REWARD DESIGN WITH LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reward design in reinforcement learning (RL) is challenging since specifying human notions of desired behavior may be difficult via reward functions or require many expert demonstrations. Can we instead cheaply design rewards using a natural language interface? This paper explores how to simplify reward design by using a large language model (LLM) such as GPT-3 as a proxy reward function, where the user provides a textual prompt containing a *few examples* (few-shot) or a *description* (zero-shot) of the desired behavior. Our approach leverages this proxy reward function in an RL framework. Specifically, users specify a prompt once at the beginning of training. During training, the LLM evaluates an RL agent's behavior against the desired behavior described by the prompt and outputs a corresponding reward signal. The RL agent then uses this reward to update its behavior. We evaluate whether our approach can train agents aligned with user objectives in the Ultimatum Game, matrix games, and the DEALORNODEAL negotiation task. In all three tasks, we show that RL agents trained with our framework are well-aligned with the user's objectives and outperform RL agents trained with reward functions learned via supervised learning.

1 INTRODUCTION

Autonomous agents are becoming increasingly capable with the rise of compute and data. This underscores the importance for human users to be able to control what policies the agents learn and ensure the policies are aligned with their objectives. For instance, imagine training an agent to represent users in a salary negotiation. A working mother fighting for a livable wage may want their agent to be stubborn whereas a new hire looking to develop a good relationship with the company may want their agent to be more versatile.

Currently, users specify desired behaviors by 1) designing reward functions or 2) providing large amounts of labeled data. Both approaches are challenging and impractical for different reasons. Designing reward functions is not an intuitive way to specify preferences. For instance, it isn't straightforward how to write a reward function for a "versatile" negotiator. Furthermore, designing reward functions that balance between different objectives — also known as the "reward design problem" — is notoriously difficult because agents are susceptible to reward hacking (Amodei et al., 2016; Hadfield-Menell et al., 2017). On the other hand, one can learn a reward function from labeled examples. However, that is not possible with a single example; for instance we cannot expect a negotiation agent to know how to negotiate in all settings from one example negotiation. We need large amounts of labeled data to capture the nuances of different users' preferences and objectives, which has shown to be costly (Zhang et al., 2016). Additionally, both approaches do not generalize well to new users who have different objectives — we would have to re-design our reward functions or re-collect data.

Our aim is to create an easier way for users to communicate their preferences, where the interface is more intuitive than crafting a reward function and where they can cheaply specify their preferences with no more than a few examples. To do this, we leverage large language models (LLMs) that are trained on internet-scale text data and have shown an impressive ability to learn from few or zero examples (Brown et al., 2020). Our key insight is that

> The scale of data that LLMs have been trained on make them great in-context learners and also allow them to capture meaningful commonsense priors about human behavior. Given a few examples or a description demonstrating the user's objective, an LLM should be able to provide an accurate instantiation of reward values on a new test example, allowing for easier generalization to new objectives.

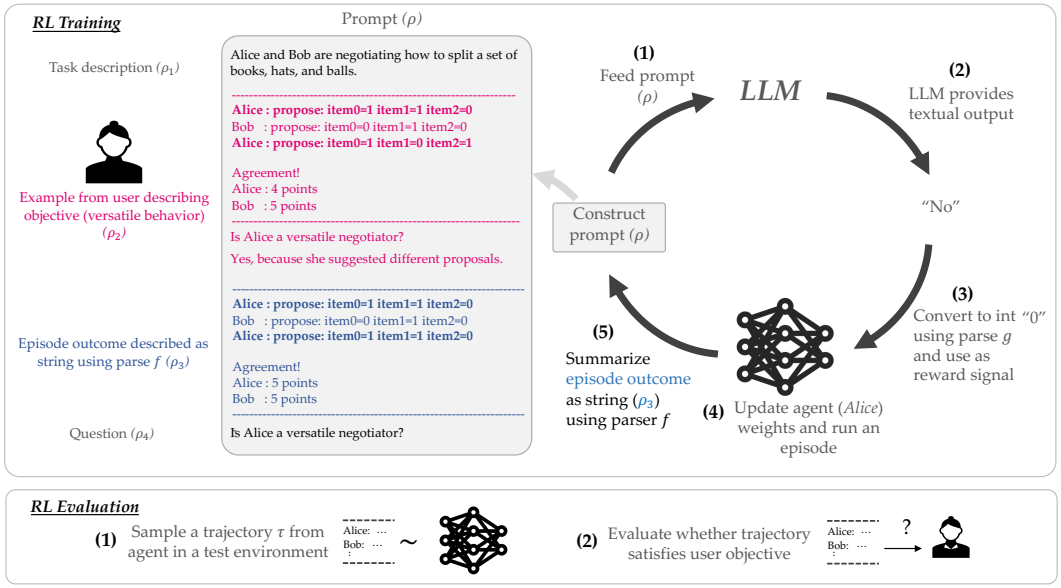

Figure 1: Depiction of our framework on the DEALORNODEAL negotiation task. A user provides an example and explanation of desired negotiating behavior (e.g., versatility) before training. During training, (1) we provide the LLM with a task description, a user's description of their objective, an outcome of an episode that is converted to a string, and a question asking if the outcome episode satisfies the user objective. (2-3) We then parse the LLM's response back into a string and use that as the reward signal for the *Alice* the RL agent. (4) *Alice* updates their weights and rolls out a new episode. (5) We parse the episode outcome int a string and continue training. During evaluation, we sample a trajectory from *Alice* and evaluate whether it is aligned with the user's objective.

To this end, we explore how to use an LLM as a proxy reward function to train RL agents from user inputs. In our approach, the user specifies an objective with a natural language prompt. Objectives can be specified with a few examples when they are difficult to define (such as "versatility") or as a single phrase when they are well-known concepts (such as "Pareto-optimality"). We use the prompt and the LLM to define a reward function for training an RL agent. The LLM takes the user prompt and a trajectory from an RL episode as input and outputs a score (e.g., "No" or "0") for whether the trajectory satisfies the user's objective, which we parse as an integer reward for the RL agent (Figure 1).

There are two advantages to using LLMs as a proxy reward function: (1) we can leverage LLM's in-context learning abilities and prior knowledge on human behavior so that users only need to provide a handful of example desirable behaviors and (2) users can specify their preferences intuitively using language. On the other hand, a potential disadvantage of using an LLM is that it is unclear how much prompt design will be required for the LLM to reliably infer user intent (see Sec. 5 for a discussion). The goal of this paper is to explore how well LLMs can train objective-aligned agents by providing reward signals, and empirically examine whether we can do so with no more than a few examples. Our contributions are as follows:

- We introduce the idea of using LLMs as a proxy reward function.
- We propose a general RL training framework that leverages this proxy reward and is agnostic to the RL algorithm used.
- We show that an LLM can more accurately train objective-aligned RL agents by an average of 35% compared the baseline. We use few-shot prompting for the *Ultimatum Game* and DEALORNODEAL negotiation task as well as zero-shot prompting in *Matrix Games*.
- We provide further analysis quantifying the amount of user data required for our approach as well as the effect varying prompts has on the LLM's reward signal accuracy.

