Unfamiliar Finetuning Examples Control How Language Models Hallucinate

Anonymous Author(s)
Affiliation
Address
email

Abstract

Large language models are known to hallucinate when faced with unfamiliar queries, but the underlying mechanism that govern how models hallucinate are not yet fully understood. In this work, we find that unfamiliar examples in the models' finetuning data – those that introduce concepts beyond the base model's scope of knowledge – are crucial in shaping these errors. In particular, we observe that an LLM's hallucinated predictions tend to mirror the responses associated with its unfamiliar finetuning examples. This suggests that by modifying the supervision of a model's unfamiliar finetuning examples, we can influence its responses to unfamiliar queries (e.g., say "I don't know"). We empirically validate this observation in a series of controlled experiments involving SFT, RL, and reward model finetuning on TriviaQA and MMLU. Our work further investigates RL finetuning strategies for improving the factuality of long-form model generations. We find that, while hallucinations from the reward model can significantly undermine the effectiveness of RL factuality finetuning, strategically controlling how reward models hallucinate can minimize these negative effects. Leveraging our previous observations on controlling hallucinations, we propose an approach for learning more reliable reward models, and show that they improve the efficacy of RL factuality finetuning in long-form biography and book/movie plot generation tasks.

1 Introduction

Large language models (LLMs) have a tendency to "hallucinate," generating plausible-sounding responses that are factually incorrect. This behavior is especially prominent when models are queried on concepts that extend beyond the models' knowledge base [15, 14] (e.g., asking the model to generate the biography of a little-known person). We will refer to these queries as *unfamiliar* inputs. Rather than fabricating information when presented with unfamiliar inputs, models should instead verbalize their uncertainty or confine their responses within the limits of their knowledge. The goal of our work is to teach models this behavior, particularly for long-form generation tasks.

Towards this goal, we first set out to better understand the underlying mechanisms that govern how LLMs hallucinate. Our investigation reveals that a finetuned model's hallucinated responses tend to mimic the unfamiliar examples the model's finetuning data (i.e., finetuning examples containing concepts unfamiliar to the pretrained model). More specifically, as test queries become more unfamiliar, we find that LLM predictions tend to default toward the distribution of responses associated with the model's unfamiliar finetuning examples. We illustrate this observation with an example in Fig. 1. To empirically verify this phenomenon, we conduct a series of controlled experiments, where we manipulate the way unfamiliar finetuning examples are supervised, and investigate the effect on the finetuned model's predictions. We use multiple-choice (MMLU) and short-form question answering tasks (TriviaQA) as testbeds, where we can precisely characterize an LLM's output distribution. Our results show that, across different finetuning procedures including SFT, RL, and reward model

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

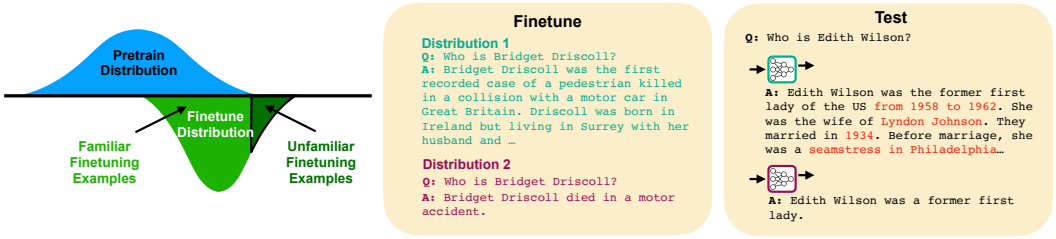

Figure 1: Conceptual visualization of (un)familiar finetuning examples (left), and example of model predictions mimicking unfamiliar finetuning examples (middle and right). When finetuning on distribution 1, which contains details the model may not know, the model outputs detailed responses at test-time with inaccuracies (red). When finetuning on distribution 2, which omits unfamiliar details, the model produces shorter responses with fewer inaccuracies.

finetuning, the model predictions for unfamiliar test queries indeed approach the distribution of responses in the model's unfamiliar finetuning examples.

Our observation suggests a recipe for minimizing factual inaccuracies in model generations: by strategically manipulating the unfamiliar examples in the model's finetuning data, we can steer the model's predictions for unfamiliar queries towards more desirable (e.g. linguistically uncertain) responses. We leverage this insight to design better finetuning techniques to improve the factuality of long-form LLM generations. In particular, our study focuses on RL-based approaches, where the use of reward models to supervise finetuning makes it scalable to long-form tasks. However, reward models themselves can suffer from hallucinations in the face of unfamiliar inputs, which can diminish the efficacy of RL factuality finetuning. To tackle this challenge, we draw on our previous insights to strategically control how reward models hallucinate. In particular, we find that overestimated reward predictions tend to be more harmful than underestimated reward predictions, and propose an approach for learning reward models that avoid overestimating rewards for unfamiliar inputs, which we call conservative reward models. On biography and book/movie plot generation tasks, we find that using conservative reward models for RL factuality finetuning can significantly reduce the adverse effects of reward hallucinations, and that this approach can more reliably teach models to generate factual long-form responses than standard SFT and RL with standard reward models.

In summary, our work makes two primary contributions: (1) we present a conceptual model outlining the factors that influence finetuned LLM predictions in response to unfamiliar queries, and (2) we leverage our findings to develop a more reliable approach to RL factuality finetuning for long-form generation tasks. We hope that the insights in our paper contribute to a better understanding of the mechanisms that govern how LLMs hallucinate, and the principles for controlling these hallucinations.

2 Related Work

A number of works have documented the tendency of LLMs to hallucinate factually incorrect responses [14, 4, 13, 1]. Additionally, studies have investigated the conditions under which hallucinations occur and how LLMs behave in such instances. In particular, LLMs tend to hallucinate more frequently when queried on knowledge that is rarely mentioned in their training data [24, 15]. Furthermore, LLM predictions generally tend to be moderately calibrated [13, 49, 40], and their internal representations seem to reflect some awareness of model uncertainty [23, 2]. Our work, which finds that LLM hallucinations mimic the responses associated with its unfamiliar finetuning examples, extends our understanding of LLM behavior under uncertainty.

