Learning from Stories: Using Natural Communication to Train Believable Agents

Brent Harrison, Siddhartha Banerjee, and Mark O. Riedl
School of Interactive Computing, Georgia Institute of Technology
Atlanta, Georgia, USA

Abstract

In this work we introduce *Quixote*, a system that allows non-programmers to train believable virtual agents and robots using the sociocultural knowledge present in stories. Quixote uses a corpus of exemplar stories to engineer a reward function that can be used to train virtual agents to exhibit desired behaviors using reinforcement learning. We show the effectiveness of our system with a case study in a modified gridworld environment called Pharmacy World. In this case study, we examine the performance of Quixote under ideal conditions and evaluate how humans perceive the policies that our system produces. In addition, we examine how Quixote performs in this environment when ambiguities in natural language correspondence causes difficulty in mapping story events to agent actions.

1 Introduction

Authoring virtual agents and robots is a difficult task that requires a non-trivial amount of programming knowledge to perform. Even using a self-contained machine learning tool to train agents and robots will require some amount of programming/computer science knowledge. This makes it difficult for non-programmers to train virtual agents and robots. In this work, we make this task more accessible to non-programmers by enabling them to train virtual agents and robots through natural communication.

Methods such as *apprenticeship learning* [1] or *learning from demonstration* (LfD) [2] seek to make agent training more accessible to non-programmers by allowing people to provide a corpus of exemplar demonstrations to help the agent learn optimal behavior. This improves on upon traditional learning-based approaches to agent and robot training as providing these demonstrations typically does not require extensive programming knowledge. These demonstrations, however, typically come in the form of complete trajectories in the environment that the agent will be acting in, thus making it a requirement that authors have prior knowledge of the agent's environment. Also, many times these demonstrations are generated by people performing physical actions in a real space. These types of demonstrations can be especially problematic for those with disabilities or the elderly to provide. As such, it can still be difficult for trainers to provide demonstrations in some cases.

In this work we introduce the idea of using a natural source of human communication to train machine learning algorithms: stories. Our system, which we call *Quixote*, uses stories told by humans to train virtual agents to exhibit specific behaviors. This task is similar to that of LfD algorithms except that our technique learns from stories told about a task rather than from explicit demonstrations of said task. The primary difference between stories and demonstrations, as well as the primary challenge in dealing with stories, is that stories are more unconstrained than demonstrations since authors have no prior knowledge about specifics of the environment. Also, many different stories could all correctly describe the same task, or authors could skip steps that they feel are obvious or do not need mentioning. Further, storytelling is non-Markovian in that some events that occur are influenced by events that happened far in the past. This can make it especially difficult to utilize story information since most agent environments are assumed to be Markovian.

Quixote addresses many of these issues by first cleaning an initial story corpus using the technique outlined by Li *et al* [7]. This allows for Quixote to reconcile the exemplar stories with each other and fill in any gaps that may exist, which makes learning a more manageable task. From there, Quixote uses this new corpus to define the space of acceptable behaviors that is then turned into a reward function that can be used to train reinforcement learning agents.

To explore the effectiveness of our system, we present a case study in which we use stories to train a reinforcement learning agent in Pharmacy World, a modified grid world in which the agent attempts to acquire drugs from a pharmacy. In this case study, we examine how Quixote performs under ideal circumstances in which the stories told map directly onto the environment that the agent exists in. We also perform a user study to further evaluate the quality of the behaviors exhibited by our system. Finally, we examine how Quixote performs in non-ideal cases and provide insight into how our technique performs when some aspects of the story cannot be mapped onto the environment or it is unclear how to do so.

2 Related Work

Recently there has been an increased focus on *interactive machine learning*, which seeks to augment machine learn-

ing algorithms with the ability to learn from human feedback or demonstrations directly. The type of interactive machine learning that is most closely related to our own work is *Inverse Reinforcement Learning* (IRL). IRL attempts to learn the reward function that best describes a corpus of policy examples [13] or policy trajectories [1]. Early work in this area required either complete policy examples or complete trajectories in order to learn. This requirement was relaxed through the introduction of techniques such as Bayesian IRL [14] and maximum entropy IRL [17]. There has also been work on relaxing the assumption that all example policies or trajectories are correct [3]. Researchers have also sought to derive behaviors from natural language commands [8; 10].

