

# **HUMANS TEACHING INTELLIGENT AGENTS WITH VERBAL INSTRUCTION**

A Dissertation  
Presented to  
The Academic Faculty

By

Samantha Krening

In Partial Fulfillment  
of the Requirements for the Degree  
Doctor of Philosophy in the  
School of Institute for Robotics and Intelligent Machines

Georgia Institute of Technology

May 2019

Copyright © Samantha Krening 2019

# **HUMANS TEACHING INTELLIGENT AGENTS WITH VERBAL INSTRUCTION**

Approved by:

Dr. Karen M. Feigh, Advisor  
School of Aerospace Engineering  
*Georgia Institute of Technology*

Dr. Mark Riedl  
School of Interactive Computing  
*Georgia Institute of Technology*

Dr. Charles Isbell  
School of Interactive Computing  
*Georgia Institute of Technology*

Dr. Sonia Chernova  
School of Interactive Computing  
*Georgia Institute of Technology*

Dr. Ayanna Howard  
School of Interactive Computing  
*Georgia Institute of Technology*

Date Approved: December 13, 2018

Common sense is not so common

*Voltaire*

Dedicated to John Winterrowd.

My friend.

He fought his last fight. Semper Fidelis.

## ACKNOWLEDGEMENTS

I would like to show my gratitude for the incredible support shown by my advisor, professors, colleagues, friends, and family.

First, I'd like to thank my advisor. Dr. Feigh has been amazing and steadfast in her support, both personally and professionally, through the ups and downs of this process. She's always there to get chat about ideas and get me back on track when new, shiny ideas try to distract me. She has an incredible talent for editing and organizing technical content that has been of great benefit to this work.

I'd also like to thank my phenomenal committee members: Dr. Charles Isbell, Dr. Mark Riedl, Dr. Sonia Chernova, and Dr. Ayanna Howard. Charles and Mark are on the IML grant that funds my research, so have been with me since the start. Sonia taught me a lot in her HRI class and is always willing to meet and chat. Ayanna was very encouraging in the robotics project course where I was able to experiment with some early ideas of this work. I'm very fortunate to have such an engaged committee.

I am lucky enough to have had some wonderful mentors over the years. I'd like to thank Dr. Hanspeter Schaub, my M.S. thesis advisor, for his support and encouragement to pursue my doctorate. I'd also like to thank Dr. Dirk Grunwald for his unwavering support ever since my freshman year of undergraduate studies - he introduced me to programming and changed my life.

Thanks to the ONR for funding my research and making this work possible.

Next I want to thank my fellow students. Brent Harrison and Kaushik Subramanian were instrumental in helping me transition from aerospace to machine learning. Thanks to the entire IML team for bouncing ideas off of each other, especially Himanshu Sahni, Yannick Schrockner, and Thomas Cederborg. Thanks to Danny Nguyen for his work with the Praat NLP software, and Lara Martin for her NLP introduction. All of the members of Robowomen provided friendly ears and an escape from research. Thanks to all of the

members of the Cognitive Engineering Center for being the first to listen to presentations and provide constructive feedback.

I want to thank my friends for always being there for me, especially Amber Roberts and Jana Jordan. Thank you, Yosef Razin, for lunches out of the lab, good perspective, and better conversation. Thanks to Sasha Lambert for dragging me away from my research for walks.

Finally, I want to thank my family. None of this would be possible without the unconditional love and support of my parents. Thanks to my siblings for always listening and conspiring to make me laugh.

## TABLE OF CONTENTS

<b>Acknowledgments</b> . . . . .	v
<b>List of Tables</b> . . . . .	xiv
<b>List of Figures</b> . . . . .	xvi
<b>Summary</b> . . . . .	xix
<b>Chapter 1: Introduction</b> . . . . .	1
1.1 Thesis Statement . . . . .	1
1.1.1 Thesis Foci . . . . .	2
1.2 Agent Overview . . . . .	3
1.2.1 Methods of Human Instruction . . . . .	3
1.2.2 Language as an Interface . . . . .	4
1.2.3 Interaction Algorithms . . . . .	5
1.3 Contributions . . . . .	5
1.4 Organization . . . . .	7
<b>Chapter 2: Literature Review</b> . . . . .	9
2.1 Reinforcement Learning (RL) . . . . .	9
2.1.1 Object-Oriented RL . . . . .	10

2.1.2	Bayesian Q-Learning . . . . .	11
2.2	Learning from Human Teachers . . . . .	11
2.2.1	Learning from Critique . . . . .	14
2.2.2	Learning from Explanations and Advice . . . . .	14
2.3	Natural Language Processing (NLP) . . . . .	16
2.3.1	Automatic Speech Recognition (ASR) . . . . .	16
2.3.2	Sentiment Analysis . . . . .	16
2.3.3	Prosody . . . . .	17
<b>Chapter 3:</b>	<b>Application of NLP to IML . . . . .</b>	<b>18</b>
3.1	Introduction . . . . .	18
3.1.1	Research Questions . . . . .	18
3.1.2	Development . . . . .	18
3.1.3	Method Summary . . . . .	19
3.2	Sentiment Analysis as an Advice Filter . . . . .	19
3.2.1	A Question of Ethics . . . . .	23
3.2.2	Conclusion on Sentiment from Study 1 . . . . .	24
3.3	Sentiment Analysis as a Critique Filter . . . . .	24
3.3.1	Conclusion on Sentiment from Study 3 . . . . .	26
3.4	Prosody . . . . .	26
3.4.1	Conclusion on Prosody from Study 3 . . . . .	29
3.5	RQ Results Summary . . . . .	30
<b>Chapter 4:</b>	<b>Study 1: Learning from <i>A Priori</i> Explanations . . . . .</b>	<b>32</b>



4.1	Introduction . . . . .	32
4.1.1	Research Questions . . . . .	34
4.1.2	Algorithm Development . . . . .	35
4.1.3	Method Summary . . . . .	35
4.2	Algorithm Development: Object-Focused Human Advice . . . . .	35
4.2.1	Advice . . . . .	36
4.2.2	Warnings and Multiple Objects . . . . .	38
4.3	Study 1: Method . . . . .	40
4.3.1	Mario Domain . . . . .	41
4.3.2	Familiarization . . . . .	42
4.3.3	Human explanations . . . . .	42
4.3.4	Post-Processing . . . . .	44
4.4	Study 1: Results and Discussion . . . . .	45
4.4.1	Observations on the nature of explanations . . . . .	46
4.4.2	Object-Focused Advice . . . . .	49
4.5	Study 1: Conclusion . . . . .	54
4.6	RQ Results Summary . . . . .	55

**Chapter 5: Study 2: Template of Design and Verification of IML Algorithms  
with NAA Example . . . . . 57**

5.1	Introduction . . . . .	57
5.1.1	Research Questions . . . . .	58
5.1.2	Algorithm Development . . . . .	59
5.1.3	Method Summary . . . . .	59

5.2	Algorithm Development: Newtonian Action Advice . . . . .	59
5.2.1	Combining Supervised and Reinforcement Learning to allow for personalization by end-users . . . . .	62
5.2.2	Choosing the friction parameter . . . . .	64
5.3	Study 2: Method . . . . .	66
5.3.1	Constructing Oracles . . . . .	66
5.3.2	Human-Subject Experiment . . . . .	67
5.4	Study 2: Results and Discussion . . . . .	68
5.4.1	Simulations . . . . .	68
5.4.2	Human Subject Results . . . . .	73
5.5	Study 2: Conclusions . . . . .	74
5.6	RQ Results Summary . . . . .	75

**Chapter 6: Study 3: Moving Beyond Oracles - The Comparative Human Experience of Learning from Advice and Critique . . . . . 76**

6.1	Introduction . . . . .	76
6.1.1	Research Questions . . . . .	78
6.1.2	Development . . . . .	78
6.1.3	Method Summary . . . . .	79
6.2	Differences Between Critique and Action Advice . . . . .	79
6.2.1	Rhetoric . . . . .	79
6.2.2	Immediate Applicability of Feedback . . . . .	80
6.2.3	Issues of language processing . . . . .	80
6.3	Perceived Intelligence - Interaction Method Parallels with Morphology . . .	82

6.4	Study 3: Method . . . . .	84
6.4.1	Human Experience Measures . . . . .	87
6.4.2	Objective ML measures . . . . .	88
6.5	Study 3: Results . . . . .	88
6.5.1	Human Experience Overview . . . . .	88
6.5.2	Frustration . . . . .	91
6.5.3	Transparency . . . . .	95
6.5.4	Perceived Intelligence . . . . .	96
6.5.5	Perceived Performance . . . . .	103
6.5.6	Immediacy . . . . .	103
6.5.7	External Verification of Long Response Data Analysis . . . . .	104
6.5.8	Objective metrics on ML Performance . . . . .	105
6.5.9	Evaluating the User's Path . . . . .	107
6.6	Study 3: Discussion . . . . .	111
6.6.1	Perceived Intelligence: More than simply managing expectations . .	111
6.6.2	Summary of participants' long responses . . . . .	112
6.6.3	Design considerations . . . . .	113
6.6.4	Thoughts on using critique for IML applications . . . . .	114
6.6.5	Limitations . . . . .	115
6.7	Study 3: Conclusion . . . . .	116
6.8	RQ Results Summary . . . . .	116

<b>Chapter 7: Study 4: Effect of Advice Variations - Teasing Apart What People Like About Advice . . . . .</b>	<b>118</b>
--	------------

7.1	Introduction . . . . .	118
7.1.1	Research Questions . . . . .	119
7.1.2	Algorithm Development . . . . .	120
7.1.3	Method Summary . . . . .	120
7.2	Algorithm Development: Newtonian Action Advice Variations . . . . .	120
7.2.1	Algorithm 1: Newtonian Action Advice, 5 Steps . . . . .	120
7.2.2	Algorithm 2: Newtonian Action Advice, 1 Step (no generalization) . . . . .	120
7.2.3	Algorithm 3: Newtonian Action Advice, 5 Steps, probabilistic . . . . .	121
7.2.4	Algorithm 4: Newtonian Action Advice, 5 Steps, time delay . . . . .	121
7.3	Study 4: Method . . . . .	122
7.3.1	Radiation World Domain . . . . .	122
7.3.2	Human Experience Measures . . . . .	123
7.3.3	Objective ML measures . . . . .	124
7.4	Study 4: Results and Discussion . . . . .	124
7.4.1	Human Factors Metrics . . . . .	125
7.4.2	Objective Metrics . . . . .	131
7.4.3	Open-Ended Responses from Participants . . . . .	134
7.4.4	Design recommendations . . . . .	138
7.5	Study 4: Conclusion . . . . .	139
7.6	RQ Results Summary . . . . .	140
	<b>Chapter 8: Concluding Discussion . . . . .</b>	<b>142</b>
8.1	Design Recommendations for IML algorithms . . . . .	142

8.2	IML algorithms . . . . .	143
8.3	Design and Verification Process for IML algorithms using Human Factors .	144
8.4	Application of NLP tools to IML . . . . .	145
8.5	Summary . . . . .	146
<b>Appendix A: Study 3 Questionnaire - (NAA) Advice Agent . . . . .</b>		<b>148</b>
<b>Appendix B: Study 3 Questionnaire - (Policy Shaping) critique Agent . . . . .</b>		<b>150</b>
<b>Appendix C: Study 3 Questionnaire - After Training Both Agents . . . . .</b>		<b>152</b>
<b>Appendix D: Study 4 Questionnaire - (NAA) Advice Agent . . . . .</b>		<b>154</b>
<b>Appendix E: Study 4 Questionnaire - After Training All Four Agents . . . . .</b>		<b>156</b>
<b>References . . . . .</b>		<b>162</b>
<b>Vita . . . . .</b>		<b>163</b>

## LIST OF TABLES

3.1	Free-Form Explanations after Reclassification . . . . .	22
3.2	Structured Explanations after Reclassification . . . . .	22
3.3	Critique Given in Study 3 and Associated Sentiment Classification . . . . .	25
3.4	Correlations between individual speech features and frustration . . . . .	27
4.1	Example of Multiple Objects and Warnings. Initial Q-values with indicators.	40
4.2	Example of Multiple Objects and Warnings. Learned Q-values. . . . .	40
6.1	Paired T-tests on Human Experience . . . . .	90
6.2	One-sample T-tests on Human Experience . . . . .	90
6.3	Interpretation of Cohen's kappa statistic . . . . .	104
6.4	Cohen's kappa for Study 3 . . . . .	105
6.5	Paired T-tests on ML Performance . . . . .	105
6.6	Wilcoxon Signed Rank tests on ML Performance . . . . .	106
6.7	Percentage of Participants who tried to 1) always follow the Shortest path, 2) always follow the AvoidRadiation path, or 3) switched between the Shortest and AvoidRadiation paths. . . . .	110
7.1	Frustration: Repeated-Measured ANOVA . . . . .	127
7.2	Perceived Performance: Repeated-Measured ANOVA . . . . .	127
7.3	Transparency: Repeated-Measured ANOVA . . . . .	128

7.4	Immediacy: Repeated-Measured ANOVA . . . . .	129
7.5	Perceived Intelligence: Repeated-Measured ANOVA . . . . .	130
7.6	Training Time: Repeated-Measured ANOVA . . . . .	131
7.7	Amount of Advice: Repeated-Measured ANOVA . . . . .	132
7.8	Earned Avg Reward: Repeated-Measured ANOVA . . . . .	133
7.9	Number of Steps: Repeated-Measured ANOVA . . . . .	133
7.10	Percentage of participants who mentioned in their long responses certain features that contributed to the user experience. . . . .	135
7.11	Cohen's kappa for Study 4 . . . . .	136

## LIST OF FIGURES

1.1	Human-Agent Information Flow . . . . .	3
2.1	Reinforcement Learning Diagram [2] . . . . .	13
3.1	Sentiment Classification Decision Tree. . . . .	22
3.2	Prosody Feature Importance Ranking. D1: First Derivative. D2: Second Derivative. . . . .	28
3.3	Ensemble Random Forest Model Results using Top Four Most Important Features. . . . .	29
4.1	Work Flow Chart . . . . .	40
4.2	A Participant’s View of the Mario Bros. Game . . . . .	41
4.3	Comparison of Explanation Formats. . . . .	42
4.4	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 <i>jump</i> . 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. . . . .	49
4.5	Cumulative Reward from Survey. . . . .	51
4.6	Reward for Chasms from Survey. . . . .	52
4.7	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. . . . .	53



4.8	Comparing the Reward of Good, Mediocre, Adversarial, and No Advice when a coin is northeast of Mario from Survey. . . . .	53
4.9	Impact of Warnings on Reward for Coins and Participant 3 from Survey. . .	54
5.1	Simple Force Model. Actions are an external force acting on the agent, and ‘friction’ determines the amount of time the action will be followed after the advice is given. . . . .	60
5.2	Radiation World Initial Condition . . . . .	67
5.3	Advice given to simulation to avoid radiation. . . . .	68
5.4	How the amount of advice provided impacts performance. (advice given 20, 50, and 90 percent of the time) . . . . .	69
5.5	Minimal advice for two paths. . . . .	69
5.6	Generalization through time. (advice given 20, 50, 90 percent of the time) .	72
5.7	Comparison of Newtonian Action Advice and Policy Shaping . . . . .	72
5.8	Comparison of the human experience. . . . .	73
6.1	Advice applies to the next action after language processing . . . . .	82
6.2	The credit attribution problem with critique . . . . .	82
6.3	Radiation World Initial Condition . . . . .	84
6.4	Agent Setup . . . . .	85
6.5	Human Factors Metrics. For all metrics save frustration, higher values represent a better human experience (better performance, greater transparency, more immediate, and more intelligent). For frustration, higher values indicate a higher-level of frustration, and therefore a worse human experience. .	89
6.6	Frustration Direct Comparison . . . . .	90
6.7	Transparency Direct Comparison . . . . .	95
6.8	Perceived Intelligence Direct Comparison . . . . .	97

6.9	Perceived Performance Direct Comparison . . . . .	103
6.10	Immediacy Direct Comparison . . . . .	104
6.11	ML Performance Comparison. . . . .	106
6.12	Radiation World with the 1) Shortest Path to Goal and 2) Path to Goal that Avoids Radiation. . . . .	108
6.13	Histogram: for each participant, the mean distance (averaged across episodes) from the Shortest path. . . . .	109
6.14	Example of a participant training the agent to follow different paths in dif- ferent episodes using the Newtonian Action Advice agent. . . . .	111
6.15	Summary of factors that impact frustration, transparency, and perceived intelligence based on participants' free responses. . . . .	112
7.1	Radiation World Initial Condition . . . . .	122
7.2	Human Factors Results for NAA Variations . . . . .	125
7.3	Human Factors Results for NAA Variations . . . . .	125
7.4	Frustration Results for NAA Variations . . . . .	126
7.5	Perceived Performance Results for NAA Variations . . . . .	127
7.6	Transparency Results for NAA Variations . . . . .	128
7.7	Immediacy Results for NAA Variations . . . . .	129
7.8	Perceived Intelligence Results for NAA Variations . . . . .	130
7.9	Objective Metric Results for average values of NAA Variations . . . . .	131

## SUMMARY

The widespread integration of robotics into everyday life requires significant improvement in the underlying machine learning (ML) agents to make them more accessible, customizable, and intuitive for ordinary individuals to interact with. As part of a larger field of interactive machine learning (IML), this dissertation aims to create intelligent agents that can easily be taught by individuals with no specialized training, using an intuitive teaching method such as critique, demonstrations, or explanations. It is imperative for researchers to be aware of how design decisions affect the human's experience because individuals who experience frustration while interacting with a robot are unlikely to continue or repeat the interaction in the future. Instead of asking how to train a person to use software, this research asks how to design software agents so they can be easily trained by people.