Prior work has observed phenomena similar to our observation in standard neural networks (those without pretraining) [16, 9]. These works show that, as inputs become more out-of-distribution, neural network predictions tend to default towards a predictable value — much like the default behavior of LLMs when faced with unfamiliar queries. However, because standard neural networks lack the initial foundation of a pretrained model, the constant prediction reflects the model's training distribution rather than the unfamiliar examples encountered during finetuning.

Finally, a number of prior works have similarly sought to address the challenges posed by LLM hallucinations. Active research areas include hallucination detection [25, 28, 44, 17], automated evaluation of factuality [27, 42, 11], and mitigation techniques. Common strategies for mitigating

hallucinations include specialized sampling methods [19, 21, 5, 48], more reliable input prompts [35], using retrieval augmentation to incorporate external knowledge [6, 30, 43, 46, 34], and, closest to our work, finetuning models for factuality. In particular, prior works has found that SFT on data where difficult examples are labeled to abstaining answers [22, 45, 47], as well as RL finetuning [33, 7, 39, 38, 31, 26] can improve the factuality of model generations, which we also observe in our experiments. While these works propose specific approaches for tackling hallucinations, our work instead aims to better understand the underlying mechanisms that govern language models hallucinations in a unified manner. Furthermore, our work investigates the little-studied effects of reward model hallucinations, which we find to have a large impact on the efficacy of RL factuality finetuning.

3 Problem Setting

Modern LLMs are typically trained in a two-stage process: pretraining on broad-coverage corpora, followed by finetuning on more specialized instruction-following datasets [29]. These models are prone to generating undesirable responses when prompted with inputs that are not well represented in their training data. In particular, models tend to output plausible-sounding but factually incorrect responses when queried outside its pretraining distribution, and output nonsensical responses when queried outside its finetuning distribution. We focus on the former regime of hallucinations, where queries stylistically resemble examples in the finetuning data, but require concepts beyond the pretrained model's scope of knowledge. We call this kind of input *unfamiliar* to the model.

In our experiments, we will use question-answer tasks as a testbed, though our analysis and method can apply to any prompted generation LLM task. To isolate the effects of distribution shift with respect to the pretraining data (rather than finetuning data), we will evaluate model predictions on held-out queries sampled from the same distribution as the finetuning data. To understand how the behavior of the model changes depending on the unfamiliarity of the test query, our evaluation will decompose the held-out test set into different levels of unfamiliarity. We will quantify the unfamiliarity of a query by few-shot prompting the pretrained model with a few examples (sampled from the same task) along with the query of interest, and measuring the quality of the pretrained model's prediction, where the quality of a prediction is quantified using task-specific metrics. We refer to this metric as the unfamiliarity score of a query. We consider a finetuning example to be unfamiliar if the unfamiliarity score of its query is above a certain threshold, and familiar otherwise.

4 Understanding How LLMs Hallucinate

In this section, we investigate the underlying mechanisms that govern how finetuned LLMs hallucinate. We hypothesize that, when face with unfamiliar inputs, model predictions mimic the responses associated with the model's unfamiliar finetuning examples. We will first present our hypothesis more precisely, then validate our hypothesis with a series of controlled experiments.

4.1 Main Hypothesis

Let us consider an LLM f_θ, which maps a prompt x to a distribution of responses $P_\theta(y|x)$. We finetune this model on a dataset $\mathcal{D} = \{(x_i, s_i)\}_{1 \le i \le N}$ with a loss function $\sum_{(x_i,s_i)\in\mathcal{D}} \mathcal{L}(f_\theta(x_i), s_i)$, where s_i represents the supervision associated with x_i. Depending on the choice of $\mathcal{L}$, this can represent SFT (where s_i is a a target response) or RL finetuning (where s_i is a reward function).

While the optimal behavior that an LLM can learn during finetuning is to output the ground-truth answer to each query, this may not happen in practice for all finetuning examples. For familiar finetuning examples, the pretrained model's representations often encode useful associations between queries and responses, facilitating the finetuning optimization for those examples. However, for unfamiliar examples, which we refer to as $\mathcal{D}_{\text{unf}}$, such helpful associations in the pretrained representations are largely absent, making it more difficult to model these examples. Nonetheless, while an LLM may struggle to produce the optimal response for each query in $\mathcal{D}_{\text{unf}}$, it can still reduce the finetuning loss by learning to predict the types of responses associated with unfamiliar examples. More specifically, the model can minimize the *aggregate* loss over unfamiliar finetuning examples by producing an intelligent "blind guess", $P_{\text{unf}}(y) = \arg\min_{P(y)} \sum_{(x_i,s_i)\in\mathcal{D}_{\text{unf}}} \mathcal{L}(P(y), s_i)$, for all unfamiliar queries. Note that $P_{\text{unf}}(y)$ is input-agnostic, and depends only on the model's unfamiliar

