Exploring the Path from Instructions to Rewards with Large Language Models in Instance-Based Learning

Chase McDonald[1], **Tyler Malloy**[1], **Thuy Ngoc Nguyen**[2], **Cleotilde Gonzalez**[1]
[1] Carnegie Mellon University
[2] University of Dayton
coty@cmu.edu

Abstract

A prominent method to model human learning is through experiential learning, where decisions are influenced by the outcomes observed in previous actions. The decisions-from-experience approach often excludes other forms of learning in humans, such as learning from descriptive information. In humans, descriptive information can enhance learning by providing a denser signal, achieved through understanding the relationship between intermediate decisions and their future outcomes, instead of relying solely on observed outcomes. To account for experiential and descriptive information, we propose the use of large language models (LLMs) to convert descriptive information into dense signals that can be used by computational models that learn from experience. Building on past work in cognitive modeling, we utilize task instructions and prompt an LLM to define and quantify the critical actions an agent must take to succeed in the task. In an initial experiment, we test this approach using an Instance-Based Learning cognitive model of experiential decisions in a grid-world task. We demonstrate how the LLM can be prompted to provide a series of actions and relative values given the task instructions, then show how these values can be used in place of sparse outcome signals to improve the model's learning of the task significantly.

Introduction

Human learning is commonly modeled through decisions from experience or trial-and-error interactions with an environment. In this paradigm, agents often learn tabula rasa with the need to discover all the effects that their actions can have in the environment. There are a number of reasons why such an approach may be undesirable. For instance, the learning agent will need to explore unsafe actions, and learning is significantly slower and inefficient in such models when prior knowledge about the world is not incorporated into the decision making process.

When humans learn a new task, they are capable of learning from non-experiential sources of information, such as instructions (Walsh and Anderson 2011b). In this work, we propose the use of LLMs to parse instructions to extract reward signals that guide experiential learning. Specifically, we show how an LLM can be prompted with task instructions to provide both a set of steps that an agent should complete to maximize their task success and the values associated with each step. We then demonstrate how these rewards can be used in place of a sparse outcome signal in a task to significantly improve learning.

We briefly review related concepts and literature, then describe and discuss our methods, experiment, results, and paths for future investigation.

Temporal Credit Assignment Temporal credit assignment is the process of attributing credit to past actions responsible for the experienced rewards or future outcomes. This process is crucial for learning and making decisions in dynamic environments over time with feedback delays (Sutton and Barto 2018; Fu and Anderson 2008). Many studies in reinforcement learning and cognitive science have explored different aspects of temporal credit assignment, including integrating hierarchical structures and intrinsic motivation to enhance the learning process (Kulkarni et al. 2016), as well as whether existing credit assignment mechanisms are well aligned with human behavior (Gershman and Daw 2017; Walsh and Anderson 2011a; Nguyen, McDonald, and Gonzalez 2023).

Temporal credit assignment can be bolstered by providing appropriate signals to the agents—in the form of intrinsic rewards—that provide supplementary information about the effect that actions in a sequence have on observed outcomes. This has received significant attention in the reinforcement learning literature, from explorations of intrinsic rewards for cooperative social learning (Hughes et al. 2018) to long-term credit assignment (Zheng et al. 2020).

Intrinsic Motivation Studies in psychology have shown that individuals do not always solely aim to maximize their utility (Dovidio 1984). In such cases, it is often assumed that participants or agents are intrinsically motivated. Hence, intrinsic rewards could be an alternative to external rewards, particularly in environments with sparse signals for rewards or success (Chentanez, Barto, and Singh 2004). In studying both human intrinsic motivations and modeling these motivations computationally, a key challenge is in knowing how to construct these internal rewards. Among numerous alternative sources (e.g., from social preferences in behavioral economics (List 2006) to preferences for self-efficacy (Blain and Sharot 2021)), past work has shown that in goal-oriented

settings, such supplementary motivations may come from sub-goals based on an individual's understanding and breakdown of their overall goal (Huang, Jin, and Zhang 2017).

With the understanding that humans have intrinsic motivations guided by their preferences and knowledge that drive their behavior, we can follow extensive existing work in computational modeling and use intrinsic signals to improve learning over temporally extended tasks. In order to do so, we must formalize a process that takes information about the task and any relevant world knowledge and distills it into concrete intrinsic reward structures. To that end, we utilize task instructions and LLMs: the former provides the context of the task and the latter utilizes natural language understanding and reasoning to decompose the task into subgoals for the agent.

