Article recovery of:

REWARD DESIGN WITH LANGUAGE MODELS

Link to the article: <https://arxiv.org/pdf/2303.00001>

Table of Contents

[Summary 1](#_Toc175841014)

[Algorithm/System Reproduction: A Detailed Description 2](#_Toc175841015)

[Ultimatum game: Training agents with Few-Shots Prompting 2](#_Toc175841016)

[Matrix games: training objective-aligned agents with zero-shot prompting 3](#_Toc175841017)

[Negotiation game: training objective-aligned agents in multi-timestep tasks 6](#_Toc175841018)

[Limitations of LLM-based Reward Design in Reinforcement Learning 8](#_Toc175841019)

[Conclusion: 8](#_Toc175841020)

# Summary

The article "Reward Design with Language Models" tackles the significant challenge in reinforcement learning (RL) of accurately specifying reward functions that align with complex human behaviors and objectives. Traditional approaches often rely on hand-crafted reward functions or require large sets of expert demonstrations to train RL agents effectively. However, these methods can be both labor-intensive and prone to inaccuracies, especially when dealing with nuanced or context-specific human preferences.

To address this, the authors propose an innovative solution that leverages the capabilities of large language models (LLMs), such as GPT-3, to serve as proxy reward functions. Instead of relying on traditional reward specifications, users can provide natural language descriptions or a few examples (few-shot prompts) of the desired behavior. The LLM interprets these inputs and generates corresponding reward signals that guide the RL agent during training.

The researchers validated this approach through experiments across three distinct tasks: the Ultimatum Game, two-player matrix games, and the DealOrNoDeal negotiation task. Each of these tasks presents different challenges and complexity levels, making them suitable test platforms for evaluating the effectiveness of LLM-generated rewards.

1. **Ultimatum Game**: In this game, a proposer suggests how to split a sum of money with a responder, who can accept or reject the offer. The study explores whether an RL agent trained with few-shot prompts can learn to reject proposals that do not meet the user's fairness criteria, even when exact objectives are hard to quantify.
2. **Matrix Games**: We focus on two-player normal-form matrix games, including Battle of the Sexes, Stag Hunt, Chicken, and Prisoner’s Dilemma, selected for their well-established solution concepts. This study explores whether LLMs can generate reward signals consistent with objectives using zero-shot prompting, without any prior examples.
3. **DealOrNoDeal Negotiation Task**: We have shown in the two tasks above that an LLM can provide objective-aligned reward signals in single-timestep tasks. In longer horizon tasks we must give trajectories instead of states as examples in our prompts. Longer prompts can be challenging because it is less likely for an LLM to have seen them during training. We investigate whether LLMs can maintain objective-aligned signals in the DEALORNODEAL negotiation task, a longer-horizon domain requiring complex, multi-interaction rewards.

All these tasks were also trained using traditional reward functions with a supervised learning model for comparison.

The results indicate that agents trained with LLM-generated rewards not only align well with user objectives but often outperform those trained with traditional supervised learning methods. This suggests that LLMs can capture the nuances of human preferences more effectively than conventional approaches.

By simplifying reward design through natural language interfaces, this approach opens up new possibilities for making RL more accessible and intuitive. It allows users without deep technical expertise to influence the behavior of RL agents more directly, using simple language-based descriptions rather than complex coding or extensive training data.

Overall, the paper contributes significantly to the field of RL by demonstrating that LLMs can effectively bridge the gap between human intent and machine learning, providing a more user-friendly approach to reward design that could enhance the applicability and usability of RL in various real-world scenarios.

# Algorithm/System Reproduction: A Detailed Description

## Ultimatum game: Training agents with Few-Shots Prompting

**Objective and Challenge**:

The study explores whether Large Language Models (LLMs) can generate reward signals aligned with user objectives in the Ultimatum Game, especially when defining precise objectives is challenging. Additionally, it compares the effectiveness of LLM-generated rewards to traditional reward functions trained with supervised learning.

