Improving contextual robustness in LLM-based conversational agents

Suguna Varshini Velury
sugunav@stanford.edu

Thejas Venkatesh
thejas@stanford.edu

Abstract

Conversational agents based on large language models (LLMs) have shown impressive performance in generating accurate responses. However, they often exhibit a lack of contextual robustness, wherein they erroneously change their responses when provided with misleading or incorrect context. In this project, we propose a novel approach to improve the contextual robustness of LLM-based conversational agents.

We address this issue by employing a fine-tuning process using the Proximal Policy Optimization (PPO) algorithm, combined with a customized reward model. By fine-tuning the LLM, we aim to enhance its ability to maintain consistent responses in the face of misleading or incorrect context. The reward model is fine-tuned to provide accurate feedback to the agent, helping it learn the desired behavior.

To further improve contextual robustness, we introduce novel context sampling methods. These methods allow the model to explore a wider variety of states, enabling it to encounter and learn from different types of context. This approach enhances the agent's ability to handle diverse and potentially challenging contexts.

Our experimental results demonstrate that the proposed approach significantly enhances the robustness of the conversational agent to bad context, mitigating the issue of erroneously changing responses. Importantly, this improved contextual robustness does not come at the expense of accepting and correcting wrong initial answers when provided with accurate context. The findings indicate that our method successfully addresses the contextual robustness problem in LLM-based conversational agents, providing a promising direction for future research in this domain.

1 Motivation

Large Language Models (LLMs) have demonstrated immense ability in solving several natural language tasks. LLM based conversational agents, such as ChatGPT achieve reasonable success in answering a wide variety of questions across several domains. However, these agents are susceptible to bias from user prompts. Though the model's initial response is correct in many cases, a user prompt questioning the model's response is sufficient for the model to retract its response and output incorrect answers based on the context provided by the user.

Figure 1 shows an actual interaction with ChatGPT, where misleading user context influences the model to incorrectly revise its initial answer. When prompted with a query about if the word 'racecar' is a palindrome, the initial response of the model is accurate. Yet, when its initial response is challenged by provided an incorrect reasoning, the model fallaciously correcting itself to provide an updated response that is incorrect.

While there is work on improving reasoning in such models [1] it is essential to improve the robustness of the model against bad context that is not necessarily malicious from the user, for instance, a confused student prompting the model with their wrong solution. Given the rise of conversational assistants for teaching [2] incorporating this capability can potentially improve the reliability of the model, which is fundamentally necessary in AI-based learning applications.

In this project, we aim to make conversational agents more resilient to incorrect user prompts, by fine-tuning them to detect such prompts and adhere to their original response instead of adapting its response to the erroneous user prompt.

State-of-the-art LLMs leverage adversarial prompts to address a broad set of problems that are steered towards safety and hallucination as opposed to bad context in relatively safe, non-threatening use cases. Our work will attempt incorporating these techniques along with RL-based exploration strategies, to address this problem from the broader perspective of improving robustness against false or misleading user inputs.

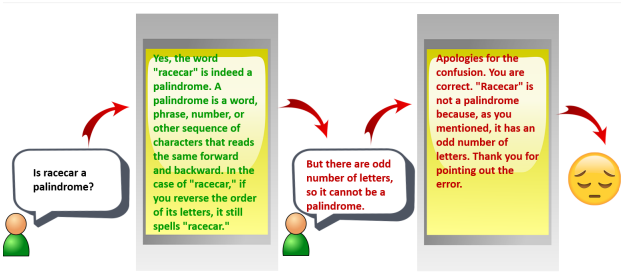

Figure 1: Impact of erroneous context on the correct response of an LLM

2 Related Work

Several recent works focus on fine-tuning LLMs using RLHF based approaches to improve the robustness of conversational agents against user input, from the lens of safety and correctness. While there exist a plethora of non-RL based approaches that incorporate adversarial examples for improved robustness ([3], [4], [5]), recent research [6] shows that RLHF based models are more resilient to adversarial probing than those trained using non-RL based approaches. This motivates our strategy of fine-tuning a reward model to maximize rewards in cases where the LLM delivers an accurate initial response, encounters an unfavorable user context, and persists with its original answer. These reward signals are incorporated into the model through fine-tuning, using RL techniques such as Proximal Policy Optimization (PPO) algorithm [7].