LLMS, Learning, & Decision Making LLMs gained prominence through their performance in natural language settings; however, there is a growing body of literature that demonstrates their efficacy in control and decision making tasks, and in providing useful quantifications that may be difficult for humans to produce.

To the latter point, Park et al. (2023) develop a social simulation that relies on LLM prompting for agents to select actions and communicate with one another. As part of their agent architecture, they utilize LLM queries to ask how relevant and important particular pieces of information are to the decision at hand—they automate these processes that are often difficult to formalize in computational experiments by simply querying the LLM. They demonstrate that these values result in meaningful actions from the agents and emergent social behavior in a population.

Wu et al. (2023a) develop an approach to provide intrinsic reward signals to reinforcement learning agents in Atari games by via the instruction manuals. They utilize a question-answering extraction module to summarize game instructions and a reasoning module to determine if particular agent interactions are valuable in the current state of the game. Based on the reasoning modules assessment (a "Yes" or "No" response if reward should be provided), fixed intrinsic rewards or penalties are provided to a reinforcement learning agent during training. They demonstrate that such an approach can provide significant gains in terms of learning speed and overall performance when compared to baselines without their instruction-based reward.

In our work, we combine the aforementioned ideas to build a mechanism for developing goal-based intrinsic rewards that allow an agent to utilize non-experiential information. This allows for improved temporal credit assignment—and in turn, learning—in sequential tasks with sparse rewards.

Methods

Experimental Task We use a modified version of the Door Key task from the MiniGrid environment (Chevalier-Boisvert et al. 2023). The environment is formalized as a partially observable Markov Decision Process (POMDP), which is represented by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R}, \Omega, \gamma)$. Here, $\mathcal{S}$ represents the state space of the environment, $\mathcal{A}$ the action space, $\mathcal{O}$ the observation space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ the transition function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the reward function, $\Omega : \mathcal{S} \to \mathcal{O}$ the observation function, and $\gamma \in [0, 1)$ the discount factor. In a POMDP, the goal is for a policy, at every step $t \in \{0, ..., T\}$ to select an action $a_t \in \mathcal{A}$ after observing observation $o_t \in \mathcal{O}$ to maximize the discounted return $\sum_{i=t}^{T} \gamma^{i-t} r_i$, where $r_t \in \mathbb{R}$ is the reward observed after taking an action and the environment transitioning to a new state.

In our modified Door Key task, there is an agent, a key, a door, and two targets: one green and one purple. In each episode, the agent must navigate through the environment to collect either the green or the purple target. The episodes terminate either when a target has been collected or the time horizon T has been reached. The action space $\mathcal{A}$ is comprised of six actions: Move North, Move South, Move East, Move West, Pick Up/Drop, and Toggle. The agent faces one of four cardinal directions and navigates the environment through one of the four Move actions. If the agent attempts to move in a direction that it isn't currently facing, its orientation will change to the desired direction. If the orientation is aligned with the desired movement, the agent will move one unit in the desired direction—given that they are unobstructed. When an agent is facing and one unit away from the key, it can use the Pick Up/Drop action to pick it up. The same action will drop the key if the space in front of the agent is empty. When holding the key, the agent can unlock and open the door by using the Toggle action when facing the door. The same action will close the door, but it cannot be locked again. Finally, to collect a target, the agent must use the Toggle action when facing a target.

Observations of the environment take a simple form: in each observation o_t, the agent observes its coordinates in the grid, its current direction, an indicator if the agent is holding a key, and an identifier for the object in the cell directly in front of the agent.

For our preliminary investigation, we only use a single layout depicted in Figure 1. It is important to note that the observation representation is not generalizable across grids (e.g., the coordinates are only useful insofar as their surroundings are fixed), and is constructed as a simple representation for our demonstration.

Instance-Based Learning Theory Actions of the agent in the Door Key task are determined using a cognitive model of learning and dynamic decision making based on Instance Based Learning Theory (IBLT). This theory is related to the ACT-R cognitive architecture through the activation function, which is ultimately used to predict the estimated utility of performing an action in a state based on the utility outcomes of similar past experiences held in declarative memory (Thomson et al. 2015). We refer to a computational model implementing IBLT as an IBL model.

In IBLT, declarative memory consists of instances $k = (o, a, x)$ represented by the observation that describes the state of the environment o, the action performed by the agent a, and the utility outcome of that action x.