The problem that we solve with Quixote is fundamentally different than the problem posed in IRL. While we are attempting to derive reward functions based on a corpus of examples, we make different assumptions about what these examples represent, which leads to a different understanding of how to approach the problem. In this work, we assume that our example stories define a space of believable behaviors. Rather than design a reward function that can reproduce all of these examples, we are trying to create a reward function that produces behaviors that fall within this space.

There is also a branch of research that explores how human feedback about agent learning can be integrated into agent training. These systems integrate human reward signals in order to shape agent behavior [6; 5; 9]. These systems each have users give feedback to an agent while it is learning by means of giving it positive or negative feedback. Our work differs from this work in that we do not seek to dynamically shape agent behavior. Quixote aims to derive what correct behavior is by looking at examples of desired behavior.

3 Reinforcement Learning Background

The Quixote system uses a set of exemplar stories to construct reward functions that can be used to train reinforcement learning agents. Reinforcement learning [15] is a technique that is used to solve a Markov decision process (MDP). A MDP is a tuple $M =< S, A, T, R, \gamma >$ where S is the set of possible world states, A is the set of possible actions, T is a transition function $T : S \times A \rightarrow P(S)$, R is the reward function $R : S \times A \rightarrow \mathbb{R}$, and γ is a discount factor $0 \leq \gamma \leq 1$.

Reinforcement learning first learns a policy $\pi : S \rightarrow A$, which defines which actions should be taken in each state. In this work, we use Q-learning [16], which uses a Q-value $Q(s, a)$ to estimate the expected future discounted rewards for taking action a in state s. Reinforcement learning allows the agent to fill in any gaps that may exist in the stories due to authors not having prior knowledge about the agent's environment. Thus, the agent is able to take several actions, should it need to, in between plot points. Reinforcement learning also allows the agent to deviate from the stories it has been told if doing so will allow it to more efficiently reach a goal state.

4 The Quixote System

A high level flowchart of the Quixote system can be seen in Figure 1. The Quixote system works by first taking in a set of exemplar stories and cleaning it to filter out noise and determine the many possible ways that a task can unfold. Quixote then converts this corpus into a trajectory tree which encodes every story in this corpus. This tree is then used to determine a reward function which is used to train a reinforcement learning agent to exhibit the desired behaviors. In this section, we will describe each of these steps in greater detail.

4.1 Automatic Story Corpus Cleaning

Initial input into the Quixote system is a set of exemplar stories written in natural language about a given task. Since human authors created these stories, it is likely that this initial corpus requires some amount of cleaning. For example, this set of exemplar stories may contain different stories that still correctly describe the same scenario. Also, some authors may skip events in the story or use different language to convey the same event. Some of these stories can also contain errors or events that do not relate to the scenario being described. To help mitigate these problems, we use the approach proposed by Li *et al* [7] to generate a clean set of stories.

This technique first involves clustering natural language sentences according to semantic similarity. These clusters are referred to as *events*. If any sentences do not cluster into any event then they are discarded. This way the technique helps to reduce the noise caused by errors or variable language in the corpus of exemplars. Second, this technique learns how events can be ordered as well as different ways in which the story can be told. Thus, noise as a result of misodering events is filtered out.

4.2 Trajectory Tree Creation

The Quixote system begins by using a set of stories to derive a trajectory tree. This is done to enable the agent to track its progress through a story trajectory. Without this, the agent becomes especially susceptible to repeatable actions. In stories it is not uncommon for authors to talk about events that are normally repeatable (such as entering or exiting buildings). If the agent cannot monitor its progress through the story then it is possible that it will repeat such actions infinitely (in order to maximize reward).

Trajectory tree creation begins with a corpus of stories that have been cleaned using the technique described in the previous section. We then use these stories to create a *trajectory tree* [4]. A trajectory tree is a structure that encodes each story in this corpus. To create the tree we do the following for each story in the initial corpus of exemplars. Stories are added to the trajectory tree by iterating over each event and searching for it in the current node's set of children. If it exists there then that child becomes the current node. If it is not a child of the current node then it is added as a child and then that node becomes the current node. Thus, every traversal of the tree from root to node is a unique story that exists in the cleaned corpus.