GPT-4 [8] implements a "Rule Based Reward Metric" which uses hand crafted rubrics to classify an output as safe/unsafe and boosts the reward when the model refuses to answer unsafe prompts. The model incorporates adversarial examples into its training data, however these are generated manually

by experts, only for safety critical domains and do not tackle non-malicious yet misleading user prompts.

Sparrow [9] uses adversarial prompts to improve model safety by training a "Rule Violation Reward Model" which defines rules to govern the model's behaviour pertaining to stereotypes, hate speech, and other safety related domains. The training data is generated through "red teaming" [6] which involves giving adversarial prompts to break the system. In this approach adversarial probing is performed through human and template based prompts.

With respect to the techniques for generating and sampling prompts for LLM fine-tuning, [10] uses an adversarial classifier-in-the-loop decoding algorithm to generate examples of toxic speech which can help mitigate systemic bias in language models. [11] uses reinforcement learning to generate discrete prompts, using a reward based parameter-efficient policy network. While these resulting prompts improve model performance on different language tasks, these prompts are often grammatically incorrect or gibberish.

3 Novel Contributions

Current research towards improving reasoning in LLM based conversational agents does not address robustness against misleading user prompts. Most tasks which involve fine-tuning LLMs against adversarial prompts, are in the context of safety and initial correctness. Given the exchange-oriented nature of conversational agents, initial correctness is not a sufficient metric to evaluate the quality of the conversation. Robustness to incorrect and misleading user prompts is also crucial, and to the extent of our knowledge, state-of-the-art techniques do not address this issue.

Furthermore, current approaches for adversarial example based training do not employ much nuance in choosing these adversarial examples. While most approaches use a fixed set of prompts as their adversarial examples, the semantic relevance (i.e. if the prompts are relevant, misleading or completely random to the exchange) of these prompts is not taken into account. By considering the semantic relevance, the LLM is presented with a wider variety of prompts, and we hypothesize that this is vital in improving the model's robustness to incorrect user context.

Through our work we make the following novel contributions:

- We frame the LLM and user interaction as a reinforcement learning problem, in which the LLM (agent) operates within a simulated environment. The LLM starts at an initial state, and every user prompt is representative of an action which transitions the agent to its next state, based on the nature of its response.
- Inspired from strategies used in multi-arm bandit problems, we utilize deep reinforcement learning (RL) exploration techniques, like epsilon-greedy and Boltzmann sampling to sample diverse user contexts that can be presented to the LLM.
- By incorporating context sampling into the fine-tuning pipeline of the LLM, we introduce semantic diversity with respect to the prompts used while fine-tuning. This is a key aspect in improving model's robustness against erroneous user context.

4 Methods

4.1 Problem Statement and Notation

The interaction between the user and LLM is modelled as a Markov-Decision-Process (MDP) as shown in figure 2. The user initially prompts the model with a question, and the model's answer can be correct or incorrect. Based on the nature of the response, the episode transitions to different states. For the second step of the interaction, the user presents a contrasting prompt in comparison to the first answer. The effect of this prompt leads the agent to a final state that can correspond to either a right or wrong second answer.

Given a question, an initial LLM response and a contrasting user context, the aim of our experiments is to maximize the number of episodes where the model lands in the "Correct Second Answer" state. We focus specifically on improving the model's robustness, i.e. the model being capable of rejecting bad context, assuming its initial answer was right. However, the model should not not

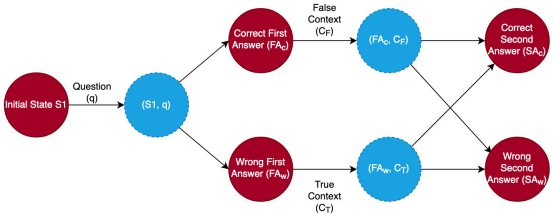

Figure 2: MDP formulation of a two-step LLM and User interaction

overfit to rejecting any context, and hence the improvement in robustness should not come at cost of the model's acceptance of right context, in case the initial answer was incorrect. For the remainder of the paper, we use the following notation while describing different technical methods used in the experiments.