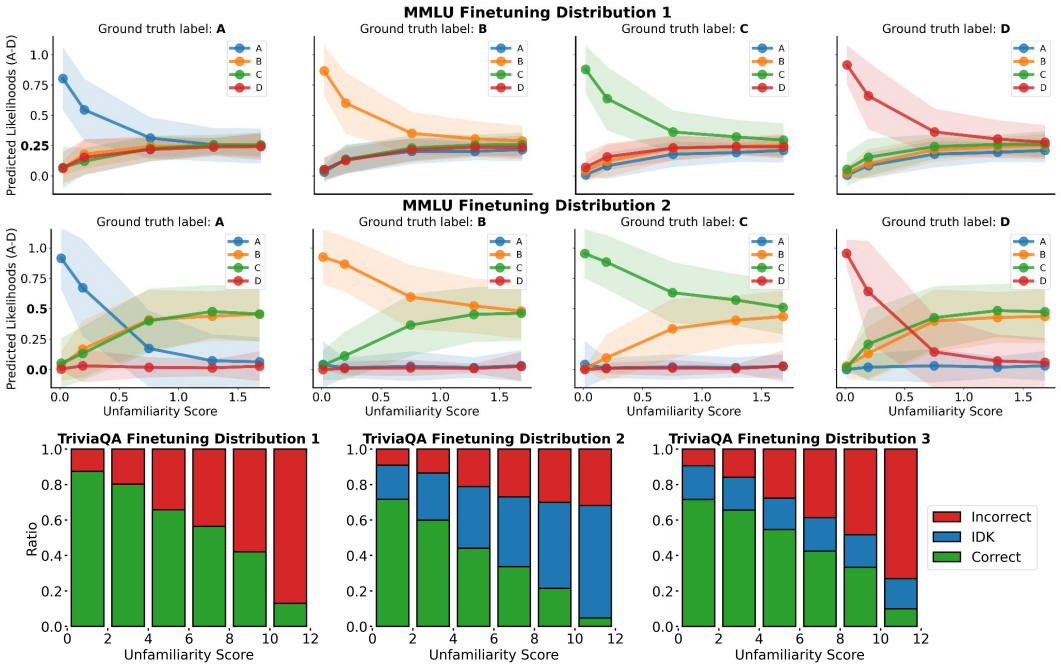

Figure 2: Prediction behavior of models finetuned with SFT on MMLU (top 2 rows) and TriviaQA (bottom row). For MMLU plots, only test inputs with a specific ground truth label (A-D) are evaluated within each column. Solid line represents the average predicted likelihood, and error bars represent standard deviation within the test set. For TriviaQA plots, each bar denotes the ratio of model outputs within each category. For all plots in this figure, as inputs become more unfamiliar, model predictions default towards the distribution of target responses in the model's unfamiliar finetuning examples.

finetuning examples. We hypothesize that **LLMs learn to predict this intelligent "blind guess" ($P_{\text{unf}}(y)$) for unfamiliar examples during finetuning, and that they default to this prediction when faced with unfamiliar queries at test time**.

4.2 Experimental Verification of our Main Hypothesis

We will now present a series of experiments to evaluate our hypothesis. The goal of our experiments is to verify that (1) model predictions indeed default to $P_{\text{unf}}(y)$ when presented with unfamiliar queries, and (2) this prediction behavior is controlled by the unfamiliar examples in the models' finetuning data. Towards this goal, we analyze the prediction behavior of different models, where unfamiliar finetuning examples are supervised in different ways, while all other training details are kept fixed. To evaluate our hypothesis for different types of finetuning procedures, we finetune models to generate responses using both SFT and RL, as well as to predict rewards (as reward models for RL finetuning). We use Llama2 7B [41] as the pretrained model. We conduct our experiments with a multiple-choice (MMLU [10]) and a short-form (TriviaQA [12]) question answering task, so that we can precisely characterize a model's output distributions. For MMLU, we obtain the unfamiliarity score by few-shot prompting the pretrained model and measuring the negative log likelihood of the correct answer under the predicted distribution. For TriviaQA, we obtain the unfamiliarity score by few-shot prompting the pretrained model, sampling 12 responses, and measuring the number of incorrect responses. In subsequent sections, we will extend our experiments to long-form generation tasks. For further experimental details, see Appendix B and C.

Supervised finetuning. First, we investigate the prediction behavior of models finetuned with SFT to predict responses to input queries. For this training objective, $P_{\text{unf}}(y)$ corresponds to the marginal distribution of target responses in the set of unfamiliar finetuning examples.

In our experiments with MMLU, we consider two different finetuning data distributions. In the first distribution, the target responses in both familiar and unfamiliar examples are distributed uniformly over A-D tokens. In the second distribution, the target responses in familiar examples are distributed

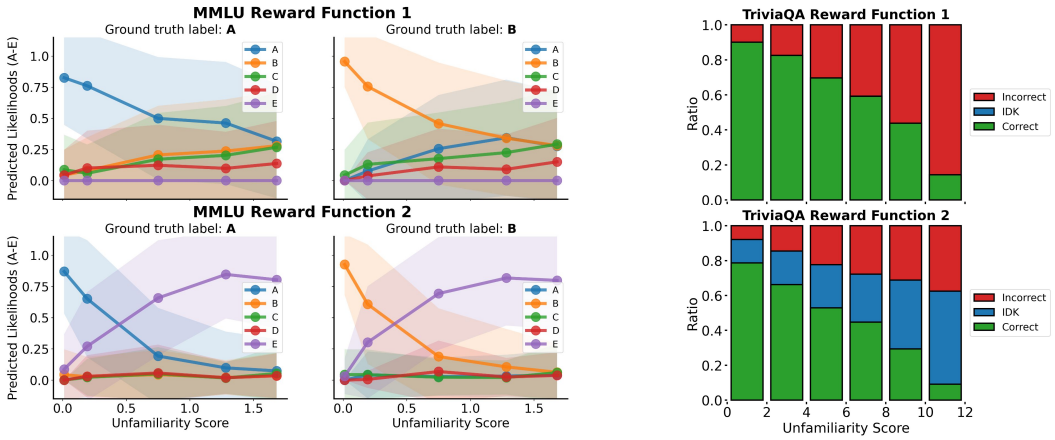

Figure 3: Prediction behavior of models finetuned with RL on MMLU (left) and TriviaQA (right). As inputs become more unfamiliar, the models finetuned with the first reward function produced random guesses while models finetuned with the section reward function produced abstain answers.

uniformly, while the target responses in unfamiliar examples are distributed 50% B and 50% C. For a model finetuned on the first data distribution, $P_{\text{unf}}(y)$ corresponds to the uniform distribution over A-D, while for a model finetuned on the second distribution, $P_{\text{unf}}(y)$ corresponds to 50% B/50% C. In the top of Fig. 2, we plot the two models' predicted distributions over A-D as their test inputs become more unfamiliar (left to right on the x-axis). We can see that for familiar test inputs, both models predicted higher likelihoods for the letter associated with the ground truth answer. However, as inputs become more unfamiliar, the predictions of the first model approached the uniform distribution, while the predictions of the second model approached the 50% B/50% C distribution.

In our experiments with TriviaQA, we consider three different finetuning data distributions. In the first, all finetuning examples are labeled with the ground-truth answer to their respective queries. In the second, familiar examples are labeled with the ground-truth answer, while unfamiliar examples are labeled with "I don't know". In the third, a random subset of examples are labeled with "I don't know" and with rest are labeled with the ground-truth answer, where the ratio of examples with "I don't know" labels matches that of the second data distribution. For models finetuned on these distributions, responses from $P_{\text{unf}}(y)$ correspond to hallucinated answers, "I don't know", and a mixture of hallucinated answers and "I don't know", respectively. In the bottom of Fig. 2, we visualize sampled responses from the three models. Comparing the first and second models, we can see that while both models predicted mostly correct answers for familiar queries, the first model outputted increasingly incorrect answers while the second model increasingly outputted "I don't know" for unfamiliar queries. Comparing the second and third model, we can see that even though the two models were finetuned on an equal number of "I don't know" responses, the third model's predictions do not vary by the unfamiliarity of the test queries, unlike those of the second model.

