  
 798 variance corresponds to low values on the diagonal (annotators disagree with themselves) and low  
 799 bias corresponds to high values in the first column (high agreement with the mode of humans). As  
 800 in Figure 9, we see that our pool of annotators  $p_{sim}^{eval}$  has low bias and high variance. Figure 10 also  
 801 shows that the largest source of variability between annotators comes from the underlying model, as  
 802 illustrated by the clusters that arise from GPT4, ChatGPT and Davinci003 annotators.

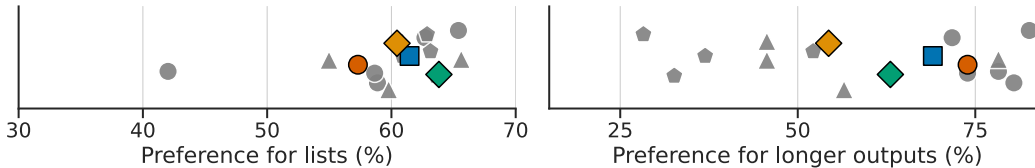


Figure 11: Humans and our simulated annotators prefer outputs that are longer and have lists.

803 **Humans and simulated annotators prefer longer outputs that contain lists.** One natural question  
 804 is whether simulated and human annotators have biases towards different type of outputs, which  
 805 would cause models in both frameworks to be qualitatively different. We identify two stylistic  
 806 features, the length and the presence of lists, for which humans have a strong preference and analyze  
 807 whether simulated annotators match those preferences. We found that humans prefer longer outputs  
 808 62% of the time, while our simulated annotators prefer those 64% of the time. Similarly, humans  
 809 prefer outputs with lists 69% of the time, while our simulated annotators prefer those 63% of the



810 time. This shows that our simulated annotators match well the stylistic preferences of humans, which  
 811 suggests that models trained in our sandbox are optimizing similar preferences as those trained with  
 812 human feedback and they will likely exhibit similar behaviors.

#### 813 E.4 Additional analysis of model outputs

814 In this section, we provide a qualitative analysis of the model outputs. In Table 3, we show three  
 815 example outputs from SFT 10k, PPO<sub>human</sub>, and PPO<sub>sim</sub>. As mentioned in ??, we find that the model  
 816 outputs after LPF training often become much longer. In Table 4 and Table 5, we show the average  
 817 length of model outputs of various systems in the number of characters, which qualitatively illustrates  
 818 this phenomenon. While longer responses indeed lead to better human preference, we continue to  
 819 see hallucinations in the output (e.g. in the example question about Berlin) and it is not clear to us  
 820 whether our LPF models improve on the more critical aspects such as faithfulness, knowledge, etc.

#### 821 E.5 Label noise ablations for simulated annotators

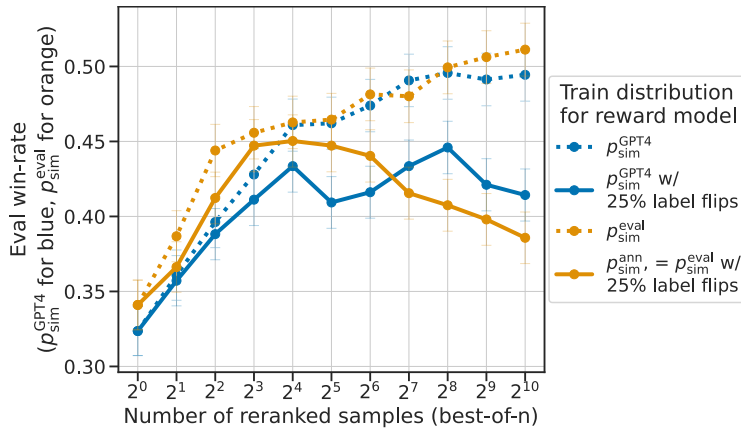


Figure 12: Label noise is the most crucial ingredient for inducing overoptimization.

822 In this section, we ablate the different components of  $p_{sim}^{ann}$  that add variability along two axes: ran-  
 823 domizing across different simulated annotators, and adding label noise. To ablate the randomization  
 824 across different annotators, we compare to the simple GPT-4 prompt  $p_{sim}^{GPT-4}$  with added label noise.  
 825 To ablate the label noise, we compare to  $p_{sim}^{eval}$ , which is  $p_{sim}^{ann}$  without the label noise. We train reward  
 826 models on these preference distributions and compare the performance of best-of- $n$  sampling.