4.3 Reward Assignment

Using this trajectory tree, we assign rewards to actions or states that exist inside the agent's environment. In order to do this we must first determine how events in the trajectory tree correspond to actions or states. There is no guarantee

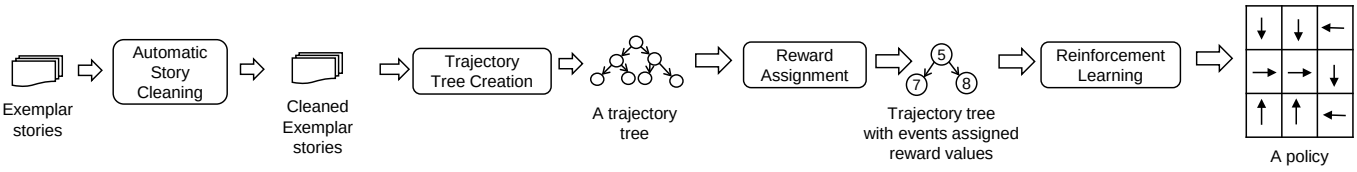

Figure 1: The Quixote system workflow.

that all story events correspond to agent actions/states since storytellers typically have no prior knowledge of the agent's capabilities or its environment. This means that actions may exist in the story corpus that do not exist for the agent or there may be actions that the agent needs to execute that do not exist in the story corpus. The mapping between events and actions/states can be done in many different ways, such as manually authoring the mapping or using natural language processing techniques to learn a mapping. We explore this topic in greater detail in Section 6.2.

Once this mapping is made, we incorporate the tree as part of the agent's world state. As the agent explores its environment it also keeps track of what story events it has completed and what story events it needs to complete in the future. Rewards are assigned to those actions or states that advance the story as determined by the trajectory tree. When the agent receives a reward, it also advances in the trajectory tree.

To our knowledge, there is no definitive technique for assigning reward values to story events. In all RL, agent behavior is highly sensitive to the environment that the agent exists in, making it difficult to determine a reward function that is applicable to every environment. We can, however, weight rewards based on their relative importance compared to other events in the stories. Here, we define an event's importance as the percentage of stories from the original corpus of exemplars that the event occurs in – meaning more important events occur in more stories.

4.4 Reinforcement Learning

Once the reward function has been specified, we use reinforcement learning to find the optimal policy through a given environment. By using reinforcement learning, the agent is able to fill in any gaps that may exist in the story corpus. These gaps exist because authors are not required to have any prior knowledge about the agent's capabilities or specifics of the environment. So, what only takes one action to complete in the story corpus may take several intermediate actions in the MDP. Reinforcement learning allows the agent to learn for itself the most efficient sequence of actions to move from one plot event to another.

As mentioned in Section 4.3, we use the trajectory tree to help determine the reward that an agent will receive. Conceptually, encoding the trajectory into the world state breaks up the learning problem into subtasks based on how the tree branches. The subtask that the agent learns is how to optimally get from its current story event to one of its children in the trajectory tree. The agent only needs to determine at any time what is the optimal policy to get to the next story event. Therefore, the reinforcement learning agent is iteratively solving simple MDPs rather than solving a single complex MDP. This allows the agent is able to learn non-Markovian behaviors (since they will be encoded in the trajectory tree) and avoid infinite rewards due to rewarding repeatable actions (since performing the action will transition the agent into a "different" MDP with different rewards).

This does affect the size of the MDP, however. Since the trajectory tree determines how the MDP task is divided into subtasks, the size of the MDP grows linearly with the number of branches in the trajectory tree. While learning, the reinforcement learning agent must, essentially, explore each branch of the trajectory tree, which will increase training time. This is acceptable as Quixote is meant to be run as an offline process and should only need to be run once per environment.

5 Case Study

To show the effectiveness of the Quixote system, we have chosen to perform a case study in a modified gridworld called *Pharmacy World*. In this case study, we explore how effective Quixote is when all events in the trajectory tree correspond to agent actions and states.

