Learning From Explanations Using Sentiment and Advice in RL

Samantha Krening, Brent Harrison, Karen M. Feigh, *Senior Member, IEEE*, Charles Lee Isbell, Jr., Mark Riedl, and Andrea Thomaz, *Member, IEEE*

***Abstract*—In order for robots to learn from people with no machine learning expertise, robots should learn from natural human instruction. Most machine learning techniques that incorporate explanations require people to use a limited vocabulary and provide state information, even if it is not intuitive. This paper discusses a software agent that learned to play the Mario Bros. game using explanations. Our goals to improve learning from explanations were twofold: 1) to filter explanations into advice and warnings and 2) to learn policies from sentences without state information. We used sentiment analysis to filter explanations into advice of what to do and warnings of what to avoid. We developed object-focused advice to represent what actions the agent should take when dealing with objects. A reinforcement learning agent used object-focused advice to learn policies that maximized its reward. After mitigating false negatives, using sentiment as a filter was approximately 85% accurate. object-focused advice performed better than when no advice was given, the agent learned where to apply the advice, and the agent could recover from adversarial advice. We also found the method of interaction should be designed to ease the cognitive load of the human teacher or the advice may be of poor quality.**

***Index Terms*—Advice, reinforcement learning (RL), sentiment.**

I. Introduction

A GOAL of interactive machine learning is to enable people to naturally and intuitively teach robots how to perform tasks. We cannot expect every person to become an expert in machine learning. If robots and intelligent agents could learn from natural human instruction, robotic behavior could be customized by end-users with no machine learning expertise.

While there are many ways people teach, including demonstrations and critique, this paper focuses on learning from explanations. Learning from natural language explanations can decrease the amount of time and effort required by a human teacher. Giving a few simple sentences is less work than demonstrating all possible situations or monitoring an agent to provide critique. Because people are naturally skilled at harsh dimensionality reduction, learning from language automatically builds human-agent interaction that plays to the strengths of both the human teacher and robotic student. Ideally, the person concisely tells the robot what is most important to pay attention to, and the robot uses its computational power to develop policies that maximize performance. Learning from explanations is helpful because people may not be able to provide demonstrations if they are elderly, injured, or the task is not safe for people to physically attempt. Additionally, a person does not have to be present to teach the robot—telecommunication or written instructions work just as well. Another attractive quality of learning from natural language explanations is it is generalizable across domains. Even if the domain changes, people will use the same language with the same meanings and structure to describe new tasks.

There are many challenges when learning from natural language explanations. People do not limit themselves to a domain-specific vocabulary—people make their own labels for objects and actions. Different people use different words for the same concept. In addition to describing things to do or avoid, explanations also tend to include information that is not actionable, like background knowledge. Many natural language explanations also do not include state information, which makes it very difficult to determine where and when to use the advice.

A primary problem addressed in this paper focuses on an area of learning from explanations that has been discussed little in previous research—how to learn from human explanations that lack state information. This paper contributes to this through the development of object-focused advice, a method in which human advice is tied to objects instead of specific states and is generalized over the object's state space. Consider this explanation from Mario: "Mario should jump on enemies." While this advice would easily be understood by a human student, it proves problematic for reinforcement learning (RL) agents. The teacher did not specify state information like, where the enemy needs to be with respect to Mario and what Mario's velocity should be. Knowing that Mario should jump on an enemy is valuable information, but how can an agent make use of it if no state information is provided?

Solving this challenge is worthwhile since people often describe tasks by talking about objects. The following is an explanation a person might give to describe how to play Mario that links an *object*, like an enemy, to an action that should be used around that object, like jumping on an enemy.

Manuscript received January 31, 2016; revised July 17, 2016 and September 29, 2016; accepted October 17, 2016. Date of publication November 14, 2016; date of current version March 9, 2017. This work was supported by ONR under Grant N000141410003.

S. Krening, B. Harrison, K. M. Feigh, C. L. Isbell, and M. Riedl are with the Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: skrening@gatech.edu).

A. Thomaz is with the Department of College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCDS.2016.2628365

We define this to be advice because it tells the agent what actions to take.

> The *goal* is to reach the end of the level to the right quickly. Mario should jump right on *enemies*. Mario should jump to collect *coins*, and jump over *chasms*.

The human teacher does not need to provide state information, like where the enemy is with respect to Mario. A person might also provide warnings in an explanation to teach the agent what actions to avoid.

> Do not fall into *chasms*.

Another problem addressed in this paper is enabling an agent to categorize each sentence in an explanation as *advice* of what to do or a *warning* of what not to do. Afterward, the advice and warnings are used to shape the agent's initial behavior. Autonomously categorizing sentences into advice and warnings allows us to use natural language explanations that are not formatted or restricted to a limited vocabulary. Most agents that attempt to learn from explanations require the input to be in a specific format, mainly advice of what actions to take in specific states. We aim to learn from natural language explanations that do not necessarily include state-specific information or follow a rigid format.

A contribution of this paper is to use sentiment analysis to filter natural language explanations into advice and warnings. Sentiment analysis, or opinion mining, is a way to computationally classify text into positive or negative opinions. This is a novel application since sentiment analysis has not traditionally been used to inform action selection. This paper provides a new understanding to how sentiment analysis can be utilized.

Using sentiment analysis as a filter allows people to explain tasks to agents without restricting the human teacher to a specific vocabulary or sentence structure. Consider the following explanations of how to deal with enemies in the Mario Brothers domain.

> It would be bad to walk right into an enemy. Jump on enemies.

Sentiment analysis classifies the first sentence as negative, so the agent is warned not to walk right when dealing with an enemy. The second sentence is classified as positive, so the agent treats jumping on enemies as advice of what to do. Sentiment classifications of "positive" and "negative" are used in a semantic sense, not syntactic.

We tested our sentiment filter and object-focused advice in a human-subject experiment conducted in the popular game domain Mario Brothers. After creating a method to mitigate false negatives, the sentiment filter was approximately 85% accurate. Three types of explanations were tested with an increasing level of structure and information provided to each participant. We found that the cognitive load of the explanation format adversely affected the quality of the advice. The results show that object-focused advice performs better than when no advice is given, the agent can learn where to apply the advice in the state space, and the agent can recover from adversarial advice. Also, providing warnings in addition to advice improved the agent's performance.

Section II contains related work about learning from human teachers, sentiment analysis, and RL. Section III develops object-focused advice. Section IV describes the human-subject experiment conducted in the Mario Brothers game, a popular domain for machine learning research. Readers unfamiliar with Mario are encouraged to read a description in Section IV-A. Section V includes the results and discussion. This paper concludes with Section VI.

II. Background and Related Work

Much of this paper is inspired by human development. RL is a form of machine learning influenced by behavioral psychology in which an agent learns what actions to take by receiving rewards or punishments from its environment [18], [21]. Skinner [19] wrote about "selection by consequences," comparing the evolution of living things through natural selection with the shaping of individual behavior through reinforcement. The probability people will repeat an action in a given circumstance is increased or decreased if they receive positive or negative reinforcement. Since one way people learn is by interacting with their environment, we chose to mirror this method when choosing a machine learning algorithm to teach our agent.

Deep learning is biologically inspired by the human brain. Deep learning models learn how to represent the input in increasing levels of abstraction. Using deep learning for sentiment analysis allows an agent to grasp the sentiment of words and sentences by looking at a large corpus of how people have used language. Classifying sentiment using deep learning is much closer to how the human brain interprets sentiment because it uses the structure and context within a sentence, not just words in isolation.