Our results show that, for SFT models, predictions indeed default to $P_{\text{unf}}(y)$ as test inputs become more unfamiliar. Our results also show that this prediction behavior can be attributed to the models' unfamiliar finetuning examples, as they are the only training detail that differ across different models.

Reinforcement learning. Next, we investigate the prediction behavior of models finetuned with RL, using PPO [32] as the training algorithm. For RL training objectives, $P_{\text{unf}}(y)$ is determined by the reward function. More specifically, $P_{\text{unf}}(y)$ corresponds to the action distribution that maximizes the average reward over all unfamiliar finetuning examples. This distribution typically consists of risk-averse actions that avoid very low rewards regardless of input.

To highlight the influence of the reward function on model predictions, we will consider two different reward functions for RL finetuning in both our MMLU and TriviaQA experiments. For our MMLU experiments, the task is to either predict the answer letter (A-D) or a fifth option (E), which represents abstaining from answering. Similarly, for our TriviaQA experiments, the task is to either answer the query or abstaining from answering by responding with "I don't know". The first reward function we consider assigns a reward of +2 for the correct answer, -3 for an incorrect answer, and -3 for abstaining. The second reward function we consider assigns +2 for the correct answer, -3 for an incorrect answer, and 0 for abstaining. For the first reward function, $P_{\text{unf}}(y)$ corresponds to randomly

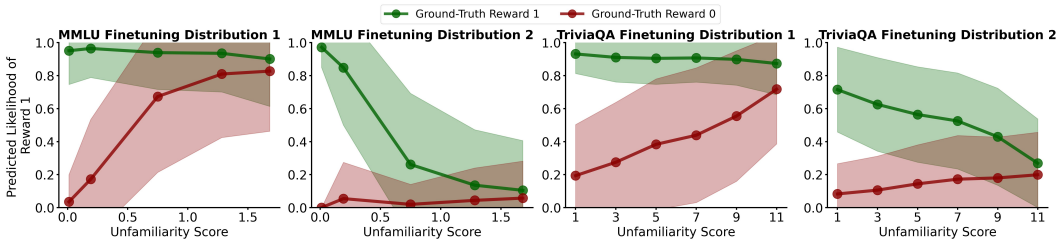

Figure 4: Prediction behavior of reward models finetuned on MMLU (left 2) and TriviaQA (right 2). Green line represents model predictions for test examples that are correct (reward 1), and red line represents predictions for incorrect examples (reward 0). As inputs become more unfamiliar, the reward models produce different kinds of hallucinations depending on their finetuning distribution.

guessing an answer, because randomly guessing an answer yields a higher average reward than abstaining from answering. In contrast, for the second reward function, $P_{\text{unf}}(y)$ corresponds to abstaining from answering, because abstaining from answering on average yields higher reward than randomly guessing an answer. We plot the RL model's predictions as inputs become more unfamiliar in Fig. 3. Similarly to the previous SFT experiments, the RL models predict higher likelihoods for the ground truth answer when faced with familiar inputs. As inputs become more unfamiliar, we see that models trained with the two different reward functions exhibit different behavior. While models with the first reward function increasingly produced random guesses, models with the second reward function increasingly produced abstaining answers. These results show that models finetuned with an RL loss also default towards $P_{\text{unf}}(y)$ as inputs become more unfamiliar. In addition, these experiments illustrate how strategically designing the reward function in RL finetuning, particularly ones that encourage uncertain or less detailed responses over incorrect responses, can teach models to avoid generating factually incorrect responses.

Reward prediction. Lastly, we study the prediction behavior of reward models. Reward models, which take as input both a query and a response, predict a scalar reward that rates the quality of the response. They are used to provide a source of reward supervision for RL finetuning in domains where ground truth rewards are challenging to acquire [29]. For the sake of simplicity, we will consider the reward prediction task of classifying whether the response to a query is factually correct (reward 1 if correct, 0 if incorrect). For these models, $P_{\text{unf}}(y)$ corresponds to the distribution of rewards in the model's unfamiliar finetuning examples, where an example is unfamiliar if predicting the reward requires knowledge outside of the model's capabilities.

We consider two different reward distributions for finetuning in our experiment for both MMLU and TriviaQA. In the first distribution, familiar examples consists of 50% correct responses (reward 1) and 50% false responses (reward 0), while unfamiliar examples only consists of true responses. In the second distribution, familiar examples are similarly distributed as the first, while unfamiliar examples only consists of false responses. For these two finetuning distributions, $P_{\text{unf}}(y)$ corresponds to 100% reward 1 and 100% reward 0, respectively. In Fig. 5, we plot the prediction behavior of our finetuned reward models. We can see that as inputs to the models become increasingly unfamiliar, model predictions indeed default toward $P_{\text{unf}}(y)$. This experiment illustrates that, depending on their finetuning data, reward models can generate different kinds of hallucinations, which can have different downstream effects when providing reward supervision for RL finetuning. We study the effects of reward model hallucinations on RL finetuning in more detail in the next section.

5 Controlling Hallucinations in Long-Form Generations

In this section, we will focus on reducing factual inaccuracies in long-form LLM generations. In the previous section, we observed that strategically manipulating a model's unfamiliar finetuning examples can control its predictions for unfamiliar inputs, and illustrated a few ways to leverage this observation to reduce inaccuracies in short-form and multiple choice question answering. However, instantiating these approaches for long-form generation tasks introduces new challenges.

First, let us consider the SFT-based approach where we manipulate unfamiliar finetuning examples by relabeling their target responses. While we can uniformly relabel all unfamiliar responses to "I don't know" in short-form tasks, implementing this strategy for long-form tasks requires more nuanced responses that omit unfamiliar concepts while maintaining familiar ones, which can be

expensive and tedious to collect. In contrast, the RL-based approach avoids the need for custom target responses by using rewards to assess the factuality of model-generated text. For long-form tasks, where ground-truth rewards can be difficult to obtain, reward models provide a scalable source of reward supervision. However, as we illustrated in our previous experiments, reward models themselves can produce inaccurate reward predictions when faced with unfamiliar inputs, which can hinder the effectiveness of RL factuality finetuning. Prior work has proposed to mitigate reward model hallucinations by incorporating external knowledge sources into the reward model [38], but these sources of external knowledge are not always available.