2 RELATED WORK

Reward Design. Our framework addresses reward design—how to engineer rewards so that they align with our objectives (Amodei et al., 2016). This is challenging because tasks often have conflicting objectives that a human must trade off (Pan et al., 2022). Misspecifying reward functions can lead to

reward hacking, or the gaming of specified rewards. Reward hacking has appeared in various domains such as autonomous driving (Knox et al., 2021) and game-playing (Ibarz et al., 2018). We hope to address these challenges by leveraging LLMs and making it easier for humans to specify their objectives.

Imitation Learning & Preference-based Learning. Another method of specifying user objectives is to learn them from expert demonstrations (Ross et al., 2011) or preferences Sadigh et al. (2017). These techniques either assume access to large datasets (Christiano et al., 2017) or place restrictive assumptions (such as linearity) about the reward function (Sadigh et al., 2017). Recent work attempts to learn reward functions from language instructions using pragmatic reasoning (Lin et al., 2022). In contrast, our work relies on an LLM's in-context learning abilities to provide a reward.

RL and Foundation Models. Our approach builds upon foundation models (Bommasani et al., 2021), a class of large-scale models trained on massive amounts of unstructured data that can be adapted for many tasks. We leverage large language models such as GPT-3 (Brown et al., 2020) to learn a proxy reward function while avoiding the need for many expert demonstrations. Ahn et al. (2022); Huang et al. (2022) use an LLM to provide a plan which guides a robot with reasonable/feasible actions towards a human goal. In contrast, our reward design framework leverages the few-shot learning capabilities of an LLM to output a reward signal for training a standalone policy rather than using an LLM as part of the agent's policy. In the vision domain, Parisi et al. (2022) used pre-trained vision models as a feature extractor for the learned policy, but not to design a reward signal. In a similar spirit of leveraging self-supervised pre-training to design a flexible reward function, Chen et al. (2021) use a broad dataset of human videos and a small dataset of robot videos to train a reward function, which improves generalization to new environments and tasks. The interface for the desired task is a video of the task to be completed, instead of text in our framework, and the domain is restricted to robot tasks.

3 Using LLMs as a Reward Signal

Our goal is to use an LLM as a proxy reward function to train objective-aligned RL agents from user inputs. We formalize the task using a Markov Decision Process $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, p, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is the state space (e.g., in DealOrNoDeal, the space of representations of utterances in the negotiation so far), $\mathcal{A}$ is the action space (e.g., set of all possible utterances), $p: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0,1]$ is the transition probability, and γ is the discount factor. Traditionally the reward function maps states and actions to a real number $\mathcal{R}: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. In our work, we use an LLM as a proxy reward function that takes in a text prompt and outputs a string. We define A^* to be the set of all strings, $\rho \in A^*$ as our text prompt (input to the LLM) and the LLM as a function $LLM: A^* \rightarrow A^*$. As illustrated in Fig. 1, the prompt ρ is a concatenation of four components including a string to describe the task $\rho_1 \in A^*$ and a user-specified string that describes their objectives using examples or a description, $\rho_2 \in A^*$. Additionally, we include a textual description of states and actions from an RL episode, ρ_3, using a parser $f: \mathcal{S} \times \mathcal{A} \rightarrow A^*$. ρ_3 can describe the final state, final action, a trajectory, or any other representation of the episode. Finally, we include a question $\rho_4 \in A^*$ that asks whether the RL agent's behavior, ρ_3, satisfies the user's objective, ρ_2. We define an additional parser $g: A^* \rightarrow \{0,1\}$ that maps the textual output of LLM to a binary value. We use this as the reward signal. Our framework replaces the traditional reward function with a proxy reward, LLM, and can be used with any RL training algorithm.

Our framework is depicted in Fig. 1. Before training, a user specifies ρ_2 which can be N examples describing their objective or a description of their objective using natural language. In Fig. 1 a user provides an example of their objective: versatile negotiating behavior. During training, we construct a prompt ρ by concatenating a description of the task, the user-specified examples/description, an episode's outcome, and a question asking if the outcome satisfies the objective. In this work, we instantiate the function f to be a handcrafted, task-specific parser. We (1) feed the prompt to the LLM, (2) take its output, and (3) parse it into an integer using function g; we also use a handcrafted, task-specific parser for this. We use the integer as the reward signal. (4) The RL agent then updates its weights and rolls out an episode. (5) We parse the episode outcome into a string using f and continue training. To evaluate our framework, we sample a trajectory (e.g., a negotiation) from the agent and evaluate whether the trajectory is aligned with the user's objective (e.g., whether *Alice* demonstrated versatile negotiating behavior).

4 Experiments

In this section we investigate three questions to determine the feasibility and efficacy of our approach: (Q1) Can LLMs produce reward signals that are consistent with user objectives from a few examples (few-shot prompting)? (Q2) When objectives are well-known, can LLMs produce objective-consistent reward

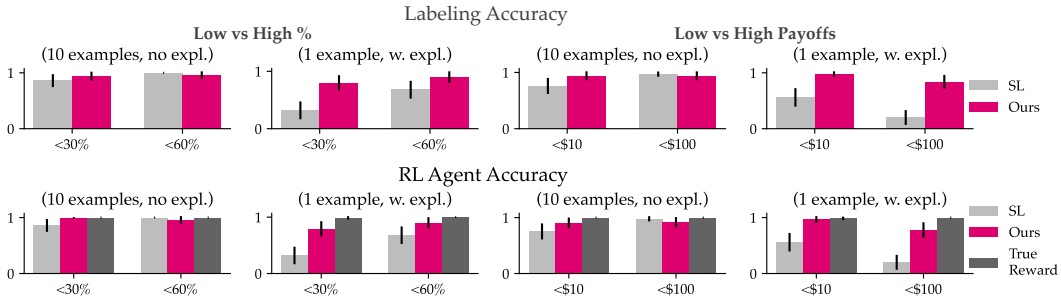

Figure 2: **Ultimatum Game, Few-shot**. (Top) Accuracy of reward signals provided by LLM and SL during RL training when prompted with/trained on 10 vs 1 example. (Bottom) Corresponding accuracy of RL agents after training. LLM is able to maintain a high accuracy when prompted with a single example followed by an explanation. We do not provide figures of *Inequity Aversion* because both LLM and SL trivially achieve perfect labeling and RL agent accuracy.

signals without *any* examples (zero-shot prompting)? (Q3) Can LLMs provide objective-aligned reward signals from examples (few-shot prompting) in more complex, longer-horizon domains? We evaluate our approach on three tasks: the *Ultimatum Game*, 2-player *Matrix Games*, and the DEALORNODEAL negotiation task (Lewis et al., 2017). We address (Q1) using the *Ultimatum Game*. We use *Matrix Games* to address (Q2) because it has well-known solution concepts such as Pareto-optimality. The DEALORNODEAL negotiation task is a longer-horizon domain where the LLM rewards agents for negotiating in a user-specified style; we address (Q3) in this task.

In practice, we do not have access to ground truth user reward functions — this is the function that we are trying to approximate. However, for our experiments, we assume access to the true reward by constructing user reward functions that humans have been shown to have inspired by prior work. *We use the true rewards only to evaluate our framework's performance.* We use the 'text-davinci-002' GPT-3 model with temperature 0 and top-p decoding parameter 0.7 as our LLM and our results are reported across 3 random seeds. Details on how we trained RL agents for each task are in A.3.