A. *Sentiment Analysis*

Sentiment analysis has been used to determine whether people think movies, books, music, consumer products, political campaigns, etc., are good or bad [16]. Much of the work in sentiment analysis has used a bag-of-words method in which each word in a document is scored. The accumulated score of the text determines if the document is classified as positive or negative. Since each word is scored separately, word order and context are ignored, which leads to less-accurate results. This paper uses Stanford's deep learning sentiment analysis software, which builds a representation of an entire sentence instead of looking at words independently [13].

Stanford's sentiment tool uses recursive neural tensor networks and the Stanford Sentiment Treebank [20]. The Stanford Sentiment Treebank is a corpus of fully-labeled parse trees based on the dataset of movie reviews from rottentomatoes.com [15].

B. *Reinforcement Learning*

RL is a form of machine learning in which agents learn what actions to take in situations by interacting with their environment—specifically, by receiving a signal of rewards and punishments [21]. This paper incorporates human advice into an RL agent.

Markov decision processes (MDPs) learn policies by mapping states to actions such that the agent's expected reward is maximized. An MDP is a tuple (S, A, T, R, γ) that describes S, the states of the domain; A, the actions the agent can take; T, the transition dynamics describing the probability that a new state will be reached given the current state and action; R, the reward earned by the agent; and γ, a discount factor in which $0 \le \gamma \le 1$.

Object-oriented MDP (OO-MDP) are an extension of MDPs. An OO-MDP is a model-based representation that uses a fixed-length feature vector of object relations [6]. For example, one feature in the Mario domain would be a binary relation indicating if an enemy is east of Mario at each time step. A drawback of OO-MDPs is all relations must be defined by a designer. Also, because the feature vector is fixed in length, the agent cannot adapt to new objects in the environment. Object-focused Q-learning (OF-Q) is a modified version of OO-MDPs.

OF-Q with an off-policy TD control Q-learning algorithm was used to train the Q-values for each object's policy. Unlike OO-MDPs, OF-Q is model-free and does not have a fixed-length feature vector [5]. The number of objects and relations in the feature vector can vary through time. Every object class has its own policy and reward signal. Since this paper focuses on the efficacy of using sentiment as a filter on natural language explanations, a well-understood tabular algorithm was used. In (1), s_{o_t} and a_t are the object's state and action chosen at time t, $Q(s_{o_t}, a_t)$ is the Q-value for a given object state and action, r is the reward received after carrying out action a_t, α is the learning parameter, and γ is the discount parameter for expected future rewards

$$Q\big(s_{o_t}, a_t\big) \leftarrow (1-\alpha)Q\big(s_{o_t}, a_t\big) + \alpha\Big[r + \gamma \max_a Q\big(s_{o_{t+1}}, a\big)\Big]. \quad (1)$$

RL agents must tradeoff between *exploring* and *exploiting*—whether to search for better policies or carry out what the agent has already learned. If the agent decides to exploit the policy, the action with the maximum Q-value over all objects in the state space is chosen since it is expected to yield the greatest reward

$$\pi(s) = \arg\max_a \max_o Q(s_o, a). \quad (2)$$

While natural language explanations can be incorporated into many machine learning algorithms, we used OF-Q to represent tasks for two main reasons [5]. First, OF-Q can directly use object-based human instruction, which improves the transparency between the human teacher and robotic learner. Second, object-based algorithms provide a method to solve high-dimensional state spaces, like Mario.

C. Learning From Human Teachers

Three commonly studied methods of human instruction are explanations, demonstrations, and critique [4]. Explanations transfer knowledge through language. Demonstrations provide ideal actions to take in situations. Critique is positive or negative feedback that informs students how good or bad their actions were.

Most machine learning methods that learn from human teachers force people to provide state-specific information. That level of detail is often not intuitive or natural, and precludes the possibility of learning from natural language that lacks state information. When an agent learns from demonstrations, the agent learns a mapping from states to actions [3]. An agent that learns from demonstrations is affected by how much of the state space was explored in the demonstrations and how well the person performed. Paired states and actions from demonstrations can also be used with apprenticeship and inverse RL to approximate the reward function the teacher was following [1]. Critique is linked to the current state by informing an agent how good or bad its actions were, which affects the probability the action will be taken in the same state in the future [7]. Critique can also be used directly as a reward signal to tell the agent how positive or negative its actions were in certain states [4]. All of these approaches link human input to specific states.

Various forms of advice have been developed in other work, including linking one condition to each action [12], and linking a condition to rewards [11]. Several connect conditions to higher-level actions that are defined by the researcher instead of primitive actions [8], [10], [12]. Reference [2] creates policies using demonstrations and advice. Reference [14] parses language into a graphical representation and finally to primitive actions. Reference [12] has the person provide a relative preference of actions, whereas the agent determines the order of preferred actions in our work. Reference [17] explored learning multiple interpretations of instructions. Similar to this paper, the advice in [2] does not require people to give specific numbers for continuous state variables, but uses a set of predefined advice operators. The advice developed here links one action to each object. This allows each action to be used multiple times in one domain; for example, in the Mario domain the agent may be advised to jump to the right quickly for chasms and enemies. Extracting advice concerning primitive actions allows the researcher to include less domain-specific knowledge in the agent.

Many researchers incorporate advice using IF-THEN rules and formal command languages [10], [12]; if the state meets a condition, then the learner takes the advice into account. Formal command languages and IF-THEN rules require advice that is state specific and contains numbers. "When the agent is within 10 m of this object, do this action." Developing a parser is labor intensive, and prior knowledge like distance calculations must be encoded. This paper is different because the advice is object specific, the agent learns which part of the state space the advice applies, and a person does not need to provide numbers. This allows an agent to learn from a few simple sentences that nonexperts can provide. "Jump when Mario encounters a chasm." No state-specific information is provided by the person, like: where is the chasm with respect to Mario? How far away from the chasm should Mario jump? What should Mario's speed be near the chasm?

Most methods are permanently influenced by the advice. Reference [10] can adjust for bad advice by learning biased function approximation values that negate the advice. Reference [12] uses a penalty for not following the advice that decreases with experience. This paper differs because the advice is followed a set number of times for each object and each state in addition to exploration. After the advice is followed a set number of times, the exploitation action selections are based entirely on experience, and advice is no longer considered by the agent. If it was good advice, it will be reflected in the Q-values and will continue to be the policy.

III. Object-Focused Human Advice

Object-focused advice ties actions to objects instead of specific states and generalizes the advice over the object's state space [9]. Before the agent starts learning, a person instructs the agent what action to take when dealing with an object. For example, in the Mario Bros. game, a person could advise the agent to *jump right* (action) when encountering a *coin* (object). The person does not need to specify state information, like where a coin needs to be with respect to Mario, in order to take the advised action. The agent will take the action the human advised a specified number of times, regardless of its experience, before following normal exploitation. The job of the sentiment filter is to tell the agent whether a sentence should be treated as advice or a warning.

It is likely this type of general, object-focused advice will only apply to some subset of the state space. The advice of jumping right will gain a reward if the coin is to the right of Mario, but will not work if the coin is to Mario's left. The agent determines the applicable parts of the state space and how good the advice is through experience. Following human advice occurs only during exploitation and does not interfere with exploration. Another way of thinking about this is following advice initially supplants exploitation with a form of exploration directed by a human. This is separate from, and in addition to, ϵ-greedy exploration. Since ϵ-greedy exploration is used, object-focused advice has the convergence properties of ϵ-greedy exploration.

