Paper Review of Language-Instructed Reinforcement Learning for Human-AI Coordination

Yaro Kazakov

September 22, 2024

Abstract

In collaboration with Yuchen Cui, we aim to understand the long-term consequences of large language models (LLMs) for reinforcement learning (RL) in multi-agent environments. Our goal is to propose further research and potential improvements.

1 General Idea of the Paper

The paper emphasizes that the objective is not to follow or find an optimal strategy but rather to adhere as closely as possible to a human-defined policy. These strategies often tend to be equilibrium policies, meaning optimal or near-optimal joint policies. The authors mention that, **humans often prefer specific subsets of policies — particular equilibria in multi-agent games — that align well with our capabilities and common sense**, while reinforcement learning policies that do not incorporate any form of human priors often converge to policies that are difficult for humans to collaborate with.

COMMENT: I am not sure if this statement applies to all environments and humans. Common-sense strategies are not always optimal or sub-optimal.

Examples include Edward Thorp and the optimal blackjack strategy, which contrasted with how people played before. Another example is OpenAI's Deep RL bots that defeated the Dota 2 champions in 2019. Human players considered the bots' actions "weird" and not aligned with "common sense." However, humans later adopted many of those strategies in human-vs-human games, improving their overall gameplay.

My point is that we should use human instructions as a good starting point or prior, but also explore other policies, given this prior. The authors of this paper provide two examples where humans guided the agent toward near-optimal strategies, but these are specific to the environments the authors used as examples. What if there is an environment where humans perform sub-optimally, but no one realizes it? This might be an idea for regularizing with LLM priors in future research.

2 Problems with RL-Human Interactions

It is unclear how existing RL algorithms can reliably produce policies that are most natural to humans. For the authors' Alice and Bob game, it is easy to see that, from the RL algorithms' perspective, there are numerous equally optimal joint policies that achieve perfect results by learning an arbitrary mapping from Alice's past action sequences to Bob's decision-making.

If a researcher approached this problem without LLMs, **they would produce a diverse set of policies** and then train a common best-response strategy that may generalize to any human partner.

Problems with this approach:

1. It implicitly requires the underlying RL algorithm to generate policies near the equilibria that humans prefer, which may be problem-dependent.

2. It incurs a much higher computational cost to produce enough policies to facilitate generalization. In the Alice-Bob game, a common best response trained against all possible optimal policies still requires many episodes of exploration to identify its partner's policy when paired with a human.

How the authors address these issues:

1. Humans can better understand and coordinate with a policy if it can be concisely summarized in natural language.

2. In most real-world coordination scenarios, humans communicate or even negotiate to reach agreements on how they should collaborate, such as determining which conventions or equilibrium to follow.

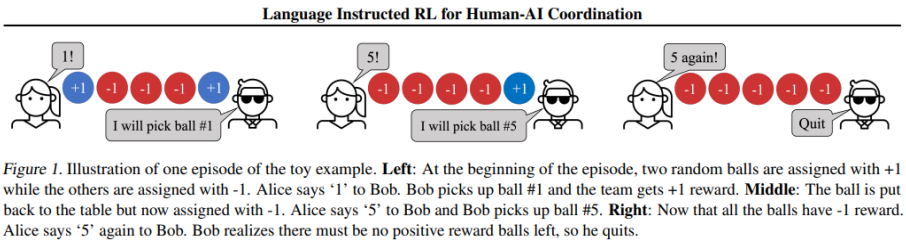

Figure 1. Illustration of one episode of the toy example. **Left**: At the beginning of the episode, two random balls are assigned with +1 while the others are assigned with -1. Alice says '1' to Bob. Bob picks up ball #1 and the team gets +1 reward. **Middle**: The ball is put back to the table but now assigned with -1. Alice says '5' to Bob and Bob picks up ball #5. **Right**: Now that all the balls have -1 reward. Alice says '5' again to Bob. Bob realizes there must be no positive reward balls left, so he quits.

Figure 1: Alice-Bob Optimal Strategy as Defined by a Human

These two important aspects have not been considered in existing multi-agent reinforcement learning (MARL) methods, and we aim to incorporate them to guide RL agents toward more human-like policies.

COMMENT: My guess is that the long-term goal of the authors is to create an agent capable of working in complex environments with humans through verbal communication. This approach seems to combine MARL, RLHF (Reinforcement Learning from Human Feedback), Inverse Modeling (IM), and Human-AI interaction. **What is the communication overhead?**

3 Specifics of the Framework

The authors introduced InstructRL, a framework that enables humans to specify the types of strategies they expect from their AI partners through natural language instructions.

They used pretrained large language models to generate a prior policy conditioned on the human instruction, and used the prior to regularize the RL objective.