Agent actions are determined by maximizing the value $V_{k,t}$ of an available action a in an instance k performed at

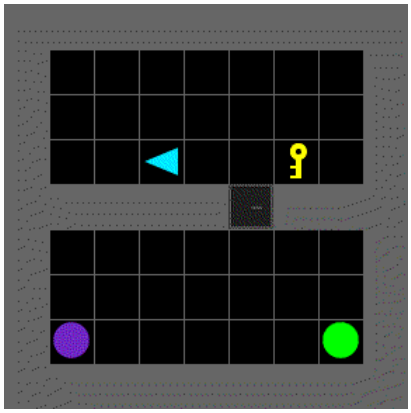

Figure 1: A rendering of the full game state of the modified Door Key task. The blue triangle is the agent pointing west, three units to its east is a key, and a locked door is represented by the dark gray block. The light gray cells are impassable walls. The purple and green circles represent targets, which upon collection earn the agent a reward of 0.4 and 0.1, respectively.

time-step t, calculated using the "blending" function (Gonzalez, Lerch, and Lebiere 2003):

$$V_{k,t} = \sum_{i=1}^{n_{k,t}} p_{i,k,t} x_{i,k,t} \tag{1}$$

where $n_{k,t}$ are the previously generated instances held in procedural memory, $x_{i,k,t}$ are the outcomes of those instances, and $p_{i,k,t}$ is the probability of retrieving an instance in memory, calculated by Equation 2.

$$p_{i,k,t} = \frac{\exp\left(\Lambda_{i,k,t}/\tau\right)}{\sum_{j=1}^{n_{k,t}} \exp\left(\Lambda_{j,k,t}/\tau\right)} \tag{2}$$

Further, $\Lambda_{i,k,t}$ is given by Equation 3.

$$\Lambda_{i,k,t} = \ln\left(\sum_{t' \in T_{i,k,t}} (t - t')^{-d}\right) + \sigma \ln \frac{1 - \xi_{i,k,t}}{\xi_{i,k,t}} \tag{3}$$

Here, d and σ are decay and noise parameters, and $T_{i,k,t} \subset \{0, ..., t-1\}$ is the set of previous timesteps where instance k was stored in memory. The $\xi_{i,k,t}$ term is used to capture noise in the individual differences in memory recall. Because of the relationship between noise σ and temperature τ in IBLT, the temperature parameter τ is typically set to $\sigma\sqrt{2}$. In our experiments, we use all default parameters of $d = 0.25$ and $\sigma = 0.5$. We also set the default utility to 1.0 to encourage exploration through an optimistic prior Sutton and Barto (2018). The default utility is used to predict the utility of an instance when there are no similar instances in memory to estimate the expected utility.

A key aspect of applying IBLT to modeling decision making is determining the utility outcome of actions, which can either be directly provided by the learning environment or determined by the cognitive modeler. Nguyen, McDonald, and Gonzalez (2023) investigate several methods for temporal credit assignment in IBL models, demonstrating the relative efficacy of each. We adopt the simple method of assigning outcomes at the end of each episode with their exponentially discounted future return. Formally, the ith choice is assigned outcome $x_i = \sum_{t=i}^{T} \gamma^{t-i} r_t$.

LLM Reward Model We construct a reward model by prompting a pre-trained LLM. For our experiments, we use OpenAI's `gpt-3.5-turbo`. Drawing on previous successes in utilizing LLMs in control settings (e.g., Wu et al. (2023a,b); Park et al. (2023); Ahn et al. (2022)), we query the LLM to produce a plan or sub-goals that are critical for success in the task. In contrast to Wu et al. (2023a), where the LLM decides whether or not a fixed reward should be provided to an agent, we also query the LLM to provide the value of the reward. Specifically, we frame our query in terms of steps that the agent must achieve in order to reach its goal of reward maximization. The full query is as follows, with `{instructions}` being replaced by the text provided in Appendix :

```
Your job is to evaluate a game and
generate a step-by-step plan for the
player to achieve the maximum score
possible.
The instructions are:
  ⟨begin instructions⟩
  {instructions}
  ⟨end instructions⟩
Based on the instructions, identify the
most important steps that the player
must meet in order to maximize their
reward, and provide a value from 0 to 1
that represents the value of that step
to reward maximization.
The list of steps should be in the
following format:
  1. Player State:  ⟨step in terms of
  the state of the game or player
  observation⟩. Value: ⟨value between 0
  and 1⟩
```

Due to the brevity of our task instructions, context length is not prohibitive and we are able to input the instructions in their entirety along with our reward prompt. In alternative contexts, such summarization techniques may be required, as in, e.g., Wu et al. (2023a,b).