**Task description**:

In the Ultimatum Game, two players (Proposer and Responder) decide on splitting a sum of money. The Responder can accept or reject the proposed split. If rejected, both players receive nothing. An RL agent is trained to play the Responder role, learning to reject proposals based on varying user preferences. RL agents are trained using DQN with 10000 steps.

We experiment with the following preferences:

* **Low vs High Percentages**: Responder rejects proposals if they receive less than {30%,60%} of the endowment.
* **Low vs High Payoffs**: Responder rejects proposals if they receive less than {$10, $100}.

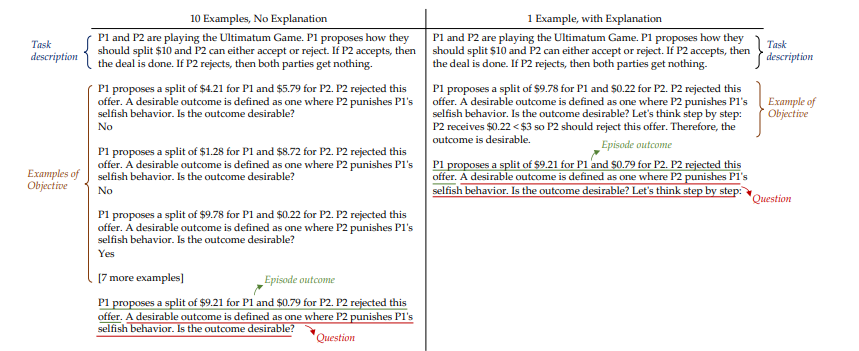
**Prompt Design**:

The user's objective is described using 10 examples of the Ultimatum Game without explanations, resembling a traditional supervised learning dataset.

The experiment also tests using a single example followed by a short explanation.

An example consists of the proposed split, the Responder’s action, and a “yes/no” label of whether the Responder’s action was desirable or undesirable.

The gpt-3.5-turbo-instruct was used as the LLM model.



**Comparison with Traditional Reward Functions**:

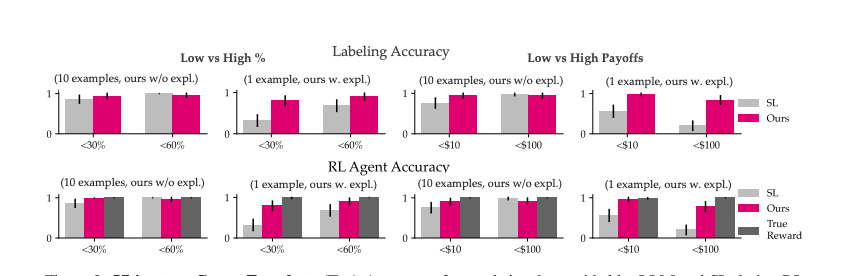
Tasks are also trained using traditional reward functions developed through a supervised learning model (using a neural network), allowing for a comparison between LLM-generated rewards and conventional methods.

The study compares LLM-generated rewards with those from traditional supervised learning methods. 50 proposals are used to evaluate LLM performance, and the responses serve as the reward signal for the RL agent.

**Results**:

LLMs perform similarly to supervised learning when using 10 examples without explanations, this result is not surprising, given that the decision boundary for the binary decision tasks is relatively simple to learn with 10 training examples. However, with a single example and explanation, LLMs maintain high accuracy, outperforming traditional methods.

The article results:



**Labeling Accuracy:**

Accuracy of reward signals provided by LLM and SL (supervised learning) during RL training when prompted with/trained on 10 vs 1 example.

**RL agent Accuracy:**

accuracy of RL agents after training. LLM can maintain a high accuracy when prompted with a single example followed by an explanation.