We first (1) construct a prior policy by querying large language models (LLMs) based on the (2) instructions and (3) short descriptions of the current observation. So, is the output of the **prior policy a text**?

3.1 LLM Prior Policy

For instance, in the Say-Select example in Figure 1, we can give the instruction "Select the same number as Alice" to the Bob agent. The description of the current observations could be "Alice said 1." We then train an RL agent, where its objective is regularized using the generated LLM prior as a reference policy.

COMMENT: Regularization based on a reference policy usually involves applying a penalty for the difference between the proposed policy and the reference.

We construct the prior policy by letting an LLM predict the probability of possible actions given the observation and the instruction. To do so, we essentially need to evaluate:

$p_{LLM}[lang(a_t)|lang(\tau^i t), inst]$

$pLLM$ = softmax($\beta * logit$), where β is an optional scaling factor and the logit is a function of the language components, i.e., logit = $f(inst, lang(\tau), lang(a_t))$
For actions that have homogeneous descriptions, such as in Say-Select, the logit function f can simply be the prediction loss. A reminder that prediction loss is:

Loss = $-\sum_{i=1}^{n} y_i \cdot \log(\hat{y}_i)$
Note that we would iterate over the true and observed probabilities of all actions in a given state. We want the loss to approach 0.

COMMENT/POINT OF IMPROVEMENT: This is a great idea. Problems may arise when it's difficult to map to a_t. LLMs are non-deterministic and can hallucinate. While the instructions remain constant, the wording of the observations might change slightly. This paper tries to emulate an interaction between a human and a robot. Suppose, even in simple settings, a human says/prompts: "I'd recommend you pick 1," "Pick one," "One is good," or "Good to go with 1." Are we sure that these minor changes in the wording of the observation won't cause the model to hallucinate or map us to a different action? This challenge can be overcome with fine-tuning and using more advanced language models, but this prior reliance on the inherently stochastic nature of LLMs can be tricky.

The authors also mention that: It is easy to convert actions to language descriptions in the games considered in this paper. However, it is worth noting that many environments contain actions that cannot be easily abstracted into language, e.g., in a robotics setting, where the actions are the continuous joint angles of a robot arm.

COMMENT: This is true, but even in simpler settings, models can generate language interpretations of actions that do not exist in your $lang(a_t)$ space.

SUGGESTION: What if we instead use LLMs to convert back to dictionaries or history lists? That is, a human says/prompts something, and we use function calling to only produce a parameterized probability of possible actions?

SUGGESTION: Additionally, what if we experiment with the current settings and introduce variable instructions?

SUGGESTION: The authors suggest that in real-world scenarios that require grounding in the physical environment, we may use image captioning models. Instead of words, we could experiment with descriptive languages of the scenes or extend the framework to allow humans to specify instructions in video format.

SUGGESTION: "The LLM prior p_{LLM} itself is not sufficient to solve complex tasks. For example, a moderate-sized LM with roughly 6B parameters cannot figure out when to quit in Say-Select in Figure 1, and even the largest LM to date cannot play Hanabi." Why use a 6B model for this task? A larger model can definitely follow human instructions. **Offer this to Yuchen.**

3.2 Regularization

Regularization has been widely used in RL to encourage exploration (Mnih et al., 2016) or to **encourage an RL policy to stay close to a given prior**. In this paper, the author wants to similarly regularize an RL agent to guide the equilibria towards desirable behaviors. They consider two types of regularization techniques for **Q-learning** (check my blog on this) and **PPO**, respectively.

For Q-learning with priors, the authors reference this, which I don't have access to. I assume the authors reference the Q-learning update rule for prior knowledge and prior probabilities. The p_{LLM} becomes:

$$a = \epsilon\text{-GREEDY}(Q_\theta + \lambda \log p_{LLM})$$

and the training update rule becomes

$$Q_\theta(\tau_t^i, a_t) \leftarrow r_t + \gamma Q_\theta(\tau_t^{i+1}, a_{t+1}),$$

where

$$a_{t+1}^{'} = \arg\max_a \left[Q(\tau_{t+1}^i, a) + \lambda \log p_{LLM}(\tau_{t+1}^i, a) \right].$$

Note how this algorithm is different to the standard Q-learning:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right)$$

Choosing action using the exploration policy:

$$a_t = \epsilon\text{-GREEDY}(Q)$$

For PPO (Schulman et al., 2017b;a), we add a KL penalty term to the objective:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[R(\tau) + \lambda \text{KL}(\pi_\theta || p_{LLM}) \right].$$

The policy loss becomes:

$$\mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_t \left[-\log \pi_\theta(a_t | \tau_i^t) A_t + \lambda \text{KL}[\pi_\theta(\tau_i^t) || p_{LLM}(\tau_i^t)] \right] \right].$$

COMMENT: Read the paper on PPO or get a hang of it by reading supplementary material. PPO seems to provide better stability and better transition under the gradient policy ascent than Q-learning.