In order to provide a reward to the agent, we construct an environment parser that identifies when each step in the plan has been accomplished and provides the associated reward to the agent. In an idealized setting, this process would be conducted via an additional LLM query, where the environment state or agent trajectory would be input into the model along with a prompt that elicits an evaluation of each step in the plan. The latency and cost associated with many queries make such an automated process prohibitive.

Within each episode, we consider the full set of steps provided by the initial LLM query. Once a step has been evaluated to have been completed, the associated value is returned to the agent as a reward and that step is no longer considered until the subsequent episode. An example set of steps and values, utilized in our experiments, is provided below:

```
 1. Player State: Identify the location
of the Green target. Value: 0.2
2. Player State: Identify the location
of the Purple target. Value: 0.2
3. Player State: Identify the location
of the key. Value: 0.1
4. Player State: Identify the location
of the door. Value: 0.1
5. Player State: Collect the Green
target. Value: 0.3
6. Player State: Collect the Purple
target. Value: 0.5
7. Player State: Collect the key.
Value: 0.2
8. Player State: Open the door using
the Toggle action. Value: 0.3
9. Player State: Navigate to the Green
target. Value: 0.4
10. Player State: Navigate to the
Purple target. Value: 0.6
```

The setting is sufficiently simple that these intrinsic rewards may have been hand-crafted. Our demonstration focuses on the efficacy of the elicitation of these steps and values from an LLM. We also note, in this LLM response, there is redundancy in the steps: steps (1) and (2) are operationalized in the same way as (9) and (10): identifying the location of the targets requires the agent to navigate to them. In our implementation, the first occurrence of navigation to the target is validated as identification in (1) and (2), and any subsequent arrivals are validated with steps (9) and (10).

The LLM reward model $\mathcal{R}_{LLM} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ can be formalized as the following. Let the context $\mathcal{C}$ by defined as the combination of the task instructions and previously described query. The LLM model $\mathcal{M}$ takes as input the context and produces a set of k rewarded steps $\mathcal{P} = \{p_1, ..., p_k\}$ and the values associated with each step $\mathcal{V} = \{v_1, ..., v_k\}$. Finally, let $\mathcal{E} : \mathcal{S} \times \mathcal{P} \to \{\text{True}, \text{False}\}$ be environment parser that takes an input the state $s \in \mathcal{S}$ of the POMDP and a step $p \in \mathcal{P}$ and returns a boolean indicating whether or not that step has been satisfied. In each environment episode, we initialize the set of remaining steps $\mathcal{P}_{remaining} := \mathcal{P}$ to be the full set of steps provided by the LLM. At every timestep t, we check whether any remaining step has been satisfied $\mathcal{E}(s_t, p_k) \forall p_k \in \mathcal{P}_{remaining}$. For any p_k that is satisfied, the reward v_k is returned to the agent and p_k is removed from the remaining set. That is, $\mathcal{P}_{remaining} \leftarrow \mathcal{P}_{remaining} \backslash \{p_k\}$. In our experiments, the reward model entirely replaces the environmental reward function $\mathcal{R}$ when used.

Results

For our experiment, we ran 50 independent trials of an IBL model learning in two conditions in a single configuration of the task, as shown in Figure 1. Our two conditions are defined by the source of reward: either the environmental reward dictated by POMDP or the LLM reward. Each trial consists of 200 episodes, each having a maximum of $T = 250$ timesteps. Each episode ends when the maximum number of timesteps elapses or a target is collected. Our outcome of interest is the environmental reward in both conditions.

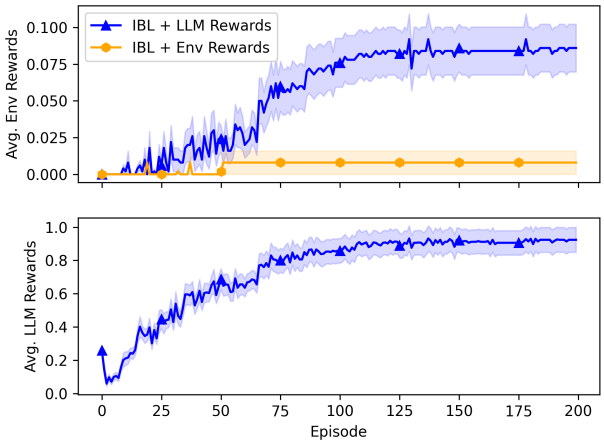

Figure 2: The top panel shows the average environment reward earned in each episode by the IBL model that utilizes only environment rewards and the IBL model that utilizes only the LLM reward. The bottom panel shows the LLM reward earned by the IBL agent that received that reward signal. The shaded region shows the standard error over 50 independent trials.