When creating a robotic system, designers must make numerous decisions concerning the mobility, morphology, intelligence, and interaction of the robot. This dissertation focuses on the design of the interaction between a human and intelligent agent, specifically an agent that learns from a human's verbal instructions. Most research concerning interaction algorithms aims to improve the traditional ML metrics of the agent, such as cumulative reward and training time, while neglecting the human experience. My work demonstrates that decisions made during the design of interaction algorithms impact the human's satisfaction with the ML agent. I propose a series of design recommendations that researchers should consider when creating IML algorithms.

This dissertation makes the following contributions to the field of Interactive Machine Learning: (1) *design recommendations for IML algorithms* to allow researchers to create algorithms with a positive human-agent interaction; (2) two new *IML algorithms* to foster a pleasant user-experience; (3) a 3-step *design and verification process for IML algorithms using human factors*; and (4) new methods for the *application of NLP tools to IML*.

# **CHAPTER 1**

## **INTRODUCTION**

The widespread integration of robotics into everyday life requires significant improvement in the underlying machine learning (ML) agents to make them more accessible, customizable, and intuitive for ordinary individuals to interact with. As part of a larger field of interactive machine learning (IML), this dissertation aims to create intelligent agents that can easily be taught by individuals with no specialized training, using an intuitive teaching method such as critique, demonstrations, or explanations. It is imperative for researchers to be aware of how design decisions affect the human’s experience because individuals who experience frustration while interacting with a robot are unlikely to continue or repeat the interaction in the future. Instead of asking how to train a person to use software, this research asks how to design software agents so they can be easily trained by people.

When creating a robotic system, designers must make numerous decisions concerning the mobility, morphology, intelligence, and interaction of the robot. This dissertation focuses on the design of the interaction between a human and intelligent agent, specifically an agent that learns from a human’s verbal instructions. Most research concerning interaction algorithms aims to improve the traditional ML metrics of the agent, such as accumulated reward and training time, while neglecting the human experience. My work demonstrates that decisions made during the design of interaction algorithms impact the human’s satisfaction with the ML agent. I propose a series of design recommendations that researchers should consider when creating IML algorithms.

### **1.1 Thesis Statement**

Verbal communication is a rich medium that is currently underutilized in teaching Reinforcement Learning (RL) algorithms. This dissertation will demonstrate that 1) the nature

of the language instruction used by a human teacher to train an RL algorithm affects a person's experience and satisfaction of teaching an agent; 2) the information people verbally provide is efficiently communicated to the agent; and 3) the RL agents taught using verbal instruction achieve a reasonable level of performance.

### 1.1.1 Thesis Foci

The three foci of the thesis statement are discussed below.

*1) **Human Experience.** The nature of the language instruction used by a human teacher to train an RL algorithm affects a person's experience and satisfaction of teaching an agent.* Through a series of studies, my work will demonstrate how and why decisions made during the design of interaction algorithms impact the human's experience with the agent. A selection of design decisions studied include: what teaching method the agent can use; what format of explanation an agent should ask a human for; how the human's feedback is incorporated; whether the agent generalizes feedback through time; and whether the agent allows instructions to inform about future actions.

*2) **Efficiency.** The information people verbally provide is efficiently communicated to the agent.* This work will explore how a small amount of verbal instruction can be used to train an agent. Also, to more efficiently use verbal instruction I will utilize sentiment analysis to expand the vocabulary available to the instructor. Finally, this work will explore how agents should ask for explanations to most efficiently obtain high-quality information from operators.

*3) **Performance.** The RL agents taught using verbal instruction achieve a reasonable level of performance.* I will demonstrate that designing RL agents that improve the human experience with the agent does not significantly detract from the agent's capabilities.

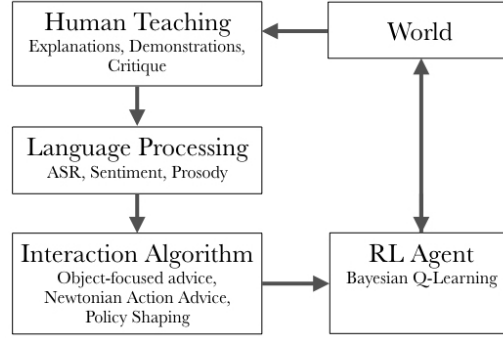


Figure 1.1: Human-Agent Information Flow

## 1.2 Agent Overview

The three foci of the thesis are supported by four studies. In each of the studies, a human provides instruction to a machine learning agent. Throughout the dissertation, an ‘agent’ is an instantiation of a machine learning algorithm.

Figure 1.1 shows the flow of information for all agents designed or studied in this work. First, verbal instruction is provided by a human teacher and sent to language processing. The processed human input is sent to an interaction algorithm that allows the instructions to be interfaced with an RL algorithm. The RL agent takes actions and learns from a reward signal. In certain cases, people might watch how the agent acts in the world and modify their instructions accordingly.

### 1.2.1 Methods of Human Instruction

The three methods of human instruction used in this dissertation are: explanations, advice (i.e. verbal demonstrations), and critique. In a classroom setting, an explanation would be a professor lecturing on material, a demonstration would be a professor going through an example problem, and critique would be returning graded homework assignments.

In this work, **explanations** are simple sentences provided by a human teacher to describe how to perform a task. For example, in the Mario domain an explanation might be, “Move right toward the goal. Jump on enemies. Do not fall into chasms.” Explanations are

provided before the agent tries to learn the task.

**Advice**, which can be thought of as a verbal demonstration, is when a human teacher acts like a backseat driver and interactively tells the agent what action to take immediately in the future. For example, if the person thinks the agent should move left, the person’s advice would be “move left.”

**Critique** is binary positive or negative feedback such as, “good job,” or “don’t do that.” Similar to advice, critique is provided interactively in real-time by the human teacher. Unlike advice, critique informs the agent whether its past actions were productive or detrimental.

My dissertation investigates if the method of human instruction can affect a human teacher’s experience with the agent. For example, does the teaching method impact how intelligent people perceive the agent to be? Is one teaching method more rhetorically positive than another? Understanding how interaction methods affect a human teacher’s experience is important because it sheds light on design decisions that can negatively impact a person’s perception of the robot’s capabilities.

### 1.2.2 Language as an Interface

Language was chosen as an interaction method because language is a universal interface - a new interaction method will not need to be created for each robot or domain.

Learning from natural language instructions poses many challenges and is not a part of interactive machine learning that has been extensively studied. First, 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, which complicates grounding words to states and actions. Second, explanations also tend to include information that is not actionable, like background knowledge. Third, many natural language explanations do not include state information, which makes it very difficult to determine where and when to use the advice. The challenge is further compounded because, in

English, written language is more formal than spoken. Finally, human teachers fluctuate rapidly between levels of abstraction.

### 1.2.3 Interaction Algorithms

Three interaction algorithms were studied in this work. I designed **Object-focused advice** to learn from explanations in Study 1. **Policy Shaping** is an algorithm that learns from positive and negative critique. It was created by previous researchers to learn from people clicking + and - buttons, but I slightly modified it to work with interactive verbal critique in Studies 2 and 3. I designed **Newtonian Action Advice** to learn from advice for Studies 2, 3, and 4.

## **1.3 Contributions**

To support the thesis statement, this dissertation makes the following contributions to the field of Interactive Machine Learning:

- **Design Recommendations for IML algorithms** IML algorithms have historically been designed to improve objective metrics rather than the human experience. Researchers design new algorithms to decrease training time or increase the earned reward. For example, Policy Shaping was created to use critique as policy information, which was shown to be more efficient than using critique as a reward signal.

My dissertation studies IML algorithms from a human factors perspective in order to learn how to design algorithms that improve the human's experience. Throughout this work, I studied how the interaction and nature of the ML algorithm can affect a person's experience with the agent. I identified several aspects of the interaction of IML algorithms that impact the human's experience with the agent and turned these into design recommendations. These recommendations will allow IML researchers to design algorithms with a positive human-agent interaction.



- **IML algorithms** Because the IML field has not employed user-focused design, IML algorithms do not create a positive human-agent interaction. For example, Policy Shaping uses critique to increase or decrease the probability an action will be taken again if the same state is seen in the future. It does not immediately change the agent’s behavior. This causes frustration and feelings of powerlessness in users [1].

I have developed two IML algorithms to foster a positive human experience. Newtonian Action Advice, presented in Chapter 5, connects action advice to an RL agent with the goal that it improve the human’s experience compared to other interactive algorithms. The algorithm leverages a simple physics model to provide an agent that acts in a way people expect and find non-frustrating. I also created Object-Focused advice, developed in Chapter 4, that looks at how to ground simple instructional explanations to an RL algorithm, creating a method for generalizing advice across the state space since people do not provide state-specific information.

- **Verification Process for IML algorithms using Human Factors** Many of the existing machine learning algorithms that learn from human instructions are evaluated using oracles (i.e. simulated feedback) and focus on how quickly the agent learns. While this is valuable information, it ignores important aspects of the human-agent interaction such as frustration. An oracle will never get frustrated with, confused by, or lose trust in an agent. I suggest that validating interaction algorithms with oracles and analyzing traditional RL metrics such as cumulative reward and training time is only one step in the verification process. In addition to RL metrics, interaction algorithms should be validated by measuring peoples’ experiences with agents using human factors, such as frustration. If people become very frustrated with the agent because it does not learn as they expect, people will not want to use the agent regardless of the efficiency of an algorithm’s theoretical learning curves.

I propose a 3-step design and verification method in which 1) the algorithm is de-

signed to create a positive user experience, 2) the algorithm is tested in simulation, and 3) the algorithm is tested in a human-subject experiment and compared to other algorithms with human factors metrics in addition to machine learning metrics.

- **Application of NLP tools to IML** I have developed methods of applying sentiment and prosody to IML.

Sentiment analysis is an NLP tool that has traditionally been used to classify book, movie, and product reviews into positive and negative. Sentiment analysis has not been used for action selection before. I developed two new ways of using sentiment analysis: to filter natural language critique into positive and negative, and to filter natural language advice into ‘what to do’ and ‘what not to do’ actions. When applied to critique, sentiment analysis also allows people to provide instructions without being restricted to a limited set of words, which creates a more natural and intuitive interface for the human.

In linguistics, prosody is an analysis of the intonation, stress, tempo, rhythm, and emotion of a speaker’s utterances. Prosody contains significant information in the audio signal that is lost when algorithms transcribe the speech to text with automatic speech recognition and then operate only on the semantic meaning. This dissertation developed a method to use prosody as an objective measure of frustration by comparing the linguistic feature values to the frustration metric reported in questionnaires.

## **1.4 Organization**

Chapter 2 contains a review of related literature.

Each of the inner chapters has a common structure. The Introduction contains a list of research questions as well as a summary of the algorithm development and evaluation method. The Introduction is followed by Algorithm Development, Method, Results and Discussion, and the Conclusion. After the Conclusion is the RQ Results Summary, which

is a list of the answered research questions, how each question supports a thesis focus, and which study validated each question.

Chapter 3 covers methods of applying NLP tools to IML, including using sentiment analysis to classify verbal critique and action advice. This chapter can be read in order or referred to while reading later chapters covering the associated studies.

Chapter 4 details Study 1, a human-subject experiment that tests Object-focused advice, an IML algorithm that connects explanations to Reinforcement Learning. Study 1 investigates how to learn from explanations that do not contain state information, and what format an explanation should take.

Chapter 5 presents Newtonian Action Advice, an IML algorithm that incorporates a human’s verbal action advice with Reinforcement Learning. Study 2 provides oracle simulations of Newtonian Action Advice and compares its theoretical performance to Policy Shaping and Bayesian Q-learning. The chapter also acts as a template for IML design and verification.

Chapter 6 delves into Study 3, a human-subject experiment comparing two teaching interaction methods: Policy Shaping critique and Newtonian Action Advice. The study identified several factors in the design of IML algorithms that impact the user experience.

Chapter 7 follows up on the advice vs. critique study in Chapter 6 by isolating and testing three factors that impact the human teacher’s experience with the RL agent in Study 4.

Chapter 8 discusses how this dissertation will impact the IML field.

## CHAPTER 2

### LITERATURE REVIEW

This chapter overviews fundamental concepts of this dissertation, including reinforcement learning, learning from human teachers, and natural language processing. Some specific topics will be referenced and explained throughout the dissertation.

#### 2.1 Reinforcement Learning (RL)

Reinforcement learning (RL) is a branch of machine learning in which intelligent agents learn from the environment which actions to take by receiving a signal of rewards and punishments [2]. RL has been a topic of study in behavioral psychology [2, 3]. B. Skinner compared the evolution of living things through natural selection with the shaping of individual behavior through reinforcement [4]. People will likely repeat an action in a circumstance if they receive positive reinforcement, and will likely not repeat the action if given negative reinforcement.

Most RL algorithms are modeled as Markov Decision Processes (MDPs), which learn policies by mapping states to actions such that the agent's expected reward is maximized. An MDP is a tuple  $(S, A, T, R, \gamma)$  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  $\gamma$ , a discount factor in which  $0 \leq \gamma \leq 1$ .

The followed subsections will introduce Object-Focused Q-Learning used in Study 1 and Bayesian Q-Learning used in Studies 2-4.

### 2.1.1 Object-Oriented RL

Object-Oriented Markov Decision Process (OO-MDPs) are an extension of MDPs. An OO-MDP is a model-based representation that uses a fixed-length feature vector of object relations [5]. 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.

Researchers have explored using Object-Oriented MDPs as a state vector [6]. If an element in the state vector was true, it corresponded to a simple reward of +1. The state was grounded directly to a reward function. Natural language commands were used, but a strict grammar had to be used. The final behavior of an RL agent is very sensitive to the reward function, so a simple +1 based on elements of the state is not guaranteed to produce desired behavior. The state representation is at a high level of abstraction and is not learned from low-level input (like pixels).

While an OO-MDP is an interesting method of connecting natural language to RL, it is not flexible enough or guaranteed to produce the right behavior, so I used Object-Focused Q-Learning, which is a modified version of OO-MDPs. Object-Focused Q-Learning (OF-Q) with an Off-Policy TD Control Q-Learning algorithm was used to train the Q-values for each object’s policy in Study 1. Unlike OO-MDPs, OF-Q is model-free and does not have a fixed-length feature vector [7]. 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 Study 1 focuses on the efficacy of using sentiment as a filter on natural language explanations, a well-understood tabular algorithm was used. In Equation 2.1,  $s_{o_t}$  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  $a_t$ ,  $\alpha$  is the learning parameter, and  $\gamma$  is the discount parameter for expected future rewards.

$$Q(s_{o_t}, a_t) \leftarrow (1 - \alpha)Q(s_{o_t}, a_t) + \alpha[r + \gamma \max_a Q(s_{o_{t+1}}, a)] \quad (2.1)$$

Reinforcement learning agents must trade off 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.

$$\pi(s) = \arg \max_a \max_o Q(s_o, a) \quad (2.2)$$

While natural language explanations can be incorporated into many machine learning algorithms, I used Object-focused Q-learning (OF-Q) to represent explanations in Chapter 4 for two main reasons [7]. 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 the Mario game domain.