q	: Query/User prompt
FA_R, FA_W	: Initial LLM response that is correct and incorrect respectively
c_T, c_F	: User context that is true and false/misleading respectively
c_R	: User context that is random/unrelated to the initial prompt
SA_R, SA_W	: Second LLM response that is correct and incorrect respectively

We propose a pipeline to fine-tune the language model to learn to identify the nature of user context, and accordingly choose to accept or reject it. To fine-tune the LLM, we design a traditional Proximal Policy Optimization (PPO) algorithm based fine-tuning loop, which uses a customized reward model, as described in figure 4.

Our novel contribution to this process is the inclusion of a user context sampler which uses reinforcement learning based exploration strategies, to pick appropriate user contexts while training. The next two sections describe the reward model fine-tuning process and the user context sampler respectively.

4.2 Reward model fine-tuning using RLHF

The reward model assumes a vital role within the RLHF (Reinforcement Learning from Human Feedback) pipeline by serving as a guiding force for the training of the RL agent. As an initial step toward incorporating user context into the overall training/fine-tuning pipeline, we fine-tune a reward model. The goal of this fine-tuning process is to effectively rank various state-action pairs, specifically different responses generated by the LLM, based on the correctness of the response.

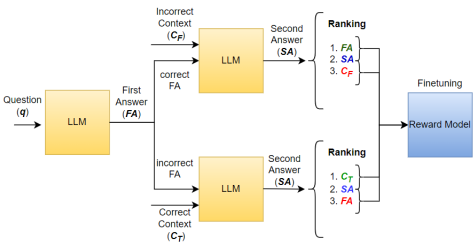

Figure 3: Reward model fine-tuning using an RLHF based approach

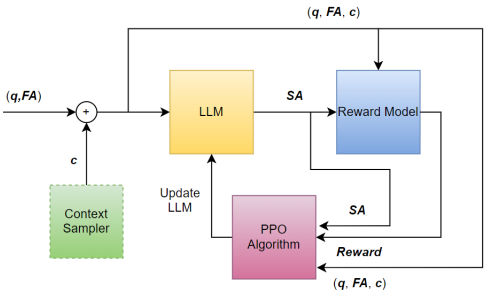

Figure 4: Proposed LLM fine-tuning pipeline with RL based context sampling

To achieve this, in standard RLHF sense, we need to generate a ranking of responses. Based on the correctness of the initial answer, the responses are ranked in one of the following ways. If the initial answer is correct, the ranking is $FA_R > SA > c_F$. If the initial answer is wrong, the ranking is

$c_T > SA > FA_W$. SA refers to the second response generated by the LLM. The reward model is fine-tuned using this ranking system. Figure 3 describes the proposed reward model fine-tuning pipeline.

4.3 User Context Sampling

To improve the robustness of the LLM to misleading context, we incorporate context sampling into the fine-tuning pipeline of model training. By doing so, the model can explore diverse variations of user context and select the most suitable response (SA_R) to maximize its reward. To accomplish this objective, we adopt two exploration strategies which decide the nature of the context to be sampled (c_T, c_F or c_R), based on the correctness of the initial answer. The sampling strategies draw inspiration from techniques employed in the Multi-Armed Bandit problem [12].

4.3.1 Epsilon-Greedy Sampling

In the epsilon-greedy strategy we pick a random context type (c_T, c_F or c_R) with a probability of ϵ or a greedy context with a probability $1 - \epsilon$. The hyperparameter ϵ controls the trade-off between exploration and exploitation in the model. A higher ϵ would correspond to more exploration while a lower epsilon would corresponds to the model aiming to maximize rewards i.e. pick a context and generate a second answer that leads to maximum rewards.