The results of the experiment in terms of average reward over time are shown in Figure 2. The baseline IBL model with the environmental reward fails to learn to reach a target in the vast majority of trials. The baseline model reaches a maximum average reward of 0.008 ± 0.003. On the other hand, the IBL model with the LLM reward is able to make significant improvements over the baseline by increasing the average reward tenfold to 0.084 ± 0.011.

In the bottom panel of Figure 2, we can observe how the LLM reward signal continuously increases before the environment signal makes significant gains: the LLM reward provides a guiding signal to the agent that helps it explore the task space effectively: reaching and picking up the key, finding and opening the door, and locating the target all provide a reward that enables the agent to reach higher environmental rewards than it would otherwise.

Discussion

Our preliminary experiment demonstrates the promise of utilizing LLMs for intrinsic reward signals in instance-based learning models solely from task instructions and the environment state. Indeed, our results demonstrate a significant improvement in the success of an IBL model when utilizing the reward scheme defined by our LLM-based reward model and the value of incorporating non-experiential information into the reward scheme. It is important to note that the specifics of our experiment and results are dependent on the model, model parameters, and the environment; however, it has nonetheless demonstrated the feasibility of our approach.

Future Directions Our work leaves many potential future avenues for inquiry. In particular, we have shown a proof-of-concept for using LLMs as a source for a dense, intrinsic reward via non-experiential information in cognitive modeling. This initial step was done in a simplistic environment with a reward scheme that could be easily designed by a modeler. The natural next step is to advance to more complex environments and representations that would prove more burdensome for a modeler to explicitly construct. In the same vein, ablation experiments should be conducted to determine how valuable the steps are in conjunction with the LLM reward values versus predefined rewards (e.g., selecting a constant value for all steps identified by the model). Furthermore, we use a simple prompting scheme for the LLM and we receive relatively simple steps. A more advanced prompting chain may be necessary in more complex environments and may yield more nuanced responses (e.g., in our setting we would likely have seen increased performance if the steps had additional conditions, such as reaching the door only when a key had been picked up).

As previously discussed, we manually design an environment parser to check if the steps have been met by the agent to earn the LLM rewards. As the cost and latency of LLM queries decrease, it will likely become more feasible to query the model at every step to allow it to determine which goals have been met, entirely automating the process we've described here. Furthermore, as the feasibility of per-environment step LLM queries improves, this approach can be further refined to develop new steps as the episodes progress, conditioning them on the agent's previous behaviors to refine the reward structure as the agent learns.

LLMs have provided a new toolset for computational cognitive modeling: they will allow us to incorporate new kinds of reasoning, understanding, and non-experiential information in settings where it was previously difficult or infeasible. Our experiment provides evidence of the efficacy of this tool in one of many possible use cases.

Task Instructions

The full task instructions provided to the model are:

```
You are playing a game in a
grid-based environment. Your
goal is to navigate through the
environment and to maximize your
reward by collecting one of two
targets: Purple or Green. There is
one Green and collecting it will
give you a reward of 0.1. There
is one Purple and collecting it
will give you a reward of 0.4. To
collect either a Green or Purple
target, you must rotate toward that
target and identify a path to it.
If you collect one target, you will
be unable to collect the other.

In the game, there are also walls,
doors, and keys. In order to open a
door, you must first find a key and
pick it up, then navigate to the
door and use the Toggle action to
open it.

You may navigate by moving North,
South, East, or West. When there is
a target directly in front of you
(one unit away in the direction you
are facing) you must use the Toggle
action to collect the target. If
a target is not directly in front
of you, the collect action does
nothing. Once a single target is
collected, the game is complete and
you will restart.

There are six possible actions:
Move North, Move East, Move South,
Move West, Pick Up/Drop, and
Toggle. You will also be facing one
of the four cardinal directions.
If you are facing a direction,
selecting the Move action in that
direction will advance you a single
unit in that direction. Otherwise,
it will change your rotation to
face that direction and you will
not move.

In the game, you must identify
target locations through
exploration. You will only have
access to your current position,
rotation, what item you are holding
(if any), and an indicator of what
object is directly in front of you.
Identifying a target (or any other
object) requires that you navigate
to an adjacent cell and face the
target, such that it is directly in
front of you.
```

References

Ahn, M.; Brohan, A.; Brown, N.; Chebotar, Y.; Cortes, O.; David, B.; Finn, C.; Fu, C.; Gopalakrishnan, K.; Hausman, K.; et al. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

Blain, B.; and Sharot, T. 2021. Intrinsic reward: potential cognitive and neural mechanisms. *Current Opinion in Behavioral Sciences*, 39: 113–118.