### 2.1.2 Bayesian Q-Learning

Bayesian Q-Learning was the underlying ML agent used for both the critique and action advice conditions in Chapters 5, 6, and 7. Bayesian Q-Learning is an RL algorithm in which the utility of state-action pairs are represented as probabilistic point estimates of the expected long term discounted reward [8].

## **2.2 Learning from Human Teachers**

Interactive Machine Learning (IML) is a subfield of machine learning in which a human provides instructions (e.g. explanations, demonstrations, or critique) to help the agent learn a task. IML agents typically use RL algorithms for several reasons. First, RL provides a way for the agent to learn by interacting with its environment, including a person. Peo-

ple learn by interacting with their environment, so the IML field chooses RL methods that replicate this process. Second, a human teacher is unlikely to provide exhaustive instructions for the entire state space. RL allows an agent to learn policies based on the agent's experience and earned reward in addition to human input. Similarly, human instructions can be used to help an RL agent visit high-value states earlier than the agent's exploration policy, causing an RL agent to reach a high-scoring policy with less training time. The human's instructions can be incorporated into an RL algorithm in many ways, which will be detailed in Section 2.2. Like other works in IML [9, 10], this work incorporates human instruction into RL agents; however, I focus on the human teacher's experience.

Three commonly studied methods of human instruction are explanations, demonstrations, and critique [11]. Explanations transfer knowledge through non-interactive 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. Interaction algorithms have been created that use different methods of human instruction. Object-Focused Q-Learning (OF-Q) was used in Study 1 to utilize human explanations. Policy Shaping was used for critique in Studies 2-3, and Newtonian Action Advice was created for action advice in Studies 2-4.

There are many ways to connect human instructions with RL. Figure 2.1 shows a simple RL diagram in which an agent is in a state and takes an action in the environment; the agent is now in a new state and earns a punishment or reward. The cycle continues since the agent must decide which action to take in the new state. Explanations and advice can be converted to (state, action) pairs, so when the agent is in a state, it takes the advised action. Critique can be used directly as a reward signal [12] or as policy information about whether to repeat an action if the same state is seen again in the future [13].

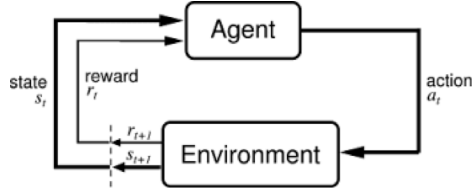


Figure 2.1: Reinforcement Learning Diagram [2]

The way the human instructions are incorporated into the RL algorithm defines the nature of the interaction between the human and agent, including whether people have to repeat themselves, how quickly the agent incorporates the instructions, and how transparent the agent’s decisions appear. One of the main purposes of this dissertation is to study how to design the connection between the human and RL agent to create a human-friendly experience.

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 [14]. An agent that learns from demonstrations is effected 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 reinforcement learning to approximate the reward function the teacher was following [15]. 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 [13]. Critique can also be used directly as a reward signal to tell the agent how positive or negative its actions were in certain states [11]. All of these approaches link human input to specific states.



### 2.2.1 Learning from Critique

Many IML algorithms learn from critique. Initially, critique was incorporated into RL as a reward signal [12]. This was shown to be less than optimal because people provide an unpredictable signal and stop providing critique once the agent learns the task [16]. Later, it was shown that it is more efficient to use critique as policy information rather than a reward signal [17, 18]. An example of this is Policy Shaping, an IML algorithm that enables a human to teach an agent using critique, which is incorporated into a Bayesian Q-learning agent as policy information [13]. Cederborg, Grover, Isbell, and Thomaz [19] investigated how to interpret silence while learning from critique with Policy Shaping.

There are some basic behavioral templates that people expect in a teaching situation. For example, if I were to tell my nephew, “Stop,” I would expect him to immediately change his behavior. Algorithms like Policy Shaping, which more efficiently use critique as policy information instead of directly as a reward signal, do not adhere to this teaching template. Chapter 6 discusses how this causes people to be confused and frustrated, and explains how Policy Shaping using critique creates a worse user experience than Newtonian Action Advice [1].

### 2.2.2 Learning from Explanations and Advice

Various forms of advice have been developed in other work, including linking one condition to each action [20], and linking a condition to rewards [6]. Several connect conditions to higher-level actions that are defined by the researcher instead of primitive actions [20, 21, 22]. Argall, Browning, and Veloso [23] creates policies using demonstrations and advice. Meriçli, Klee, Paparian, and Veloso [24] parses language into a graphical representation and finally to primitive actions. Maclin, Shavlik, Torrey, Walker, and Wild [20] has the person provide a relative preference of actions, whereas the agent determines the order of preferred actions in our work. Sivamurugan and Ravindran [25] explored learning multiple interpretations of instructions. Tellex, Kollar, Dickerson, Walter, Banerjee, Teller, and Roy

[26] represents natural language commands as probabilistic graphical models. Similar to this work, the advice in [23] does not require people to give specific numbers for continuous state variables, but uses a set of predefined advice operators. The advice developed in Study 1 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.

Many researchers incorporate advice using IF-THEN rules and formal command languages [20, 21]; 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 meters of this object, do this action.” Developing a parser is labor intensive, and prior knowledge like distance calculations must be encoded. Our work 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 non-experts 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. Kuhlmann, Stone, Mooney, and Shavlik [21] can adjust for bad advice by learning biased function approximation values that negate the advice. Maclin, Shavlik, Torrey, Walker, and Wild [20] uses a penalty for not following the advice that decreases with experience. The Newtonian Action Advice (NAA) developed in Chapter 5 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. With NAA, the advice can be overwritten by new, contradictory advice in the future.

## **2.3 Natural Language Processing (NLP)**

### 2.3.1 Automatic Speech Recognition (ASR)

Automatic Speech Recognition (ASR) is how the human’s verbal instructions are transcribed to written text. This dissertation used the Sphinx ASR software [27].

### 2.3.2 Sentiment Analysis

Sentiment analysis is an NLP tool that has traditionally been used to classify book, movie, and product reviews into positive and negative [28]. Sentiment analysis has not been widely used for action selection. This dissertation develops two new applications for sentiment analysis that are discussed in Chapter 3. In Study 1, I developed a method for using sentiment analysis to classify natural language advice into advice of ‘what to do’ and warnings of ‘what not to do’ [29]. In Study 3, I developed a method for using sentiment to filter natural language critique into positive and negative. Used in this new manner, sentiment analysis allows people to provide critique without being restricted to a limited set of words.

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 work uses Stanford’s deep learning sentiment analysis software, which builds a representation of an entire sentence instead of looking at words independently [30]. This sentiment tool uses Recursive Neural Tensor Networks and the Stanford Sentiment Treebank [31]. The Stanford Sentiment Treebank is a set of labeled data corpus of fully-labeled parse trees trained on the dataset of movie reviews from rottentomatoes.com [32].

Many approaches to learning from language instruction require people to provide instructions using specific words, often in a specific order or format [24]. Thomason, Zhang, Mooney, and Stone [33] worked to get around limitations like keyword search by creating

an agent that learns semantic meaning from the human. In this work we created a method of using sentiment analysis to filter verbal critique into positive and negative, which furthers the goal of allowing people to provide verbal instructions without being limited to a specific dictionary of words.

### 2.3.3 Prosody

In linguistics, prosody is an analysis of the intonation, stress, tempo, rhythm, and emotion of a speaker's utterances. Prosody contains significant information in the audio signal that is lost when algorithms transcribe the speech to text with automatic speech recognition and then operate only on the semantic meaning. Chapter 3 develops a model to use prosody as an objective measure of frustration by comparing the linguistic feature values to the frustration metric reported by human teachers in questionnaires.

Prosody has been used as a feature within an ML classifier to objectively detect whether a patient is suffering from depression [34]. Kim, Leyzberg, Tsui, and Scassellati [35] showed that humans teaching robotic students in unscripted interactions varied their affective vocalizations based on the robot's performance. Thomason, Nguyen, and Litman [36] investigated which prosodic features correlate with student entrainment in a tutoring dialogue system.

The software package Praat is being used to analyze prosodic features of the audio data collected in Study 3 [37].

## **CHAPTER 3**

### **APPLICATION OF NLP TO IML**

This chapter covers three new applications of NLP tools to IML. These methods were tested in Studies 1 and 3, details of which can be found in Chapters 4 and 6.

#### **3.1 Introduction**

##### 3.1.1 Research Questions

- Can sentiment analysis from the NLP field be used to filter natural language explanations into advice of what to do and warnings of what not to do?
- Can sentiment analysis from the NLP field be used to classify natural language critique into positive and negative?
- Can sentiment analysis from the NLP field be used to allow human teachers to provide critique using an unrestricted vocabulary?
- Can prosodic features be used as an objective measure of frustration in place of NASA-TLX subjective questions?

##### 3.1.2 Development

This chapter will develop methods to 1) use sentiment analysis to filter advice, 2) use sentiment analysis to filter critique, and 3) use prosodic features as an objective metric for frustration.

### 3.1.3 Method Summary

The use of sentiment as an advice filter will be tested in Study 1. The use of sentiment as a critique filter as well as prosody as a frustration metric will be verified in Study 3.

## **3.2 Sentiment Analysis as an Advice Filter**

A problem addressed in Study 1 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.

A contribution of this work 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 work 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.

I tested the sentiment filter and Object-focused advice in a human-subject experiment conducted in the popular game domain Mario Brothers. For details on Study 1, please see Section 4.3.

I started by classifying entire sentences as either positive/neutral or negative. I 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, I 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 I 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.”*

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, I split each sentence into clauses and determined the sentiment of each clause. If at least half the clauses were positive/neutral, I 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 Figure 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 3.1 and 3.2 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 nega-



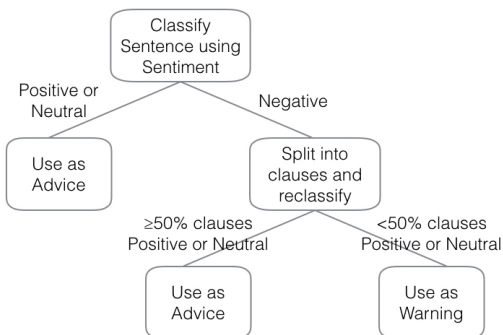


Figure 3.1: Sentiment Classification Decision Tree.

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

Table 3.1: Free-Form Explanations after Reclassification

tive 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.

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.

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

Table 3.2: Structured Explanations after Reclassification

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.

### 3.2.1 A Question of Ethics

Another thing to consider: is there an ethical question? Here, sentiment is applied to determine 'what to do' and 'what not to do.' This is essentially the same as the field of ethics. Ethics determines what actions we should take or not take. The Merriam-Webster dictionary defines ethics as "the discipline dealing with what is good and bad and with moral duty and obligation." The sentiment analysis tool is acting as the ethical decision-maker for the agent.

If we retrain the sentiment analysis model for violent game domains in which the agent kills other actors, we are essentially building an agent that thinks killing is good, because the sentiment classification says so. No one really cares if an agent kills in a video game because the consequences are not real to us. An agent cannot distinguish between real

scenarios and simulations; the consequences of punishments and rewards are always real for the agent, regardless of domain.

Using sentiment analysis as the ethical decision-making ‘module’ of A.I. agents is an interesting thought, but with the potential for easy abuse. Depending on how accurate the sentiment analysis model is, the agent may have a good chance of acting in a way most people would consider a fairly ethical manner in line with mainstream behavior. However, retraining the sentiment model could radically change the agent’s behavior; whoever chooses the data to retrain the model would be capable of determining the agent’s ethical philosophy, at least in the short term.

### 3.2.2 Conclusion on Sentiment from Study 1

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.

## **3.3 Sentiment Analysis as a Critique Filter**

During Study 3 in Chapter 6, one of the agents learned from critique. A person would watch the agent act and provide positive and negative critique, such as “good job,” or “bad job.” It is not intuitive to restrict people to only saying “good” and “bad” because 1) the human teachers need to be taught exactly which words they can say before interacting with the agent, and 2) the human teachers need remember to conform to that behavior while interacting with the agent. Study 1 (Section 4.4.2) shows that increasing the cognitive load by requiring a certain format of an explanation leads to poor quality of instructions given

by the human, even in a non-interactive setting. In order to create an intuitive interaction for the human and get high-quality instructions, researchers should try to create agents that learn from an unrestricted vocabulary.

Using sentiment analysis to filter critique into positive or negative allowed people to provide verbal natural language critique without restricting their vocabulary. For example, a participant could give varied critique such as, “Good job,” “That’s great,” “That is a bad idea,” and “You’re wasting time”. The critique agent used Policy Shaping to incorporate the positive and negative feedback.

Examples of critique provided during the experiment along with the sentiment classification are provided in Table 3.3. Sentiment analysis allows the agent to understand that statements like “they’re dead” and “turn around” are bad and mean the agent should make different decisions in the future. One drawback of allowing people to use an unrestricted vocabulary is it opens the door to ambiguous feedback. If someone angrily says, “that’s it,” the feedback would indicate that the person is very frustrated and fed up with the agent; however, if someone says “that’s it” in an encouraging manner, the feedback would be positive. Sentiment analysis that solely uses the transcribed text on individual sentences has no way to distinguish correctly between the two.

Table 3.3: Critique Given in Study 3 and Associated Sentiment Classification

Critique	Sentiment
“yes that’s right”	positive
“turn around”	negative
“good job”	positive
“keep going back”	positive
“okay come on”	negative
“that’s it”	negative
“they’re dead”	negative
“good I’m happy”	positive

Details of the critique agent and experimental method can be found in Section 6.4.

### 3.3.1 Conclusion on Sentiment from Study 3

Sentiment analysis can be used to filter natural language critique into positive and negative without restricting the vocabulary available to the human teacher.

## **3.4 Prosody**

The purpose of this section is to determine whether speech features can be used to predict frustration. Currently in human-subject research, people are required to complete a task and then score aspects of their experience, such as frustration, using subjective metrics like the NASA-TLX questionnaire. It would be helpful if there existed a real-time, objective metric for frustration based on nothing more than an analysis of the user's voice. Eventually, field of IML may benefit from incorporating emotional analysis into agents, a feat which has not yet been accomplished.

During Study 3 in Chapter 6, the audio was recorded of each participant training each agent. After participants trained an agent, they scored their frustration of working with the agent. Details of the study can be found in Section 6.4.

The audio recordings were analyzed using the Praat NLP software, which took the recordings as input and output a time history of NLP features including the pitch, intensity, and the first to fifth formants. *Pitch* is the frequency of the sound wave; females generally have higher-pitched voices than males. *Intensity* is the directional power carried by sound waves, and is related to the perceived loudness of a sound. In general, a higher intensity translates to a louder sound. However, a sound wave outside of a human's hearing range may have a non-zero intensity but zero loudness because it is not perceivable by humans [38]. A *formant* is a concentration of energy around a frequency. Many formants exist simultaneously in a vocal signal and are numbered incrementally from the lowest formant frequency; formants are separated by a span of approximately 200-1100 Hz. Formants vary through time based on factors such as if the person is vocalizing an

oral or nasal vowel or consonant. For example, antiresonances (i.e. high impedance or resistance at a frequency) in the vocal tract can lead to missing or attenuated formants with oral consonants, while resonance in the nasal cavity can lead to additional formants with nasal vowels [39]. Changes to the constriction of the vocal tract, larynx depression, as well as the position of the tongue and lips can change the formant frequencies.

For each NLP feature, the minimum, maximum, average, and standard deviation were calculated for the feature and its first and second derivatives. This resulted in 84 speech features for every audio recording, which had one associated frustration score.

First, I checked each of the 84 features separately to see if any were correlated with frustration. Four features were correlated with frustration: the minimum and standard deviation values of both the third and fifth formants. However, these were not strong correlations since all were less than 0.43, as seen in Table 3.4.