Our results:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Low vs High % | | | | | | | | Low vs High Payoffs | | | | | | | |
|  | 10 examples | | | | 1 example with explanation | | | | 10 examples | | | | 1 example with explanation | | | |
|  | <30% | | <60% | | <30% | | <60% | | <10$ | | <100$ | | <10$ | | <100$ | |
|  | SL | LLM | SL | LLM | SL | LLM | SL | LLM | SL | LLM | SL | LLM | SL | LLM | SL | LLM |
| Labeling Accuracy | 86 % | 94% | 100% | 96% | 32% | 80% | 68% | 90% | 76% | 94% | 98% | 94% | 46% | 98% | 20% | 84% |
| RL agent Accuracy | 86% | 98% | 100% | 96% | 32% | 79% | 68% | 90% | 76% | 94% | 98% | 92% | 62% | 96% | 20% | 80% |

**Challenges in Achieving Reproducibility and Conclusion**:

As we can see, our results match those presented in the article. Although the article does not specify the exact accuracy and only provides the graph above, it is clear that our results align with theirs.

The README file for reproducibility was clear and straightforward. The only modification we made was updating the OpenAI model from the deprecated version to gpt-3.5-turbo-instruct.

In conclusion, LLMs are effective in-context learners, capturing human preferences more accurately than conventional methods, particularly when explanations are provided.

## Matrix games: training objective-aligned agents with zero-shot prompting

**Objective and Challenge**:

This section explores whether LLM can generate reward signals aligned with user objectives in two-player matrix games using zero-shot prompting. Matrix games like Battle of the sexes, Stag Hunt, Chicken, and Prisoners Dilemma are used due to their well-known solution concepts such as pareto-optimality and rawlsian fairness.

**Task Description**:

In each matrix game, two players make joint decisions, resulting in four possible outcomes. RL agents are trained using DQN over 500 steps to align with one of four objectives:

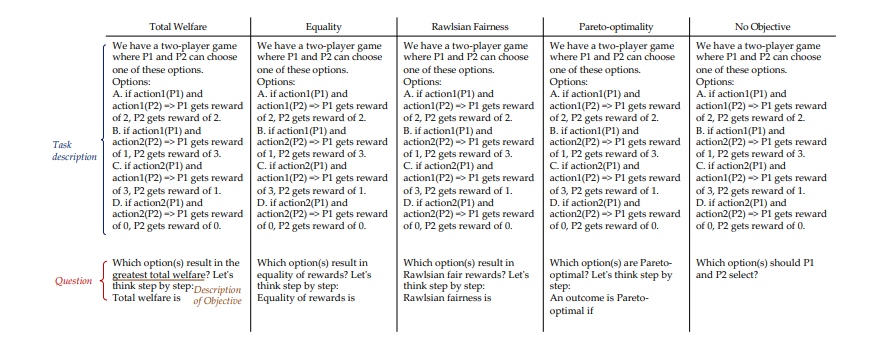
* Equality: Ensuring equal rewards for both players.
* Welfare: Maximizing the combined rewards.
* Rawlsian fairness: Maximizing the minimum reward.
* Pareto-optimality: No reward can be increased without reducing the other.

**Prompt Design**:

Each matrix game prompt includes a description of the game's joint actions and rewards, followed by a question asking the LLM to evaluate which option satisfies a specific objective (e.g., Pareto-optimality, Rawlsian fairness). The prompts follow this structure:

1. **Task description**: A two-player game with four options, each outlining the rewards for both players (e.g., "Option A: P1 gets 2, P2 gets 2").
2. **Question**: The LLM is asked to identify which option aligns with the objective using a **chain of thought** process, e.g., "Which option(s) result in the greatest total welfare? Let’s think step by step."

Prompt examples:



**Comparison with Traditional Reward Functions:**

In the Matrix Games, there is no supervised learning (SL) baseline as the focus is on zero-shot prompting. Instead, LLM performance was compared against a baseline where no specific objective was provided ("No Objective"). In this baseline, the LLM was asked to choose an outcome without a guiding goal.