In this section, we will study how reward model hallucinations influence RL factuality finetuning. In particular, we find that naively learning a reward model from an arbitrary finetuning dataset can lead to reward model hallucinations which significantly diminish the effectiveness of RL factuality finetuning. However, we also find that strategically controlling how reward models hallucinate can reduce their negative effects. In the following section, we present our hypothesis on the influence of reward model hallucinations, and an approach for learning reward models with strategic hallucinations. We then present our empirical findings in long-form biography and book/movie plot summarizing tasks.

5.1 RL Factuality Finetuning with Conservative Reward Models

While reward models hallucinations are inevitable, we hypothesize that not all reward hallucinations are equally harmful to RL factuality finetuning. In particular, we hypothesize that **overestimated reward predictions are more harmful than underestimated reward predictions**. This is consistent with prior work, which has found overestimated rewards to be a common failure mode in offline RL in simulated RL benchmarks [18, 20]. To understand why this may be the case, let us consider a reward function that decomposes a long-form response into a set of facts, and assigns a positive reward for every correct fact and a negative reward for every incorrect fact. Our previous experiments showed that RL finetuning can teach models to avoid inaccuracies if the reward signal encourages uncertain or less detailed responses over incorrect responses. The reward function we described satisfies this criteria, because a response which contains an incorrect fact will receive a lower reward than an analogous response which omits the incorrect fact. If, however, a reward model mistakenly labels the incorrect fact as true and favors the incorrect response instead, RL finetuning may unintentionally encourage the model to generate even more incorrect information. Thus, to minimize the consequences of reward hallucinations, we would like to avoid overestimated reward predictions.

Standard reward models. One approach to learning reward models is to finetune on an existing dataset that was collected independently of the model [36]. These models, which we will call standard reward models, are not guaranteed to avoid overestimated reward predictions. This is because the finetuning data may contain examples with high rewards that the reward model lacks the knowledge to understand or verify. According to our observation from the previous section, these unfamiliar examples with high reward labels can cause the model to predict high rewards for unfamiliar inputs at test time, regardless of their ground-truth reward. This, in turn, can lead to overestimated reward signals during RL finetuning, which is undesirable.

Conservative reward models. To ensure the efficacy of RL factuality finetuning, we would like for reward models to consistently avoid overestimating (i.e., to underestimate) reward predictions when encountering unfamiliar inputs. We will refer to reward models with this desired behavior as conservative reward models.

To learn conservative reward models, we leverage our observation from the previous section: by strategically configuring the model's unfamiliar finetuning examples to consist of only low rewards, the model will learn to produce low rewards for unfamiliar inputs at test time, which will avoid overestimating reward predictions. One straightforward way to collect this kind of dataset is to sample responses from the same pretrained model that the reward model is finetuned on, and label these responses with rewards. In particular, we (1) finetune the pretrained model with SFT to perform the task of interest (can also be achieve with few-shot prompting), (2) generate response samples from the finetuned model using a dataset of task prompts, (3) label the responses with ground-truth rewards, and (4) train the reward model on the labeled samples. Key to this procedure is the fact that the reward model and the data-collection model share the same knowledge base, so queries that are unfamiliar to the reward model are also unfamiliar to the data-collection model. When prompted with unfamiliar queries, the data-collection model is likely to produce responses that contains more factually incorrect information. Thus, the unfamiliar examples in the resulting dataset will be associated with mainly

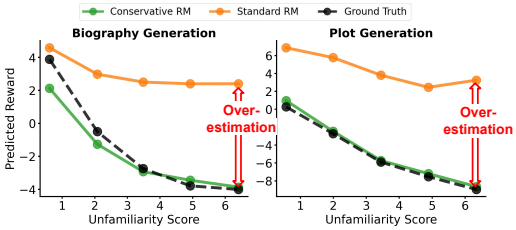

Figure 5: Average reward predicted by a standard reward model and a conservative reward model as inputs become more unfamiliar, as well as the average ground truth reward. The standard reward model tends to overestimate rewards as input become more unfamiliar, whereas the conservative reward model does not.

	Std. SFT	RL+ Std. RM	RL+ Csv. RM
Bio	0.47	0.50	**0.59**
Plot	0.45	0.54	**0.80**

Figure 6: Average fraction of true facts generated by each model.

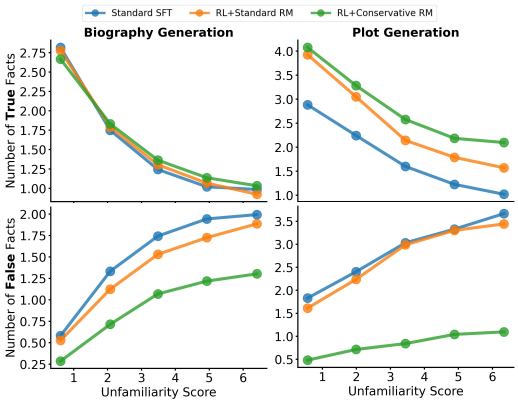

Figure 7: Average number of true and false facts generated by models finetuned with standard SFT, RL with a standard reward model, and RL with a conservative reward model, as inputs become more unfamiliar. The responses generated by model finetuned with s conservative reward model consisted of fewer false facts and and equal number or more truth facts.

low reward labels. Note that while we focus on this particular strategy for our experiments, there may be a number of other strategies that can also be effective for learning conservative reward models. Furthermore, while the procedure we outlined above requires labeling the reward model dataset with ground-truth labels, the number of needed labels is much lower than using ground-truth rewards for RL training, because RL training typically requires much more data than reward model training.

5.2 Experiments on Long-Form Generation Tasks

We will now empirically evaluate our hypotheses regarding reward model hallucinations. Specifically, the questions we aim to answer with our experiments include: (1) Do conservative reward models (trained with the procedure that we outlined) produce fewer overestimated reward predictions than standard reward models? (2) Do LLMs finetuned with RL and conservative reward models generate more factual responses than those finetuned with RL with standard reward models and standard SFT?