3.3 Experiment

In this game, we use `instructQ` for Bob, while allowing Alice to use vanilla Q-learning.

COMMENT: Why does Alice need to use vanilla Q-learning at all? Doesn't she emulate a human?

We set the instruction for Bob: `inst = "I should select the same number as my partner"`. We map Bob's observation to text `lang(`τ_i^t`)` by converting Alice's most recent action (1 through 5) from integer to string. Note that the RL policy still observes the last two actions. For Bob's actions, we map them to the strings "0", "1", . . . , "5", with 0 for quitting and the remaining 1 through 5 for selecting the corresponding ball. We combine all these components to create the following prompt:

1. `inst`
2. My partner selected `lang(`τ_i^t`)`
3. So should I select

COMMENT: It would be interesting to see how prompt-specific this part is.

The authors feed the prompt to an open-sourced GPT-J model with 6 billion parameters and use the prediction loss for the action strings as logits (Ahn et al., 2022).

COMMENT: Confirm with Yuchen that this means outputting probabilities of each word? Sometimes GPT-J can output it in the form of logits directly.

They apply `SOFTMAX` to the logits with $\beta = 1$ to get the prior policy p_{LLM}. They use tabular Q-learning

with no neural network as the state space is small enough, and a regularization weight $\lambda = 0.25$ for `instructQ`. Details on the hyperparameters are in Appendix A.1.

COMMENT: A neural network would be better for a more sophisticated problem then? Check with Yuchen.

COMMENT: The learning rate was mentioned in the Appendix. It is not in the formula. I guess the author mentioned it for the vanilla Q-learning for Alice? The update rule for the regularized version doesn't include the learning rate, right?

4 Result

With `instructQ`, Alice and Bob always (10 out of 10 seeds) converge to the intuitive joint policy.

COMMENT: These results are fascinating because the first two tables are constructed based on the Q-learning algorithms that had not been able to identify a human-defined strategy, meaning quitting when Alice says the same number twice. Now I understand the purpose of `instructQ`. The optimality of the first two strategies cannot be defined in human terms as "common sense" or "sensible." The first two strategies are essentially randomly fitted rewards.

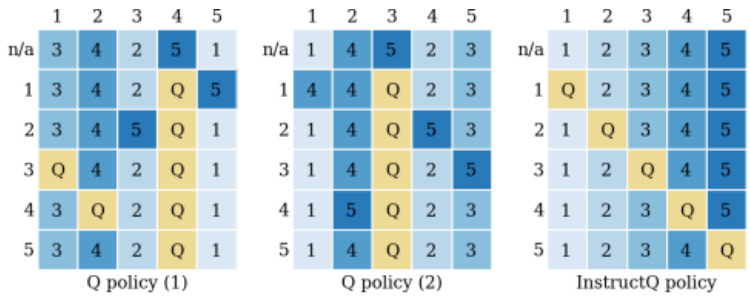

Figure 3. Bob's policy trained with different methods. Row values are Alice's actions *two steps ago* and column values are Alice's actions *one step ago*. The value in each cell is Bob's action when observing Alice's past two actions. Here Bob's actions are 1 through 5 (shown in different shades of blue) for selecting different balls and "Q" (shown in yellow) refers to Bob quitting. **Left** and **Middle**: Two policies from vanilla Q-learning but with different seeds. **Right**: Policy from instructQ with $\lambda = 0.25$. We note that all three policies shown here are optimal in self-play, but only the InstructQ policy is the intuitive policy that follows `inst`= "I should select the same number as my partner".

Figure 2: Optimality based on Q-learning and InstructQ

5 Hanabi

The idea is the same as before, but the environment is more complex.

Q-learning and PPO are as good as the `instructQ` policies in terms of collected reward!

The downside of vanilla Q-learning and PPO is the lack of interpretability and the inability for humans to enjoy the optimal strategy. **SUGGESTION:** Try a better language model first.

SUGGESTION: Different question style? What can we do instead of YES/NO questions?

We pre-compute the logits for all $\text{lang}(\tau_t^i)$ and $\text{lang}(a_t)$ pairs and cache them before training RL.

COMMENT: Discuss this idea with Yuchen.

The LLM prior policy only needs to provide coarse guidance to bias the RL agent to converge to desired equilibria. Second, we do not want the RL agent to suffer from the inherent biases of the LLM because they can lead to sub-optimal outcomes.

COMMENT: Discuss this too. How coarse is coarse? How well would it converge to the policy if the priors were more detailed?