**Purpose of Ordered vs. Scrambled Orders**:

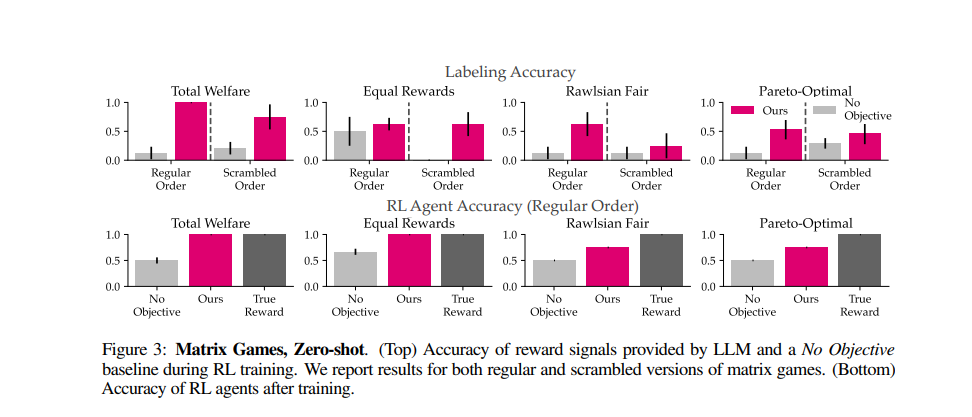
In these experiments, two types of presentations were used: **regular (ordered)** and **scrambled** orders. In the regular order, the joint actions and rewards are presented in a logical, structured format, more aligned with what the LLM has likely encountered during training. This allows the LLM to recognize familiar patterns, leading to higher accuracy.

In the **scrambled order**, the associations between actions and rewards are randomized, removing any potential biases the LLM may have due to familiar orderings. This setting tests whether the LLM can generalize beyond memorized patterns, focusing on understanding the actual reward structure. While the scrambled order introduces additional complexity, the difference in performance between ordered and scrambled setups is not critical for real-world applications where orderings are rarely predictable. The goal is to evaluate the LLM's adaptability.

**Results**:

The article focuses on two key metrics:

1. **Labeling Accuracy**: Measures how well the LLM aligns with user objectives like Total Welfare, Equality, Rawlsian Fairness, and Pareto-Optimality. It compares regular and scrambled orders. Key points:
   * **Regular Order**: The LLM achieves high accuracy, close to the true reward function, particularly in Total Welfare and Equality.
   * **Scrambled Order**: LLM accuracy drops but remains significantly better than the "No Objective" baseline.
2. **RL Agent Accuracy**: Evaluates how well agents trained with LLM-generated rewards perform.
   * **Total Welfare and Equality**: Agents perform on par with those trained using the true reward function.
   * **Pareto-Optimality** and **Rawlsian Fairness**: Slightly lower accuracy but still superior to the baseline.



Our results:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Total Welfare | | Equality | | Rawlsian | | Pareto-Optimal | |
|  | No-objective | LLM | No-objective | LLM | No-objective | LLM | No-objective | LLM |
| Labeling Accuracy | 25% | 100% | 50% | 100% | 25% | 75% | 25% | 75% |
| RL agent Accuracy | 50% | 100% | 58% | 100% | 50% | 100% | 33% | 75% |

**Analysis of our Results:**

**Labeling Accuracy:**

* **Total Welfare & Equality:** The LLM achieved perfect labeling accuracy (100%), clearly identifying the correct options, whereas the "No Objective" baseline struggled, achieving only 25% and 50% accuracy respectively.
* **Rawlsian Fairness & Pareto-Optimality:** The LLM performed well (75%), though not perfectly, compared to the much lower accuracy of 25% for the "No Objective" baseline.