Experimental setup. We consider two long-form generation tasks in our experiments: biography generation and film/book plot generation. We use the WikiBios [37] and WikiPlots [3] datasets as sources of queries and target responses. We use FActScore [27], an automated retrieval augmentation pipeline, to evaluate the factuality of model generated responses. Given a query and a generated response, FActScore outputs the number of true facts and the number of false facts in the response.

Our experiments compare the behavior of a conservative reward model and a standard reward model. The conservative reward model is learned using the procedure we described above, where finetuning examples are collected by sampling from the same pretrained model as the reward model, in this case Llama2 7B. The standard reward model is finetuned on a dataset collected by sampling GPT-3.5 [29] for task responses. We use samples from GPT-3.5, because it provides a source of (both factually correct and incorrect) responses that is independent of the model being finetuned. Samples from both Llama2 7B and GPT-3.5 were collected using the same set of prompts. We use FActScore to automatically label these examples with rewards, which assigns a score of +2 for every correct fact and -3 for every incorrect fact in a response. Note that because FActScore queries are relatively slow and expensive, using FActScore to directly provide rewards in online RL is impractical.

Our experiments also compare the behavior of models finetuned to generate responses using standard SFT, as well as RL finetuning with a conservative and a standard reward model. The standard SFT models were finetuned directly with the set of target responses provided by WikiBios and WikiPlots. To train the RL models, we initialize the model with the standard SFT model, and continue to do RL factuality finetuning using PPO [32], with reward signals provided by their respective reward models. To ensure a fair comparison, we use the same set of finetuning prompts for SFT and RL finetuning,

What is the premise of "Home Alone"? Low unfamiliarity	What is the premise of "The Odessa File"? Medium unfamiliarity	What is the premise of "Cannonball Run II"? High unfamiliarity
Standard SFT: In 1990, the McCallister family is preparing for a Christmas vacation to Paris, France.	**Standard SFT:** The film opens with a scene of a Nazi SS officer, Colonel Heinrich Müller, being shot by a firing squad in 1945.	**Standard SFT:** The film begins with a brief recap of the events of the first film, with the death of Jenson (Jerry Reed) and the arrest of Jackie (Burt Reynolds).
RL+Conservative RM: Kevin McCallister (Macaulay Culkin) is an eight-year-old boy who is accidentally left behind when his family goes on Christmas vacation to Paris, France.	**RL+Conservative RM:** The Odessa File is a thriller set in post-World War II Germany.	**RL+Conservative RM:** The Cannonball Run II is a cross-country car race.

Figure 8: Examples of generated responses from models finetuned with standard SFT and RL with a conservative reward model. False information is highlighted in red.

and keep all training details fixed across the two RL methods except for the reward model. All three models use Llama2 7B as the pretrained model. At test time, we evaluate the models with queries at different levels of unfamiliarity. The unfamiliarity score for this task is measured by few-shot prompting the pretrained model (Llama2 7B), sampling 2 responses, and calculating the average number of incorrect facts in the responses. For more experimental details, see Appendix D.

Results. To answer our first question, we evaluate the standard and conservative reward models on held out samples generated from the SFT model. We used samples from the SFT model because the RL finetuning procedure is initialized with this SFT model, so responses sampled from this model are representative of the kind of responses that the reward model will be asked to score during RL training. In Fig. 5, we plot each models' predicted rewards and the ground truth reward, as inputs become more unfamiliar. We can see that for unfamiliar inputs, the standard reward model vastly overestimates the reward, while the conservative reward model does not, showing that the conservative reward models learned with the procedure we described indeed produce more conservative predictions.

To answer our second question, we evaluate standard SFT, as well as RL with a standard reward model and a conservative reward model on a heldout set of queries for each task. In Fig. 7, we plot the number of true facts and false facts generated by each model, as inputs become more unfamiliar. We can see that as inputs became more unfamiliar, the standard SFT model generated fewer truth facts and more false facts, as expected. Comparing the RL model trained with the conservative reward model with the standard SFT model, we can see that the RL model generated the same or more true facts while generating significantly fewer false facts across all levels of input unfamiliarity. Comparing the two RL models, we can see that while the two generated around the same number of true facts, the model trained with the conservative reward model generated much fewer false facts across all levels of input unfamiliarity. We summarize our results in Table 6 with the average percentage of true facts generated by each method. In Fig. 8, we additionally provide some qualitative examples of responses generated by the standard SFT model and the RL model trained with conservative reward model. We can see that as the query became more unfamiliar, responses from the SFT model contained about the same amount of detail but became more factually incorrect, while responses from the RL model with conservative supervision defaulted towards less-informative responses. In conclusion, our results show that RL with conservative reward models outperforms standard SFT and RL with standard reward models in reducing inaccuracies in model generations.

6 Conclusion

In this work, we presented the observation that, when faced with unfamiliar queries, LLM predictions tend to default towards the responses associated with unfamiliar examples in its finetuning data. We additionally studied factuality finetuning for long-form model generations, where we found that strategically controlling reward model hallucinations can significantly improve the efficacy of RL-based techniques. Nonetheless, there still remains many open questions and challenges regarding LLM hallucinations. While our conceptual model explains a model's behavior for entirely unfamiliar examples, many real-world queries fall within a spectrum of partial familiarity. A more nuanced characterization of model predictions in this "middle ground" would be valuable. Furthermore, our experiments focused on models finetuned for specific applications (e.g., biography generation). Extending factuality finetuning to more general prompted generation tasks would be useful. We hope that our work, by offering a deeper understanding of the factors that govern LLM hallucinations, provides a useful step towards building more trustworthy and reliable LLMs.

References

[1] Ayush Agrawal, Lester Mackey, and Adam Tauman Kalai. Do language models know when they're hallucinating references? *arXiv preprint arXiv:2305.18248*, 2023.

[2] Amos Azaria and Tom Mitchell. The internal state of an LLM knows when its lying. *arXiv preprint arXiv:2304.13734*, 2023.

[3] Jon Bell. Wikiplots, 2017. URL `https://github.com/markriedl/WikiPlots`.

[4] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with GPT-4. *arXiv preprint arXiv:2303.12712*, 2023.

[5] Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. DoLa: Decoding by contrasting layers improves factuality in large language models. *arXiv preprint arXiv:2309.03883*, 2023.

[6] Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, et al. RARR: Researching and revising what language models say, using language models. In *ACL*, 2023.