Table 3.4: Correlations between individual speech features and frustration

Speech Feature	Accept/Reject	p-value	$\rho$
Formant 3 Minimum	Reject	0.030	0.320
Formant 3 $\sigma$	Reject	0.032	-0.317
Formant 5 Minimum	Reject	0.004	0.422
Formant 5 $\sigma$	Reject	0.017	-0.350

Since none of the individual features were strongly correlated with frustration, I created a supervised learning model that used several features to predict frustration. I used an ensemble random forest regression model trained on the top four most important features, as seen in Figure 3.2. Importance was calculated using the *scikit* package in python. As the minimum and standard deviation values of both the third and fifth formants were individually correlated with frustration, it is unsurprising that these features were in the top seven most important features.

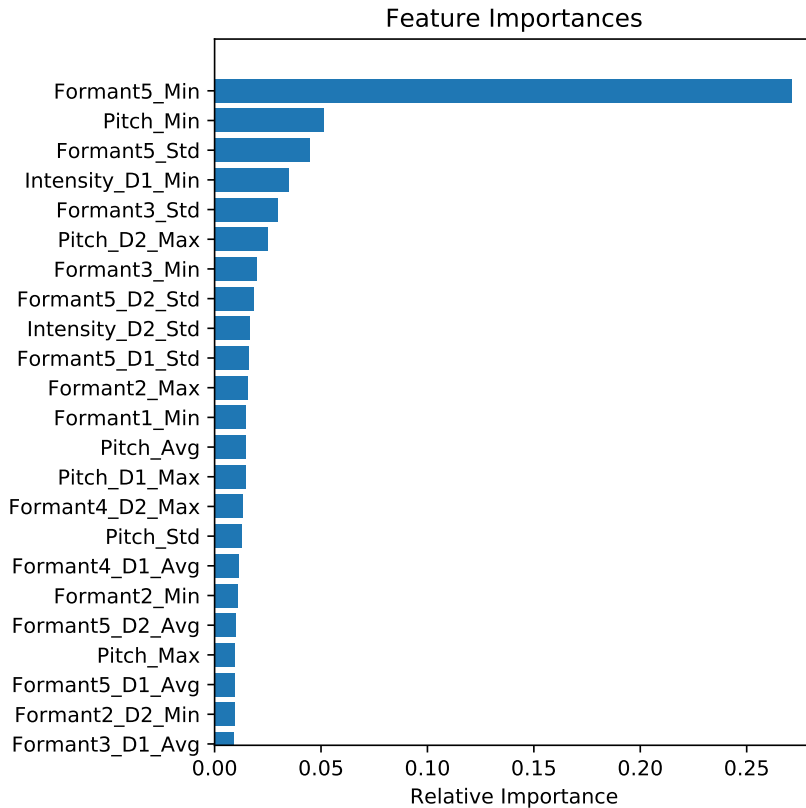


Figure 3.2: Prosody Feature Importance Ranking. D1: First Derivative. D2: Second Derivative.

The top four features with the highest calculated importance were: the minimum value of the fifth formant, the minimum pitch, the standard deviation of the fifth formant, and the minimum value of the first derivative of the intensity.

Multiple studies have found that a high pitch and/or intensity (i.e. a “raised voice”) are important predictors of frustration [40, 41, 42]. However, the results here show this is not the case. A low pitch paired with an unchanging intensity (loudness), be it high or low, are key features to predicting frustration. In my results, high pitch was ranked as the 20th feature by importance, and high intensity was even further down the list. This discrepancy indicates it is likely that people express frustration differently toward an IML agent than they would to another person.

It is algorithmically convenient that three of the top four features were minimum values as it is easy to keep track of ‘low watermarks’ in real time - only the lowest value needs to be stored. There is a method of keeping a running value of the standard deviation without storing a time history in memory; I recommend that approach be used for space considerations.

The four features with the highest relative importance were used to train an ensemble random forest regression model. The results are seen in Figure 3.3. The test data had a Spearman correlation value between the recorded and predicted values of 0.935.

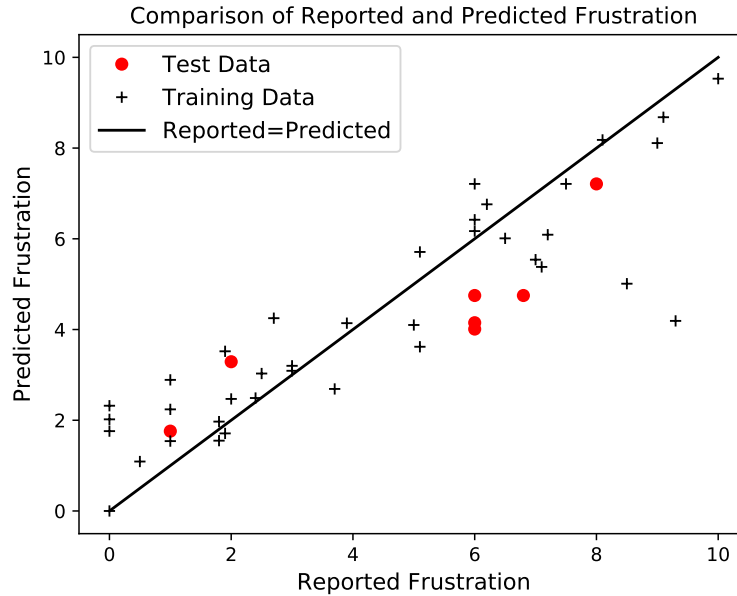


Figure 3.3: Ensemble Random Forest Model Results using Top Four Most Important Features.

### 3.4.1 Conclusion on Prosody from Study 3

An audio signal can be analyzed to predict frustration using NLP calculations rather than subjective metrics. This will allow IML algorithms in the future to have an objective, real-time estimate of the human’s frustration based solely on how the person is speaking. The most important speech features for predicting frustration with IML agents differs from those used for predicting frustration with another person. It is likely that people express



frustration differently toward an IML agent than they would to another person, which is important to take into account when building the frustration prediction model.

### 3.5 RQ Results Summary

- Sentiment analysis can be used to filter natural language explanations into advice of what to do and warnings of what not to do.
  - *Validation:* Study 1.
  - *Supports thesis:* Part 2 (efficiency) - this work enables us to more efficiently use verbal instruction.
- Sentiment analysis from the NLP field can be used to classify natural language critique into positive and negative.
  - *Validation:* Study 2.
  - *Supports thesis:* Part 2 (efficiency) - this work enables us to more efficiently use verbal instruction.
- Sentiment analysis from the NLP field can be used to allow human teachers to provide critique using a nearly unrestricted vocabulary.
  - *Validation:* Study 3.
  - *Supports thesis:* Part 2 (efficiency) - this work expands the vocabulary available to the instructor when using verbal critique. Part 1 (human experience) - a less-restricted vocabulary creates a more intuitive experience for the human.
- As few as 4 prosodic features can be used to train a model to predict frustration based on objective measures in place of NASA-TLX subjective questions.
  - *Validation:* Study 3.

- *Supports thesis:* Part 2 (efficiency) - this work explored a method to efficiently use verbal instruction to objectively measure frustration instead of relying on subjective questionnaires after task completion.
- *Future work:* Future IML researchers can determine the best methods of incorporating prosodic frustration measures into IML algorithms.

## CHAPTER 4

### STUDY 1: LEARNING FROM A *PRIORI* EXPLANATIONS

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 chapter discusses a software agent that learned to play the Mario Bros. game using explanations. The goals to improve learning from explanations were two-fold: to filter explanations into advice and warnings, and to learn policies from sentences without state information. I used sentiment analysis to filter explanations into advice of what to do and warnings of what to avoid. I developed Object-focused advice to represent what actions the agent should take when dealing with objects. An RL 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. I 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.

#### 4.1 Introduction

While there are many ways people teach, including demonstrations and critique, this chapter 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 computa-

tional 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 chapter 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 work 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 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. 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*.

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.

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.

#### 4.1.1 Research Questions

This work investigates methods to turn natural language explanations into actionable instruction for the IML agent to use.

- Can we learn policies from NL sentences that are not state specific?

- Can we link objects to actions and generalize over the state space?
- Does the format of asking a person for an explanation impact the quality of the explanation?

#### 4.1.2 Algorithm Development

In this chapter, an algorithm, Object-focused advice, will be developed to connect human explanations to RL.

#### 4.1.3 Method Summary

The research questions in this chapter were tested in Study 1, which was a human-subject experiment conducted in the Mario Bros. domain.

### **4.2 Algorithm Development: 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 [43]. 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,  $\epsilon$ -greedy exploration. Since  $\epsilon$ -greedy exploration is used, Object-focused advice has the convergence properties of  $\epsilon$ -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.

#### 4.2.1 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 twenty-five 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 twenty-five 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 Equation 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 Equation 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,  $\epsilon$ -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.  $\epsilon$  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



---

**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:   for each object  $\in$  objects do
7:     If this object has never been seen by the agent,
8:     create a new object policy
9:     If this object has never been seen in this state,
10:    add a state entry to the object's policy
11:   if  $\text{rand}(0, 1) > \epsilon$  then ▷ Exploit
12:     Initialize action and  $Q_{max} \leftarrow -\infty$ 
13:     for each object  $\in$  objects do
14:        $q_i \leftarrow \max_i(Q(\text{object.state}, a_i))$ 
15:       if  $q_i > Q_{max}$  then
16:          $Q_{max} \leftarrow q_i, \text{action} = a_i$ 
17:   else ▷ Explore
18:      $\text{action} = \text{random}\{\text{actions}\}$ 
19:   objectsOld = objects
```

---

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.

#### 4.2.2 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 4.1 and 4.2. 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 4.1 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 +2,000, warnings are -2,000, 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 ( $2,000 - 2,000 = 0$ ). 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 ( $Q = 2,000$ ) is better than walking right ( $Q = 0$ ), which is in turn better than moving left ( $Q = -2,000$ ). 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 4.2 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 4.2, the *total* row shows the agent has learned that the warning to avoid walking right into an enemy is much more

Table 4.1: 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	<b>2,000</b>	0	-2,000

Table 4.2: Example of Multiple Objects and Warnings. Learned Q-values.

Object	JumpRight	Right	Left
Coin	10	10	-1
Enemy	50	-50	0
Total	<b>60</b>	-40	-1

severe than walking left near a coin.

### 4.3 Study 1: Method

Two 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 (see Section 3.2) and assess the performance of Object-focused advice.

The experiment had four phases: familiarization, free-form explanations, structured explanations, and 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 of 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 Figure 4.1. The cumulative reward for each object was analyzed to evaluate the agent’s performance over 500 trials.



Figure 4.1: Work Flow Chart

### 4.3.1 Mario Domain

The experiment was conducted using the Mario Bros. platform from the 2009 Mario AI Competition [44], as seen in Figure 4.2. 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.



Figure 4.2: A Participant’s View of the Mario Bros. Game

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 22x22 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

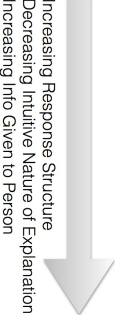
of the available information was used in the representation. Environmental objects like coins were included in the state space if they appeared in the 3x3 grid directly surrounding Mario’s location, while enemies were included if they appeared anywhere on the screen. This allowed us to determine how 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.

#### 4.3.2 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.

#### 4.3.3 Human explanations

Explanations were collected from human participants in three successive trials, in order of increasing structure and provided information, as seen in Figure 4.3. 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 subsections.



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

Figure 4.3: Comparison of Explanation Formats.

### *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.

### *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.

### *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.

#### 4.3.4 Post-Processing

##### *Sentiment Analysis as a 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 3.2.

##### *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.

### *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 Section 4.2. 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  $\epsilon_{min} = 0.15$ .  $\epsilon$ -greedy exploration was used.

## **4.4 Study 1: 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.



#### 4.4.1 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.

##### *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.

##### *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.

### *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.

### *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.

### *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 don’t 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.

#### 4.4.2 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.

##### *Advice from Explanation Formats*

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, <u>don'tRunInto</u>	jump, <u>don'tRunInto</u>						
	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	<u>walkThrough</u>		jump	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	jump			waitProceedCautiously	jump	<u>jump</u>	
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

Figure 4.4: 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.

Figure 4.4 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 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 Figure 4.4. In the free-form explanations, the advice had a somewhat varied vocabulary including *collect*, *jump*, *right*, *hitBottom*, *fireball*, *don't 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 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.

#### *Total Cumulative Reward*

Figure 4.5 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.

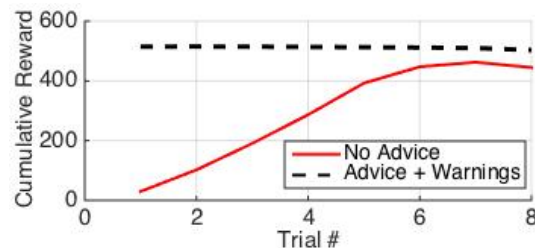


Figure 4.5: Cumulative Reward from Survey.

#### *Performance over an object's entire state space*

Figure 4.6 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 3x3 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.

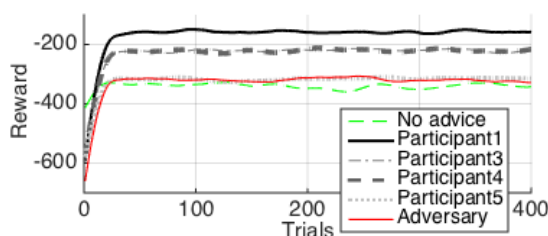


Figure 4.6: Reward for Chasms from Survey.

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.

#### *Object-level generalization: learning where advice applies*

Figure 4.7 shows the agent learns to which part of the 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.

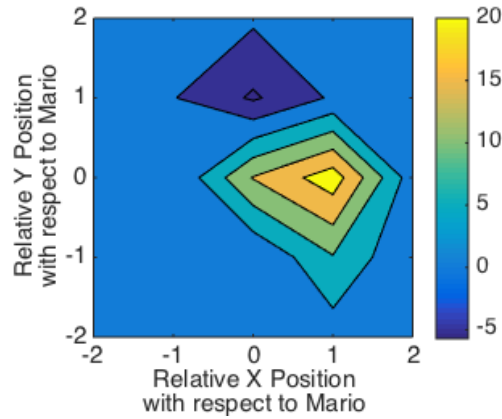


Figure 4.7: Visualizing Object-level Generalization in a Policy for Goombas from Survey. The color scale represents Q-values showing when to jump quickly to the right.

#### *Performance in a specific subset of an 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's review the performance in one specific subset of an object's state space. Figure 4.8 shows the results when a coin is northeast of Mario.

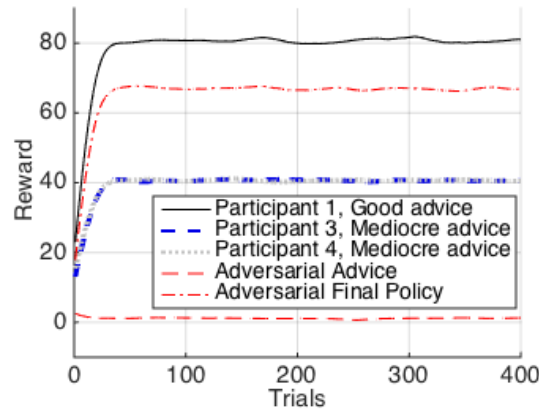


Figure 4.8: Comparing the Reward of Good, Mediocre, Adversarial, and No Advice when a coin is northeast of Mario from Survey.

It can be more difficult to recover from mediocre advice than adversarial (Figure 4.8). With adversarial advice, the agent recognizes quickly that the advice is harmful by earning 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  $\epsilon$ -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.

### *Advice+Warnings*

Figure 4.9 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.

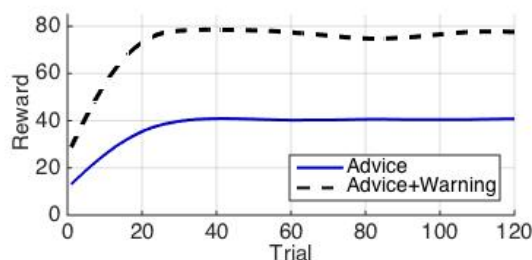


Figure 4.9: Impact of Warnings on Reward for Coins and Participant 3 from Survey.

## **4.5 Study 1: Conclusion**

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 aren’t state-specific since they reflect much of non-expert instruction.

#### **4.6 RQ Results Summary**

- Policies can be learned from sentences that do not contain specific state information, such as numbers.
  - *Validation:* Study 1.
  - *Supports thesis:* Part 2 (efficiency) - OF-advice allows a small amount of instruction to be used to train an agent. Part 3 (performance) - OF-advice uses human instruction without detracting from the agent’s capabilities.
- Using OF-advice, we can link objects to actions and generalize over the state space.
  - *Validation:* Study 1.
  - *Supports thesis:* Part 2 (efficiency) - OF-advice allows a small amount of instruction to be used to train an agent.
- The requested format of an explanation impacts the quality of the explanation.
  - *Validation:* Study 1. Explanations were collected from human participants in

three successive trials, in order of increasing structure and provided information.

- *Supports thesis*: Part 1 (human experience) - this work explored what format of explanation an agent should ask a human for.

## **CHAPTER 5**

### **STUDY 2: TEMPLATE OF DESIGN AND VERIFICATION OF IML**

#### **ALGORITHMS WITH NAA EXAMPLE**

Many of the existing machine learning algorithms that learn from human instructions are evaluated using simulated feedback and focus on how quickly the agent learns. While this is valuable information, it ignores important aspects of the human-agent interaction such as frustration. In this chapter, I present the Newtonian Action Advice agent, a new method of incorporating human verbal action advice with Reinforcement Learning (RL) in a way that improves the human-agent interaction. In addition to simulations, I validated the Newtonian Action Advice algorithm by conducting a human-subject experiment. The results show that Newtonian Action Advice can perform better than Policy Shaping, a state-of-the-art IML algorithm, both in terms of RL metrics like cumulative reward and human factors metrics like frustration.

### **5.1 Introduction**

This chapter introduces an algorithm, Newtonian Action Advice, which incorporates a human’s verbal action advice with Reinforcement Learning (RL). The algorithm leverages a simple physics model to provide an agent that acts in a way people expect and find non-frustrating.

I validate the algorithm by first constructing oracles to simulate human feedback to compare Newtonian Action Advice (NAA) with Policy Shaping and Bayesian Q-learning. In addition to the simulations, I conducted a human-subject experiment in which participants trained both NAA and Policy Shaping agents, and then reported on the experience of working with both agents.

I suggest that validating interaction algorithms with oracles and analyzing traditional

RL metrics such as cumulative reward and training time is only the first step. In addition to RL metrics, interaction algorithms should be validated by measuring peoples' experiences with agents using human factors, such as frustration. Ideally, the interaction algorithms should be designed with the goal of creating a positive human experience, because individuals who experience frustration while interacting with an agent are unlikely to continue or repeat the interaction in the future.

The results show that Newtonian Action Advice can perform better than Policy Shaping, both in terms of RL metrics like cumulative reward and human factors metrics like frustration. NAA can learn faster using less human instruction than Policy Shaping. NAA creates a better human experience than Policy Shaping. Compared to Policy Shaping, participants found the Newtonian Action Advice agent to be less frustrating, clearer and more immediate in terms of how the agent used human input, better able to complete the task as the participant intended, and more intelligent.

#### 5.1.1 Research Questions

This chapter develops the Newtonian Action Advice algorithm.

In particular, this chapter is presented as a template for IML algorithm design and verification in which researchers: 1) design for a positive human experience, 2) test the algorithm with oracles, and 3) verify the algorithm with a human-subject experiment measuring human factors.

- What is the theoretical efficiency of the Newtonian Action Advice agent, and how does it compare to Policy Shaping?
- How well does the NAA algorithm perform when trained using human teachers instead of oracle simulations?

### 5.1.2 Algorithm Development

In this chapter, an algorithm, Newtonian Action Advice, will be developed to connect human action advice (e.g. “move left”) to RL.

This chapter also acts as a template for creating IML algorithms.

### 5.1.3 Method Summary

The NAA algorithm was first tested with oracle simulations and compared to Bayesian Q-learning and Policy Shaping in Study 2. Then, the NAA algorithm was tested in a human-subject experiment and compared to Policy Shaping using human factors metrics. The human-subject experiment is detailed in Chapter 6, but is briefly included here to show a template of IML design that includes comparing algorithms with human factors metrics.

## **5.2 Algorithm Development: Newtonian Action Advice**

Newtonian Action Advice is a teaching interaction algorithm I designed to enable an RL agent to learn from human action advice. The theory is a metaphor of Newtonian dynamics: objects in motion stay in motion unless acted on by an external force. In the Newtonian Action Advice model, a piece of action advice provided by the human is an external force on the agent. Once a person provides action advice (ex: “Go right”), the agent will immediately move in the direction of the external force, superseding the RL agent’s normal action selection choice of exploration vs. exploitation. The model contains natural friction that ‘slows down’ the agent’s need to follow the human’s advice. The friction ensures that after some amount of time, the agent will resume the underlying RL algorithm’s exploration policy. The advice does not necessarily need to specify directional motion; the metaphor of advice as a force pushes the agent to follow the advice as opposed to the RL’s action selection mechanism.

Newtonian Action Advice was designed to behave in a manner that is more natural and

intuitive for the human teacher than other IML algorithms such as Policy Shaping. The force model allows each piece of advice to be generalized through time. If a person says, “go right,” the Newtonian Action Advice agent will move right and keep moving right until the ‘friction’ causes the agent to resume normal exploration. The simplicity of the force model is a feature to improve the human experience; people deal with Newtonian mechanics in their everyday life and are used to objects moving in a Newtonian manner. For example, if a ball is thrown straight up in the air the same way multiple times, it will always rise to a certain height and fall back to the ground. The motion is predictable, and one does not need formal physics training to recognize and expect the motion. I expect that loosely mimicking this will help create a user-friendly experience.

If advice was followed in one state, the Newtonian Action Advice algorithm will cause the agent to follow the same advice if the state is seen in the future. This means that a person will only have to provide advice once for a given situation. Also, only the latest advice is saved for a state, so people can correct any mistakes they made or change the desired policy in real time.

Bayesian Q-Learning was used as NAA’s underlying RL algorithm. This choice was primarily made so the Newtonian Action Advice algorithm could be more directly compared to Policy Shaping. The structure of the NAA algorithm is such that the BQ-L algorithm could be exchanged for a different RL algorithm.

In algorithm 2, *NAA* is the main procedure. At each time step, the agent listens for

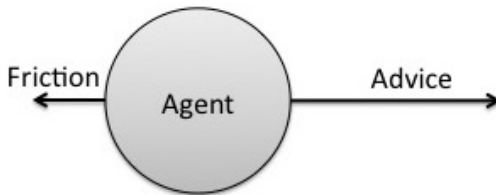


Figure 5.1: Simple Force Model. Actions are an external force acting on the agent, and ‘friction’ determines the amount of time the action will be followed after the advice is given.

---

**Algorithm 2** Newtonian Action Advice algorithm

---