5.1 Pharmacy World Domain

The Pharmacy World domain is a modified gridworld that is based on the innocuous activity of going to the pharmacy to purchase drugs and returning home with them. Pharmacy World contains five different locations each positioned somewhere in the gridworld: a house, a bank, a doctor's office, a clinic, and a pharmacy. Each of these locations, except for the house, contains items that can be used to enable or disable certain actions. The bank contains money that can be used to purchase either weak or strong drugs from the Pharmacy. The doctor's office and the clinic both contain prescriptions that can be used with the money to purchase strong drugs.

The actions that the agent can take in Pharmacy World include simple movement actions, such as moving and entering/leaving a building, and actions that are used to retrieve objects. This is a stochastic environment because one particular action (*Get Examined*) fails 25% of the time. Generally, items can be acquired either by stealing them or by first interacting with people at various locations (such as the Bank Teller at the bank). The goal in Pharmacy World is to return to the house with either the strong or the weak drugs, with the strong drugs being preferred.

5.2 Simulation Study

The first study done in Pharmacy world was to determine if the optimal policy learned by Quixote exists within the space

of acceptable behaviors defined by the cleaned corpus of stories. We will discuss this study and its results below.

Determining Rewards

In order to assign rewards in Pharmacy World, we first must have a cleaned corpus of stories defining acceptable behaviors. For this case study we chose to manually author this set of stories. We do this because the story cleaning process process has been explored in great detail [7] and empirically evaluated. Manually authoring these stories gives us an efficient means to control the evaluation of the novel parts of our system. In total, we manually authored 213 stories that can be used to generate a trajectory tree containing 827 nodes.

Once the trajectory tree has been created, we manually map the plot events in the tree onto actions/states that exist inside Pharmacy World. Each node in the trajectory tree directly corresponds to an action available in Pharmacy World, except for the act of receiving or not receiving a prescription. This node actually represents whether the *Get Examined* action fails or succeeds, so that is where rewards are assigned.

In these experiments, we used the base reward value 10 every time the agent moved to a new event in the trajectory tree. This, in practice, produced acceptable policies for Pharmacy World, but is likely domain specific. We leave the problem of automatically determining this value to future work. The base reward value was then further weighted by event importance as discussed earlier. For each other possible state, we assigned a reward value of -1.0.

Training

We used Q-learning in conjunction with ϵ-greedy exploration for training. For this study we define ϵ to be 0.8 and then slowly decay it over $200,000$ learning episodes. In practice, this was a sufficient number of episodes for Q-learning to converge. In addition, we use parameters $\gamma = 1.0$ and $\alpha = 0.5$. To evaluate the learned behavior, we examined the policy that the agent learned in order to verify that it existed within the space of accepted stories defined by the corpus of stories.

Results

After training for $200,000$ episodes, we examined the policy that the agent learned. Since there is a source of stochasticity in Pharmacy World, this resulted in 3 possible classes of trajectories through the environment depending on the outcome of the *Get Examined* action.

The first case that the agent could encounter is the one in which the *Get Examined* action succeeds on the first attempt. In this case, the agent first navigates to the clinic and gets examined in order to receive the prescription. Then, the agent navigates to the bank and requests and withdraws the money. Having obtained the money and the prescription, the agent then moves to the pharmacy, requests and purchases the strong drugs, and finally returns home.

In the second case the first *Get Examined* action fails while the second one succeeds. The agent's policy is the same as in the first case except that the agent goes to the doctor's office to get examined after retrieving the money from the bank. This action ultimately succeeds, and then the agent navigates to the pharmacy to purchase the strong drugs and returns home.

The final case is the one in which the agent is unable to obtain a prescription at all due to both *Get Examined* actions failing. In this case, the agent's policy is the same as the policy in the second case, except that the agent then chooses to purchase the weak drugs and return home.

Each of these trajectory classes falls within the realm of acceptable behavior as defined by the story corpus we used to generate the reward function. Thus, by introducing this reward function we were able to prevent the agent from exhibiting abnormal, and possibly psychotic-appearing behaviors that may result from simpler reward functions (such as stealing the drugs rather than purchasing them).

5.3 User Study

In order to further evaluate the effectiveness of Quixote, we performed a user study to determine how humans perceived the policies learned by our agent. The details of this study and the results are discussed in this section.

Methodology

In this study humans watched replays of agents acting in a graphical representation of Pharmacy World and then chose which one they felt acted in a more humanlike manner.