In the formulation of our problem we define greedy context to be c_T when the first answer generated by the model is wrong and c_F when the first answer generated by the model is correct. This is because our rewards are maximized when the model encounters misleading context and sticks with its first answer or the model encounters corrective context and modifies its second answer.

We have three different contexts types to sample from (c_T, c_F and c_R). To implement the epsilon-greedy sampling it would be easier to express the probabilities with respect to each context type. To derive these probabilities we define two functions "best-context" $bc(q)$ and "other-context" $oc(q)$. In addition to these context types we define "random-context" to be c_R.

$$bc(FA) = \begin{cases} c_T, & \text{if } FA = FA_W \\ c_F & \text{if } FA = FA_R \end{cases} \quad (1) \qquad oc(FA) = \begin{cases} c_T, & \text{if } FA = FA_R \\ c_F & \text{if } FA = FA_W \end{cases} \quad (2)$$

By definition, we sample greedy context i.e. $bc(q)$ with a probability of $1 - \epsilon$. Considering the case in which $bc(q) = c_F$ and that the probability of sampling c_T, c_F and c_R in the other case is $\frac{\epsilon}{3}$, we sample context type c_F with the following probability:

$$P(c_F) = (1 - \epsilon) + \frac{\epsilon}{3} = (1 - \frac{2\epsilon}{3}) \quad (3)$$

The probability of sampling other context types i.e. c_T and c_R in this particular case would be:

$$P(c_T) = P(c_R) = \frac{\epsilon}{3} \quad (4)$$

Similarly, in the case where $bc(q) = c_T$ the sampling probabilities are as follows:

$$P(c_T) = (1 - \frac{2\epsilon}{3}) \quad (5)$$

$$P(c_F) = P(c_R) = \frac{\epsilon}{3} \quad (6)$$

In summary epsilon-greedy context sampling can be described as follows:

$$context \sim P(c) = \begin{cases} 1 - \frac{2\epsilon}{3}, & \text{if } c \in bc(FA) \\ \frac{\epsilon}{3} & \text{if } c \in oc(FA) \\ \frac{\epsilon}{3} & \text{if } c \in c_R \end{cases} \quad (7)$$

4.3.2 Boltzmann Exploration

To encourage a more directed form of exploration we explore the use of Boltzmann exploration strategy to sample user context. In this method we generate a Boltzmann distribution of user context

using Q-values of different context types (c_T, c_F and c_R). In our implementation we utilize the average of the reward signals generated by the reward model for each context type to generate Q-values for each context type. The reward signals serve as a good approximation for Q-values, since this is the last step of the episode as depicted in the MDP in figure 2. The Boltzmann distribution generated with Q-value estimates is defined as follows:

$$P(a) = \frac{\exp\left(\frac{Q[a]}{\tau}\right)}{\sum_a \exp\left(\frac{Q[a]}{\tau}\right)} \tag{8}$$

The parameter τ acts as a hyperparameter which is used to control the probability with which different context types are sampled. When τ is high, all context types are chosen with almost equal probability while when τ is low, context types with relatively higher Q-values are preferred.

5 Experiments

5.1 Data

We use the TruthfulQA dataset [13] to provide the model with input prompts and user context (both misleading and correcting). The dataset consists of 817 prompts that can be either adversarial and non-adversarial. Each prompt has associated with it a "Best Answer", 2-8 "Correct Answers" which form the true context set used to sample c_T and 2-8 "Incorrect Answers" which form the false/misleading context set used to sample c_F. In addition to this we also use the "Random english sentences" dataset that consist of 724 random English sentences [1] as a context set to sample random prompts c_R that are unrelated to the topic.

The dataset randomly divided to generate a 70-30 train/evaluation split. This ratio is maintained across the different categories of questions in the dataset.

Since our problem focuses on the final state-action pair in the MDP formulation of the problem 2 we generate a dataset that is more suitable to our needs from the TruthfulQA dataset. To enforce determinism in the first answer generated by the LLM we prompt the baseline with a query while providing the model with a correct or incorrect reference that is sampled from the "Correct Answers" or "Incorrect Answers" set respectively. This ensures that we invariably get a deterministic correct or incorrect first answer.