```
1: procedure NAA
2:   for each time step do
3:     Listen for human advice
4:     if human advice given then
5:       newAdvice(state, advice)
6:       action = actionSelection()
7:       Take action and get reward
8:       Update Bayesian Q-learning policy with reward
9:   procedure NEWADVICE(state, advice)
10:    agent.adviceJustGiven  $\leftarrow$  True
11:    agent.advisedAction  $\leftarrow$  advice
12:    agent.adviceDictionary[state] = advice
13:   procedure ACTIONSELECTION(state)
14:    if agent.adviceJustGiven == True then
15:      chosenAction  $\leftarrow$  agent.advisedAction
16:      if state  $\notin$  agent.adviceDictionary then
17:        agent.adviceDictionary[state] = chosenAction
18:        timesNewAdviceFollowed += 1
19:        if timesNewAdviceFollowed >= timesToFollowAdviceThreshold then
20:          agent.adviceJustGiven = False
21:          timesNewAdviceFollowed = 0
22:      If advice has been given for this state, choose between the human advice and the
        algorithm
23:      Otherwise, if advice has not been given for this state in the past, use the action
        recommended from the BQL algorithm
23:   return chosenAction
```

---



advice. If advice is given, the agent updates its internal advice dictionary. The agent then chooses and takes an action, receives a reward, and updates the Bayesian Q-learning policy.

The *New Advice* procedure adds the new (state, advice) pair to the agent’s dictionary and sets a parameter that will tell the action selection procedure to follow the new advice.

The *Action Selection* procedure first checks to see whether advice has recently been given and should still be followed. If the advice is being generalized through time (due to low friction) and the new (state, advice) pair has not been added to the dictionary, it will be added at this time. If this state is revisited in the future, the recent advice given for a previous state will be applied as if it had been given for this state, too. The timer that keeps track of the friction parameter threshold is updated. If the timer indicates that the advice has been followed for long enough, parameters will be reset so the agent will return to the Bayesian Q-learning’s action selection for the next time step. If advice has previously been given for this state, the agent must choose between the human advice and the BQL suggestion. For this work, I always choose the human’s advice. If a researcher wants to encourage more exploration, a different method can be chosen (i.e. an algorithm similar to  $\epsilon - greedy$  applied to human vs. agent action selection instead of exploration vs. exploitation). However, I have found that following advice in a probabilistic manner increases frustration and uncertainty since the agent seems to disregard advice. In the case that no advice has been given for or generalized to the current state, the action is chosen from the Bayesian Q-learning’s action selection method.

### 5.2.1 Combining Supervised and Reinforcement Learning to allow for personalization by end-users

When designing an interactive machine learning algorithm, one first must ask: what is the goal of IML? Is the goal to use human instruction to decrease training time? Or is the goal to enable people to teach an agent to perform a task in the way the human intends?

If the goal of IML is to use human instruction to decrease the amount of time it takes

to train the RL agent, at first glance it seems like the best of both worlds. We get to use powerful RL algorithms that are capable of learning from their environment instead of having policies hard-coded or models from datasets built a priori; and we get to decrease training time by getting useful human input.

However, if the goal is to get the agent to perform a task that a non-expert specifies in the way the human wants the task done, then we have a problem.

The policy an RL agent learns is very sensitive to the reward function. In this work, as well as most IML research, the reward function is provided by the researcher prior to the experiment. Given a reward function, an RL algorithm will learn a policy to maximize the reward, but the policy learned may be very alien to a human mind. The agent will technically complete the task efficiently, but not in a way that makes a lot of sense to a human. Given a reward function and human input, the RL algorithm may initially learn a policy that conforms to the human's instructions, but eventually might learn a policy that solely maximizes cumulative reward. This may satisfy a goal of decreasing training time since the human will have shown the RL agent high-earning states earlier than it would have explored, but the policy may still be baffling to a non-expert. The human teacher may feel like their instructions were disregarded in the long-term, creating feelings of frustration and powerlessness when they cannot directly control the agent's policies.

Newtonian Action Advice can be seen as a way to combine supervised and reinforcement learning. A human provides information that is used to create policies that are static to the agent (supervised learning), but can be overwritten by the human. The agent uses reinforcement learning to determine the best course of action for parts of the state space in which human advice is not given. Not only does this enable the agent to use human input to decrease training time, it also empowers the human teacher to customize how the agent performs the task. It is possible the learned policy will be near-optimal instead of optimal from an objective analysis of the cumulative reward; but the agent's performance will be in greater accordance with the human teacher's instructions, which will increase the human's

satisfaction with the agent's performance.

### 5.2.2 Choosing the friction parameter

When calculating the algorithm's 'friction' parameter, it is more straightforward to think of the parameter as the number of steps each piece of advice should persist for,  $S_{des}$ . Increasing  $S_{des}$  causes each piece of advice to be followed for a longer time, which causes a metaphorically lower friction in the model.

Let:

$S_{des}$  (*steps*) is the desired number of steps advice should persist for

$S_{min}$  (*steps*) is the minimum number of steps advice should persist for

$S_{max}$  (*steps*) is the maximum number of steps advice should persist for

$U$  (*steps/second*) is the domain update rate

$\Delta t_{des}$  (*second*) is the desired time between given advice

$\Delta t_{min}$  (*second*) is the minimum time between given advice

$\Delta t_{max}$  (*second*) is the maximum time between given advice

Equations 5.1-5.3 show how to calculate the minimum, maximum, and desired values of the friction parameter. Two items should be noted for  $S_{min}$ . First, the value of  $\Delta t_{min}$  has a lower bound based on human limitations. You cannot expect people to provide advice infinitely quickly. It is nonsensical to provide advice that only lasts for a fraction of a second; if  $\Delta t_{min}$  is too small, it may occur that the agent follows the advice for such a short amount of time that it is not perceivable by the human. We suggest  $\Delta t_{min} \geq 0.5(\text{seconds})$ . Second, the advice must last for at least one time step, so  $S_{min} \geq 1(\text{step})$ .

$$S_{des} = \Delta t_{des} * U \quad (5.1)$$

$$S_{min} = \Delta t_{min} * U \quad (5.2)$$

$$S_{max} = \Delta t_{max} * U \quad (5.3)$$

Depending on the domain, task, and nature of the actions, we suggest a starting value of  $\Delta t_{des}$  to be between 2-8 seconds.

Equation 5.4 shows the bounds on the friction parameter. The value of  $S_{min}$  has the potential to run into a hard boundary based on human limitations, while  $S_{max}$  is a flexible boundary based on desired behavior.

$$(1step) \leq S_{min} \leq S_{des} \leq S_{max} \quad (5.4)$$

Instead of calculating the friction parameter using the desired time between when advice is given, the average duration of each action can be used.

Let:

$\Delta a$  (*actions*) is the desired number of actions between when advice is given

$spa_{avg}$  (*steps/action*) is the average number of steps it takes to complete an action

$tpa_{avg}$  (*seconds/action*) is the average time it takes to complete an action

If the human teacher is instructing the agent to take primitive actions, then  $spa_{avg} = 1(step)$ . If the human is providing instructions for higher order actions, then  $spa_{avg} \geq 1(step)$ . An action must last for the duration of at least one time step, so  $\Delta a \geq 1$ . Equations 5.5 and 5.6 show expressions for  $S_{des}$  depending on whether the researcher has easier access to  $spa_{avg}$  or  $tpa_{avg}$ .

$$S_{des} = \Delta a * spa_{avg} \quad (5.5)$$

$$S_{des} = \Delta a * tpa_{avg} * U \quad (5.6)$$

If primitive actions are being used, it is likely that the ideal  $S_{des}$  parameter will be fairly large because the human teacher will want a given piece of advice to be followed for several consecutive time steps. If higher order actions are being used, a smaller  $S_{des}$  may

be beneficial because it will already take the agent several time steps to carry out the higher order action.

### 5.3 Study 2: Method

We validated our NAA algorithm first with oracles to test the theoretical performance of the algorithm, and then with a human-subject experiment to compare the human teacher’s experience with another IML interaction algorithm, Policy Shaping.

Many of the existing machine learning algorithms that learn from human feedback are evaluated using oracles and focus on how quickly the agent learns. While this is valuable information, it ignores other important aspects of the human-agent interaction such as how humans react to the agent. For example, an oracle will never get frustrated with the agent or confused by its actions. If the interaction method affects how frustrated people are, regardless of the underlying machine learning algorithm, then perhaps that interaction method should be avoided. To that end, interactive machine learning agents should not solely be designed to optimize theoretical learning curves from simulations, but also to create a positive experience for the human teacher.

Both the simulations and human-subject experiment used the same task domain. The oracle or human participant was required to teach agents to rescue a person in Radiation World, a game developed in the unity minecraft environment (Figure 7.1). In the experimental scenario, there has been a radiation leak and a person is injured and immobile. The agent must find the person and take him to the exit while avoiding the radiation.

#### 5.3.1 Constructing Oracles

I first tested the Newtonian Action Advice algorithm with simulations that used a constructed oracle to simulate human feedback. Each oracle was instantiated with a probability,  $p_{advice}$ , that determined how often to check for advice from the oracle. If the simulation was testing the algorithm’s performance with  $p_{advice} = 20\%$ , then at every time step a ran-

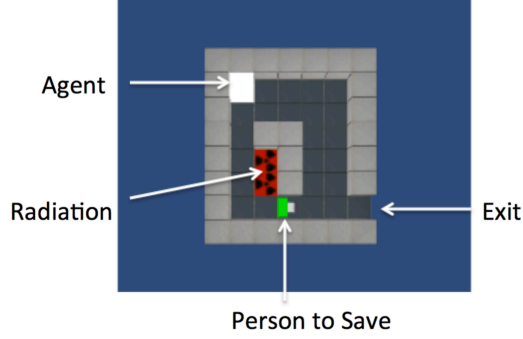


Figure 5.2: Radiation World Initial Condition

dom decimal,  $d$ , would be chosen between 0 and 1. The agent would check for advice if  $p_{advice} \leq d$ , and would otherwise not check for advice. I provided the advice for the oracles to test several cases, including maximum friction, two cases of minimal advice, and decreased friction.

The same oracle algorithm controlling when to check for instructions was used to test the Policy Shaping agent. The same advice dictionary was used for the NAA and Policy Shaping oracles. The advice dictionary was converted to critique for the Policy Shaping agent in the following manner: if the agent took the advised action for the state, the critique was positive; otherwise, the critique was negative.

### 5.3.2 Human-Subject Experiment

I conducted a repeated measures human-subject experiment in which I investigated the effect of two different interaction methods, NAA and Policy Shaping, on the human's experience of teaching the agent. Both the interaction methods shared the same underlying Bayesian Q-learning algorithm.

The method of the human-subject experiment is detailed in Section 6.4.

## 5.4 Study 2: Results and Discussion

### 5.4.1 Simulations

#### *No generalization through time (extreme friction)*

I simulated how the percentage of time advice is followed impacts performance. The simulation was set up so the NAA agent did not generalize a given piece of advice to other states immediately after the advice was given, meaning that one piece of advice counted for only one time step (maximum friction with  $S = 1(step)$ ). The oracle was built with advice given for every square in the grid (Figure 5.3).

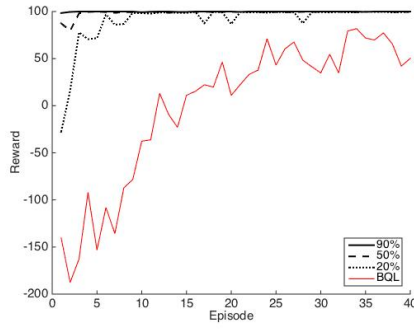
Incorporating human instruction by using the Newtonian Action Advice algorithm allows the RL agent to achieve a higher level of performance in many fewer episodes than without human input. As advice is given for a greater number of individual times steps (increasing from 20% to 90%), the agent accumulates more reward and completes each episode with fewer actions (Figure 5.4). The case with no human input is shown as BQL on the figures.



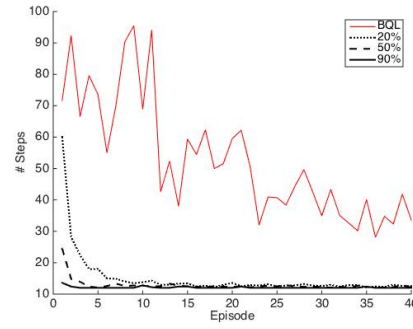
Figure 5.3: Advice given to simulation to avoid radiation.

#### *Minimal Advice - shortest path*

The minimal advice to take the shortest path (which takes the agent next to the radiation) is comprised of only two pieces of action advice equivalent to a human saying, “First

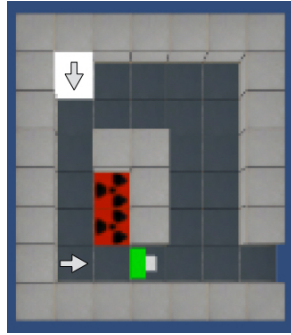


a: Reward

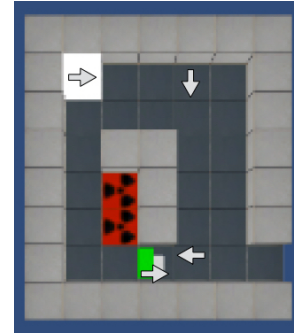


b: Number of Steps to complete an episode

Figure 5.4: How the amount of advice provided impacts performance. (advice given 20, 50, and 90 percent of the time)



a: Minimal advice given to simulation to complete task with minimal steps.



b: Minimal advice given to simulation to avoid radiation.

Figure 5.5: Minimal advice for two paths.

move *down*. Then go *right*.” The minimal advice used to create the oracle in this case is represented in Figure 5.5a.

Given only two pieces of advice, the NAA agent was able to complete the episode in 10 steps achieving a reward of 102.0 every single episode. The NAA agent was set to follow each piece of advice for  $S = 5(\text{steps})$  before returning to the BQ-L baseline action selection.

The NAA model with a decreased friction parameter is what allows a human teacher to say “down, right,” instead of, “down, down, down, down, down, right, right, right, right, right.” It makes for a much better and more intuitive user experience to provide less instruction and not have to constantly repeat advice.