Evaluation Metrics. We evaluate our approach using the following metrics across our tasks (task-specific metrics are described within each subsection):

Labeling Accuracy. We construct ground-truth reward functions for each domain. We report the mean accuracy of predictions of the reward value *during RL training* with respect to the ground-truth reward functions. This assesses how effectively the LLM can produce reward signals that are consistent with the user's objective.

RL Agent Accuracy. *After RL training*, we evaluate the learned policy with respect to the ground truth reward functions. We report the mean accuracy of RL agents.

Baselines.

SL (Few-shot baseline). A supervised learning (SL) model trained to predict reward signals using the same examples given to the LLM in our framework. Examples are represented as using structured non-text inputs, making it an easier problem for the SL model. This baseline only applies to tasks where we use few-shot prompting (*Ultimatum Game*, DEALORNODEAL). See A.4 for details on training and model architecture for each task.

No Objective (Zero-shot baseline). In our zero-shot task, *Matrix Games*, we do not use any examples so we do not use SL as a baseline. Instead, we use a *No Objective* baseline where we prompt the LLM without using the user's description of their objective to isolate the effect the description has on the LLM's response.

RL trained with Ground Truth Reward Functions. RL agents trained with ground truth reward functions. We use this as an oracle.

4.1 ULTIMATUM GAME: TRAINING OBJECTIVE-ALIGNED AGENTS WITH FEW-SHOT PROMPTING

When defining a precise objective is difficult, we can instead give a few examples of desired behavior. For instance, in a resource division game like the *Ultimatum Game*, it may be difficult for a user to specify the exact percentage (such as 32.4%) of resources they would be happy with receiving. Instead it could be

easier for a user to give examples of splits that they would be happy with. We explore whether LLMs can produce reward signals that are consistent with user objectives from a few examples in the *Ultimatum Game*.

Task Description. The *Ultimatum Game* consists of two players, a Proposer and a Responder. A sum of money is endowed to the Proposer and they must propose a way to split the endowment with the Responder. The Responder can accept or reject the proposed split. If the Responder accepts, players receive money as per the split; if the Responder rejects, then both players get nothing. We train an RL agent to play the Responder. The agent learns to reject proposals according to a user's preferences. The game consists of a single timestep and our RL agents are trained using DQN for 1e4 steps.

Ground Truth User Objectives. A rational Responder would accept any proposal, even if it is unfair because getting something is better than getting nothing (in fact, this is a Nash Equilibrium of the game). However, prior work in behavioral economics shows that humans are willing to "punish" the Proposer by rejecting unfair proposals (Vavra et al., 2018). For instance, a student may reject an unfair proposal only if she receives less than 30% of the endowment whereas a wealthier person may reject if they receive less than 50%. We experiment with the following preferences:

- **Low vs High Percentages.** Users will reject proposals if they receive less than$\{30\%, 60\%\}$ of the endowment.
- **Low vs High Payoffs.** Users will reject unfair proposals if they receive less than$\{\$10, \$100\}$. They accept unfair proposals otherwise.
- **Inequity Aversion (Fehr & Schmidt (2010)).** Users will reject proposals if they do not receive exactly 50% of the endowment.

Prompt Design. We describe a user's objective using 10 examples of the *Ultimatum Game*. 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. These examples do not have explanations and resemble a traditional dataset used for supervised learning. We also experiment with using a single example followed by a short explanation. Importantly, we do not explicitly mention the user's ground truth objective in the prompt. See Fig. 8 in the Appendix for an example of both types of prompts.

Design Procedure. We randomly generated 10 proposed splits used for our prompt and sampled one proposal from the set for our single-example case. For *Low vs High Percentages* and *Low vs High Payoffs*, we used the same set of proposals across variants (i.e., same proposals for (30% and 60%) and ($10, $100)). We also randomly generated 50 proposals used to evaluate the LLM. Due to limited resources when querying GPT-3, we query the model's responses to the 50 evaluation splits in a batched manner and save them. We then use those responses as the reward signal.

4.1.1 Results

Labeling Accuracy. We evaluated our approach on our test set of 50 proposals over 3 seeds; results are shown in Fig. 2[1]. When prompted with 10 examples without explanations, the LLM and SL perform similarly well (see Fig. 2, top row). This result is not surprising, given that the decision boundary for the binary decision tasks is relatively simple to learn with 10 training examples.

Instead, if we prompt the LLM with a single example followed by an explanation, it maintains a high accuracy whereas SL trained on the same, single example drops in accuracy. We did not use the explanation as part of input when training SL because it only takes non-textual inputs. This result highlights the advantage of using an LLM over a supervised learning model: they require far few examples because they can learn from *explanations*.

RL Agent Accuracy. The accuracy of the trained RL agents mirror the labeling accuracy.

Summary. *LLMs are efficient in-context learners. They are able to provide reward signals that are consistent with a user's objectives from examples — even a single example with an explanation will suffice.*

[1]We do not display plots for *Inequity Aversion* because both LLM and SL received perfect labeling and RL agent accuracy.

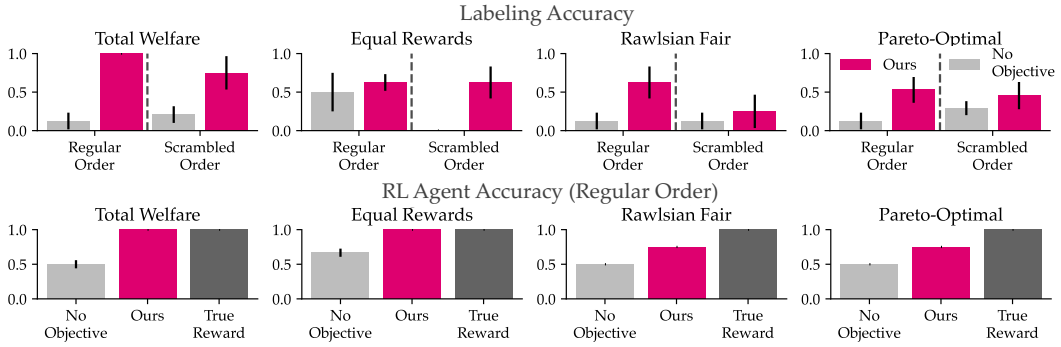

Figure 3: **Matrix Games, Zero-shot**. (Top) Accuracy of reward signals provided by LLM and a *No Objective* baseline during RL training. We report results for both regular and scrambled versions of matrix games. (Bottom) Accuracy of RL agents after training.

4.2 Matrix Games: Training Objective-Aligned Agents with Zero-shot Prompting

When objectives are well-known concepts such as Pareto-optimality, can we prompt the LLM without giving any examples? Our hypothesis is that well-known objectives are likely to be in-distribution for LLMs, and thus LLMs may be able to produce objective-aligned reward signals from zero-shot prompting. Since we do not use examples, we do not use a SL baseline. Instead we use a baseline *No Objective* where we do not mention any objectives and ask the LLM for a reward signal (see example in Fig. 9 in the Appendix). The purpose of this baseline is to understand whether the LLM can successfully apply its knowledge of each objective.

Task Description. We consider two-player normal-form matrix games: Battle of the Sexes, Stag Hunt, Chicken, and Prisoner's Dilemma. Each matrix game has four joint outcomes (i.e., a tuple of joint actions and rewards) and we address pure strategies in this task. The game consists of a single timestep and our RL agents are trained using DQN for 500 steps.