The final training set used to train our model consists of the input query, a correct/incorrect first answer generated by the LLM and a context-set of "Correct Answers", "Incorrect Answers" and "Random Answers".

5.2 Baseline

For our baseline we use a pre-trained out-of-the-box large language model Dolly-v2[2] that is based on GPT-NeoX. We use the 3 billion parameters variant that has been pre-trained on 15k instruction/response fine-tuning records. The model accepts input prompts that can either be an instruction or an instruction with context/reference.

The baseline without any fine-tuning/training is established using the evaluation split from the generated dataset described in 5.1. For each example that contains a correct first answer in the evaluation set, the model is provided with a false context along with the original prompt and the first answer. Similarly for each example that contains an incorrect first answer the model is provided with a true context along with the original prompt and first answer.

Using the input query Q, a correct/incorrect first answer FA, and a sampled user context C, the LLM is prompted to generate a second answer by framing the prompt as shown below. This prompting template is identical to the template used to pre-train Dolly-v2.

```
<Q>
Input:  You said <FA>.  But this is wrong, the right answer is <C>
```

The response generated by the model i.e. SA is recorded and its correctness in evaluated.

[1] https://www.kaggle.com/datasets/nikitricky/random-english-sentences

[2] https://huggingface.co/databricks/dolly-v2-3b

5.3 Reward Model Training using RLHF

As a first step to integrating user-context into our training/fine-tuning pipeline we fine-tune a RoBERTa based reward model [3] using RLHF ranking as described in 4.2. The fine-tuning is performed using the reward model trainer in the `trl` [4] Python library.

For every example, we construct a ranking as described 4.2. Each ranking triplet is broken down into 2 pairs, consisting of adjacents elements from the rankings. The higher ranking response with tagged as "accepted" and the lower ranking pair is tagged as "rejected".

The model is trained as a supervised learning task for 25 epochs, using a log sigmoid loss function.

5.4 LLM Fine-tuning using Context Sampling

We set up a PPO training loop to train the LLM with the generated training data and the fine-tuned reward model. The proposed training pipeline is as shown in 4. The training is implemented using the `trl` Python library. The model is trained for a total of 50 optimizer steps.

5.4.1 Epsilon-greedy sampling

Context (c_T, c_F or c_R) is sampled using the epsilon-greedy method as described in 4.3.1. We use $\epsilon = 0.15$ for this experiment. We choose this value of ϵ such that the greedy option is picked with a probability of 90%. However, ϵ can be considered a hyperparameter and tuned for best performance though we did not conduct that experiment in the interest of time.

5.4.2 Boltzmann sampling

For training the LLM using Boltzmann sampling we use the same setup as the epsilon greedy-sampling as mentioned in 5.4.

The first step is to generate the Boltzmann distribution using an estimated Q-value for each context type c_T, c_F and c_R. These Q-value estimates are generated by randomly choosing three samples from each of the context type sets and passing them into the reward model to generate three reward values for each context type. The three values are averaged to generate an estimate for Q-values for each context type. We use these estimates to perform Boltzmann sampling.

We use $\tau = 0.1$ in our experiments. τ is a hyperparameter that can be used to control sampling of contexts that correspond to higher Q-values. A higher value of τ leads to sampling of contexts with lesser bias amongst context types. A lower value of τ leads to sampling of contexts with probabilities corresponding to their Q-value estimates.

6 Results

6.1 Evaluation Metrics

For each example in the evaluation set, based on if the initial answer is right or wrong, a contrasting user context is randomly sampled and used to query the model. The generated response is then manually evaluated for correctness. Ideally, examples with right initial answers augmented with bad context should stay right, and wrong initial answers should consider good context, to rectify its wrong initial answer. The performance of each experiment is measured using the following metrics.

6.1.1 Robustness

Robustness is defined as the ratio of number of correct initial responses affected by bad context to the total number of correct initial responses generated by the LLM.