### *Minimal Advice - avoiding radiation*

The minimal advice to take a path that avoids the radiation is comprised of only four pieces of action advice equivalent to a human saying, “First move *right*. Then go *down*. Move *left* then immediately *right* after rescuing the injured person.” This advice, which was used to construct the oracle for this case, can be seen in Figure 5.5b.

Given only four pieces of advice, the NAA agent was able to complete the episode in 12 steps achieving a reward of 100.0 every single episode. The NAA agent was set to follow each piece of advice for  $S = 5(\text{steps})$  before returning to the BQ-L baseline action selection.

This case is an example of the optimal vs. customized discussion in Section 5.2.1. The learned policy was near-optimal instead of optimal from an objective analysis of the cumulative reward since the path to avoid the radiation was slightly longer, but the agent’s performance was in accordance with the human teacher’s advised path.

### *Generalization through time (friction effect)*

I studied how the algorithm performs as the friction of the NAA model is decreased (i.e. the  $S$  parameter is increased). Sections 5.4.1 and 5.4.1 have already shown that a small amount of advice paired with a lowered friction can enable the NAA agent to perform optimally or near-optimally from the very first episode. To test the friction effect more rigorously, I built an oracle with the same advice given for every square as Section 5.4.1 (Figure 5.3).

When advice is given 20% of the time, the agent with a lower friction  $S = 5(\text{steps})$  initially performs better than a higher friction  $S = 1(\text{step})$ . However, in later episodes the agent with lower friction earns a lower cumulative reward while taking more steps to complete each episode compared to the high friction agent. In general, these results indicate that, while lowering the friction can increase initial performance, it can also cause a lower-performing policy to be learned by the agent.

But what is really going on in this case? We have seen in Sections 5.4.1 and 5.4.1 that

minimal advice paired with a lowered friction enables the agent to perform optimally or near-optimally from the very first episode. Why would providing more advice ( $p_{\text{advice}} = 20\%$ ) harm the agent’s performance, particularly when Section 5.4.1 showed that increasing advice increases performance? The core issue is a limitation of the oracle. At every time step, the oracle listens for advice with a probability of 20%. This is not how a human would provide advice. The decreased performance in this case occurs when the agent spends time repeatedly banging into walls after advice has been generalized to a wall state instead of the oracle providing advice for that wall state. The probability  $p_{\text{advice}} = 20\%$  is low enough that this behavior is not corrected for many episodes. Human teachers who observed this behavior would quickly provide an extra piece of advice to make sure the agent did not fruitlessly waste time.

When people decrease advice, they tend to limit themselves to the most important pieces of advice, such as the minimal advice cases. The oracle has no way to know which advice is the most important, and so provides advice in a way that is not indicative of human behavior. A possible solution to this problem is to build more elaborate oracles that more accurately represent human behavior. There are three main issues with this approach: 1) the use of and response to an algorithm will vary across individuals, so multiple contradictory oracles would need to be constructed, 2) an oracle’s ability to provide a type of input does not mean a human is likely or able to provide that input in reality, and 3) it is very unlikely that even the most elaborate oracle could simulate the human’s response to the agent, such as frustration. A more practical solution to this problem is to test algorithms with human-subject experiments.

This case shows why IML researchers should verify interaction algorithms with human-subject experiments in addition to simulations. If I had analyzed these results without understanding the limitations of the oracle, I might have discarded parameterizations using a lower friction.

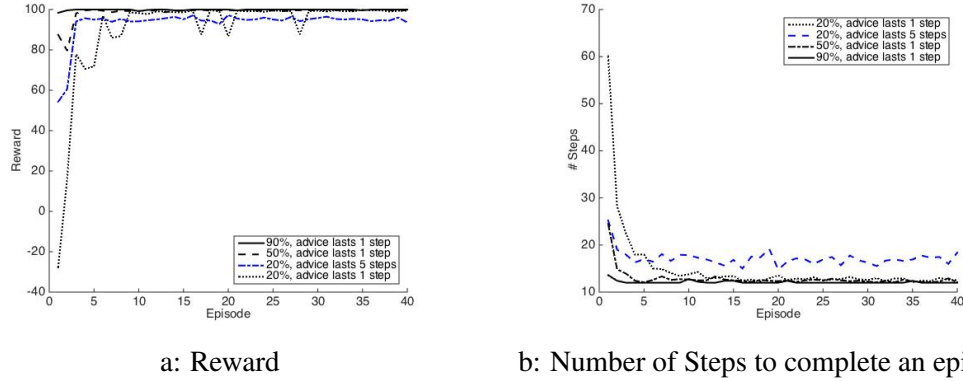


Figure 5.6: Generalization through time. (advice given 20, 50, 90 percent of the time)

### Comparison of Newtonian Action Advice and Policy Shaping

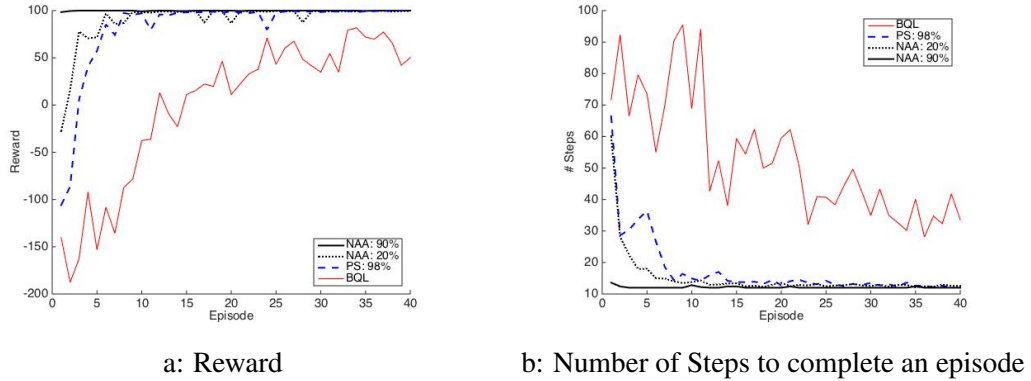


Figure 5.7: Comparison of Newtonian Action Advice and Policy Shaping

Figure 5.7 shows that, given equivalent input, Newtonian Action Advice can learn faster using less human instruction than Policy Shaping in this domain. Even when Policy Shaping used advice 98% of the time, it learned slower than the NAA agent that was using input only 20% of the time. The oracle used the same setup as Sections 5.4.1 and 5.4.1 (Figure 5.3).

When learning from human teachers in practice, however, the performance of each agent is entirely dependent on the instruction provided by the person. Neither agent is guaranteed to perform better than the other. If the human provides no instructions, the Policy Shaping and NAA agents perform equally since they both reduce to a Bayesian Q-

learning algorithm. In order to investigate how the performance of the agents varied with real human teachers, as well as how the human experience was impacted by interacting with each agent, I conducted a human-subject experiment.

#### 5.4.2 Human Subject Results

The results of the human-subject study are briefly described here to demonstrate how IML algorithms can be compared using human factors metrics rather than relying solely on objective ML metrics. This human-subject experiment will be discussed in detail in Chapter 6.

Immediately after training each agent, participants were asked to score aspects of their experience training the agent, including frustration, perceived performance, transparency, immediacy, and perceived intelligence (Figure 5.8). Paired t-tests were conducted for each metric in which the null hypothesis was the pairwise difference between the two paired groups had a mean equal to zero. I found that all measured aspects of the human experience differed significantly between the two agents.

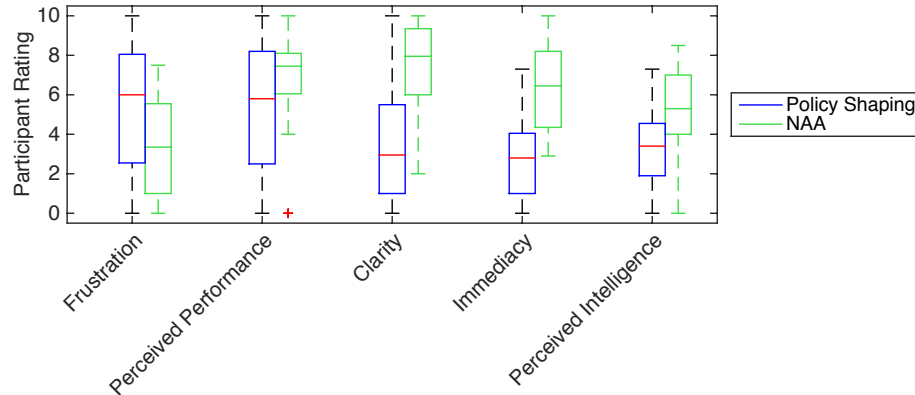


Figure 5.8: Comparison of the human experience.

In summary, compared to Policy Shaping participants found the Newtonian Action Advice agent to be less frustrating, clear and more immediate in terms of how the agent used human input, better able to complete the task as the person intended, and more intelligent.

In addition to creating a better human experience, the NAA agent also performed bet-

ter than Policy Shaping in terms of objective RL metrics. The average training time was smaller for the NAA agent. The number of steps the agent took to complete each episode was smaller for NAA. The average reward was higher for NAA than Policy Shaping. However, the amount of input provided by the human teachers was statistically equal for the two interaction algorithms.

The nature of the agent’s response to the human’s instruction created significantly different human experiences between the two agents. Consider that both agents used the same underlying BQL algorithm, and so would perform equally with no human input. Also, people provided a statistically equal number of instructions for both agents, so both were equally effortful for the person. And yet, the peoples’ perceptions of the agents differed across all subjective metrics, including frustration, perceived intelligence, and transparency. A thorough analysis of the participants’ responses in Chapter 6 shows that the main factors that influenced the participants’ frustration levels were whether the agent’s behavior made the person feel powerless, whether the agent’s choices were transparent, the complexity of the instruction format, and whether the agent immediately acted on the person’s instructions. The objective metrics like training time and reward were not factors mentioned by the participants. This tells us that the human experience cannot be improved solely by designing algorithms to optimize the objective metrics that are available in oracle testing.

## **5.5 Study 2: Conclusions**

This chapter presented Newtonian Action Advice, an IML algorithm that integrates a human’s interactive action advice (ex: “move left”) with reinforcement learning. For equivalent human input, Newtonian Action Advice performs better than Policy Shaping, both in terms of RL metrics like cumulative reward and human factors metrics like frustration.

This chapter also acts as a template for IML algorithm design and verification in which researchers: 1) design for a positive human experience, 2) test the algorithm with oracles, and 3) verify the algorithm with a human-subject experiment measuring human factors.

While the results of the human-subject experiment were briefly included here to demonstrate the full template for IML design and verification, the next chapter delves more deeply into these results.

## 5.6 RQ Results Summary

- Newtonian Action Advice can learn optimal policies at least an order of magnitude faster than when no advice is provided to the underlying BQL agent. NAA can learn optimal policies from the very first episode given minimal advice. A low friction parameter can cause a sub-optimal policy to be learned. Given equivalent input, NAA can learn faster with less input than Policy Shaping.
  - *Validation:* Study 2.
  - *Supports thesis:* Part 2 (efficiency) - NAA allows a small amount of instruction to be used to train an agent. Part 3 (performance) - NAA can use action advice without detracting from the agent’s capabilities.
- NAA creates a better human experience than Policy Shaping. Compared to Policy Shaping, participants found the NAA agent to be less frustrating, clearer and more immediate in terms of how the agent used human input, better able to complete the task as the participant intended, and more intelligent.
  - *Validation:* Study 3.
  - *Supports thesis:* Part 1 (human experience) - this work analyzed human factors metrics to determine how real people responded to the agent.

## CHAPTER 6

### STUDY 3: MOVING BEYOND ORACLES - THE COMPARATIVE HUMAN EXPERIENCE OF LEARNING FROM ADVICE AND CRITIQUE

A goal of interactive machine learning (IML) is to enable people with no specialized training to intuitively teach intelligent agents how to perform tasks. Toward achieving that goal, I am studying how the design of the interaction method for a Bayesian Q-Learning algorithm impacts aspects of the human’s experience of teaching the agent using human-centric metrics such as frustration in addition to traditional ML performance metrics. I posit that the type of feedback a robot can learn from affects the perceived intelligence of the robot, similar to its physical appearance. This study investigated two methods of natural language instruction: critique and action advice. I conducted a human-in-the-loop experiment in which people trained two agents with different teaching methods but, unknown to each participant, the same underlying reinforcement learning algorithm. The results show an agent that learns from action advice creates a better user experience compared to an agent that learns from binary critique in terms of frustration, perceived performance, transparency, immediacy, and perceived intelligence. I identified nine main characteristics of an IML algorithm’s design that impact the human’s experience with the agent, including: using human instructions about the future, compliance with input, empowerment, transparency, immediacy, a deterministic interaction, the complexity of the instructions, accuracy of the speech recognition software, and the robust and flexible nature of the interaction algorithm.

#### 6.1 Introduction

This chapter focuses on the question: *Does the interaction method affect the person’s experience with the agent, and if so, why?* Specifically, I analyze whether the method of natural language feedback given to a ML agent impacts the human teacher’s satisfaction. Human

experiences are important because individuals who experience frustration while interacting with a robot are unlikely to interact with the robot in the future.

Traditional research into IML often fails to account for the human experience. Instead, algorithms are tested with computer simulations of human input, i.e. oracles, and verified using objective ML metrics, such as cumulative reward. While a good starting point, this method should be considered incomplete for any system that is intended to learn from humans, as it ignores important aspects of the human-agent interaction that are difficult to gauge using oracles alone. The use of oracles instead of human-subject experiments overly simplifies the human interaction and ignores how humans react to the agent. Oracles will never get frustrated with the agent or confused by its actions. I suggest that researchers should design and evaluate IML algorithms using human subjects and human factors metrics for testing and verification in addition to the traditional ML analyses in order to measure and improve the human teacher's experience.

To illustrate how the results from an oracle can fail to capture the depth of differences experienced by a human teacher, I performed a repeated measures human-subjects experiment in which participants taught two agents how to play a simple game with different teaching methods: critique and action advice. For the first teaching method, people trained the agent using positive and negative verbal critique such as, "good job," and "don't do that." The second teaching method enabled people to teach the agent using action advice such as, "move right," and "go left." The same underlying machine learning algorithm, Bayesian Q-Learning, was used for both agents. The critique agent used Policy Shaping as an interaction algorithm, while the action advice agent used the Newtonian Action Advice algorithm.

As predicted, human teachers had difference experiences teaching the two agents. Participants had a much better experience with the action advice agent compared to the critique agent. Participants found the action advice agent to be more intelligent, less frustrating, clearer and more immediate in terms of how the agent used human input, better able



to complete the task as the person intended, and more intelligent than the critique agent. Compared to the critique agent, the advice agent had a shorter average training time, a smaller number of steps to complete each episode, and a higher average reward. People provided approximately the same amount of instruction for both the advice and critique agents.

### 6.1.1 Research Questions

This chapter investigates how and why an interaction algorithm impacts the human teacher's experience.