827 Figure 12 shows the results of the ablation, demonstrating clearly that added label noise provides  
 828 the majority of the overoptimization effect. In particular, the two options that do not add label noise,  
 829  $p_{sim}^{GPT-4}$  and  $p_{sim}^{eval}$ , keep increasing win-rates with more samples. This result suggests that modeling  
 830 intra-annotator variability via label noise may be an important component to understanding learning  
 831 from human preference data.

## 832 F Details on methods implementation and hyperparameters

### 833 F.1 PPO

834 We follow an existing PPO implementation for fine-tuning language models,<sup>2</sup> but also introduce  
 835 modifications. First, off-the-shelf PPO implementations for language model fine-tuning tend to  
 836 normalize the estimated advantage for each minibatch. We found this led to training instabilities for  
 837 small minibatch sizes and instead normalize the advantage across the entire batch of rollouts obtained  
 838 for each PPO step. Second, we initialize the value model from the reward model as opposed to the  
 839 SFT model, following more recent documented practice [47] (the authors did not release code). Our  
 840 preliminary experiments showed that initializing from reward worked much better than initializing  
 841 from SFT for maximizing the surrogate reward.

<sup>2</sup><https://github.com/openai/lm-human-preferences>

Table 3: Comparing system output before and after training with PPO, we observe that outputs often become longer. Example showing output from PPO-tuned models trained with the human preference  $p_{\text{human}}^{\text{ann}}$  and with the simulated preference  $p_{\text{sim}}^{\text{ann}}$ .

|  |
|--|
| What do you think about ChatGPT?   |
| <p>SFT 10k: I think ChatGPT is a promising technology that has the potential to revolutionize the way we interact with computers. It has the ability to generate natural language text quickly and accurately, which can be used for many different applications. I think it's an exciting technology that we'll continue to see more and more uses for in the future.</p> <p>PPO<sub>human</sub>: I think ChatGPT is a fascinating development in natural language processing. It is an artificial intelligence system that is able to generate human-like text, allowing it to converse with humans on a variety of topics. It is able to learn from large datasets and improve its performance over time, allowing it to produce text with increasing accuracy and sophistication. I think it has a lot of potential applications, from natural language processing and machine translation, to automated customer service and natural language processing. It is an incredible step forward in AI technology, and I look forward to seeing what new developments come out of it in the future.</p> <p>PPO<sub>sim</sub>: I think ChatGPT is a really cool technology that can generate natural language text based on an input. It allows for users to quickly generate text responses without needing to write code or create a custom natural language processor. It also is very good at understanding context and can generate text based on context and prompts. I think it has a lot of potential applications, such as automated customer service, natural language processing, and text generation. I think it's pretty amazing how quickly the technology is advancing and it's really cool to see what's possible with AI and machine learning. I'm really excited to see what the future holds for ChatGPT and other AI technologies.</p>   |
| Who made Berlin?   |
| <p>SFT 10k: The city of Berlin was made by the German people.</p> <p>PPO<sub>human</sub>: The city of Berlin was made by various people over the course of its history. It was first established as a medieval trading post in the 12th century, and subsequently built up by a variety of different rulers. The city was heavily damaged during the Thirty Years' War, but was rebuilt and expanded by the Hohenzollern dynasty in the 17th and 18th centuries. During the 19th century, Berlin became the capital of the German Empire, and was significantly expanded and modernized under the leadership of Mayor Otto von Bismarck. The city was also divided into East and West sectors following World War II, with each sector being ruled by separate Cold War-era governments.</p> <p>PPO<sub>sim</sub>: The city of Berlin was made by various people over the centuries. The earliest known inhabitants of the area were the Tubal and Jabal tribes, who arrived in the region around 2300 BC. These tribes were later replaced by Celtic tribes, who founded the city of Berlin in the 1st century BC. The city was then conquered by the Romans in the 1st century AD and was renamed as 'Berlinium'. During the late Middle Ages, Berlin became an important city in the Holy Roman Empire and was ruled by various princes and electors. In the 17th century, Berlin gained its current shape under the rule of Prince-Elector Frederick William I, who established the city's defensive ring of bastions and set up the first permanent settlements outside the city walls. In the 18th and 19th centuries, Berlin experienced a cultural and economic boom, becoming the capital of the Kingdom of Prussia in 1701 and the seat of the German Empire in 1871. The city was also the site of many historical events, including the foundation of the Weimar Republic in 1919 and the fall of the Berlin Wall in 1989. Today, Berlin is a bustling city with a population of over 3.7 million people, making it the third-largest city in Germany.</p> |
| How many black holes are known to exist at the time?   |
| <p>SFT 10k: Nearly 100 million black holes are known to exist at the time.</p> <p>PPO<sub>human</sub>: At the time, there are believed to be several million black holes in the universe. This number is constantly increasing as astronomers discover more and more black holes through observation and study. Most black holes are found at the centers of many different galaxies, and are thought to be formed when the most massive stars in the galaxy run out of nuclear fuel and explode in a supernova. The resulting core of the star is so densely packed that nothing, not even light, can escape its powerful gravitational pull, giving it the name "black hole".</p> <p>PPO<sub>sim</sub>: At the time of this writing, there are over 1000 known black holes in the universe.</p>  |