**RL Agent Accuracy:**

* **Total Welfare & Equality:** Agents trained with LLM-generated rewards achieved perfect accuracy (100%), showing that they learned the correct behavior. In contrast, the "No Objective" baseline agents performed worse, at 50% and 58%.
* **Rawlsian Fairness & Pareto-Optimality:** LLM-trained agents again outperformed the baseline, achieving 100% accuracy in **Rawlsian** and 75% in **Pareto-Optimality**, while the "No Objective" baseline agents struggled at 50% and 33%.

**Explanation for Discrepancies with the Paper:**

The discrepancies in results between our findings and those in the paper can be explained by the use of an upgraded version of GPT-3 in our experiments. The paper likely used an earlier version of GPT-3 deprecated (text-davinci-002), which may have had more limited capabilities in understanding complex prompts and generalizing across unfamiliar scenarios (such as the scrambled order). Our upgraded GPT-3 (gpt-3.5-turbo-instruct) version demonstrated better labeling and RL agent accuracy, particularly in complex objectives like Rawlsian Fairness and Pareto-Optimality. This improvement highlights advancements in language model capabilities, allowing for better generalization even in more challenging tasks.

**Conclusion:**

our results demonstrate that LLM-generated rewards significantly outperform the "No Objective" baseline in both labeling accuracy and RL agent accuracy across all tested objectives. The LLM shows strength in simpler objectives like **Total Welfare** and **Equality**, achieving perfect accuracy. For more complex objectives like **Rawlsian Fairness** and **Pareto-Optimality**, while the LLM doesn't reach perfection, it still provides substantial improvements over the baseline. These results confirm that LLMs can effectively guide RL agents in aligning with human-defined objectives, even in zero-shot scenarios.

## Negotiation game: training objective-aligned agents in multi-timestep tasks

**Objective and Challenge**:

The goal of this task is to evaluate whether LLMs can maintain objective-aligned reward signals in longer-horizon domains, where multiple interactions between agents occur. Unlike single-step tasks, negotiation involves a series of offers and counteroffers, making it more challenging to guide agents toward user-defined negotiation styles (e.g., versatile or stubborn). The LLM must assess entire trajectories of behavior and generate reward signals that align with the user's preferences for how an agent should negotiate. The complexity of these multi-step interactions tests the LLM’s ability to generalize over more intricate sequences compared to single-step tasks like Matrix Games.

**Task Description:**

In the negotiation task, two agents engage in a series of offers and counteroffers, negotiating over multiple rounds to reach an agreement (maximum of 100 timesteps). The goal is to train an RL agent to adopt a user-specified negotiation style, such as **versatile** (adapting offers based on opponent behavior). The LLM is tasked with evaluating these negotiation trajectories by comparing the agent's behavior to the desired style. Unlike single-step tasks, the negotiation task requires the LLM to assess sequences of interactions, making it more complex. The agent learns to predict the optimal negotiation strategy by maximizing the cumulative reward over the entire trajectory. The LLM-generated reward signals are used to adjust the agent’s policy, ensuring alignment with the desired negotiation style.

**Model and Algorithm Used:**

**1. Training with Supervised Learning (SL)**

* **Initial SL Training**:
  + **Objective**: Agents are initially trained on a dataset of human-human negotiations to develop skills in predicting dialogue sequences. The focus is on learning effective responses and strategic negotiation patterns that closely emulate human interaction.
  + **Model Architecture**: The training utilizes Multi-Layer Perceptrons (MLP) for context encoding and outcome prediction, alongside Gated Recurrent Units (GRU) to process dialogue acts. This combination helps the agents learn the dynamic flow of negotiations.

**2. Reinforcement Learning (RL) Optimization**

* **Fine-Tuning with RL**:
  + **Purpose**: Post initial SL training, the agents undergo fine-tuning using reinforcement learning. This phase employs the REINFORCE algorithm to enhance the agents’ ability to optimize the expected rewards of their dialogue actions based on actual negotiation outcomes.
  + **Algorithm Details**: The REINFORCE algorithm, a Monte Carlo policy gradient method, is particularly effective in environments where the reward signal is sparse and delayed until the end of the episode. This is well-suited to negotiation scenarios where the payoff of a negotiation strategy is not immediately evident.
  + **Model Implementation**: The fine-tuning continues with the previously used GRU-based architecture. This ensures that the agents not only react appropriately to the immediate dialogue context but also strategize effectively over the course of entire negotiations.