- Does the interaction method (e.g. Policy Shaping critique vs. Newtonian Action Advice) impact the human teacher's experience with the agent (perceived intelligence, perceived performance, frustration, immediacy, transparency)?
- Does the interaction method (e.g. Policy Shaping critique vs. Newtonian Action Advice) impact the objective metrics of the teacher's interaction with the agent (reward, training time, amount of human input provided, number of steps for the agent to complete each episode)?
- Which design decisions made during the creation of an interaction algorithm impact the human teacher's experience?
- Is action advice more rhetorically positive than critique?

### 6.1.2 Development

The elements were developed for use in this Study. The Newtonian Action Advice algorithm used in this experiment is developed in Section 5.2. The method of using sentiment analysis to filter verbal critique into positive and negative was developed in Chapter 3. This chapter also develops a method of comparing IML algorithms using human factors metrics.

### 6.1.3 Method Summary

The research questions in this chapter were tested in Study 3, which was a human-subject experiment conducted in the Radiation World domain.

## **6.2 Differences Between Critique and Action Advice**

In this section, I discuss differences between advice and critique, including rhetoric, timing, and whether the instructions apply to the future or past.

### 6.2.1 Rhetoric

I suggest that the rhetoric of critique is inherently negative, while advice is more positive. Since action advice exclusively informs the agent about what to do next, there is a sense of rhetorical forgiveness - the human does not judge the agent's behavior, but rather offers a helping hand and advises the agent what to do next to move to a better situation. Critique, on the other hand, is about placing credit and blame on past actions without encouraging or allowing a person to provide a potential solution of what to do next [45].

Suppose that the agent was driving a car and got stuck in the mud. Rhetorically, the critique teaching method is like a passenger crossing his arms, glaring at the driver, and saying, "You are a terrible driver." This may be true, but does not help the situation. The action advice teaching method would be like a passenger who, without judging, suggested using twigs, leaves, or a car mat to create traction, and told the driver slowly accelerate out of the rut.

The inherent rhetoric of different teaching methods has never been applied to IML algorithm design before, but I believe it is a characteristic that can impact the human experience. We do not want to design algorithms that force human teachers to be the judgmental backseat driver in perpetuity as this would likely result in endless frustration. I believe in creating a more positive interaction between the agent and human teacher.

### 6.2.2 Immediate Applicability of Feedback

Critique provides rewards or punishments about past behavior, while advice supplies actions that should be taken by the agent in the immediate future. The critique will only affect the Policy Shaping agent's behavior if the critiqued past state is revisited in the future. Because the critique agent applies feedback to past instead of future actions, there is no way to 'close the loop' so the human can immediately tell if their feedback was correctly understood and applied. Action Advice, on the other hand, provides instructions that the agent can immediately apply. Consequently, it will likely seem to the human that the action advice agent is more responsive than the agent learning from critique. Additionally, people give instructions about the future, even if expressly told not to [18]. In these terms, I expect action advice to be more human-friendly than critique.

### 6.2.3 Issues of language processing

There are three main issues associated with language processing when using verbal instruction as an interaction medium that impact the advice and critique algorithms unequally: language processing time, accuracy of the ASR software, and credit attribution. The advice agent suffers more with the language processing time and accuracy of the transcription software, while critique is impaired by credit attribution.

1. With the action advice agent, people are sensitive to the *language processing time* because the agent's next action is effected. Human teachers are aware of the processing time because it acts as a lag between when the person gives advice and when the agent follows the advice. A long processing time would degrade the human experience. With the critique agent, however, people are unaware of the processing time because their instructions are used to inform past behavior.
2. Human teachers are expected to be more sensitive to the *accuracy of the automatic speech recognition software* with the action advice agent because the human can im-

mediately tell if the agent understood based on the next action. People are unaware of the speech software accuracy with the critique agent because the next actions are not impacted.

3. One severe issue with critique is the *credit attribution problem*. There is no definitive way to know which past action(s) the human teacher intended to reward or penalize. Action advice does not have this problem because the human's action instructions are simply applied to the next action. The credit attribution problem may be more apparent to the algorithm designer than the human teacher - the human will know the critique agent is not performing well, but will not necessarily know if, and how much, credit attribution is to blame.

Figure 6.1 shows how language processing impacts the advice agent. At some point in time, the human teacher starts giving verbal action advice, such as “move left.” After the person stops speaking, the ASR software transcribes the speech to text. Additional language processing, such as sentiment analysis, may be performed on the text depending on the agent setup. After language processing, the advice is applied to the next action in time. The human teacher is sensitive to longer language processing times since it that would cause the agent to not follow the advice for several actions. Assuming a relatively uniform time step, people soon learn the lag in the system. People are also sensitive to the ASR accuracy because they will immediately tell if the verbal instruction was incorrectly transcribed when the agent does not follow their advice. Significant inaccuracies in speech recognition degrade the human experience. Finally, there is no credit attribution problem with an advice agent.

Figure 6.2 shows how language processing impacts the critique agent. At some point in time, the human teacher starts to give verbal critique, such as “Good job.” Once the person stops speaking, the language is processed. After the language is processed into positive and negative critique, the critique is used to update the Policy Shaping agent. The human teacher is not aware of either the language processing time or the ASR accuracy

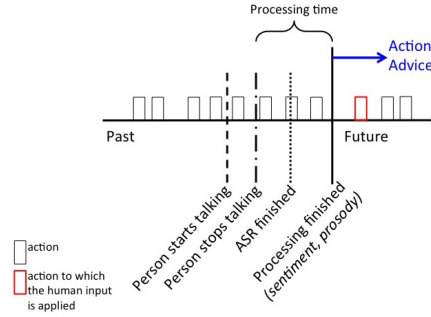


Figure 6.1: Advice applies to the next action after language processing

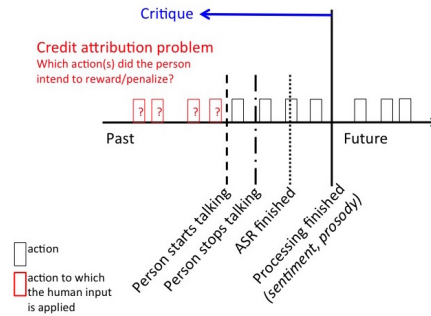


Figure 6.2: The credit attribution problem with critique

since critique is applied past actions instead of the future. The human teacher is also not necessarily aware of the credit attribution problem, even though it is a severe issue with critique. It may appear at first glance that the three issues of language processing do not impact the human's experience of the critique agent as much as the advice agent. However, the teacher does not know if 1) the critique has been heard, 2) the critique has been correctly interpreted, and 3) to which action(s) the critique has been applied.

### 6.3 Perceived Intelligence - Interaction Method Parallels with Morphology

Many factors have been shown to effect perceived intelligence. A robot's animacy, or life-like behavior, is strongly correlated to its perceived intelligence [46]. Multiple studies have shown that a person's accent affects how intelligent that person is perceived to be by others [47, 48]. HRI researchers have investigated standardized tools for measuring perceived intelligence [49]. Human teachers change their behavior toward a robot as they develop a mental model of the robot [50].

Many studies have shown a robot’s morphology, or physical appearance, affects how intelligent people expect the robot to be [51, 52, 53]. People expect a humanoid robot to interact with a human-level of intelligence, whereas a robotic dog is expected to have the lower-level of intelligence of a dog. I suggest that the method of interaction built into a robotic agent also affects the perceived intelligence of the robot, similar to its physical appearance.

Scaffolding is an instructional technique in which teachers can change their type of instruction based on how confused the student is [11]. When a teacher explains a concept, she might ask an open-ended question (ex: What color is this?). If the student looks confused, the teacher switches to yes-or-no questions (ex: Is this red?). The teacher gives the student the best chance of getting a correct answer by decreasing the infinite number of responses in the open-ended question to a 50/50 chance of a confused student getting a correct answer.

The two methods of natural language instruction I studied were *critique* and *action advice*. When teaching using critique, a person gives the agent positive or negative feedback (good, bad). When teaching with action advice, a person would tell the agent what actions to take next (up, down, left, right).

Concerning our verbal interaction methods, action advice is similar to an open-ended question (What action should I take?), whereas critique is similar to the yes-or-no question (Am I doing good or bad?). If an agent can only understand binary input of good or bad from a teacher, I hypothesize people will associate the agent with a less intelligent or struggling student that cannot answer open-ended questions on their own. Similarly, I think an agent that uses the more complex input of action advice from a teacher will be perceived as more intelligent and capable, even if the underlying ML algorithm used by the agent is the same.

## 6.4 Study 3: Method

I conducted a repeated-measures human-subject experiment in which I investigated the effect of two different teaching methods, critique and action advice, on the human's satisfaction with the agent. The experiment collected data from 24 participants who were not experts in machine learning, and in many cases were not associated with a university. The age of the participants varied from 18-65 years old.

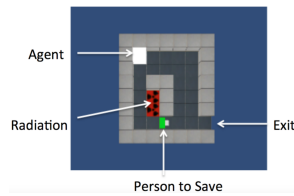


Figure 6.3: Radiation World Initial Condition

Each participant trained two agents with different interaction methods - critique and action advice - but an identical underlying Bayesian Q-learning algorithm. The task required participants to teach each agent to rescue a person in Radiation World, which is a simple game developed in the unity minecraft environment as shown in Figure 7.1. In the experimental scenario, there has been a radiation leak and an injured person is located somewhere in the grid unable to move. The agent must find and rescue the person, take him to the exit, all while avoiding the radiation. The task is complete if the agent takes the person to the exit. The task is failed if the agent enters the radiation or exits without the person.

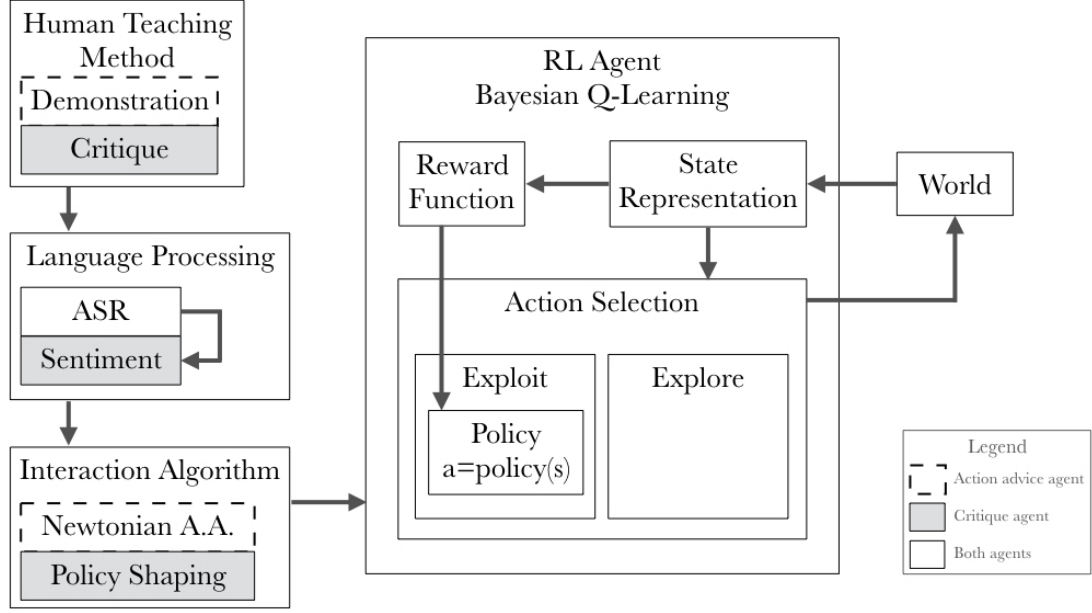


Figure 6.4: Agent Setup

The agent setup is shown in Figure 6.4. Both agents learned from verbal instruction. The action advice agent used what can be thought of as verbal demonstrations, while the critique agent learned from positive and negative critique. The verbal instructions were transcribed to text using ASR software. The critique agent had an additional language processing step of using sentiment analysis to convert the text to positive or negative critique. After language processing, the processed instructions were sent to an interaction algorithm. The advice agent used Newtonian Action Advice, while the critique agent used Policy Shaping. Both interaction algorithms worked with the Bayesian Q-learning algorithm to determine action selection.

Neither agent is guaranteed to perform better than the other. If the human provides no instructions, the critique and advice agents perform equally. With human teachers, however, the performance of each agent is entirely dependent on the instruction provided by the person. In terms of theoretical algorithmic performance for an equivalent set of human instructions, the Newtonian Action Advice agent with a friction parameter of  $S = 5(\text{steps})$  in this experimental scenario is likely to have a higher cumulative reward for the initial



episodes, but the Policy Shaping agent is likely to earn a higher cumulative reward for the later episodes [54].

To teach the critique agent, participants were instructed to provide positive or negative critique if the agent did something they considered good or bad. Using sentiment analysis as a filter allowed people to provide verbal natural language critique without restricting their vocabulary. For example, a participant could give varied critique such as, “Good job,” “That’s great,” “That is a bad idea,” and “You’re wasting time”. The critique agent used Policy Shaping to incorporate the positive and negative feedback.

For the action advice agent, people were instructed to tell the agent to move in a desired direction. For example, if participants wanted the agent to move down, they should say, “down.” The only four words recognized by the action advice agent were, “up,” “down,” “left,” and “right.” The action advice agent used the Newtonian Action Advice algorithm to incorporate the advice with the Bayesian Q-learning algorithm’s action selection. The friction value used in the experiment was  $S = 5(\text{steps})$ .

The experiment split participants into two groups randomly. The first group trained the critique agent first, and the second group trained the advice agent first. Participants were told to stop training when either the agent was performing as they intended or the participant was too frustrated to continue or wanted to stop for any reason. Consequently, the training time varied for each participant and teaching method. After a participant finished training an agent, the participant completed a questionnaire concerning the experience (*see Paired Tests*). After training both agents, the participants filled out a questionnaire comparing the two agents (*see One-Sample Tests*). At the end of the experiment, participants were asked if they had any questions or additional thoughts concerning the experiment in an exit interview.