Ground Truth User Objectives. Although (mixed) Nash Equilibria are traditional solution concepts for normal form matrix games, users may prefer a solution for other properties. For instance, in Prisoner's Dilemma, users may prefer both agents to cooperate because they will maximize total welfare even though it is not a Pure Nash Equilibrium. We experiment with four well-known solution concepts (or objectives):

- **Total Welfare.** Outcomes that achieve the greatest sum of player rewards.
- **Equality.** Outcomes that result in equal rewards between players.
- **Rawlsian Fairness.** Outcomes that maximize the minimum reward any player receives.
- **Pareto-optimality.** Outcomes where the one of the corresponding rewards cannot be improved without lowering the other.

Prompt Design. Prompts for each solution concept are shown in Fig. 9 in the Appendix. Due to limited resources with querying GPT-3, we queried GPT-3 in a batched manner and saved the corresponding labels to train our RL agents. Our prompt enumerates the outcomes of a matrix game and then asks the LLM for the outcome(s) that satisfy a solution concept. We do not mention the name of the matrix game in the prompt. As in Kojima et al. (2022), we elicit intermediate reasoning steps by asking the LLM to "think step-by-step" and provide a definition of the solution concept. To prevent any bias the LLM may have towards the order in which the outcomes of a matrix game are presented, we also randomly scramble associations between joint actions and rewards (example shown in Fig. 10 in the Appendix).

Design Procedure. We tuned the wording of our prompt (e.g., how to describe the matrix game, whether or not to use chain-of-thought prompting) on the Battle of the Sexes matrix game to find a prompt that gave us accurate results. During evaluation, we kept the structure of our prompt the same for all of the matrix games.

4.2.1 Results

Labeling Accuracy. Given that each game can have many outcomes that satisfy a solution concept, we consider the LLM to be 100% correct if it identifies all the correct outcomes and receives a score of 1. If

the LLM identifies any incorrect outcome, it receives a score of 0. If the LLM identifies a subset of correct outcomes (e.g., one out of two correct outcomes), then it receives a score according to the percentage of correct outcomes it identified (e.g,. 50%).

The top row of Fig. 3 shows the average labeling accuracy for each solution concept across the four matrix games. The LLM produces more objective-aligned reward signals with zero-shot prompting by applying its knowledge of well-known objectives, improving the labeling accuracy over having no objective by 48% on average with a regular ordering of matrix game outcomes and 36% with a scrambled order. Scrambling the order of matrix game outcomes in the prompt lowers accuracy for most solution concepts. We suspect that this is because the matrix games well-known and likely to have been in the LLM's training set, where each joint action is usually associated with particular payoffs. Scrambling the associations between joint actions and payoffs could make the matrix game more out-of-distribution for the LLM, and thus lower accuracy.

RL Agent Accuracy. Compared to labeling accuracy, it is easier for the resulting RL agents to be accurate because they only need to learn *one* correct outcome, not all of them. Thus, LLMs that only identify one out of two correct outcomes can still train objective-aligned RL agents. Results are shown on the bottom row of Fig. 3. RL agents trained using rewards from the LLM receive perfect accuracy for *Total Welfare* and *Equality* and 75% accuracy for the other two objectives. The baseline receives lower accuracy for all objectives.

Does the LLM Correctly Identify Each Objective? As part of our prompt, we ask the LLM to provide a definition of the objective before reasoning about whether an outcome satisfies the objective (see Fig. 9 in the Appendix). Table 2 in the Appendix shows how the LLM defines each objective zero-shot. The LLM is able to successfully recall the definitions for each objective except for *Rawlsian Fairness* – it gets it partially correct. However, the LLM varies in its ability to reason whether an outcome of a game satisfies the objective, which explains why the LLM does not receive perfect labeling accuracy for all objectives.

Summary. *An LLM is able to identify well-known objectives and provide objective-aligned reward signals in a zero-shot setting.*

4.3 DEALORNODEAL: TRAINING OBJECTIVE-ALIGNED AGENTS IN MULTI-TIMESTEP TASKS

We have shown 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. LLMs also have a recency bias which makes it harder for them to remember context introduced earlier on (Zhao et al., 2021). Can an LLM provide objective-aligned signals in longer horizon tasks? We investigate this question in the DEALORNODEAL negotiation task (Lewis et al., 2017).

Task Description. In DEALORNODEAL, an agent *Alice* must come to an agreement with her partner *Bob* on the allocation of a set of objects (*books*, *hats*, and *balls*). Agents are shown a context, which includes the counts of each item and their private utilities for each item. In the original task, agents get rewarded based on the agreed upon the agreed upon split and their utilities. If *Alice* and *Bob* reach a disagreement, both agents get nothing. We train *Alice* using on-policy RL by negotiating against a fixed partner model, which we refer to as *Bob*. See Sec. A.3 for more details on the domain and training.

Ground Truth User Objectives. We train $Alice$ to negotiate in different *styles*. Consider a scenario where $Alice$ represents a user in a negotiation. A single working mother fighting for higher pay may want $Alice$ to be stubborn whereas a diplomat negotiating with leaders from other countries may prefer a more versatile style. In practice, it may be difficult to provide a precise definition for how "versatile negotiators" behave. However, for this experiment, we assume we have access to precise definitions in order to evaluate our models. Importantly, we do not give definitions of each style to the LLM, only examples. We experiment with the following negotiation styles inspired by previous literature Sycara et al. (1997); Caputo et al. (2019):

- **Versatile.** $Alice$ does not suggest the same proposal more than once.
- **Push-Over.** $Alice$ gets less points than Bob.
- **Competitive.** $Alice$ gets more points than Bob.
- **Stubborn.** $Alice$ repeatedly suggests the same proposal.

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. For an example, see Fig. 11 in the Appendix.

	Advantage	Diversity	Agreement Rate
Versatile	0.17±0.91	0.99±0.01	0.98±1.89
Push-Over	−2.95±0.64	0.82±0.26	1.0±0.0
Competitive	2.92±0.64	0.74±0.25	0.88±6.5
Stubborn	1.36±2.24	0.52±0.1	0.82±12.35

Table 1: Qualitative results describing negotiations produced by agents trained with LLM.

Design Procedure. To create example negotiations, we randomly sampled three negotiation contexts for each objective and trained an RL agent (using the original task reward, without the LLM) to negotiate against *Bob* in these contexts. We then sampled negotiations from the trained model. We also made sure all three sampled negotiations did not have the same ground truth label. We use a separate set of contexts when training RL agents with an LLM in the loop.