For this study, we compared an agent trained using Quixote against two other agents. For the remainder of this study, we'll refer to our agent as the *Quixote agent*. As a baseline, we used an agent that was trained using Q-Learning with a naïve reward function in which the agent is only rewarded for returning home with either of the drugs. We refer to this agent as the *baseline agent*. The other agent used in the study, referred to as the *human agent*, was hand authored based on replays of humans acting out the Pharmacy World scenario. In order to create this agent, 3 human players unfamiliar with the research played through the Pharmacy World scenario under three different conditions, each meant to account for a possible outcome of the *Get Examined* action. Each author played through these conditions in a randomized order. We took the behavior exhibited by a majority of the authors and used that to create the human agent used in the study. Both Q-Learning agents (the Quixote agent and baseline agent) were trained over $200,000$ learning episodes using ϵ-greedy exploration with the parameters $\gamma = 1.0$ and $\alpha = 0.5$.

In the study, participants are semi-randomly shown pairs of replays selected from the three agents tested acting under one of the following conditions: no *Get Examined* actions fail, the first *Get Examined* action fails, or all *Get Examined* actions fail. Since we are interested in how the Quixote agent compares to the other agents, participants are always presented with replays of the Quixote agent when making a comparison. The other agent compared against was randomly selected from among the baseline agent and the human agent. The pairs of agents are presented to the user in a random order. After viewing both replays, users are asked to select if either the first replay is more humanlike than the second, the second replay is more humanlike than the first, or if they are both equally humanlike. This process repeats for each of the remaining conditions meaning that each user views two replays for each of the three possible conditions. These conditions are also presented to users in a randomized order.

Table 1: Number of responses for each condition. C1, 2, and 3 correspond to the cases where all *Get Examined* actions succeed, the first *Get Examined* action fails, and both *Get Examined* actions fail respectively.

	C1	C2	C3
Quixote vs Human	33	25	33
Quixote vs Baseline	32	42	34

Table 2: Comparison of the Quixote agent and the baseline agent. This shows the number of participants that selected each option for which agent was more humanlike.

	Quixote	Baseline	Both
Condition 1	23	2	7
Condition 2	28	10	4
Condition 3	23	8	3

Our hypothesis in this study is that the Quixote agent will exhibit more humanlike behavior than the baseline agent and comparable behavior to the human agent. We place more emphasis on outperforming the baseline agent, however, as this tells us that our technique is offering some amount of improvement over a naïve baseline.

Data Collection

Users took part in this study online over the course of several weeks. In total, 83 users participated in the study producing 199 evaluations. This means that, on average, each user completed 2.4 evaluations. The number of evaluations for each condition of the study are summarized in Table 1. We attribute the imbalance in condition 2 to the variance that occurs from showing users random replays.

Results and Discussion

A summary of responses is shown in Tables 2 and 3. To analyze the data collected, we performed a chi-square test between each condition with a null hypothesis of a uniform distribution across preferences. Comparing the Quixote agent and the baseline agent, results show that the resulting distribution of participant responses was significantly ($p < 0.05$) skewed towards the Quixote agent. This means that participants felt that the Quixote agent exhibited more humanlike behavior than the baseline. This confirms part of our hypothesis in that agents produced by Quixote are differentiable from agents that learn behavior based on a naïve reward function.

When comparing the Quixote agent and the human agent, we find that the distribution of participant responses is significantly ($p < 0.05$) skewed towards the human agent except in the first condition where the distribution is significantly ($p < 0.05$) towards both agents being equally humanlike. To better understand this result, we must examine the behaviors exhibited by each of these agents under each condition. Under the first condition, both agents act identically,which explains why these agents appear equally humanlike. In the second condition and third condition, the human agent tries to get the prescription first. When this action fails, the human makes further attempts to get a prescription before moving on to other tasks. Contrast this with the Quixote agent which first tries to get the prescription at the clinic (which fails), then withdraws the money from the bank, and then tries to get the prescription at the doctor's office.

Table 3: Comparison of the Quixote agent and the human agent. This shows the number of participants that selected each option for which agent was more humanlike.