Table 4: Average length of the model outputs of baselines and LPF models trained with human preferences. We observe that LPF training generally leads to longer outputs.

| Model         | Number of characters |
|---------------|----------------------|
| GPT-4         | 504.4                |
| ChatGPT       | 333.4                |
| Davinci001    | 286.3                |
| SFT 52K       | 383.2                |
| SFT 10K       | 277.5                |
| LLaMA 7B      | 950.5                |
| PPO 200 steps | 495.6                |
| PPO 80 steps  | 623.7                |
| PPO 40 steps  | 683.1                |
| Best-of-128   | 680.0                |
| Best-of-16    | 565.2                |
| Best-of-4     | 478.7                |
| ExpIter-128   | 524.7                |
| ExpIter-16    | 458.3                |
| ExpIter-4     | 422.1                |
| FeedMe        | 371.4                |

Table 5: Average length of the model outputs of baselines and LPF models trained with simulated preferences. We observe that LPF training generally leads to longer outputs.s

| Model        | Number of characters |
|--------------|----------------------|
| GPT-4        | 504.4                |
| ChatGPT      | 333.4                |
| Davinci001   | 286.3                |
| SFT 52K      | 383.2                |
| SFT 10K      | 277.5                |
| LLaMA 7B     | 950.5                |
| PPO 80 steps | 863.4                |
| PPO 20 steps | 637.7                |
| Best-of-128  | 704.7                |
| Best-of-16   | 570.5                |
| Best-of-4    | 483.3                |
| ExpIter-128  | 527.5                |
| ExpIter-16   | 458.3                |
| ExpIter-4    | 407.4                |

842 We tuned hyperparameters to improve training stability and reduce convergence time so that experi-  
843 ments can reliably finish with relatively tight compute budgets. In the end, we settled on a batch size  
844 of 512 for each PPO step, which consisted of 2 epochs of gradient steps each performed with a batch  
845 of 256 rollouts. We used a peak learning rate of  $10^{-5}$  which decayed to 0 throughout training. We  
846 clipped the gradient by Euclidean norm with a threshold of 1. We trained for 10 full passes over the  
847 unlabeled set, which amounts to 390 PPO steps. Performance typically peaked very early on during  
848 training (see Figure 3). We set  $\lambda$  and  $\gamma$  both to 1 for generalized advantage estimation [57]. We used  
849 a fixed KL regularizer coefficient as opposed to an adaptive one. We tuned the coefficient value for  
850 both simulated and human PPO, and settled with 0.02 for human PPO, and 0.002 for simulated PPO.  
851 We note that suitable values for the KL regularizer coefficient depend on the early stopping criteria  
852 and the scale of surrogate reward values.