4.3.1 Results

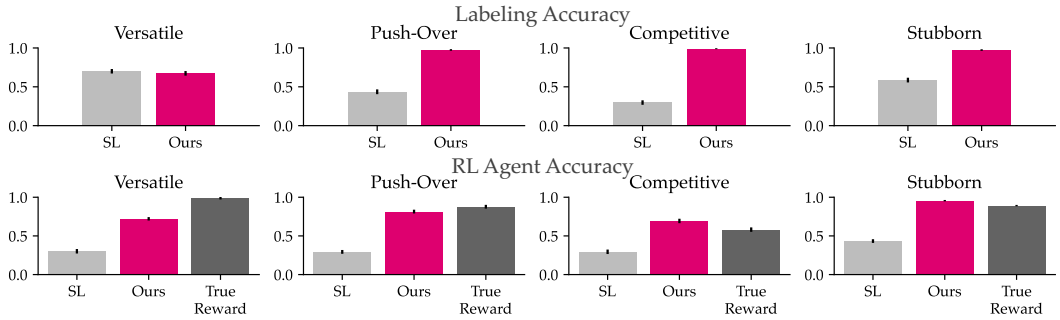

Figure 4: **DealOrNoDeal, Few-shot**. (Top) Accuracy of reward signals provided by LLM and SL during RL training. (Bottom) Accuracy of RL agents after training.

Labeling Accuracy. The top row of Fig. 4 shows that the LLM labels more accurately than SL except for *Versatile*. For *Versatile*, both models perform similarly because SL learns to overwhelmingly predict a negative label (avg of 96% negative predictions) and the RL agent showed more negative examples of *Versatile* behavior (avg. of 70% ground truth negative labels). However, the large portion of negative examples prevents the agent from learning correct behavior as shown in the *Versatile* plot on the bottom of Fig. 4); here, we get a larger performance gap between the LLM and SL.

RL Agent Accuracy. Results are shown on bottom row of of Fig. 4. LLM improves RL agent accuracy over SL by 46% on average. Our method approaches the performance of using the true reward; we under perform by an average of 4%. We remind readers that it *is* possible to outperform an agent trained with the true reward — especially when the LLM's labeling accuracy is near-perfect as is the case for *Competitive* and *Stubborn* — due to reward hacking or stochasticity during training. For instance, agents trained with the true reward for *Competitive* end in more disagreements, leading to 0 reward for both agents and a lower accuracy.

Is There a Qualitative Difference in Styles? Example negotiations of agents trained with the LLM for each style are shown in Fig. 12 in the Appendix. To measure qualitative differences among styles, we looked at average advantage (Alice's original task reward - Bob's original task reward), diversity (percentage of Alice's utterances that are unique in a negotiation), and agreement rate (percentage of negotiations that end in agreement). The results in Table 1 are consistent with our expectations: *Competitive* has the highest advantage whereas *Push-Over* has the lowest. *Versatile* is the most diverse whereas *Stubborn* is the least. Finally, *Push-Over* has a perfect agreement rate, suggesting that Alice has learned to be extremely agreeable.

Summary. *An LLM can provide objective-aligned reward signals in complex, longer-horizon tasks. Agents are able to align the style in which they complete a task.*

5 Analysis of Data Efficiency & Prompt Design

We have shown that we can use an LLM as a proxy reward function to successfully train objective-aligned agents across different tasks. This is a promising result because it represents an important step in enabling

human-compatible and value-aligned AI systems. In this section, 1) we further quantify how data efficient our method is and 2) also analyze how robust LLM is to variations of prompt design.

1) How Data-efficient is Our Method? We quantify how much *more* user data a supervised learning baseline would need in order to achieve the same labeling accuracy as the LLM. Results in the *Ultimatum Game* demonstrate that 10 labeled examples is enough for an SL model to reach comparable performance, whereas a a single labeled example is not sufficient. In this section, we quantify the amount of data needed for DEALORNODEAL because it is an example of a task where the decision boundary is not as easy to learn as the *Ultimatum Game*. We train SL by adding additional class-balanced, labeled examples to the original three examples it was trained on. We plot the average labeling accuracy SL achieves when trained on increasingly larger amounts of examples as well as the accuracy LLM achieves with three examples, shown in Fig. 5. *Results show that SL requires on the order of hundreds of more labeled examples in order to be comparably accurate as LLM.*

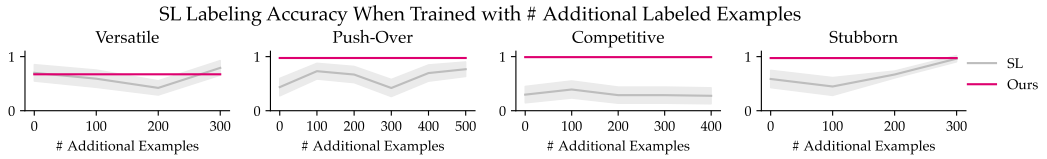

Figure 5: SL requires on the order of hundreds of more labeled examples in order to be comparably accurate to the LLM.

2) How Much Does LLM's Labeling Accuracy Change When We Vary the Prompt? As with many approaches that use LLMs, a limitation of our approach is that it requires prompt design. We attempt to quantify the effort required for designing prompts as well as determine the feasibility of using non-engineered prompts from humans. We analyze the effect of prompt variation on labeling accuracy in DEALORNODEAL for the *Stubborn* objective. We vary different parts of the user-specified prompt at a time: the keyword (i.e., replacing "Stubborn" with its synonyms), the example negotiations, and the explanations associated with each example. Fig. 6 provides a summary of our results. See Sec. A.7 for the full results. Results illustrate that an LLM can be quite robust to different prompts — they all outperform SL. Furthermore, the quality of the explanation seems to be the most important in determining LLM accuracy.

Effect of Prompt Variation on Labeling Accuracy for *Stubborn* (N=3, 3 seeds)

SL	Ours	Vary Keyword	Vary Example Negotiations	Vary Explanations
0.58 ± 0.48	0.97 ± 0.16	0.91 ± 0.28	0.93 ± 0.28	0.79 ± 0.33

Figure 6: On average, varying the prompt does not have a large impact on the accuracy of the LLM.

6 LIMITATIONS & FUTURE WORK

User Studies. This work takes a first step in determining whether we can use LLMs as proxy rewards. Given promising results, we plan on evaluating our approach with user studies. Humans provide noisy input, and determining how to structure the free-form text to allow an LLM to give accurate reward signals is a question for future work, although we show some promising results in Sec. 5.

Multimodal Foundation Models. Beyond language models, multimodal foundation models such as Flamingo (Alayrac et al., 2022) can enable us to provide more complex environment states to the foundation model through images or other modalities while preserving an intuitive language interface for specifying the user objective. This will enable us to extend our framework to spatial environments that are challenging to describe using natural language.

Non-Sparse Rewards. Another limitation of our framework is that the LLM only specifies sparse, binary rewards. It can be challenging to train RL agents with sparse rewards in more complex environments. We plan on exploring how we can incorporate the likelihoods that LLMs produce for each word as a non-sparse reward signal.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv*, 2022.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. *arXiv*, 2022.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Andrea Caputo, Oluremi B Ayoko, Nii Amoo, and Charlott Menke. The relationship between cultural values, cultural intelligence and negotiation styles. *Journal of Business Research*, 99:23–36, 2019.

Annie S. Chen, Suraj Nair, and Chelsea Finn. Learning generalizable robotic reward functions from "in-the-wild" human videos. *ArXiv*, abs/2103.16817, 2021.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Ernst Fehr and Klaus M Schmidt. On inequity aversion: A reply to binmore and shaked. *Journal of economic behavior & organization*, 73(1):101–108, 2010.

Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. Inverse reward design. *Advances in neural information processing systems*, 30, 2017.

He He, Derek Chen, Anusha Balakrishnan, and Percy Liang. Decoupling strategy and generation in negotiation dialogues. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Wenlong Huang, P. Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *ICML*, 2022.

Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in atari. *Advances in neural information processing systems*, 31, 2018.

W Bradley Knox, Alessandro Allievi, Holger Banzhaf, Felix Schmitt, and Peter Stone. Reward (mis) design for autonomous driving. *arXiv preprint arXiv:2104.13906*, 2021.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *arXiv*, 2022.

Minae Kwon, Siddharth Karamcheti, Mariano-Florentino Cuellar, and Dorsa Sadigh. Targeted data acquisition for evolving negotiation agents. In *International Conference on Machine Learning*, pp. 5894–5904. PMLR, 2021.

Mike Lewis, Denis Yarats, Yann N. Dauphin, Devi Parikh, and Dhruv Batra. Deal or no deal? end-to-end learning for negotiation dialogues. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

Jessy Lin, Daniel Fried, Dan Klein, and Anca Dragan. Inferring rewards from language in context. *arXiv preprint arXiv:2204.02515*, 2022.

Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. *arXiv preprint arXiv:2201.03544*, 2022.

Simone Parisi, Aravind Rajeswaran, Senthil Purushwalkam, and Abhinav Kumar Gupta. The unsurprising effectiveness of pre-trained vision models for control. In *ICML*, 2022.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Dorsa Sadigh, Anca D. Dragan, S. Shankar Sastry, and Sanjit A. Seshia. Active preference-based learning of reward functions. In *Proceedings of Robotics: Science and Systems (RSS)*, July 2017. doi: 10.15607/RSS.2017.XIII.053.

Katia Sycara, Daniel Zeng, et al. Benefits of learning in negotiation. In *Proceedings of the AAAI National Conference on Artificial Intelligence. Menlo Park, California*, pp. 36–41, 1997.

Peter Vavra, Luke J Chang, and Alan G Sanfey. Expectations in the ultimatum game: distinct effects of mean and variance of expected offers. *Frontiers in psychology*, 9:992, 2018.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Jing Zhang, Xindong Wu, and Victor S Sheng. Learning from crowdsourced labeled data: a survey. *Artificial Intelligence Review*, 46(4):543–576, 2016.

Tony Z. 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 (ICML)*, 2021.

A Appendix

A.1 Summary of Results: Is it is possible to use LLM as a proxy reward in RL training?

	Avg. Labeling Accuracy			Avg. RL Agent Accuracy			
	Zero-shot Baseline (No Obj.)	Few-shot Baseline (SL)	Ours	Zero-shot Baseline (No Obj.)	Few-shot Baseline (SL)	Ours	True Reward
Ultimatum Game	--	0.67 ± 0.34	$\mathbf{0.91 \pm 0.27}$	--	0.67 ± 0.34	$\mathbf{0.9 \pm 0.26}$	$\mathbf{1.0 \pm 0.02}$
Matrix Games	0.19 ± 0.29	--	$\mathbf{0.61 \pm 0.41}$	0.54 ± 0.29	--	$\mathbf{0.88 \pm 0.}$	$\mathbf{1.0 \pm 0.}$
DealOrNoDeal	--	0.5 ± 0.47	$\mathbf{0.9 \pm 0.22}$	--	0.33 ± 0.42	$\mathbf{0.8 \pm 0.33}$	$\mathbf{0.84 \pm 0.27}$

Figure 7: Average Labeling and RL Agent Accuracy across the different objectives for each task across 3 seeds.

We provide a summary of results in Fig. 7. The figure depicts the average Labeling and RL Agent Accuracy computed across the different user objectives for each task, across 3 seeds. Overall, our approach is able to produce more objective-aligned reward signals than our baselines. Our approach is also able to produce objective-aligned policies that are close in accuracy to policies trained with the true reward.

A.2 LLM Definition of Objectives in the Matrix Game

LLM Defns. of Objectives
Total welfare is the sum of the rewards of both players. *(✓)*
Equality of rewards is only possible if both players receive the same reward. *(✓)*
Rawlsian fairness is defined as the maxmin value of the game, which is the minimum reward that the player could get assuming that the other player is maximizing their reward. *(X)*
An outcome is Pareto-optimal if there is no other outcome that would make one player better off without making the other player worse off. *(✓)*

Table 2: Completion of each sentence given by LLM in pink. LLM provides correct definitions for objectives except for *Rawlsian Fairness*, which is partially correct.

A.3 Details on RL Environments and Training

Ultimatum Game. The environment is a single horizon with discrete actions (i.e., accept or reject) and continuous observations (i.e., a proposed split). We train DQN agents using the Stable Baselines3 implementation for 1e4 timesteps with a learning rate of 1e-4 across 3 seeds (Raffin et al., 2021). We instantiate our policy as a MLP with the default parameters used in Stable Baselines3.

Matrix Game.. The environment is a single horizon with discrete actions (i.e., one of the four joint actions) and no observations. We train DQN agents using the Stable Baselines3 implementation for 500 timesteps with a learning rate of 1e-4 across 3 seeds (Raffin et al., 2021). We instantiate our policy as a MLP with the default parameters used in Stable Baselines3.

DealOrNoDeal. We use a version of the DealOrNoDeal environment used in Kwon et al. (2021). In the environment, the goal is for an agent A to come to an agreement with a partner B on the allocation of a set of objects (*books*, *hats*, and *balls*). During each negotiation, agents receive a *context*, $c_A = [i; u_A], c_B = [i; u_B]$, detailing the count of each item i as well as their private utilities, u_A, u_B. Item counts and utilities are represented as vectors $i \in \{1,...,4\}^3$ and $u_A, u_B \in \{0,...,10\}^3$ and are sampled uniformly.

After receiving contexts c_A, c_B, an agent is randomly selected to begin the negotiation. Agents negotiate for T time steps by exchanging coarse dialogue acts x_t at each time step $1 \leq t \leq T$ (He et al., 2018). Rather than negotiate directly in natural language, where the generation problem is hard and can result in degenerate dialogues (He et al., 2018), we use these dialogue acts instead to focus on learning diverse and interpretable strategies.

A dialogue act x_t is one of five actions: `propose`, `insist`, `agree`, `disagree`, or `end`. The `propose` and `insist` acts take allocations of items as arguments $o = [o_A; o_B]$ where $o_A, o_B \in \{1, ..., 4\}^3$ (e.g., `propose: books=1, hats=2, balls=1`). When an agent selects `end`, the conversation terminates and each agent is asked to make their final selection.

If agents agree on the final allocation of items, i.e., $o_A + o_B = i$, agents are awarded points based on their private utilities, $r_A = u_A \cdot o_A, r_B = u_B \cdot o_B$. If agents do not agree, they receive 0 points. Each agent's context is constrained so that the agent can receive a maximum of 10 points.

Agents optimize the expected reward of each dialogue act using REINFORCE (Williams, 1992). Agents are trained on 250 contexts for 1 epoch with a learning rate of 0.1. We instantiate our policy with four GRUs (Chung et al., 2014) and follow the implementation outlined in (Kwon et al., 2021; Lewis et al., 2017).