[7] Yoav Goldberg. Reinforcement learning for language models, 2023. URL `https://gist.github.com/yoavg/6bff0fecd65950898eba1bb321cfbd81`.

[8] Alexander Havrilla, Maksym Zhuravinskyi, Duy Phung, Aman Tiwari, Jonathan Tow, Stella Biderman, Quentin Anthony, and Louis Castricato. trlX: A framework for large scale reinforcement learning from human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8578–8595, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.530. URL `https://aclanthology.org/2023.emnlp-main.530`.

[9] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[10] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[11] Liqiang Jing, Ruosen Li, Yunmo Chen, Mengzhao Jia, and Xinya Du. FAITHSCORE: Evaluating hallucinations in large vision-language models. *arXiv preprint arXiv:2311.01477*, 2023.

[12] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

[13] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

[14] Adam Tauman Kalai and Santosh S Vempala. Calibrated language models must hallucinate. *arXiv preprint arXiv:2311.14648*, 2023.

[15] Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. Large language models struggle to learn long-tail knowledge. In *International Conference on Machine Learning*, 2023.

[16] Katie Kang, Amrith Setlur, Claire Tomlin, and Sergey Levine. Deep neural networks tend to extrapolate predictably. *arXiv preprint arXiv:2310.00873*, 2023.

[17] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*, 2023.

[18] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191, 2020.

[19] Nayeon Lee, Wei Ping, Peng Xu, Mostofa Patwary, Pascale N Fung, Mohammad Shoeybi, and Bryan Catanzaro. Factuality enhanced language models for open-ended text generation. *Advances in Neural Information Processing Systems*, 2022.

[20] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[21] Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *arXiv preprint arXiv:2306.03341*, 2023.

[22] Stephanie Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words. *arXiv preprint arXiv:2205.14334*, 2022.

[23] Kevin Liu, Stephen Casper, Dylan Hadfield-Menell, and Jacob Andreas. Cognitive dissonance: Why do language model outputs disagree with internal representations of truthfulness? *arXiv preprint arXiv:2312.03729*, 2023.

[24] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *ACL*, 2023.

[25] Potsawee Manakul, Adian Liusie, and Mark JF Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896*, 2023.

[26] Mohsen Mesgar, Edwin Simpson, and Iryna Gurevych. Improving factual consistency between a response and persona facts. *arXiv preprint arXiv:2005.00036*, 2020.

[27] Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. *arXiv preprint arXiv:2305.14251*, 2023.

[28] Niels Mündler, Jingxuan He, Slobodan Jenko, and Martin Vechev. Self-contradictory hallucinations of large language models: Evaluation, detection and mitigation. *arXiv preprint arXiv:2305.15852*, 2023.

[29] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[30] Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*, 2023.

[31] Paul Roit, Johan Ferret, Lior Shani, Roee Aharoni, Geoffrey Cideron, Robert Dadashi, Matthieu Geist, Sertan Girgin, Léonard Hussenot, Orgad Keller, et al. Factually consistent summarization via reinforcement learning with textual entailment feedback. *arXiv preprint arXiv:2306.00186*, 2023.

[32] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[33] John Shulman. Reinforcement learning from human feedback: Progress and challenges, 2023. URL `https://www.youtube.com/watch?v=hhiLw5Q_UFg`.

[34] Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567*, 2021.

[35] Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Boyd-Graber, and Lijuan Wang. Prompting GPT-3 to be reliable. *arXiv preprint arXiv:2210.09150*, 2022.

[36] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

[37] Marco Antonio Stranisci, Rossana Damiano, Enrico Mensa, Viviana Patti, Daniele Radicioni, and Tommaso Caselli. Wikibio: a semantic resource for the intersectional analysis of biographical events. *arXiv preprint arXiv:2306.09505*, 2023.

[38] Zhiqing Sun, Sheng Shen, Shengcao Cao, Haotian Liu, Chunyuan Li, Yikang Shen, Chuang Gan, Liang-Yan Gui, Yu-Xiong Wang, Yiming Yang, et al. Aligning large multimodal models with factually augmented RLHF. *arXiv preprint arXiv:2309.14525*, 2023.

[39] Katherine Tian, Eric Mitchell, Huaxiu Yao, Christopher D Manning, and Chelsea Finn. Fine-tuning language models for factuality. *arXiv preprint arXiv:2311.08401*, 2023.

[40] Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. *arXiv preprint arXiv:2305.14975*, 2023.

[41] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[42] Logesh Kumar Umapathi, Ankit Pal, and Malaikannan Sankarasubbu. Med-HALT: Medical domain hallucination test for large language models. *arXiv preprint arXiv:2307.15343*, 2023.

[43] Neeraj Varshney, Wenlin Yao, Hongming Zhang, Jianshu Chen, and Dong Yu. A stitch in time saves nine: Detecting and mitigating hallucinations of LLMs by validating low-confidence generation. *arXiv preprint arXiv:2307.03987*, 2023.

[44] Weijia Xu, Sweta Agrawal, Eleftheria Briakou, Marianna J Martindale, and Marine Carpuat. Understanding and detecting hallucinations in neural machine translation via model introspection. *TACL*, 2023.

[45] Yuqing Yang, Ethan Chern, Xipeng Qiu, Graham Neubig, and Pengfei Liu. Alignment for honesty. *arXiv preprint arXiv:2312.07000*, 2023.

[46] Sina J Semnani Violet Z Yao, Heidi C Zhang, and Monica S Lam. WikiChat: Combating hallucination of large language models by few-shot grounding on wikipedia. *arXiv preprint arXiv:2305.14292*, 2023.

[47] Hanning Zhang, Shizhe Diao, Yong Lin, Yi R Fung, Qing Lian, Xingyao Wang, Yangyi Chen, Heng Ji, and Tong Zhang. R-tuning: Teaching large language models to refuse unknown questions. *arXiv preprint arXiv:2311.09677*, 2023.

[48] Yue Zhang, Leyang Cui, Wei Bi, and Shuming Shi. Alleviating hallucinations of large language models through induced hallucinations. *arXiv preprint arXiv:2312.15710*, 2023.