Using object-focused advice that is independent of the object's state allows the person to perform object-level generalization and abstraction instead of the agent. Generalization is a vital part of induction because it is a way to extend the knowledge learned from one particular example to many others. Generalizing over the entire state space of an object may seem drastic, but it is a way to quickly operationalize and learn from human explanations without state information. It is unrealistic to expect people to provide detailed state information when giving advice. A person might say, "jump on the enemy," but will not say, "Hold the jump key for 10 frames when Mario is within 2.5 horizontal blocks of an enemy with a velocity of 3.2 units/frame." The agent will take the action advice of jump on the enemy, and determine to which portions of the state space, if any, the advice applies.

A. Advice

The first step is to get advice from a person that describes what actions the agent should take. Advice is given to the algorithm as two lists: one containing the objects and the other the advised actions. Advice can be provided for as many or as few objects in the state space as a person decides. If the agent encounters an object for which advice was not given, the policy is initialized without advice and the agent learns from exploration and experience.

Next, an object policy must be created for each object a person gave advice for. If a person advises the agent to jump when dealing with coins, an empty object policy table is created that sets the advised action to "jump." Whenever a new object state is encountered (like the first time Mario sees a coin to the northeast), a new state entry is made in the coin's policy table with specific state values like the x- and y-positions of the coin with respect to Mario. This new state entry includes a value that counts the number of times the advice has been followed as well as a threshold number of times the advice should be followed. This is what allows the agent to determine which part of the state space the advice applies—it tries the advice a set number of times everywhere in the state space, and the resulting Q-values reflect whether the advice is good or bad in that region of the object's state space. For example, the first 25 times Mario sees a coin to the *northeast*, Mario would follow the advice to jump and update the Q-values based on the earned reward. The first 25 times Mario sees a coin to the *southwest*, Mario would also follow the advice to jump.

To include object-focused advice in the OF-Q algorithm, an extra Q-value was created that corresponds to advice, not a specific action. This indicator Q-value is initialized to a value much larger than any reward the agent could achieve in the state space. During action selection in (2), this indicator Q-value forces the policy to choose the advice. While the advice is followed, the Q-values that correspond to each action are updated as expected. The indicator Q-value is never updated, nor does it affect the outcome of the Q updates in (1). After the advice has been followed some set number of times, the indicator Q-value is removed and the policy chooses exploitation actions based on experience.

For every time step in game execution, the agent must choose an action (Algorithm 1). First, object recognition is used to determine which objects are currently in the state. Reward allocation from the last time step is completed so the reward is applied to the proper objects' policies. Then, ϵ-greedy exploration is utilized. During exploitation, if advice has been followed for an object less than a set number of times, the large indicator Q-value will force the advised action to be chosen. ϵ is exponentially decayed at the end of each level.

Two interesting aspects of object-focused advice are its ability to recover from adversarial advice and its variable "trust" in a person. Following advice a set number of times and then relying on experience allows the agent to recover from adversarial advice, which is antagonistic input that instructs the agent to take an action expected to result in the least reward (greatest punishment). An example of adversarial advice in the Mario domain is standing still while an enemy approaches. Also, object-focused advice lets the agent's trust in the human vary across the domain by treating each piece of advice without prejudice; if a person provides one piece of good advice along with eight pieces of bad advice, the agent will use its experience to build policies that reflect the good and ignore the bad.

Algorithm 1 Get Action

```
1:  function GETACTION($reward$, $environment$)
2:      $objects = getObjectsInStateSpace(environment)$
3:      for each $object \in objectsOld$ do
4:          Reward allocation: update Q values
5:          If advice followed, increment $timesAdviceTried$
6:      end for
7:      for each $object \in objects$ do
8:          If this object has never been seen by the agent,
9:          create a new object policy
10:         If this object has never been seen in this state,
11:         add a state entry to the object's policy
12:     end for
13:     if $rand(0, 1) > \epsilon$ then                    ▷ Exploit
14:         Initialize $action$ and $Q_{max} \leftarrow -\infty$
15:         for each $object \in objects$ do
16:             $q_i \leftarrow max_i(Q(object.state, a_i))$
17:             if $q_i > Q_{max}$ then
18:                 $Q_{max} \leftarrow q_i$, $action = a_i$
19:             end if
20:         end for
21:     else                                               ▷ Explore
22:         $action = random\{actions\}$
23:     end if
24:     $objectsOld = objects$
25: end function
```

B. Warnings and Multiple Objects

Advice describes what to do, while warnings describe what not to do. Similar to advice, warnings of what not to do are incorporated by using an indicator Q-value. Instead of a large positive value, a large negative indicator value is used. object-focused advice, as previously described, chooses an action by looking at each object separately. To incorporate warnings, all objects in the state space are taken into account together by summing up the Q-values associated with each action across all objects. Choosing an action by taking multiple objects into account allows us to get an idea of the overall severity of each action.

Consider the case of both a coin and enemy in the state space shown in Tables I and II. Assume a person gave advice to move *right* for coins and *jump right* for enemies, and warned the agent to not move *left* for coins or walk *right* for enemies. If each object is considered separately and no warnings are used, the agent may follow the advice for coins to move right, which would cause Mario to walk into an enemy and be injured. This is solved by considering all objects in the state space and including warnings about which actions to avoid.

Table I shows how multiple objects are considered by summing Q-values across all objects in the state space. The indicator Q-values for advised actions are $+2000$, warnings are -2000, and the default initial value when no information is given is 0. In this example, the initial Q-values will result in the agent choosing to jump right since it has the maximum Q-value in the *total* row. Moving right has a summed Q-value of zero because the action was advised for coins but warned against for enemies ($2000 - 2000 = 0$). Moving left has the worst summed Q-value because it was warned against for coins. Combining multiple objects still produces a ranked preference of actions; jumping right ($Q = 2000$) is better than walking right ($Q = 0$), which is in turn better than moving left ($Q = -2000$). Initially, the agent does not have a sense of severity; it does not know that injuring Mario is much worse than missing a coin. This is reflected in the learned Q-values.

TABLE I
EXAMPLE OF MULTIPLE OBJECTS AND WARNINGS. INITIAL Q-VALUES WITH INDICATORS

Object	JumpRight	Right	Left
coin	0	2,000	-2,000
enemy	2,000	-2,000	0
total	**2,000**	0	*-2,000*

TABLE II
EXAMPLE OF MULTIPLE OBJECTS AND WARNINGS. LEARNED Q-VALUES

Object	JumpRight	Right	Left
coin	10	10	-1
enemy	50	-50	0
total	**60**	*-40*	-1

Table II shows the learned Q-values for the same example. Eventually, the indicator Q-values will not be added in and the agent will rely on experience. Jumping right has the largest summed Q-value ($Q = 10 + 50 = 60$). Notice that the agent expects moving left ($Q = -1 + 0 = -1$) to be better than moving right ($Q = 10 - 50 = -40$), which is not the same order as the initial action preferences. Even if the agent moves right and collects a coin, it will run into an enemy and be injured; moving left results in a small Q-value hit.

Summing an action's Q-values across multiple objects allows the agent to learn the importance and severity of all given advice and warnings. In Table II, the *total* row shows the agent has learned that the warning to avoid walking right into an enemy is much more severe than walking left near a coin.

IV. EXPERIMENTAL METHOD

The main goals of the human-subject experiment were to determine if sentiment analysis could be used as a natural language filter to inform action selection and assess the performance of object-focused advice.