## 853 F.2 Quark

854 We re-implement Quark for our needs and make several modifications. First, the original Quark  
855 formulation accumulates rollouts during training and stores them in a pool that consistently grows. We  
856 found this led to overhead that increased during training (since after each rollout batch is generated,

857 the pool is expanded and rollouts in the pool are re-sorted by their reward values). To operate under a  
858 reasonable compute budget, we discard previous rollouts once a new batch of rollouts is generated.  
859 In other words, the pool is reset once rollout is performed. This modification made the compute cost  
860 constant throughout training and thus more predictable overall. Second, we found that training on  
861 rollouts of more bins led to worse efficiency for reward optimization, and thus opted to train only on  
862 rollouts of the top-scoring bin (best-quantile variant in the original paper [40]). Preliminary ablations  
863 on a simple sentiment task showed that any potential loss in perplexity for the best-quantile variant  
864 can be compensated by turning up the KL regularizer. Lastly, we found the entropy penalty used  
865 in the original Quark formulation to give no benefit for working with instruction following. Small  
866 entropy penalty terms were enough to cause big degradations in text generation quality in terms of  
867 fluency.

868 For the official run with reported results, we used a KL regularizer coefficient of 0.05, a peak learning  
869 rate of  $3 \times 10^{-6}$  which decayed to 0 throughout training. Each Quark step had batch size 512 for  
870 rollout, and 2 epochs of gradients updates each with batch size 256. We clipped the gradient by  
871 Euclidean norm with a threshold of 1. We trained for 10 full passes over the unlabeled set, which  
872 amounts to 390 Quark steps.

## 873 G Pairwise preference simulation

### 874 G.1 Details about simulated annotators

875 For all our simulated annotators we used OpenAI API to generate outputs. We first discuss below  
876 the overall design choices for all our simulators below, and then discuss our annotator pool below in  
877 more detail. For all the actual prompts we used refer to [anonymized](#).

878 **Randomized order.** For each annotator, we randomize the ordering between the two outputs to  
879 annotate, i.e., we randomly choose which output is the first and which is the second. We found  
880 randomization to be important given that the first output is often preferred by simulated annotators.

881 **Prompts with and without inputs.** Following the Alpaca dataset [66] and self-instruct framework  
882 [73] some instructions have associated inputs, while others do not. For each annotator, we thus write  
883 two corresponding prompts, one for instructions with inputs and one for instructions without inputs.  
884 Both prompts are essentially the same but in-context examples differ in the presence of the input.

885 **Batching for GPT4.** When adding in-context examples, prompts can become relatively long, which  
886 leads to high-cost and waiting time when using GPT-4 as a simulator. To decrease cost and increase  
887 annotation speed, we amortize the cost of in-context examples by providing a batch of instruction-  
888 output pairs to annotate at once by GPT-4. For our simulated annotator we use a maximum batch size  
889 of 5 but found during development that we could fit batch size up to 20 in the context window without  
890 significantly decreasing performance. To improve performance when using batching, we found it  
891 useful to provide a few in-context examples in a batched format and to index every component of an  
892 annotation (instruction, input, output, ...).

893 **Improving parsing for ChatGPT.** Overall we found ChatGPT to be much more sensitive and  
894 harder to use as a simulator. In particular, we found it to be more sensitive to the prompt format  
895 and to often fail to generate annotations that could be parsed, e.g., by responding “Neither is better,  
896 this depends on personal preferences” despite being explicitly instructed to choose a preference. We  
897 found two tricks to be effective to make ChatGPT’s more parsable. First, we add a negative bias to  
898 tokens such as “Neither” and “Both” and a positive bias to the tokens that we hoped to match. We  
899 found the aforementioned biasing of tokens to work well but it can be problematic when using Chain  
900 of Thought reasoning. A second trick that we found to be effective is to ask ChatGPT to generate a  
901 JSON object that contains a string field with a short explanation (Chain of Thought) and a boolean  
902 field that indicates whether the first output was preferred.

903 Now that we have discussed the overall design choices for our simulated annotators, we discuss in  
904 more detail the prompts and parameters for each of our annotators.