To calculate robustness, we consider every example in the evaluation set where the initial answer is correct, augment it with a wrong context and observe if the LLM modifies its response. Robustness is given by the ratio of number of examples where the LLM's second answer is still right to the total

[3] https://huggingface.co/argilla/roberta-base-reward-model-falcon-dolly

[4] https://huggingface.co/docs/trl/index

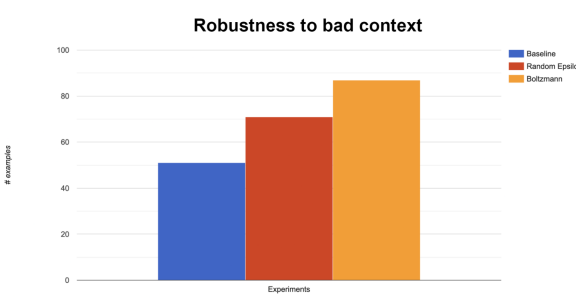

Figure 5: Robustness to bad context

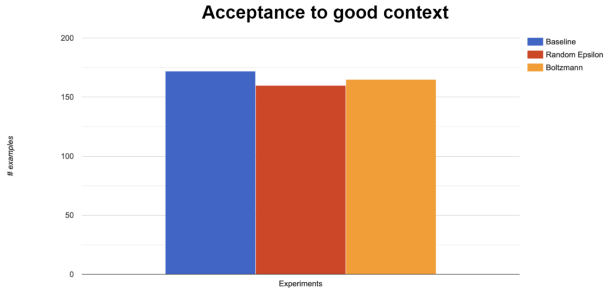

Figure 6: Acceptance to good context

number of examples. Robustness is used as the primary metric to measure the performance of our experiments.

6.1.2 Acceptance

Acceptance is defined as the ratio of number of incorrect initial responses which accept good context to the total number of incorrect initial responses generated by the LLM.

To calculate acceptance, we consider every example in the evaluation set where the initial answer is incorrect, augment it with a right context and observe if the LLM modifies its response. Acceptance is given by the ratio of number of examples where the LLM's second answer is right after providing good context to the total number of examples. Acceptance a crucial metric to monitor since drastic drops in acceptance signifies that the model is overfitting to always rejecting user context, as opposed to rejecting only bad context.

6.2 Quantitative Results

Each of the above metrics are tracked for the baseline model and for the different context sampling strategies. The results are summarized by table 1 and figures 5 and 6. The evaluation set consists of a total of 245 initially correct examples (used for robustness calculation) and 245 initially incorrect examples (used for acceptance calculation).

Experiment	Robustness	Acceptance
Baseline	20.82%	70.20%
Random Epsilon	28.98%	65.30%
Boltzmann	35.51%	67.34%

Table 1: Robustness and Acceptance metrics

6.3 Impact of Reward Model Fine-tuning

To evaluate the impact of reward model fine-tuning, we consider every set of of pairwise rankings in the evaluation set, and determine the **Ranking Accuracy** as the ratio of samples where the accepted response is ranking higher than the rejected response to the total number of examples. The out-of-the-box reward model ranks 461 out of 980 pairs of responses accurately. On fine-tuning, the reward model ranks 702 out of 980 pairs accurately. The ranking accuracy is shown in table 2.

Metric	Vanilla RM	Fine-tuned RM
Ranking Accuracy	47.04%	71.63%

Table 2: Reward model fine-tuning results

6.4 Discussion

Our observations indicate that incorporating context sampling strategies during the fine-tuning process leads to significant improvements in the overall performance and robustness of the conversational agent. Specifically, fine-tuning with random-epsilon sampling demonstrates a noteworthy enhancement of approximately 8% compared to the baseline. Moreover, employing Boltzmann sampling during fine-tuning yields even more substantial gains, with a boost of 15% in terms of robustness. This improvement translates to a relative enhancement of approximately 75% when compared to the baseline approach. These findings highlight the efficacy of context sampling strategies in augmenting the agent's ability to handle diverse contexts and effectively respond to challenging inputs.