The experiment had four phases: 1) familiarization; 2) free-form explanations; 3) structured explanations; and 4) a fill-in-the-blank survey. In post-processing, the natural language explanations were filtered through a sentiment analysis to determine if each sentence was advice of what to do or a warning of what not to do. Once advice and warnings were in the form or linking an object to an action (OF-advice), an agent was trained using the advice to shape its initial action selection. This process is shown in Fig. 1. The cumulative reward for each object was analyzed to evaluate the agent's performance over 500 trials.

A. Mario Domain

The experiment was conducted using the Mario Bros. platform from the 2009 Mario AI Competition [22], as seen

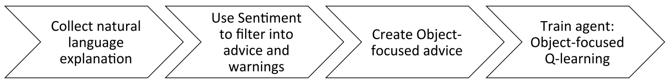
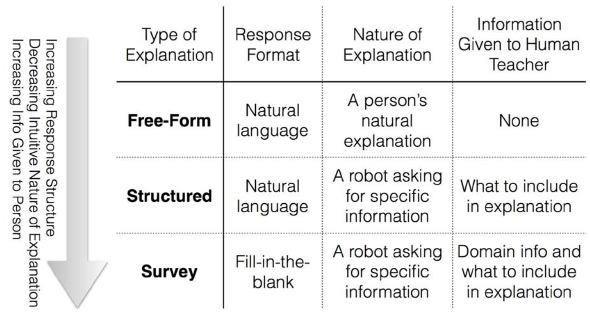

Collect natural language explanation
Use Sentiment to filter into advice and warnings
Create Object-focused advice
Train agent: Object-focused Q-learning

Fig. 1. Work flow chart.

Fig. 2. Participant's view of the Mario Bros. game.

in Fig. 2. It is a partially-observable environment in which Mario must collect rewards and avoid being harmed or killed while moving toward the goal to the right. Mario wins a level by reaching the goal, and loses by running out of time, falling into a chasm, or being repeatedly injured by an enemy. The primitive actions are right, left, jump, and speed. Multiple actions can be used at once. Momentum is incorporated into Mario's dynamics, so when some keys are held down, the results are different than pressing a key once. Before each level begins, the Mario platform generates the level using several parameters, including an integer that represents the level's difficulty. The difficulty determines the number and types of obstacles and enemies in the level. For example, a chasm is too difficult to appear in levels with a difficulty of zero.

The agent defines objects in a generalized way, which enables the agent adapt to new objects in the environment. Each object's state includes *x*- and *y*-positions with respect to Mario's location as well as an integer code from the Mario environment that indicates the type of object (coin, Goomba, etc.). The Mario Bros. platform provides a 22×22 grid of integers at every time step that shows the environmental objects surrounding Mario, who appears at the center of each grid. The platform also provides an integer code for each visible enemy as well as the continuous *x*- and *y*-positions with respect to Mario. A subset of the available information was used in the representation. Environmental objects like coins were included in the state space if they appeared in the 3×3 grid directly surrounding Mario's location, while enemies were included if they appeared anywhere on the screen. This allowed us to determine how advice performed both with a reflexive agent that looked directly around Mario and policies that looked at the whole screen. The only value of Mario's state that was included by the agent was whether Mario could jump at each time step. Representing Mario in this manner creates a state space of approximately 10^{25} states.


Increasing Response Structure
Decreasing Intuitive Nature of Explanation
Increasing Info Given to Person

Type of Explanation	Response Format	Nature of Explanation	Information Given to Human Teacher
Free-Form	Natural language	A person's natural explanation	None
Structured	Natural language	A robot asking for specific information	What to include in explanation
Survey	Fill-in-the-blank	A robot asking for specific information	Domain info and what to include in explanation

Fig. 3. Explanations.

B. Familiarization

In the first step of the experimental protocol, participants played Mario until they were comfortable and had at least played one level each at a difficulty of 0, 1, and 2. This ensured each person saw the same objects before providing advice.

C. Human Explanations

Explanations were collected from human participants in three successive trials, in order of increasing structure and provided information, as seen in Fig. 3. Taking explanations in the order of increasing structure and provided information allowed the most natural and intuitive human feedback at each step. Details of each type of explanation is provided in the following sections.

1) Free-Form Explanations: The free-form explanation was collected first since no information was given to the human teacher—this means the participant's explanation was not tainted by vocabulary, expected information, or a structure imposed by the researcher. The researcher asked the participant the following question.

> "Imagine I know nothing about how to play Mario. Can you explain to me how to play Mario?"

No guidance or instruction was provided to the participant indicating how to respond. The participants gave their explanations in natural language, and the explanations were collected as audio recordings.

2) Structured Explanations: For the second explanation, the participants were prompted to provide certain information. At this point, the responses were still in natural language. The participants could ignore, loosely follow, or attempt to fulfill the prompt. This type of structured explanation gives an approximation of a robotic agent asking an end-user to provide object-focused advice. The researcher asked participants the following question.

> "There are many objects in Mario like coins and different types of enemies. For each object, can you provide one action that you would advise someone to use when dealing with that object? Try to fill in the blank: When Mario encounters *an object*, he should *do this action*."

Lists of objects and actions were not provided to the participants—they spoke about what they remembered in natural language, including their own labels for objects and actions. The researcher collected audio recordings of the responses.

3) Survey Explanations: Finally, participants provided advice by completing a fill-in-the-blank survey. The participants were given a list of objects and actions, including pictures of the objects, and were asked to provide one advised action for each object. Participants were told they did not have to provide advice for every listed object—only the objects they thought were important. The advice was used to train agents offline; the results are in the following section.

D. Post-Processing

1) Sentiment Analysis as Filter: We used a sentiment analysis to filter natural language explanations into advice and warnings. Sentences classified with positive or neutral sentiment are considered to be advice of actions to take. Sentences with negative sentiment are treated as warnings of actions to avoid. Detailed results of the sentiment filter are found in Section V-B.

2) Object-Focused Advice and Warnings: Once each sentence in an explanation has been filtered into positive or neutral sentiment (advice) or negative sentiment (a warning), the explanation can be converted into object-focused advice. If a sentence is classified as advice or a warning and contains an object and action, the paired object and action are added to lists containing either the advice or warnings. Multiple actions can be associated with each object. To test the machine learning performance, the survey data were used in which the grounding from language to object/action was provided. Further work on grounding and ambiguity are out of the scope of our research questions.

3) Object-Focused Q-Learning: Once the object-focused advice was created, it was used to initialize the OF-Q agent. Each agent was trained using the object-focused advice and OF-Q algorithms discussed in Sections II and III. The author of this paper provided the adversarial advice, which is advice meant to minimize the agent's performance. An agent using no advice was used as a baseline for comparison.

The results were averaged over 100 trials. A sliding window average with a width of 25 trials was used. The parameters used were $\alpha = 0.1$, $\gamma = 0.95$, $\epsilon_0 = 0.8$, and $\epsilon_{\min} = 0.15$. ϵ-greedy exploration was used.

V. Results and Discussion

The experiment had five participants who provided object-focused advice; one agent was trained per participant. The author of this paper provided the adversarial advice. An agent using no advice was used as a baseline for comparison. The labels of "good," "mediocre," and "bad" were applied to participants' advice by looking at the learned policy's cumulative reward. The participants were asked to give the best advice they could, but some resulted in better or worse policies.

In OF-Q, each object class has its own reward function. Therefore, in addition to analyzing the total cumulative reward for the entire state space, we evaluate the reward for each object's state space. Policies are learned for each object.