905 **AlpacaFarm’s evaluation annotators**  $p_{\text{sim}}^{\text{eval}}$ . To try to match the bias and variance of human  
906 annotators, we use a pool of 13 simulated annotators that were developed at different stages of the  
907 project. In particular, we use the following sources of variations:

- 908 • **Models.** Five of the annotators are powered by GPT-4, four by ChatGPT, and four by  
909 Davinci003. The difference between different annotators for the same model is mostly the  
910 prompt.
- 911 • **In-context examples.** Prompts for the same models use different numbers of in-context  
912 examples.
- 913 • **Prompt format.** We use different prompt formats between and for the same model. For  
914 example different batch sizes and different formats of outputs (JSON vs raw text).
- 915 • **Preferences.** Two of the GPT4 annotators are explicitly prompted to prefer sequences that  
916 are respectively long and short.
- 917 • **Sampling.** For each annotator in the pool, we use a sampling temperature of 1.0 with top  $p$   
918 also 1.0. The high temperature means that we have variability that arises from sampling.

919 **AlpacaFarm’s training annotators**  $p_{\text{sim}}^{\text{ann}}$ . Our simulated annotators for training are the same as the  
920 evaluation annotators  $p_{\text{sim}}^{\text{eval}}$  except that we flip the output with 0.25 probability. We implement this by  
921 taking a mixture between  $p_{\text{sim}}^{\text{eval}}$  and an independent Bernoulli random variable with probability 0.5.  
922 This means that we only need to label half of the outputs for training, which makes it  $2\times$  faster and  
923 cheaper.

924 **GPT4.** For the GPT4 annotator  $p_{\text{sim}}^{\text{GPT-4}}$  we use a prompt with batch size five that corresponds to  
925 one of the prompts from our simulated pool of annotators. For  $p_{\text{sim}}^{\text{GPT-4}}$  we use temperature 0, i.e.,  
926 deterministic annotations.

## 927 H Details on human data collection

928 **Qualification.** We conducted the qualification of our annotators based on 25 qualification examples.  
929 The qualification examples were generated but an OPT 6B model that was studied in the earlier  
930 development phase of this project. The five student authors of this paper annotated a shared set of  
931 pairwise preferences. From the shared set, we selected 25 questions where the majority of the authors  
932 reached an agreement on the correct annotation. We then use these questions as a qualification test  
933 and selected the top 16 annotators whose agreement is the highest with the authors. We paid the  
934 annotators the same price for the qualification round as we did for the main qualification.

935 During the annotation process, we also compare each annotator’s preference to that of GPT-4. We  
936 identified one annotator whose agreement is around 50% with GPT-4, which is a clear outlier from  
937 other annotators. Therefore, we discontinued working with this annotator during the annotation  
938 project and removed their annotation.

939 **Annotation guideline.** We display our annotation guideline in Figure 13 and annotation interface  
940 in Figure 14. In our annotation process, we find that there are pairs that only differ in punctations or  
941 have minimal edit distance and we instruct the annotators to select a response as slightly better/worse  
942 if the difference between the pairs is marginal. As a result, around 18% of the collected preference  
943 selected the slightly better options. In our LPF experiments, we binarize the preference and treated  
944 the slightly better options the same as the normal preference labels. However, we release the more  
945 fine-grained labels as resources and leave the study to future work.

## 946 I Broader impacts

947 Furthering research on learning from human feedback can accelerate the development of more capable  
948 language models and intelligent systems. In turn, these systems may be used to cause harm or deceive  
949 by malicious actors. The simulator we propose does not account for these externalities.

Hi! We are a group of researchers working on Artificial Intelligence (AI). In this task, we will ask you to help us rate an AI model's responses to instructions.

In the area below, you will first read:

1. An instruction we give to the AI system.
2. An input that is provided along with the instruction. This is an optional input and not all instructions will have inputs.
3. Two responses from the AI system

Your task is to decide which response is better. There are several dimensions that you can think along. Consider the following questions:

1. Is the response helpful? For example, if the instruction asked for a recipe for healthy food, and the response is a useful recipe, then we can consider it helpful.
2. Is the response language natural? For example, AI responses are often repetitive, which is not natural.
3. Is the response factual/accurate? AI responses often make up new information. For example, if the response claims that Donald Trump is the current U.S. president, then you should consider it inaccurate.
4. Based on your aesthetics, which one do you prefer? For example, you might prefer one poem over another poem.
5. And so on ... ultimately, you should decide which response is better based on your judgment and based on your own preference.