	Quixote	Human	Both
Condition 1	3	9	21
Condition 2	3	13	9
Condition 3	8	20	5

The main difference between these two agents is that the human agent exhibits the idea of *conceptual locality*, or *thought flow* [12]. The human agent does not interleave general tasks such as getting the prescription, opting instead to fully complete a task before moving on. The Quixote agent, however, is willing to interleave tasks if it would be more efficient. We hypothesize that it is more humanlike to not interleave tasks in Pharmacy World. Using this finding, it is possible to further improve upon Quixote by incorporating domain-independent heuristics such as conceptual locality. This we leave for future work.

6 Event Correspondence

In this section, we explore how Quixote will perform in the more realistic situation in which natural language processing techniques are used to determine correspondence between story events and agent actions. We will examine the following cases: (a) the case where events in the trajectory tree do not exist in the agent's environment/actionset and (b) the case where events in the trajectory tree map onto several different actions in the agent's actionset.

6.1 Missing Information

In this section, we examine the performance of our technique in the case where some of the events that occur in the trajectory tree do not map onto actions that the agent can perform and, thus, must be removed from the trajectory tree.

Methodology

Since the story corpus is generated by authors with no prior knowledge of the environment that the agent will exist in, it is to be expected that there will not always be a perfect correspondence between story events and the actions an agent can take. This could also be caused by a poor natural language system missing possible matches. In order to simulate the case where the author cannot map an event onto any action, we randomly remove nodes from the trajectory tree used in Section 5.2 and then use this modified trajectory tree to generate a reward function.

For this evaluation, we choose to examine the cases where single events could not be mapped to any action and, thus, are removed from the trajectory tree. Due to the size and simplicity of Pharmacy World, we believe that examining these cases will be sufficient to get a preliminary understanding of how our technique will perform in this environment with incomplete mappings. Simulations were run with the same parameters as previous experiments.

Each policy is then evaluated based on whether or not it falls within the space of acceptable behavior outlined by the trajectory tree. The more policies that fall in this space, the more robust Quixote is to missing information.

Results and Discussion

In the case where single nodes were removed from the trajectory tree, every policy was deemed acceptable except for those learned when Withdraw Money, Buy Strong Drugs, or Buy Weak Drugs were removed from the tree. In these cases, removing the node in question resulted in a policy in which the agent resorted to stealing.

The reason that this behavior occurred is because an alternative existed for each of these actions. In Pharmacy World, the agent can either purchase these items (which involves first requesting the item) or steal them (which requires no prerequisite actions). If each step receives a negative reward, then a reinforcement learning agent will prefer to steal even though it is typically considered socially unacceptable. To prevent this, the author must assign rewards such that the negative reward received for performing an extra action is offset by the reward gained by purchasing an item.

This gives insight into one case in which missing information causes Quixote to fail: if there are multiple ways to complete a task and one is rewarded while the others are not, removing that reward will result in the agent reverting back to completing tasks as fast as possible. As such, it is important that these types of events be mapped correctly onto agent actions.

6.2 Ambiguous Events

In this section, we examine the performance of Quixote in cases where ambiguity in the natural language correspondence between events and actions leads to an event corresponding to several possible actions. For example, the story event of *Go To Bank* could correspond to both the action *Go inside Bank* and to the action *Go outside Bank* in the agent's action space. These types of ambiguities can arise when one uses natural language processing techniques to automatically map story events onto agent actions.

Methodology

In order to simulate the semantic ambiguities that might occur due to an imperfect correspondence between plot events and the actions available to an agent, we create a probabilistic mapping of story events to agent actions. In order to make this mapping, the agent's actions must have natural language labels. For this study we manually authored a set of action labels that accurately described what the action did.

Each of the words in a story event or agent action label is converted to a vector using word2vec [11] and then averaged. Events are then matched to actions based on cosine similarity of their respective vectors. While more advanced techniques exist, this method allows us to examine how Quixote performs in the face of poor mappings. For these experiments, we use a threshold value to determine whether a match has occurred or not. It is possible that a plot event can map to multiple different actions in the agent environment. For the purposes of these tests, we try different threshold values when accepting or rejecting matches.

Simulations are done using Q-Learning to train a reinforcement learning using the parameters used in previous simulation experiments. Each policy generated by a simulation run is then evaluated based on whether or not it falls within the space of acceptable behavior outlined by the trajectory tree.