In the following sections, we will discuss the nature of the explanations, show the accuracy of the sentiment filter, and discuss the agent's performance using object-focused advice.

A. Observations on the Nature of Explanations

The natural language explanations from participants were varied in many ways, including the amount of prior knowledge the agent was assumed to have, the level of detail provided, and whether primitive or higher-level actions were described. However, the similarities across explanations were intriguing. All of the participants spoke in terms of objects, not state space variables; none of the participants gave numbers to specify particulars of the state like velocity and distance, supporting the claim that it is useful to be able to learn from explanations that are not state-specific.

1) Information People Did Not Provide: The most striking observation from the natural language explanations was none of the participants provided any numbers to specify distance, relative position, velocity, etc. This reinforces the idea that it is useful for an agent to be able to learn from explanations that do not contain specific state information.

If an object or situation is considered too easy and obvious to deal with, people tend not to mention it in their explanations. Almost no one described how Mario should deal with steps in the natural language explanations.

2) Extra Information People Provided: For the survey, each participant provided advice for every available object, even though they were told they did not have to (and even if they had not encountered the object during the familiarization phase).

Several participants gave visual descriptions of how to identify what objects they were talking about—how to link their labels to objects. "You are a guy in red clothes." "Enemies look like people walking around." "A pit is when there is no floor to support you." It would be interesting to use this type of explanation, but the agent would need to start with much more background knowledge.

Some participants described a sequence of actions when dealing with objects. They wanted to advise Mario to speed up and then jump over a chasm, or go under and then jump to hit a brick. The advice developed here is a simple link from one object to one action—it cannot currently take full advantage of the nuances of natural language explanations.

Many participants provided advice from their prior knowledge of similar domains. The most common was advice explaining how to use tunnels to reach secret levels, which was not possible in the experiment's version of Mario, and was therefore never seen in the familiarization phase. One participant assumed the agent would know about right-scrolling games, and would apply that knowledge to Mario.

Most participants assumed the actions belong to the domain, not the agent. A couple of participants explained the effect of each key—each primitive action of the game. It was not assumed that these were the student's primitive actions, but rather actions the student would need to learn. If a teacher were to explain math operators like addition and multiplication, she would teach how the operators work; the operators would exist in the math domain, not the student's natural, inborn set of actions.

3) Differences in Experience: The amount of prior knowledge the agent was assumed to have varied drastically across participants. The participant with the least video game experience provided the most details, including giving advice in terms of primitive actions. The participant with the most video game experience provided the fewest details and assumed the agent had much prior knowledge, including which actions were available and what each action accomplished.

The least experienced participant often provided a piece of action advice followed immediately by the corresponding primitive action. This led to an interesting error: the natural language explanation was correct, but the given primitive action key was wrong. It is easy to accidentally say the "*s*" key instead of the "*a*" key when little meaning is associated with *s* and *a*. It is much harder to mistake the word jump for "fireball" when speaking. Enabling agents to learn from natural language explanations may reduce errors compared to attempting to make normal humans speak "computer." Agents will be able to learn from many more sources in more environments if people do not have to change their natural teaching methods.

4) Generalizations: In natural language explanations, participants tended to generalize behavior across objects by suggesting the same policy for many objects. Some generalizations were, "enemies," "obstacles," and "things coming at you." People are good at generalization. It is powerful if an agent can take a few generalized sentences and extract initial policies for many objects.

In the free-form and structured explanations, participants often discussed actions like "jump on" and "get," as in "jump on an enemy" or "get the coin." These actions are a higher-level of abstraction than the primitive actions, in which a person would have to choose between "jump right" and "jump left." It may be better to allow people to specify higher-order actions that naturally generalize across the state space instead of making them choose a primitive action. Mario can get the coin if it is anywhere on the screen by decreasing the distance between Mario and the coin; Mario can get a coin by jumping to the right if the coin is to the right of Mario. The advice developed here used primitive actions. The explanations of jump on and get might also imply a planner could be helpful instead of defining higher-level actions in the future. If the effects of the primitive actions were known, a planner could specify a sequence of actions necessary for Mario to jump such that it landed on an enemy's head or navigated to a coin's location. Alternatively, it would help to consider the possible actions in the domain as separate from the agent's actions.

5) Human-Agent Interaction: The experimenter let the participants continue speaking until they were done. They seemed to expect to be cut off or given some sort of feedback or indication that they had explained adequately and enough. Explanations died off awkwardly and uncertainly. If they were explaining directly to a robot, feedback, transparency, and gestures could help tell the teacher that the knowledge is understood, the teacher can continue, finish, or change explanation style.

Many participants provided reasons for actions, as if they needed to explain the meaning of actions to the agent or convince the agent why an action should be taken. "Jump on an enemy so you do not lose a life." Future work with this RL agent may try to convert reasons for actions into reward information—losing a life is bad, which should give the agent a negative reward, so the agent should jump on an enemy to avoid a negative reward.

B. Sentiment Analysis

One of our goals was to use sentiment analysis to filter each sentence from a natural language explanation into either advice (what to do) or warnings (what not to do). We started by classifying entire sentences as either positive/neutral or negative.

We found that positive and neutral classifications were accurate, but false negatives were a significant problem. For the free-form explanations, the sentiment analysis correctly classified approximately 86% of positive and neutral sentences. However, only 47% of sentences classified as negative are truly negative (describing warnings of what not to do). Approximately half of the sentences classified as negative are false negatives. For the structured explanations, 95% of the positive and neutral sentences were correctly classified, but 84% of the negative classifications were false negatives.

Approximately half of the free-form sentences were classified as positive and neutral while the other half were negative. Less than half of the structured explanations were classified as positive or neutral. This is interesting because before each participant gave a structured explanation they were prompted to give positive advice: "If Mario encounters *an object*, he should *do this action*." If people conformed to the prompted format to give positive advice and the sentiment classification were perfect, we would expect 100% of the structured explanations to be classified as positive or neutral. Surprisingly, people conformed to providing positive advice quite well since only 4/44 sentences were truly warnings; however, people were not as good at providing one action for an object, often providing sequences of actions.

If a warning like "Do not walk right into an enemy" is misclassified as advice, the agent will walk right whenever an enemy is in its state space, which will injure or kill Mario. The agent's initial behavior will be the opposite of what the human teacher intended. After the advice is followed a threshold number of times, the agent will rely on its experience and avoid walking right into an enemy. If advice like jump on the enemy is misclassified as a warning by the sentiment filter, the agent will avoid jumping when an enemy is in its state space, so its initial behavior will not be what the human teacher intended. The best sentiment tools are approximately 85% accurate, so there will be misclassifications. While we would prefer a perfectly accurate sentiment filter, a misclassification is not disastrous because it can be valuable for an agent to learn what not to do early so it does not repeat its mistakes in the long term.

One reason false negatives are likely to occur is people include consequences or reasoning in their explanations. The following sentence from a participant was classified as a warning (negative) even though it was meant as advice of what to do. *If you see a shell shooting at you, jump to avoid it.*

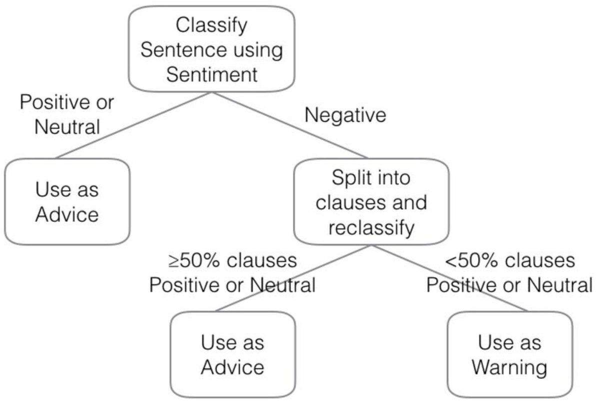

Classify Sentence using Sentiment
Positive or Neutral
Negative
Use as Advice
Split into clauses and reclassify
≥50% clauses Positive or Neutral
<50% clauses Positive or Neutral
Use as Advice
Use as Warning

Fig. 4. Sentiment classification decision tree.