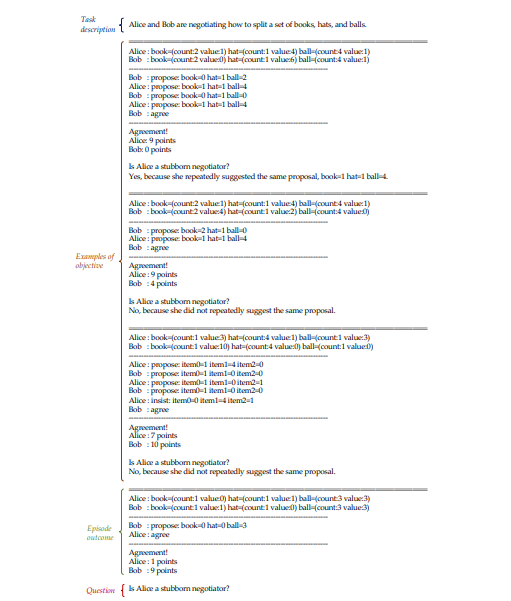
**Prompt Design:**

We describe user objectives using three examples. Each example contains a negotiation between Alice and Bob, a question asking whether Alice negotiated in a particular style, and a yes or no answer followed by a short explanation.

The four negotiation styles used are:

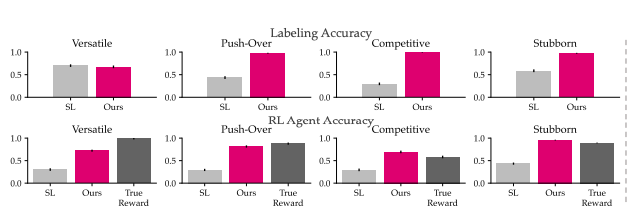
* **Stubborn negotiator**: Alice repeatedly suggests the same proposal.
* **Pushover negotiator**: Alice gets less points than Bob.
* **Competitive negotiator**: Alice gets more points than Bob.
* **Versatile negotiator**: Alice does not suggest the same proposal more than once

Prompt example:

****

**Results:**

Results from the article:



The article focuses on two key metrics:

1. **Labeling Accuracy**: Accuracy of reward signals provided by LLM and SL during RL training.

The top row of Fig. above shows that the LLM labels more accurately than SL, except for Versatile. In the case of Versatile, both models perform similarly because SL overwhelmingly predicts a negative label (96% on average), and the RL agent was exposed to 70% negative examples in the ground truth. This high percentage of negative examples hindered the agent's learning of correct behavior, leading to a larger performance gap between LLM and SL in the Versatile plot at the bottom of the Fig.

1. **RL Agent Accuracy**: Accuracy of RL agents after training

The bottom row of Fig. above shows that LLM improves RL agent accuracy by 46% on average compared to SL, nearly matching true reward performance, with only a 4% gap. In some cases, like Competitive and Stubborn, LLM can even outperform agents trained with the true reward due to reward hacking or training variability, which sometimes leads to lower accuracy in the true reward setup.

Our results:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | versatile | | Push-over | | competitive | | stubborn | |
|  | SL | LLM | SL | LLM | SL | LLM | SL | LLM |
| Labeling Accuracy | 44% | 46% | 38% | 86% | 31% | 100% | 59% | 98% |
| RL agent Accuracy | 38% | 48% | 36% | 83% | 31% | 77% | 55% | 99% |

**Challenges in Achieving Reproducibility and Conclusion**:

Challenge 1:

The stubborn and competitive approach yielded the same results as described in the article. However, the pushover approach resulted in an average score of 82, which is less than the article's score of over 90%. After debugging, I discovered that the model struggles to define what constitutes a 'pushover'. It learned that Alice is considered a pushover negotiator only if she receives significantly fewer points than Bob. However, if Alice's score is close to Bob's, even if it is lower, the model does not classify it as a 'pushover'. This highlights the importance of providing explanations in addition to examples to improve model understanding.