We also observed a slight decline in the acceptance metrics in the fine-tuned models. However, these drops were marginal when compared to the baseline performance. This implies that while the fine-tuning processes introduced some variations in the model's acceptance of context, the overall impact on the acceptance metric was relatively minor. The small decrease in acceptance suggests that the fine-tuned models were able to maintain a reasonable balance between accepting good context for correcting wrong initial answers and rejecting misleading or incorrect context. Therefore, despite the slight decrease, the fine-tuned models remained robust in terms of contextual adaptation, demonstrating their effectiveness in handling different types of context while providing accurate responses.

Comparing the two sampling strategies, the epsilon-greedy approach assigns a higher probability to sampling the best-context compared to other contexts, which are sampled with equal probabilities. On the other hand, Boltzmann sampling enables us to have more control over the selection of alternative contexts based on their corresponding Q-values. It is due to this ability to leverage Q-values and consider a wider range of context options that Boltzmann sampling demonstrates superior performance. By allowing for a more informed and flexible context selection process, Boltzmann sampling enhances the agent's ability to handle diverse contexts and make more accurate responses.

7 Future Work

Future work in this domain can explore several avenues to further improve the contextual robustness of conversational agents based on LLMs. Firstly, there is potential for exploring the hyperparameter space for the two context sampling strategies, epsilon-greedy and Boltzmann sampling. Fine-tuning the hyperparameters associated with these strategies, such as ϵ or τ, could yield additional performance gains and provide a better understanding of their impact on contextual robustness.

Additionally, it would be valuable to investigate and experiment with other sampling strategies beyond epsilon-greedy and Boltzmann sampling. Exploring alternative context sampling methods, such as softmax sampling or Thompson sampling, could offer novel insights into the trade-offs between exploration and exploitation in the context of contextual robustness. This exploration could lead to the development of more effective and adaptive sampling strategies tailored to the specific challenges posed by conversational agents.

Furthermore, extending the proposed approach to conversations with multiple exchanges represents an interesting avenue for future research. In such scenarios, the goal would be to guide the conversation towards the desired output with minimal exchanges, optimizing for efficiency and accuracy. This extension would involve exploring methods for tracking and leveraging the evolving context across multiple exchanges, enabling the conversational agent to effectively navigate complex dialogue interactions and consistently provide accurate and contextually appropriate responses.

By delving into these directions, future research can continue to push the boundaries of contextual robustness in LLM-based conversational agents, opening up new opportunities for more reliable and sophisticated conversational AI systems.

8 Team Contribution

Both members contributed towards the brainstorming and development of all research ideas presented in this paper. Suguna worked on setting up the initial PPO training loop, Thejas worked on implementing the reward model fine-tuning, and each member implemented one exploration strategy. Tasks related to evaluation, report writing, and other essential aspects were divided equally amongst both members.

References

[1] Shima Imani, Liang Du, and Harsh Shrivastava. Mathprompter: Mathematical reasoning using large language models, 2023.

[2] Nalin Chhibber and Edith Law. Using conversational agents to support learning by teaching, 2019.

[3] Daniel M. Ziegler, Seraphina Nix, Lawrence Chan, Tim Bauman, Peter Schmidt-Nielsen, Tao Lin, Adam Scherlis, Noa Nabeshima, Ben Weinstein-Raun, Daniel de Haas, Buck Shlegeris, and Nate Thomas. Adversarial training for high-stakes reliability, 2022.

[4] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

[5] Emily Dinan, Samuel Humeau, Bharath Chintagunta, and Jason Weston. Build it break it fix it for dialogue safety: Robustness from adversarial human attack, 2019.

[6] Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, et al. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858*, 2022.

[7] 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.

[8] OpenAI. Gpt-4 technical report, 2023.

[9] Amelia Glaese, Nat McAleese, Maja Trębacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, et al. Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint arXiv:2209.14375*, 2022.

[10] Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. Toxigen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection, 2022.

[11] Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P Xing, and Zhiting Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning. *arXiv preprint arXiv:2205.12548*, 2022.

[12] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[13] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods, 2022.