TABLE III
FREE-FORM EXPLANATIONS AFTER RECLASSIFICATION

True Sentiment	Classified Positive or Neutral	Classified Negative
Positive or Neutral	21 (78%)	6 (22%)
Negative	0 (0%)	3 (100%)

False negatives are also likely to occur when an object or action associated with a negative sentiment is included in an explanation. The following sentence is classified as negative even though it describes what actions the agent should take. *There are holes in the ground you should jump over.*

Positive and neutral classifications are quite accurate, but negative classifications should not be trusted without further processing. To correct the false negatives, we split each sentence into clauses and determined the sentiment of each clause. If at least half the clauses were positive/neutral, we reclassified the sentence as positive. Consider the examples from the previous two paragraphs. After being split into two clauses, *If you see a shell shooting at you,* is neutral; the clause, *jump to avoid it,* is negative. Similarly, *There are holes in the ground,* is negative, but *you should jump over,* is neutral. Both false negatives can now be reclassified correctly as advice of what to do.

The reclassification decision tree is shown in Fig. 4. If a sentence is classified as positive, it is used as advice of what actions to take. If a sentence is classified as negative, the sentence is split into clauses; each clause is classified as positive or negative. If 50% or more of the clauses are positive, the sentence is reclassified as advice of what to do. If the sentence is still classified as negative, it is used as a warning of what not to do.

Tables III and IV show the results of reclassifying sentences with negative sentiment. By splitting negative sentences from free-form explanations into clauses and reclassifying, 78% of the false negatives were correctly reclassified as positive. All of the sentences that remained negative were correctly classified. For structured explanations, splitting negative sentences into clauses and reclassifying caused 86% of false negatives to be correctly classified as positive. Two out of the three sentences that remained negative were false positives.

TABLE IV
STRUCTURED EXPLANATIONS AFTER RECLASSIFICATION

True Sentiment	Classified Positive or Neutral	Classified Negative
Positive or Neutral	19 (86%)	3 (14%)
Negative	2 (67%)	1 (33%)

Reclassifying sentences with negative sentiment by splitting each sentence into clauses increased the overall accuracy of classification from 56% to 83% for the free-form explanations. Similarly, the accuracy of the structured explanation classifications was improved from 50% to 86% by reclassifying sentences with negative sentiment.

Another approach to reducing false negatives would be to retrain the sentiment model on language specific to the desired domain. Games are generally violent. Mario's lexicon includes killing, chasms, enemies, impalement, fireballs, and shooting—not activities or objects thought of as positive in the mainstream English language.

From a traditional machine learning perspective, since the only concern is the agent's performance in a particular domain, the language model should be retrained for the domain. The data used to train the model should be representative of the data the agent will encounter in the future. However, from a human-agent interaction perspective, the answer is not as clear. Should we think of the future data as commands in a domain-specific lexicon or as language people might use? A goal of interactive machine learning is to bring the algorithm to the person instead of forcing the person to come to the algorithm. If the model is retrained for a specific domain and a person is required to speak in a limited, domain-specific vocabulary, the person's natural behavior is altered to make the algorithm work. If the model is trained on all of mainstream English and a person is allowed to say anything, the person is able to teach an agent using a more natural behavior. People are unlikely to limit themselves to a specific lexicon—they will use words they are familiar with, so it is beneficial for the sentiment model to have an understanding of the mainstream use of the language. Also, the words people choose inform what they think of situations. *The monster is chasing me* is negative, but *the boy is chasing me* is neutral.

C. Object-Focused Advice

The following sections discuss the performance of object-focused advice. First, we discuss how the quality of advice varied given different explanation formats. Then, we show the cumulative reward earned over an object's entire state space, how the agent learns where the advice applies after generalizing over the object's state, and then look closer at one particular subset of an object's state space.

1) Advice From Explanation Formats: Fig. 5 shows the object-focused advice and warnings for each participant and each form of explanation. There are many items worth noting, including the amount of actionable advice for each explanation type and the quality of the advice.

The amount of actionable advice increased with the structure of the explanation format. It is expected that free-form

Object		Coin	Ground	Tunnel	Brick	Goomba	Winged Goomba	Red Koopa	Green Koopa	Bullet Bill	Spiky	Enemy Flower	Shell	Chasm	Goal
Freeform	P1	collect		jumpOver	jump	jump, fireball	jump, fireball	jump, fireball	jump, fireball	jump, fireball	jump, fireball	jump, fireball	?		right
	P2	collect			hitBottom			jump, don'tRunInto	jump, don'tRunInto						
	P3	collect				jumpOver	jumpOver	jumpOver	jumpOver	jumpOver	jumpOver	jumpOver	?		right
	P4				collect, jump	jump	jump	jump	jump	jump	jump	jump	?		
	P5														
Structured	P1			jump	jump	jump	jump	jump	jump					jump,JSOver	jump
	P2	runInto		jump	jump	jump	jump	jump	jump	jump	jump	jump	landOn,jump		
	P3	walkThrough		jump		jump	jump	jump	jump	jump	jump	jump	?		jump
	P4			jump	jump	jump	jump	jump	jump	jump	jump	jump	jump	jumpOver	
	P5				jump			jump	jump			waitProceedCautiously		jump	
Survey	P1	SpeedRight	SpeedRight	Jump	Jump	JumpSpeed	JRS	JumpSpeed	JumpSpeed	Still	JRS	JRS	Jump	JRS	JRS
	P2	Right	Right	JumpRight	Jump	Jump	Still	Jump	Jump	Still	Jump	Still	Jump	JRS	SpeedRight
	P3	Right	Right	JumpRight	JumpSpeed	JumpRight	Still	JumpRight	JumpRight	Still	JumpRight	JRS	JumpRight	JRS	JumpRight
	P4	Right	Right	JumpRight	Jump	JumpRight	Still	JumpRight	JumpRight	Still	JRS	JRS	JumpRight	JRS	Still
	P5	JumpRight	Right	JumpRight	Jump	JumpSpeed	JumpSpeed	JumpRight	JumpRight	JRS	JumpRight	JRS	JumpRight	JRS	Jump

Fig. 5. Object-focused advice for each explanation type. Note that the responses for the free-form and survey responses are varied, while almost all of the structured responses are to *jump*. The poor performance of the structured responses is likely due to the increased cognitive load of that explanation format. Warnings are shown in red and underlined. P#=Participant#. JRS=JumpRightSpeed. JS=JumpSpeed. The question marks indicate the participant did not specify if shells were considered enemies.

explanations will contain fewer actionable sentences since the sentences can contain any information in any format. The structured explanations prompt teachers to link objects to actions, so more sentences are expected to contain actionable advice. Every survey entry will be actionable since teachers can only choose from a list of actions for each object. Each participant provided advice for every object in the survey explanation, even though it was not required and they did not see all of the objects during the familiarization phase. Even though chasms were the leading cause of death in Mario, no one provided advice about chasms in the free-form explanations, three people did in the structured explanations, and everyone did in the survey.