Chentanez, N.; Barto, A.; and Singh, S. 2004. Intrinsically motivated reinforcement learning. *Advances in neural information processing systems*, 17.

Chevalier-Boisvert, M.; Dai, B.; Towers, M.; de Lazcano, R.; Willems, L.; Lahlou, S.; Pal, S.; Castro, P. S.; and Terry, J. 2023. Minigrid & Miniworld: Modular & Customizable Reinforcement Learning Environments for Goal-Oriented Tasks. *CoRR*, abs/2306.13831.

Dovidio, J. F. 1984. Helping behavior and altruism: An empirical and conceptual overview. *Advances in experimental social psychology*, 17: 361–427.

Fu, W.-T.; and Anderson, J. R. 2008. Solving the credit assignment problem: explicit and implicit learning of action sequences with probabilistic outcomes. *Psychological research*, 72(3): 321–330.

Gershman, S. J.; and Daw, N. D. 2017. Reinforcement learning and episodic memory in humans and animals: an integrative framework. *Annual review of psychology*, 68: 101–128.

Gonzalez, C.; Lerch, J. F.; and Lebiere, C. 2003. Instance-based learning in dynamic decision making. *Cognitive Science*, 27(4): 591–635.

Huang, S.-c.; Jin, L.; and Zhang, Y. 2017. Step by step: Subgoals as a source of motivation. *Organizational Behavior and Human Decision Processes*, 141: 1–15.

Hughes, E.; Leibo, J. Z.; Phillips, M.; Tuyls, K.; Dueñez-Guzman, E.; García Castañeda, A.; Dunning, I.; Zhu, T.; McKee, K.; Koster, R.; et al. 2018. Inequity aversion improves cooperation in intertemporal social dilemmas. *Advances in neural information processing systems*, 31.

Kulkarni, T. D.; Narasimhan, K.; Saeedi, A.; and Tenenbaum, J. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29.

List, J. A. 2006. The behavioralist meets the market: Measuring social preferences and reputation effects in actual transactions. *Journal of political Economy*, 114(1): 1–37.

Nguyen, T. N.; McDonald, C.; and Gonzalez, C. 2023. Credit assignment: Challenges and opportunities in developing human-like ai agents. *arXiv preprint arXiv:2307.08171*.

Park, J. S.; O'Brien, J. C.; Cai, C. J.; Morris, M. R.; Liang, P.; and Bernstein, M. S. 2023. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Thomson, R.; Lebiere, C.; Anderson, J. R.; and Staszewski, J. 2015. A general instance-based learning framework for studying intuitive decision-making in a cognitive architecture. *Journal of Applied Research in Memory and Cognition*, 4(3): 180–190.

Walsh, M. M.; and Anderson, J. R. 2011a. Learning from delayed feedback: neural responses in temporal credit assignment. *Cognitive, Affective, & Behavioral Neuroscience*, 11: 131–143.

Walsh, M. M.; and Anderson, J. R. 2011b. Modulation of the feedback-related negativity by instruction and experience. *Proceedings of the National Academy of Sciences*, 108(47): 19048–19053.

Wu, Y.; Fan, Y.; Liang, P. P.; Azaria, A.; Li, Y.; and Mitchell, T. M. 2023a. Read and reap the rewards: Learning to play atari with the help of instruction manuals. *arXiv preprint arXiv:2302.04449*.

Wu, Y.; Min, S. Y.; Prabhumoye, S.; Bisk, Y.; Salakhutdinov, R.; Azaria, A.; Mitchell, T.; and Li, Y. 2023b. SPRING: GPT-4 Out-performs RL Algorithms by Studying Papers and Reasoning. *arXiv preprint arXiv:2305.15486*.

Zheng, Z.; Oh, J.; Hessel, M.; Xu, Z.; Kroiss, M.; Van Hasselt, H.; Silver, D.; and Singh, S. 2020. What can learned intrinsic rewards capture? In *International Conference on Machine Learning*, 11436–11446. PMLR.