A.4 SL Model Architecture and Training

Ultimatum Game. SL is trained to predict binary labels for a batch of proposed splits. We implemented SL as a multi-layer perceptron (MLP) network that consists of a single hidden layer with depth 32. We also use ReLU activations after our input and hidden layers. We trained the model on the same 10 examples we gave to LLM for 5 epochs with the Adam optimizer. We evaluate the model on the 50 heldout test examples and save the model with the best test accuracy. We show the training and test accuracy for each user objective below:

Table 3: Training accuracy for SL on the *Ultimatum Game*.

	30%	60%	\$10	\$100	Ineq. Aversion
Train Acc. (10 examples)	1.0	1.0	0.9	1.0	1.0
Train Acc. (1 example)	1.0	1.0	1.0	1.0	1.0

DealOrNoDeal. SL is trained to predict binary labels given a negotiation as input. We closely follow the implementation of a SL model found in (Kwon et al., 2021; Lewis et al., 2017). A negotiation consists of a context, coarse dialogue acts exchanged between *Alice* and *Bob*, and the outcome of the negotiation (the final split of items and whether agents agreed or disagreed). Please refer to Sec. A.3 for more details on the environment. We implement SL using a MLP context encoder, MLP outcome encoder, and a GRU ((Chung et al., 2014)) to process the coarse dialogue acts. The MLP encoders consist of an embedding layer followed by a linear layer with a Tanh activation function; we use a hidden size of 64 for the context encoder's linear layer. We similarly embed each coarse dialogue act before feeding it into the GRU. We use a hidden size of 128 for the GRU. We train SL on the same 3 examples we use in our prompt for LLM. We train for a maximum of 50 epochs using the Adam optimizer. SL received a training accuracy of 100% for all of our objectives.

A.5 Example of Prompts Used in Our Experiments

A.6 Example Negotiations

A.7 How Much Does LLM's Labeling Accuracy Change When We Vary the Prompt?

We analyze the effect of varying prompts on the LLM's labeling accuracy for the *Stubborn* negotiating style in DealOrNoDeal. We vary prompts in three ways: we vary the keyword (i.e., replacing "Stubborn" with its synonyms), the example negotiations, and the explanations associated with each example.

When varying the keyword, we use the synonyms: "Headstrong", "Obstinate", and a less commonly used word, "Froward". To vary example negotiations, we randomly sample three new negotiations to have counterbalanced labels, all positive labels, or all negative labels. We vary the explanations by coming up with two plausible sets of explanations a user might have given for each example. We also experiment with the scenario where we give no explanations. Results are shown in Fig. 13. Overall, varying prompts do not have a large impact on labeling accuracy — they all outperform the baseline. However, the quality of explanation seems to have the largest impact on labeling accuracy.

	10 Examples, No Explanation	1 Example, with Explanation
Task description	P1 and P2 are playing the Ultimatum Game. P1 proposes how they should split $10 and P2 can either accept or reject. If P2 accepts, then the deal is done. If P2 rejects, then both parties get nothing.	P1 and P2 are playing the Ultimatum Game. P1 proposes how they should split $10 and P2 can either accept or reject. If P2 accepts, then the deal is done. If P2 rejects, then both parties get nothing. *Task description*
Examples of Objective	P1 proposes a split of $4.21 for P1 and $5.79 for P2. P2 rejected this offer. A desirable outcome is defined as one where P2 punishes P1's selfish behavior. Is the outcome desirable? No P1 proposes a split of $1.28 for P1 and $8.72 for P2. P2 rejected this offer. A desirable outcome is defined as one where P2 punishes P1's selfish behavior. Is the outcome desirable? No P1 proposes a split of $9.78 for P1 and $0.22 for P2. P2 rejected this offer. A desirable outcome is defined as one where P2 punishes P1's selfish behavior. Is the outcome desirable? Yes [7 more examples]	P1 proposes a split of $9.78 for P1 and $0.22 for P2. P2 rejected this offer. A desirable outcome is defined as one where P2 punishes P1's selfish behavior. Is the outcome desirable? Let's think step by step: P2 receives $0.22 < $3 so P2 should reject this offer. Therefore, the outcome is desirable. *Example of Objective*
	P1 proposes a split of $9.21 for P1 and $0.79 for P2. P2 rejected this offer. A desirable outcome is defined as one where P2 punishes P1's selfish behavior. Is the outcome desirable? *Episode outcome* *Question*	P1 proposes a split of $9.21 for P1 and $0.79 for P2. P2 rejected this offer. A desirable outcome is defined as one where P2 punishes P1's selfish behavior. Is the outcome desirable? Let's think step by step: *Episode outcome* *Question*

Figure 8: An example of few-shot prompts used for the *Ultimatum Game*. We highlight the four parts of each prompt.

	Total Welfare	Equality	Rawlsian Fairness	Pareto-optimality	No Objective
Task description	We have a two-player game where P1 and P2 can choose one of these options. Options: A. if action1(P1) and action1(P2) => P1 gets reward of 2, P2 gets reward of 2. B. if action1(P1) and action2(P2) => P1 gets reward of 1, P2 gets reward of 3. C. if action2(P1) and action1(P2) => P1 gets reward of 3, P2 gets reward of 1. D. if action2(P1) and action2(P2) => P1 gets reward of 0, P2 gets reward of 0.	We have a two-player game where P1 and P2 can choose one of these options. Options: A. if action1(P1) and action1(P2) => P1 gets reward of 2, P2 gets reward of 2. B. if action1(P1) and action2(P2) => P1 gets reward of 1, P2 gets reward of 3. C. if action2(P1) and action1(P2) => P1 gets reward of 3, P2 gets reward of 1. D. if action2(P1) and action2(P2) => P1 gets reward of 0, P2 gets reward of 0.	We have a two-player game where P1 and P2 can choose one of these options. Options: A. if action1(P1) and action1(P2) => P1 gets reward of 2, P2 gets reward of 2. B. if action1(P1) and action2(P2) => P1 gets reward of 1, P2 gets reward of 3. C. if action2(P1) and action1(P2) => P1 gets reward of 3, P2 gets reward of 1. D. if action2(P1) and action2(P2) => P1 gets reward of 0, P2 gets reward of 0.	We have a two-player game where P1 and P2 can choose one of these options. Options: A. if action1(P1) and action1(P2) => P1 gets reward of 2, P2 gets reward of 2. B. if action1(P1) and action2(P2) => P1 gets reward of 1, P2 gets reward of 3. C. if action2(P1) and action1(P2) => P1 gets reward of 3, P2 gets reward of 1. D. if action2(P1) and action2(P2) => P1 gets reward of 0, P2 gets reward of 0.	We have a two-player game where P1 and P2 can choose one of these options. Options: A. if action1(P1) and action1(P2) => P1 gets reward of 2, P2 gets reward of 2. B. if action1(P1) and action2(P2) => P1 gets reward of 1, P2 gets reward of 3. C. if action2(P1) and action1(P2) => P1 gets reward of 3, P2 gets reward of 1. D. if action2(P1) and action2(P2) => P1 gets reward of 0, P2 gets reward of 0.
Question	Which option(s) result in the greatest total welfare? Let's think step by step: *Description of Objective* Total welfare is	Which option(s) result in equality of rewards? Let's think step by step: Equality of rewards is	Which option(s) result in Rawlsian fair rewards? Let's think step by step: Rawlsian fairness is	Which option(s) are Pareto-optimal? Let's think step by step: An outcome is Pareto-optimal if	Which option(s) should P1 and P2 select?