Something very interesting happened with the structured explanations—while the number of actionable sentences increased from 19 to 27 compared to free-form explanations, the quality and variation of the advice decreased as seen in Fig. 5. In the free-form explanations, the advice had a somewhat varied vocabulary including *collect, jump, right, hitBottom, fireball, do not run into, and jump over*. For the structured explanations, almost every piece of advice was *jump*. In Mario, jumping vertically with no horizontal velocity will eventually lead to losing the level. For the survey explanations, the variation in advice increased again.

The poor advice from the structured explanations is likely due to the increased cognitive load of the explanation format. Free-form explanations do not force people to provide specific content or formulate an answer in a particular format. People focus entirely on what to say, not how to say it. Structured explanations, while still in natural language, prompt people to provide specific content in a certain way. Now, people have to focus on not just what to say, but how to say it. Participants did a fairly good job of providing content in the desired format, as evidenced by the increase in actionable advice. However, the extra work to formulate their responses led to mostly worthless advice. Having to extemporaneously create a natural language response in a certain structure was too difficult to yield worthwhile results, even in a game domain. For the survey explanations, the cognitive load was less compared to the structured explanations. Domain information including pictures and labels for objects and actions were provided to participants. They did not have

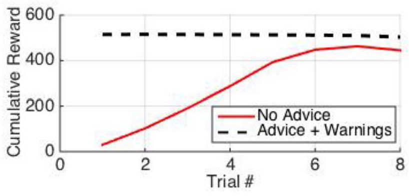


Fig. 6. Cumulative reward from survey.

to remember domain information or format responses; they simply had to fill in as many blanks as they chose. If robots ask people for information, the amount of information given to the person and the method of response should not impose a high cognitive load or the person's response may be of poor quality.

2) Total Cumulative Reward: Fig. 6 is included for completeness and shows the total cumulative reward earned by an agent with and without advice. The agent with advice is able to achieve better performance immediately.

3) Performance Over Object's Entire State Space: Fig. 7 compares the performance of participants' advice for chasms with adversarial and no advice. Good advice led to an agent with much better performance than adversarial or no advice. An agent trained with adversarial advice quickly recovers and performs as well as no advice, but not as well as good advice. After 400 trials of learning, the best advice from the experiment led to Mario falling into chasms approximately 16% of the time, while the agents using adversarial or no advice fell into chasms 34% of occurrences. Chasms are difficult for the reflexive state representation that looks at the 3×3 grid surrounding Mario. Mario's velocity and whether he is in the air are not part of the state representation. This leads to state aliasing when learning policies for chasms. The policy cannot tell if Mario is approaching the chasm quickly or slowly, which changes the likelihood a given action will succeed.

It is possible for a participant to provide poor advice for one object but good advice for another. The agent treats each piece of advice without prejudice. Even if a participant gave bad advice for chasms, the agent would not discount the rest of the advice given by the same person.

4) Object-Level Generalization (Learning Where Advice Applies): Fig. 8 shows the agent learns to which part of the

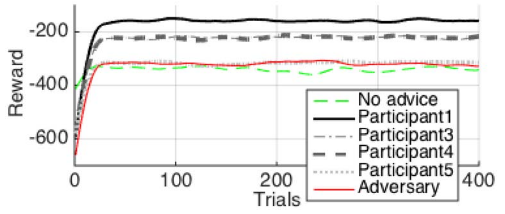


Fig. 7. Reward for chasms from survey.

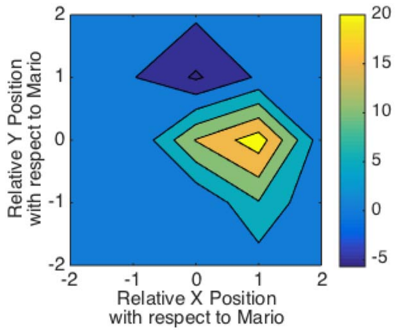


Fig. 8. Visualizing object-level generalization in a policy for Goombas from survey. The color scale represents Q-values showing when to jump quickly to the right.

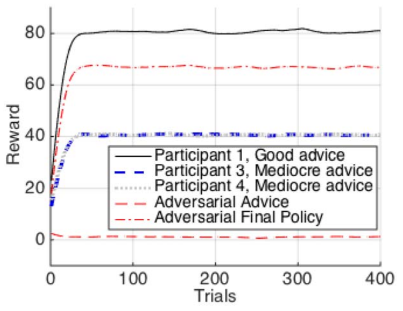


Fig. 9. Comparing the reward of good, mediocre, adversarial, and no advice when a coin is northeast of Mario from survey.

state space the advice applies. The agent was advised to jump to the right quickly when encountering Goombas. The agent learned this was a good policy when the Goomba was to the right of Mario in a "goldilocks" zone—not too close but not too far away. The agent learned jumping to the right quickly was bad advice when the Goomba was directly above Mario, because he would become injured when his head ran into the bottom of the Goomba. Using peoples' advice as object-level generalization allows the agent to quickly generalize a policy to relevant areas of the state space that would be difficult to learn about via exploration.

5) Performance in Specific Subset of Object's State Space: Now that we have seen the performance of policies across the entire state space of an object and how generalization over the state works, let us review the performance in one specific subset of an object's state space. Fig. 9 shows the results when a coin is northeast of Mario.

It can be more difficult to recover from mediocre advice than adversarial (Fig. 9). With adversarial advice, the agent recognizes quickly that the advice is harmful by earning

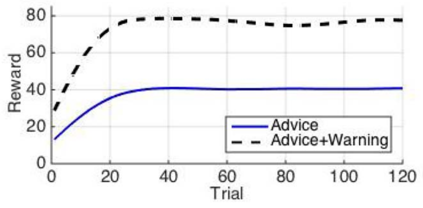


Fig. 10. Impact of warnings on reward for coins and participant 3 from survey.

negative Q-values. The most exploration occurs in the first several trials, so it is likely the agent will experience many actions with better performance than the adversarial advice. With mediocre advice, the Q-values will be positive, although not optimal. Fewer actions will earn higher Q-values, and a better policy may not be found in a timely manner. Because of the nature of ϵ-greedy exploration, it is unlikely the same advice will be followed multiple times in a row, which makes the situation more difficult in Mario's domain due to the combination of momentum in Mario's movements and state aliasing.

6) Advice+Warnings: Fig. 10 shows that incorporating "what not to do" warnings in addition to "what to do" advice increased the cumulative reward earned by the agent for different objects. Avoiding dangerous actions and considering multiple objects simultaneously during action selection improved the agent's performance.

The agent accumulated approximately twice the reward when both advice and warnings were included. Algorithmically, this implies that if a robot or software agent queries a human teacher for advice, it may improve performance by asking for both advice and warnings. Even though the experiment did not specifically ask participants to provide warnings, they were able to do so for the free-form and structured explanations. This is a result we will explore in future work.

VI. Conclusion

Sentiment analysis can be used to filter natural language explanations into advice of what to do and warnings of what not to do. Negative classifications should not be immediately trusted since there is a high likelihood of false negatives. Splitting sentences with negative sentiment into clauses and reclassifying increased the overall accuracy of the sentiment filter by approximately 30% to around 85%. While a sentiment filter can process free-form explanations, many of the sentences are not actionable and cannot be directly utilized as advice.