There are four options for you to choose from:

1. **Response 1 is better:** If you think response 1 has an advantage, then choose this option.
2. **Response 1 is slightly better:** Response 1 is very marginally better than response 2 and the difference is small.
3. **Response 2 is slightly better:** Response 2 is very marginally better than response 1 and the difference is small.
4. **Response 2 is better:** If you think response 2 has an advantage, then choose this option.

There are cases where the difference between the two responses is not clear. In this case, you can choose the second or the third option. However, in general, we ask you to choose those options as few as possible.

## Annotation Examples

To help you understand the annotation task, we provide two examples below.

### Example 1

**Instruction:** Paraphrase the following question: "What is the capital of France?" **Input:** **Response 1:** The capital of France is Paris. **Response 2:** Can you tell me the name of the city that serves as the capital of France?

In this example, response 2 paraphrases the question as asked by the instruction. In contrast, response 1 does not follow instruction as it answers the question instead. Therefore, we should choose **Response 2 is better**.

### Example 2

**Instruction:** Change the first verb to a synonym **Input:** I love surfing **Response 1:** I hate surfing **Response 2:** and am planning on going surfing tomorrow morning.

In this example, neither output is correct. Output (a) tries to follow more closely the instruction as it changes the first verb of the sentence. Therefore, we should choose **Response 1 is better**.

### Example 3

**Instruction:** Change the first verb to a synonym **Input:** I love surfing **Response 1:** I hate surfing **Response 2:** I don't like surfing

In this example, neither output is correct and the responses are very similar. In such a case, you can choose either **Response 1 is slightly better** or **Response 2 is slightly better**. However, in general we ask you to choose those options **as few as possible**.

## Payment structure

In general, we aim to pay crowd workers \$15/hour on average. Our current estimate is that it will usually not take you more than a minute to compare each pair of responses on average. Therefore, we currently pay \$0.25 per pair of responses.

### Qualification


In the qualification round, you will need to compare 24 pairs of responses and we pay \$6 for this part. We also allocate 2 minutes for you to read the annotation guideline, for which we pay \$0.50. In total, we pay \$6.50 for the qualification round.

### Main HIT

In total, we have around 20K pairs to annotate and we plan to spread the annotation over 2 weeks. In each HIT after the qualification round, you can work on 15 pairs of responses. We pay \$3.75 for each HIT.

Figure 13: Our annotation guideline.

Hi test22. [Not your MTurk id? Logout](#)

Your annotation progress for this HIT:  0/6 completed.

Sample 1 of 6.

Lets you skip forward (all samples still need to be marked):

Returns you back to the next un-marked sample to mark:

## ► Annotation Guidelines

### Instruction

Tell me about self driving cars

### Input

### AI Response 1

Self driving cars are vehicles that are capable of detecting their surroundings and navigating without any input from a human driver. They use a variety of sensors, such as cameras, radar, and lidar, to detect objects in their environment and use advanced algorithms to make decisions about how to maneuver around them. Self driving cars are becoming increasingly popular and are being used in a variety of applications, such as ride-hailing services and autonomous delivery services.

### AI Response 2

Self driving cars are cars that are equipped with sensors, cameras, and software, which allows them to navigate without the need for a driver. Self driving cars are able to monitor their environment and navigate without the assistance of a driver. Self driving cars use a combination of technologies such as GPS, radar, lidar, and computer vision to detect their environment and avoid obstacles. They are also able to update their maps and adjust their routes to account for variables such as traffic and construction.

The main benefit of self driving cars is their ability to reduce crashes and save lives. Autonomous vehicles have the potential to save up to 30,000 lives annually. They are also more energy efficient than traditional vehicles, as they do not require a driver who has to maintain a minimum speed. Autonomous vehicles also create new opportunities for increased mobility, allowing those who are unable to drive to get around.

### Rating

- Response 1 is better.
- Response 1 is only slightly better. (only pick this if it's truly close)
- Response 2 is only slightly better. (only pick this if it's truly close)
- Response 2 is better.

**Optional explanation (never required): for cases where rating is near random.**

Figure 14: Our annotation interface.