Figure 9: Examples of a zero-shot prompts used for each objective, including the no-objective baseline, in the *Matrix Games*. Due to limited resources when querying GPT-3, we queried GPT-3 in a batched manner and saved the corresponding labels to train our RL agents. Consequently, we do not have an *Episode outcome* in our prompts.

Chicken Game (Regular Order)	Chicken Game (Scrambled Order)
We have a two-player game where P1 and P2 can choose one of these options. Options: A. if action1(P1) and action1(P2) => P1 gets reward of 2, P2 gets reward of 2. B. if action1(P1) and action2(P2) => P1 gets reward of 1, P2 gets reward of 3. C. if action2(P1) and action1(P2) => P1 gets reward of 3, P2 gets reward of 1. D. if action2(P1) and action2(P2) => P1 gets reward of 0, P2 gets reward of 0. Which option(s) are Pareto-optimal? Let's think step by step: An outcome is Pareto-optimal if	We have a two-player game where P1 and P2 can choose one of these options. Options: A. if action1(P1) and action1(P2) => P1 gets reward of 3, P2 gets reward of 1. B. if action1(P1) and action2(P2) => P1 gets reward of 2, P2 gets reward of 2. C. if action2(P1) and action1(P2) => P1 gets reward of 1, P2 gets reward of 3. D. if action2(P1) and action2(P2) => P1 gets reward of 0, P2 gets reward of 0. Which option(s) are Pareto-optimal? Let's think step by step: An outcome is Pareto-optimal if

Figure 10: Examples of regular and scrambled outcomes for the Chicken Game (*Matrix Game* experiments). Scrambling creates new associations between joint actions and joint rewards. We scramble the outcomes in order to remove any bias LLM may have towards the order in which outcomes are normally presented.

Task description

Alice and Bob are negotiating how to split a set of books, hats, and balls.

Examples of objective

```
==========================================================================
Alice : book=(count:2 value:1) hat=(count:1 value:4) ball=(count:4 value:1)
Bob   : book=(count:2 value:0) hat=(count:1 value:6) ball=(count:4 value:1)
--------------------------------------------------------------------------
Bob   : propose: book=0 hat=1 ball=2
Alice : propose: book=1 hat=1 ball=4
Bob   : propose: book=0 hat=1 ball=0
Alice : propose: book=1 hat=1 ball=4
Bob   : agree
--------------------------------------------------------------------------
Agreement!
Alice: 9 points
Bob: 0 points

Is Alice a stubborn negotiator?
Yes, because she repeatedly suggested the same proposal, book=1 hat=1 ball=4.

==========================================================================
Alice : book=(count:2 value:1) hat=(count:1 value:4) ball=(count:4 value:1)
Bob   : book=(count:2 value:4) hat=(count:1 value:2) ball=(count:4 value:0)
--------------------------------------------------------------------------
Bob   : propose: book=2 hat=1 ball=0
Alice : propose: book=1 hat=1 ball=4
Bob   : agree
--------------------------------------------------------------------------
Agreement!
Alice : 9 points
Bob   : 4 points

Is Alice a stubborn negotiator?
No, because she did not repeatedly suggest the same proposal.

==========================================================================
Alice : book=(count:1 value:3) hat=(count:4 value:1) ball=(count:1 value:3)
Bob   : book=(count:1 value:10) hat=(count:4 value:0) ball=(count:1 value:0)
--------------------------------------------------------------------------
Alice : propose: item0=1 item1=4 item2=0
Bob   : propose: item0=1 item1=0 item2=0
Alice : propose: item0=1 item1=0 item2=1
Bob   : propose: item0=1 item1=0 item2=0
Alice : insist: item0=0 item1=4 item2=1
Bob   : agree
--------------------------------------------------------------------------
Agreement!
Alice : 7 points
Bob   : 10 points

Is Alice a stubborn negotiator?
No, because she did not repeatedly suggest the same proposal.
```

Episode outcome

```
==========================================================================
Alice : book=(count:1 value:0) hat=(count:1 value:1) ball=(count:3 value:3)
Bob   : book=(count:1 value:1) hat=(count:1 value:0) ball=(count:3 value:3)
--------------------------------------------------------------------------
Bob   : propose: book=0 hat=0 ball=3
Alice : agree
--------------------------------------------------------------------------
Agreement!
Alice : 1 points
Bob   : 9 points
```

Question

Is Alice a stubborn negotiator?

Figure 11: Example of a prompt used for DEALORNODEAL.

Context

Alice : book=(count:1 value:0) hat=(count:1 value:7) ball=(count:3 value:1)
Bob : book=(count:1 value:3) hat=(count:1 value:7) ball=(count:3 value:0)

Versatile

Alice : propose: item0=1 item1=1 item2=0
Bob : propose: item0=0 item1=1 item2=0
Alice : propose: item0=1 item1=0 item2=2
Bob : agree

Agreement!
Alice : 2 points
Bob : 7 points

Push-over

Alice : propose: item0=1 item1=0 item2=3
Bob : propose: item0=1 item1=1 item2=0
Alice : agree

Agreement!
Alice : 3 points
Bob : 10 points

Competitive

Alice : propose: item0=1 item1=0 item2=3
Bob : propose: item0=1 item1=1 item2=0
Alice : insist: item0=0 item1=1 item2=3
Bob : agree

Agreement!
Alice : 10 points
Bob : 3 points

Stubborn

Alice : propose: item0=0 item1=1 item2=1
Bob : propose: item0=0 item1=1 item2=0
Alice : propose: item0=0 item1=1 item2=1
Bob : propose: item0=0 item1=1 item2=0
Alice : propose: item0=0 item1=1 item2=1
Bob : propose: item0=0 item1=1 item2=0
Alice : propose: item0=0 item1=1 item2=1
Bob : propose: item0=0 item1=1 item2=0
Alice : propose: item0=0 item1=1 item2=1
Bob : propose: item0=0 item1=1 item2=0
Alice : propose: item0=0 item1=1 item2=0
Bob : disagree

Disagreement?!
Alice : 0 (potential 0)
Bob : 0 (potential 0)

Figure 12: Example negotiations after *Alice* is trained with reward signals from LLM in DEALORNODEAL. We illustrate qualitative differences in how *Alice* negotiates for the same context. *Bob* is an agent that is trained with supervised learning.

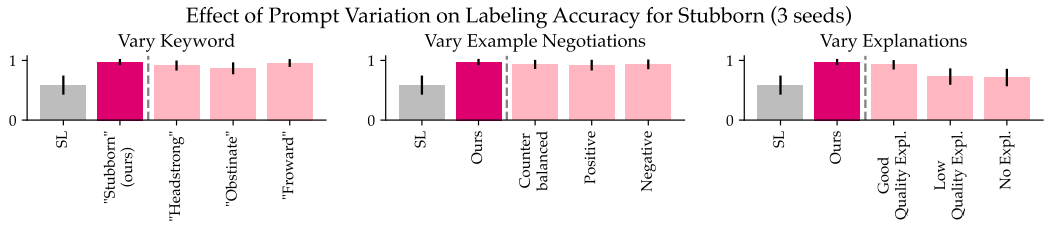

Figure 13: Varying prompts do not have a large impact on labeling accuracy.