Results and Discussion

Quixote remained mostly robust in the face of multiple matches. Dissecting the types of matches that were created helped us derive an intuition of why certain mappings led to unacceptable behavior while others did not.

Some events were mapped to actions that were not available at the same time. In these experiments, plot event *Go To Pharmacy* mapped to the actions *Go Inside Pharmacy* and *Go Outside Pharmacy*. Since these actions are not executable at the same time, this had no effect on the policy learned.

Even in the case where mapped actions are executable, we still found that acceptable behavior is produced if all mapped actions are socially acceptable. Since any of the actions is socially acceptable, the policy generated is acceptable as well.

However, when any of the executable actions is socially unacceptable, the reward function starts to have a greater impact on the acceptability of the generated policy. For these experiments, the plot event *Buy Strong Drugs* was mapped to both the actions of *Purchase strong drugs* and *Pick Up strong drugs*, which without the prerequisite action of requestion is essentially stealing. In Pharmacy World, purchasing items requires more steps than simply picking up an item. We found that unless the reward provided to Quixote for the *Purchase strong drugs* action was high enough to offset the cost of the extra step for the correct behavior (by weighting the reward based on the confidence of the match, for example), the agent learned to forego the purchase and steal.

This leads us to conclude, as with incomplete mappings, that mappings onto actions with socially unacceptable alternatives require that reward function parameters be precisely tuned to bias Quixote towards the socially correct actions.

7 Conclusions

In this work we introduce Quixote, a system for training believable agents based on stories. Our system works by using exemplar stories to define a space of acceptable behavior and then using this space to derive reward functions that encourage the agent to exhibit behaviors that fall within it. We have shown that under ideal conditions, Quixote is able to produce behaviors that fall within this space, although some difference with human behavior can occur. We have also explored how robust Quixote is to incorrect or incomplete knowledge that can result from poor mappings from the natural language contained in the story corpus onto the agent's action space. Using this technique, we allow humans to more naturally communicate with interactive machine learning algorithms which will, ideally, make it easier for non-programmers to use these algorithms.

References

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the*

twenty-first international conference on Machine learning, page 1. ACM, 2004.

[2] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[3] D. H. Grollman and A. Billard. Donut as i do: Learning from failed demonstrations. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3804–3809. IEEE, 2011.

[4] B. Harrison and M. O. Riedl. Towards learning from stories: An approach to interactive machine learning. In *Proceedings of the Workshop on Symbiotic Cognitive Systems*, 2016.

[5] K. Judah, S. Roy, A. Fern, and T. G. Dietterich. Reinforcement learning via practice and critique advice. In *AAAI*, 2010.

[6] W. B. Knox and P. Stone. Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the fifth international conference on Knowledge capture*, pages 9–16. ACM, 2009.

[7] B. Li, S. Lee-Urban, G. Johnston, and M. Riedl. Story generation with crowdsourced plot graphs. In *AAAI*, 2013.

[8] C. Lignos, V. Raman, C. Finucane, M. Marcus, and H. Kress-Gazit. Provably correct reactive control from natural language. *Autonomous Robots*, 38(1):89–105, 2015.

[9] R. Loftin, J. MacGlashan, B. Peng, M. E. Taylor, M. L. Littman, J. Huang, and D. L. Roberts. A strategy-aware technique for learning behaviors from discrete human feedback. In *Proc. of AAAI*, 2014.

[10] J. MacGlashan, M. Babes-Vroman, M. desJardins, M. Littman, S. Muresan, S. Squire, S. Tellex, D. Arumugam, and L. Yang. Grounding english commands to reward functions. In *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[12] M. J. Nelson and M. Mateas. Search-based drama management in the interactive fiction anchorhead. In *AIIDE*, pages 99–104, 2005.

[13] A. Y. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *in Proc. 17th International Conf. on Machine Learning*, 2000.

[14] D. Ramachandran and E. Amir. Bayesian inverse reinforcement learning. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 2586–2591. Morgan Kaufmann Publishers Inc., 2007.

[15] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[16] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[17] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd national conference on Artificial intelligence-Volume 3*, pages 1433–1438. AAAI Press, 2008.