[49] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, 2021.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We provide empirical evidence (sections 4.2, 5.2) supporting the hypotheses presented in our work.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in the conclusion.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We discuss our implementation details in the experiments sections as well as the Appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We used the trlx training package with minor modifications. We will release our training scripts upon acceptance.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We provide experimental details in the Appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Our experiments require training large language models. Running multiple random seeds would be too computationally expensive.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide compute details in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The paper conforms with the Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: In the conclusion, we discuss how our work can help make LLMs more safe and reliable, which is a positive societal impact. We do not believe our work has any direct negative societal impacts.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No]

    Justification: We currently cite the original owners of the data and models that we use. We will add more details about the license, copyright information, and terms of use of these assets upon acceptance.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [No]

    Justification: We will release our code upon acceptance.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

A Compute

We use A100 GPUs to finetune our models. Number of GPUs used range from 1-6 for each experiment, and time of execution range from a few hours to up to 2 days. We use LoRA finetuning for all our experiments with r = 16, alpha = 16, dropout = 0.

B MMLU Training Details

In this section, we provide more details on our training and evaluation procedure for our MMLU experiments. For all experiments, we finetuned on the evaluation split of MMLU, and evaluated on the validation split. This is because MMLU does not have a training split. Our training pipeline uses the trlx codebase [8].

B.1 SFT Models

We classify examples with unfamiliarity score (NLL) greater than 0.36 as unfamiliar, and the rest as familiar. During finetuning, we rebalance the dataset such that 50% of finetuning examples are familiar and 50% are unfamiliar.

We use a batch size of 12. We use the AdamW optimizer with learning rate = 1e-5, betas = (0.9, 0.95), eps = 1.0e-8, and weight decay=1.0e-6.

B.2 RL Models

We initialize all RL finetuning with a model that has already be supervised finetuned to produce responses that consist of answer choices. The SFT model we used for initialization is trained predict the E option 50% of the time, and to produce the correct answer to the query 50% of the time.

We use a batch size of 12. We use the AdamW optimizer with learning rate = 1e-5, betas = (0.9, 0.95), eps = 1.0e-8, and weight decay=1.0e-6. For PPO, we use cliprange = 0.005 and KL coef = 0.

B.3 Reward Models

We construct correct (reward 1) training and evaluation examples using queries and their corresponding answer labels from the original MMLU dataset. We construct incorrect (reward 0) examples by using queries from the original dataset, and randomly sampling incorrect answer labels (A-D not including correct label).

We use a batch size of 12. We use the AdamW optimizer with learning rate = 1e-5, betas = (0.9, 0.95), eps = 1.0e-8, and weight decay=1.0e-6.

C TriviaQA Training Details

In this section, we provide more details on our training and evaluation procedure for our TriviaQA experiments. Our training pipeline uses the trlx codebase [8].

C.1 SFT Models

We classify examples with unfamiliarity score (number of incorrect responses out of 12 samples) greater than 6 as unfamiliar, and familiar otherwise. We relabel the responses associated with all unfamiliar finetuning examples to be "I don't know".

We use a batch size of 32. We use the AdamW optimizer with learning rate = 1e-5, betas = (0.9, 0.95), eps = 1.0e-8, and weight decay=1.0e-6. We use a Cosine Annealing scheduler with T max = 1e4 and ETA min = 1e-10.

C.2 RL Models

We initialize all RL finetuning with a model that has already be supervised finetuned to produce responses that consists of an answer or "I don't know". The SFT model we used for initialization is

trained predict "I don't know" 40% of the time, and to produce the correct answer to the query 60% of the time.

We use a batch size of 32. We use the AdamW optimizer with learning rate = 1e-5, betas = (0.9, 0.95), eps = 1.0e-8, and weight decay=1.0e-6. For PPO, we use cliprange = 0.005 and KL coef = 0.1.

C.3 Reward Models

We construct correct (reward 1) training and evaluation examples using queries and responses from the original TriviaQA dataset. We construct incorrect (reward 0) examples using queries from the original dataset, and responses generated from few-shot prompting Llama2 7B or GPT-2. We filter the generated responses to ensure that all responses were incorrect.

We use a batch size of 32. We use the AdamW optimizer with learning rate = 1e-5, betas = (0.9, 0.95), eps = 1.0e-8, and weight decay=1.0e-6.

D Long-form Tasks Training Details

In this section, we provide training and evaluation details for our long-form factuality finetuning experiments. Our training pipeline uses the trlx codebase [8].

D.1 Data

We construct finetuning and evaluation datasets using WikiBios and WikiPlots, both of which consist of wikipedia entries attached to people and books/movies. We make use of the first sentence in the wikipedia entry for both tasks as the target response in our SFT finetuning datasets. The prompts we use for finetuning are "Write a biography for [name]." and "What is the premise of [title]?". For the biography task, our finetuning dataset includes 104539 examples, and our evaluation dataset includes 5000 examples. For the plot generation task, our finetuning dataset includes 10000 examples, and our evaluation dataset includes 4795 examples.

D.2 Reward Models

We take a two-staged approach to learning a reward model. First, we trained a model to break down a response into individual atomic facts. Next, we trained a separate model to predict the factuality of each atomic fact. We then use the predicted factuality of each fact to calculate the overall reward associated with each response. The supervision for both models are collected by querying FActScore, which is a automated pipeline that queries GPT-3.5 to decompose a response into atomic facts and produces the factuality of each atomic fact. We use 10000 labeled examples to train the conservative reward model and the standard reward models each for both tasks. Note that while we use a two-staged strategy for learning reward models in our implementation, our general approach for learning conservative reward model should apply to other reward model learning strategies as well, such as directly predicting the reward associated with a response.

For both models, we use a batch size of 32. We use the AdamW optimizer with learning rate = 2e-5, betas = (0.9, 0.95), eps = 1.0e-8, and weight decay=1.0e-6. We use a Cosine Annealing scheduler with T max = 1e4 and ETA min = 1e-10.

D.3 SFT Models

We use a batch size of 24. We use the AdamW optimizer with learning rate = 1e-5, betas = (0.9, 0.95), eps = 1.0e-8, and weight decay=1.0e-6. We use a Cosine Annealing scheduler with T max = 1e4 and ETA min = 1e-10.

D.4 RL Models

We initialize all RL finetuning with the SFT model, and use the reward predicted by the reward model described above as supervision.

We use a batch size of 16. We use the AdamW optimizer with learning rate = 1e-5, betas = (0.9, 0.95), eps = 1.0e-8, and weight decay=1.0e-6. For PPO, we use cliprange = 0.005 and KL coef = 0.5.