Once the explanations have been split into advice and warnings, object-focused advice and OF-Q can be used to train the agent to maximize its reward for each object. We presented a novel method of using human advice and warnings that links objects to actions and does not require people to specify state variables. object-focused advice allows people to generalize over an object's state space, which means people are not forced to provide numbers or particulars describing the state in explanations. A model-free approach has been described that increases performance and does not require the intensive construction of formal language translations.

The goal of object-focused advice is not to capture all the nuances and subtleties of free-form teaching, but rather to make use of human explanations without state information. It is vital to develop methods that use human explanations that are not state-specific since they reflect much of nonexpert instruction.

REFERENCES

[1] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. 21st Int. Conf. Mach. Learn.*, Banff, AB, Canada, 2004, p. 1.

[2] B. D. Argall, B. Browning, and M. Veloso, "Learning robot motion control with demonstration and advice-operators," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nice, France, 2008, pp. 399–404.

[3] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auton. Syst.*, vol. 57, no. 5, pp. 469–483, 2009.

[4] S. Chernova and A. L. Thomaz, "Robot learning from human teachers," *Synth. Lectures Artif. Intell. Mach. Learn.*, vol. 8, no. 3, pp. 1–121, 2014.

[5] L. C. Cobo, C. L. Isbell, and A. L. Thomaz, "Object focused Q-learning for autonomous agents," in *Proc. Int. Conf. Auton. Agents Multi Agent Syst.*, St. Paul, MN, USA, 2013, pp. 1061–1068.

[6] C. Diuk, A. Cohen, and M. L. Littman, "An object-oriented representation for efficient reinforcement learning," in *Proc. 25th Int. Conf. Mach. Learn.*, Helsinki, Finland, 2008, pp. 240–247.

[7] S. Griffith, K. Subramanian, J. Scholz, C. Isbell, and A. L. Thomaz, "Policy shaping: Integrating human feedback with reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 2625–2633.

[8] M. Joshi, R. Khobragade, S. Sarda, U. Deshpande, and S. Mohan, "Object-oriented representation and hierarchical reinforcement learning in infinite mario," in *Proc. IEEE 24th Int. Conf. Tools Artif. Intell. (ICTAI)*, vol. 1. Athens, Greece, 2012, pp. 1076–1081.

[9] S. Krening, B. Harrison, K. M. Feigh, C. Isbell, and A. Thomaz, "Object-focused advice in reinforcement learning," in *Proc. Int. Conf. Auton. Agents Multi Agent Syst.*, Singapore, 2016, pp. 1447–1448.

[10] G. Kuhlmann, P. Stone, R. Mooney, and J. Shavlik, "Guiding a reinforcement learner with natural language advice: Initial results in RoboCup soccer," in *Proc. AAAI Workshop Supervisory Control Learn. Adapt. Syst.*, San Jose, CA, USA, 2004.

[11] J. MacGlashan *et al.*, "Grounding English commands to reward functions," in *Proc. Robot. Sci. Syst.*, Rome, Italy, Jul. 2015.

[12] R. Maclin, J. Shavlik, L. Torrey, T. Walker, and E. Wild, "Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression," in *Proc. Nat. Conf. Artif. Intell.*, vol. 20. Pittsburgh, PA, USA, 2005, pp. 819–824.

[13] C. D. Manning *et al.*, "The Stanford CoreNLP natural language processing toolkit," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguist. Syst. Demonstrations*, Baltimore, MD, USA, 2014, pp. 55–60.

[14] C. Meriçli, S. D. Klee, J. Paparian, and M. Veloso, "An interactive approach for situated task specification through verbal instructions," in *Proc. Int. Conf. Auton. Agents Multi Agent Syst.*, Paris, France, 2014, pp. 1069–1076.

[15] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *Proc. 43rd Annu. Meeting Assoc. Comput. Linguist.*, Ann Arbor, MI, USA, 2005, pp. 115–124.

[16] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Found. Trends Inf. Retrieval*, vol. 2, nos. 1–2, pp. 1–135, 2008.

[17] M. S. Sivamurugan and B. Ravindran, "Instructing a reinforcement learner," in *Proc. FLAIRS Conf.*, Marco Island, FL, USA, 2012.

[18] B. F. Skinner, *The Behavior of Organisms: An Experimental Analysis*. New York, NY, USA: D. Appleton, 1938.

[19] B. F. Skinner, "Selection by consequences," *Science*, vol. 213, no. 4507, pp. 501–504, 1981.

[20] R. Socher *et al.*, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proc. Conf. Empir. Methods Nat. Lang. Process. (EMNLP)*, vol. 1631. Seattle, WA, USA, pp. 1631–1642, 2013.

[21] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 1. Cambridge, MA, USA: MIT Press, 1998.

[22] J. Togelius, S. Karakovskiy, and R. Baumgarten, "The 2009 Mario AI competition," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Barcelona, Spain, 2010, pp. 1–8.

Samantha Krening received the B.S. and M.S. degrees in aerospace engineering from the University of Colorado at Boulder, Boulder, CO, USA, in 2011, where she emphasized in astrodynamics and nonlinear control. She is currently pursuing the Ph.D. degree in robotics with the Georgia Institute of Technology, Atlanta, GA, USA.

She was with NASA's Jet Propulsion Laboratory in Guidance and Control for the Cassini spacecraft. Her current research interests include machine learning, human–robot interaction, learning from natural language, and explainable AI.

Brent Harrison received the B.S. degree in computer science and the B.A. degree in English from Auburn University, Auburn, AL, USA, in 2008, and the M.S. and Ph.D. degrees in computer science from North Carolina State University, Raleigh, NC, USA, in 2012 and 2014, respectively.

He is a Research Scientist with the Georgia Institute of Technology, College of Computing, Atlanta, GA, USA. His current research interests include machine learning, computational storytelling, and artificial virtual agents.

Dr. Harrison is a member of the Association for the Advancement of Artificial Intelligence.

Karen M. Feigh (M'09–SM'14) received the B.S. degree in aerospace engineering from the Georgia Institute of Technology, Atlanta, GA, USA, the M.Phil. degree in engineering from Cranfield University, Cranfield, U.K., and the Ph.D. degree in industrial and systems engineering from the Georgia Institute of Technology.

She is an Associate Professor with the School of Aerospace Engineering, Georgia Institute of Technology. Her current research interests include cognitive engineering, design of decision support systems, human automation interaction, and behavioral modeling.

Charles Lee Isbell, Jr. received the B.S. degree from the Georgia Institute of Technology, Atlanta, GA, USA, and the Ph.D. degree from the Massachusetts Institute of Technology, Cambridge, MA, USA.

He is a Professor and the Senior Associate Dean with the Georgia Tech College of Computing. His current research interests include time building autonomous agents that engage in life-long learning in the presence of thousands of other intelligent agents, including humans.

Mark Riedl received the Ph.D. degree from North Carolina State University, Raleigh, NC, USA, in 2004.

He is an Associate Professor with the Georgia Tech School of Interactive Computing, Atlanta, GA, USA, and the Director of the Entertainment Intelligence Laboratory. His current research interests include intersection of artificial intelligence, virtual worlds, and storytelling.

Andrea Thomaz (M'04) received the B.S. degree in electrical and computer engineering from the University of Texas at Austin, Austin, TX, USA, in 1999, and the M.Sc. and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2002 and 2006, respectively.

She is an Associate Professor of Electrical and Computer Engineering, University of Texas at Austin. Her current research interests include computationally model mechanisms of human social learning in order to build social robots and other machines that are intuitive for everyday people to teach.