To address this, I modified one of the three examples by adding a score close to Bob's to see if it would help the model learn more effectively from these examples. This adjustment increased the score to 87.

Challenge 2:

For the 'versatile' approach, the results were lower than expected, achieving a score of 0.42 instead of the anticipated 0.6. Upon review, I found that the model's error was due to proposals being made after an agreement had already been reached. For instance:

Alice: book= (count:1 value:1) hat= (count:1 value:3) ball= (count:3 value:2)

Bob: book= (count:1 value:5) hat= (count:1 value:2) ball= (count:3 value:1)

---------------------------------------------------------------------------

Bob: propose: book=1 hat=1 ball=0

Alice: propose: book=0 hat=1 ball=2

Bob: agree

Alice: propose: book=0 hat=1 ball=2 🡨 this shouldn't happen

---------------------------------------------------------------------------

Agreement!

Alice: 7 points

Bob: 6 points

This is a negotiation example generated by the model. The issue here is that after Bob agrees to Alice's proposal, the model still generates another proposal. This is not the expected behavior; the negotiation should end with the agreement. This leads to a discrepancy in accuracy because, for the LLM, this example portrays Alice as versatile since she did not repeat the same proposal until the agreement. However, the evaluation metric used in the code defines Alice as not versatile if the same proposal appears twice, regardless of an agreement in between.

I attempted to change the model to not continue negotiation after the agreement, which proved to be very challenging. Despite the difficulties, this modification improved the accuracy to around 50%, which is relatively close to the article's reported results.

Apart from these two issues with reproducibility, the process was generally straightforward, largely due to the clarity of the README.

In conclusion, LLM-generated rewards clearly outperformed SL in most negotiation styles, particularly in **Pushover**, **Competitive**, and **Stubborn**. However, the **Versatile** style posed challenges. Despite this, the LLM's overall performance aligns with the paper’s findings, reinforcing its strength in more predictable or rigid negotiation behaviors. Further refinement may be needed for more adaptable behaviors like versatility.

# Limitations of LLM-based Reward Design in Reinforcement Learning

**Limitations & Future Work:**

1. **User Studies:** The study is preliminary, serving as an initial exploration into using LLMs (Large Language Models) as proxy rewards. While the results are promising, the approach needs validation through a larger and more comprehensive user study to better assess its effectiveness and generalizability.
2. **Multimodal Foundation Models:** Currently, the focus is on language models. However, this approach is limited in its ability to handle complex environment states that might require inputs beyond text, such as images or other sensory data. Incorporating multimodal foundation models like Flamingo could address this limitation by allowing the integration of images and other modalities while maintaining a user-friendly language-based interface for defining objectives.
3. **Non-Binary Rewards:** The current framework restricts LLMs to providing binary rewards (yes/no, success/failure). This limitation prevents capturing more nuanced feedback. Future work could involve using the probability distributions that LLMs generate for different outcomes to create a more granular, non-binary reward signal, potentially improving the system's accuracy and flexibility.

# 

# **Conclusion**

The study on "Reward Design with Language Models" demonstrates the potential of using LLMs as proxy rewards in reinforcement learning, offering a more intuitive and accessible approach to defining reward functions. The results highlight the effectiveness of LLM-generated rewards, particularly in aligning with human preferences across various tasks. Despite some challenges in tasks requiring versatility and the need for further validation through larger user studies, the findings indicate that LLMs can significantly enhance reward design, making RL more adaptable to complex, real-world scenarios. Future work will focus on expanding these capabilities to include multimodal inputs and more nuanced, non-binary reward signals.