## Experimental Procedure

### 1. Greeting and Introduction

2. Instructions for agent 1. Train agent 1. Questionnaire about experience of training agent
  1. *See Paired Tests in the Results.*
3. Instructions for agent 2. Train agent 2. Questionnaire about experience of training agent
  2. *See Paired Tests in the Results.*
4. Questionnaire comparing the experiences of training both agents, including free response questions. *See One-Sample Tests and participant quotes in the Results.*
5. Exit interview.

#### 6.4.1 Human Experience Measures

The answers to all questions regarding human experience were collected using a sliding bar in which the selected value to the tenths decimal place was shown to the participant.

*Paired tests.* Immediately after training an agent, the participants were asked to score the intelligence of the agent on a continuous scale from [0:10]. A value of 0 indicated that the agent was not intelligent, while 10 meant very intelligent. The same scale of [0:10] was used for four additional metrics: performance, frustration, transparency, and immediacy. Values of 0 corresponded to poor performance, low frustration, non-transparent use of feedback, and a slower response time. Values of 10 meant excellent performance, high frustration, clear use of feedback, and an immediate response time, respectively.

*One-Sample Tests.* After training both agents, participants were asked to compare the intelligence of the two agents on a continuous scale from [-5:5]. A value of -5 indicated that the action advice agent was much less intelligent than critique. Zero meant the two agents were equally intelligent. A value of 5 meant the action advice agent was much more intelligent than critique. The same scale of [-5:5] was used for performance, frustration, transparency, and immediacy. Values of -5 corresponded to the advice agent performing much worse, causing much less frustration, using feedback less clearly, and responding much slower than critique. The scales are shown on Figures 6.6-6.10.

*Explanation of Responses.* After training both agents, in item 6 of the procedure participants were given the option to explain the answers that they gave on each of their five comparison ratings. For example, participants were asked, “If one agent was more frustrating than another, what made you feel that way?” These written responses were entirely free form with no priming or options provided by the experimenter.

#### 6.4.2 Objective ML measures

While participants were training the agents, objective performance metrics were logged to data files. The training time and reward earned per episode were measured as continuous data. The amount of human input provided and the number of actions it took to complete an episode were measured as ordinal count data.

### **6.5 Study 3: Results**

I will first show the results of the human teachers’ experience interacting with the agents, followed by an analysis of the objective metrics of the agents’ performances. I will conclude with an analysis of the path the user chose. For the human factor’s metrics I will begin with an overview of the quantitative analysis and then provide more detailed discussion of each metric individually.

#### 6.5.1 Human Experience Overview

Figure 6.5 illustrates the results of how participants scored aspects of the human experience. The differences indicate the human experience was not the same for the advice and critique agents. The figure shows aggregate participant ratings of how frustrated they were with each agent, whether it was clear how each agent used the human instruction, how quickly each agent responded to the instruction, whether the agent performed the task as intended, and how intelligent the person thought the agent was.

*Paired tests on Human Experience:* Paired t-tests were conducted for each of the five

human factors metrics that were collected following training each agent type. The null hypothesis was the pairwise difference between the two paired groups had a mean equal to zero. A significance of  $\alpha = 0.05$  was chosen as a reasonable limit on Type I error given the number of participants enrolled. I found significant differences between the two agents for every metric. Table 6.1 shows the results of each t-test.

*One-sample tests on Human Experience:* A series of one-sample t-tests were conducted for the five human factors metrics that were collected following both training sessions. Here the null hypothesis was the data came from a population with mean equal to zero. A significance of  $\alpha = 0.05$  was chosen as a reasonable limit on Type I error given the number of participants enrolled. I found significant differences between the two agents for every metric. Table 6.2 shows the results of each t-test. These one-sample tests provide a separate and internal verification of the paired tests. The results give us a greater assurance that the differences in the human experience were not influenced by the order in which the participant experienced the agent.

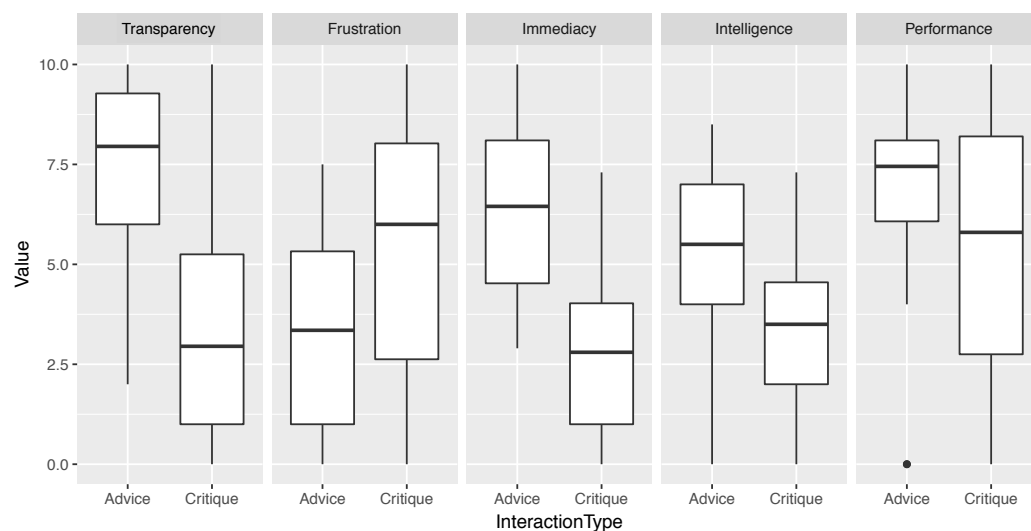


Figure 6.5: Human Factors Metrics. For all metrics save frustration, higher values represent a better human experience (better performance, greater transparency, more immediate, and more intelligent). For frustration, higher values indicate a higher-level of frustration, and therefore a worse human experience.

Overall, participants found the action advice agent to create a better human experience

Table 6.1: Paired T-tests on Human Experience

Human Factors Metric	Accept/Reject	$t(23)$	$p$	$\mu_{advice}$	$\mu_{critique}$
Frustration	Reject	-3.1364	0.0046	3.41	5.60
Transparency	Reject	5.4963	1.3738e-05	7.44	3.50
Perceived Intelligence	Reject	4.5527	1.4192e-04	5.48	3.40
Perceived Performance	Reject	2.2401	0.0350	6.85	5.26
Immediacy	Reject	6.2911	2.0291e-06	6.25	3.06

Table 6.2: One-sample T-tests on Human Experience

Human Factors Metric	Accept/Reject	$t(23)$	$p$	$\mu$
Frustration	Reject	-4.2313	3.1634e-04	-2.25
Transparency	Reject	7.3599	1.7397e-07	2.90
Perceived Intelligence	Reject	5.5087	1.3325e-05	2.387
Perceived Performance	Reject	8.1462	3.1395e-08	3.06
Immediacy	Reject	7.2958	2.0073e-07	2.94

than the critique agent. In both the paired and one-sample tests, participants found the action advice agent to be less frustrating, more immediate, more transparent, and with a higher perceived performance and perceived intelligence compared to the critique agent.

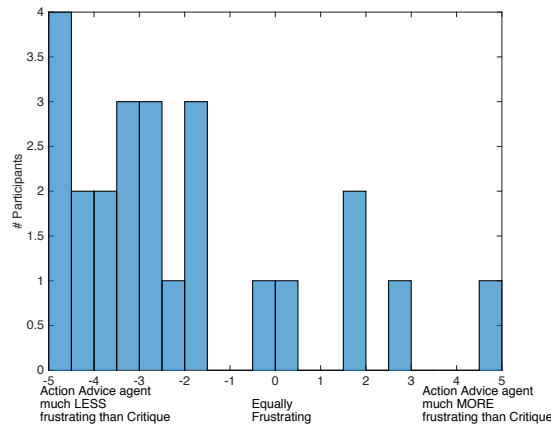


Figure 6.6: Frustration Direct Comparison

### 6.5.2 Frustration

Most participants found action advice to be much less frustrating than critique. The participants' free responses provided valuable insight as to why. I analyzed these responses and found several factors were repeated by many participants. The main factors that made people perceive one agent as more frustrating than another are listed below. I will discuss each in turn.

**Powerlessness:** whether the agent's behavior made the human operator feel powerless

**Transparency:** whether the human understands why the agent made its choices

**Complexity:** the complexity of allowed human instruction

**Compliance with input:** whether the agent did what it was told

**Probabilistic:** whether the agent incorporated input probabilistically

**Sensitivity to ASR:** the accuracy of the software that transcribes verbal speech to text

#### *Powerlessness*

The critique agent made people feel powerless, which caused higher levels of frustration compared to the advice agent. The human's lack of control over the critique agent caused higher levels of frustration compared to the action advice agent.

**P14** "In the critique case, I felt powerless to direct future actions, especially to avoid the agent jumping into the radioactive pit."

**P9** "The critique agent did not listen to me which made me frustrated. It took way longer to respond and did not end up learning how to do the task."

**P8** "Critique agent was hard to train since there wasn't much feedback as to if my critique had any affect on it at all."

**P6** "I believe the critique agent took too much time to recognize my input, and did not act upon at all."

**P3** “Repeating myself and still getting unintended results.”

**P2** “The critique agent was more frustrating to train because I had less direct control and it was not clear how it was interpreting my input.”

### *Transparency*

The critique agent caused people to be frustrated because the human teachers could not figure out how their input was being used by the agent.

**P22** “The Critique agent was very frustrating. I never understood why it made the choices it did, although that could have been because I was not telling it the best commands.”

**P8** “(The) Critique agent was hard to train since there wasn’t much feedback as to if my critique had any affect on it at all.”

**P15** “I did not understand how the critique would use my inputs.”

**P10** “Critique was more frustrating because it was very difficult to tell how it was using its knowledge base.”

**P6** “I believe the critique agent took too much time to recognize my input, and did not act upon at all.”

**P2** “The critique agent was more frustrating to train because I had less direct control and it was not clear how it was interpreting my input.”

### *Complexity*

Participants felt that the more complex action advice was less frustrating than the less complex method of critique.

**P12** “I wanted to give more complex advice to ‘help’ the Critique Agent.”

**P11** “I felt that the critique agent was more frustrating because I wasn’t able to give it specific feedback on how to improve or what it had done correctly.”

### *Compliance with input*

The action advice agent created a less-frustrating interaction because it did what it was told, unlike the critique agent.

**P11** “The action advice robot just worked. The critique robot took time to train—and even then, he was never as efficient as the action advice robot.”

**P23** “I never did get the critique agent to the injured person. It just keep going in different directions and I couldn’t figure out why.”

**P9** “The critique agent did not listen to me which made me frustrated. It took way longer to respond and did not end up learning how to do the task.”

**P6** “I believe the critique agent took too much time to recognize my input, and did not act upon at all.”

**P3** “Repeating myself and still getting unintended results.”

### *Probabilistic*

Several participants mentioned that the random or probabilistic nature of the critique agent caused them to be more frustrated. The seeming randomness of the agent’s response to instruction made them question why the critique agent was choosing certain actions. This is a particularly important lesson for ML researchers because most interaction methods incorporate human instruction in a probabilistic manner (following instructions, choosing between human and RL policies, tapering off human instructions, choosing between human and RL exploration, etc.). While a probabilistic nature of an interactive ML algorithm may be beneficial for testing algorithms in simulation (and is common practice for pure ML), it overlooks a basic and essential aspect of the human interaction experience: if I tell the agent to do something, I want it to follow my instructions in a reliable, repeatable, non-probabilistic way. I do not suggest that the underlying ML algorithm should be devoid of



a probabilistic nature, but rather that the interaction method between the human and agent should be deterministic or heavily biased toward human input in the short term.

**P24** “The Critique agent was more frustrating because its development of a mission plan based on user input seemed to be more randomized than truly crafted based on feedback. This made it frustrating to give input, as it felt like it wasn’t being used effectively.”

**P23** “I never did get the critique agent to the injured person. It just keep going in different directions and I couldn’t figure out why.”

#### *Sensitivity to speech recognition*

The few participants who found the action advice agent to be more frustrating than critique had problems with the automatic speech recognition software that transcribed their verbal feedback into text. While the ASR software used had a good average accuracy, results vary based on a person’s specific accent and speech patterns. In the future, this problem will diminish as ASR models are built using a wider array of training data.

The same ASR software was used for both advice and critique. However, the human’s perception of the advice agent is more sensitive to ASR accuracy because it is immediately apparent the agent did not understand the person if the person says “right” and the agent does not move right. Action advice allows people to provide instructions for the agent’s immediate future actions, which makes it sensitive to ASR accuracy because a person can immediately tell if the ASR software was correct. If the ASR software has poor accuracy, it is not as readily apparent for the critique agent, which applies the feedback to past actions.

**P21** “Action Advice was far more frustrating because it was not recognizing my commands (especially ”Down”). If it HAD been recognizing a very high percentage of my commands, it would have been much, much LESS frustrating than Critique was.”

**P17** “The action didn’t seem to register what I was saying as clearly.”

**P1** “Action advice was more frustrating because it often couldn’t hear me or misheard me.”

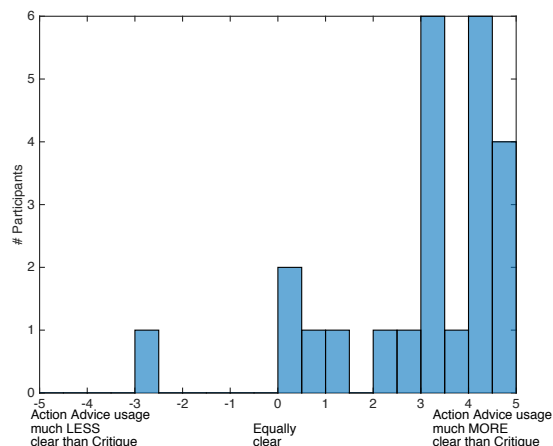


Figure 6.7: Transparency Direct Comparison

### 6.5.3 Transparency

Every participant except one found that action advice used the human instruction in a clearer manner than critique. The main factor that made people perceive one agent as more clear than another was:

**Compliance with input:** whether the agent did what it was told

#### *Compliance with Input*

**P11** “The action advice agent used my feedback more clearly because it would do exactly what I said. With the critique agent it was hard to tell how it was choosing what to do next.”

**P24** “The Action Advice agent used feedback much more clearly since it awaited instruction and did exactly what was told, so long as the audio was clear.”

- P23** “I could see that the Action Agent was at least responding by going the way I said. Even if I told the critique agent it was bad, it didn’t necessarily try something different.”
- P21** “Action Advice was using my feedback, when it understood it correctly, to directly move. I had to guess what Critique was going to do with my input.”
- P20** “I could see action agent respond to my input, but the critique agent didn’t seem to use the input.”
- P15** “(With the action advice agent,) a certain input caused an output. The critique agent seemed to be going randomly with small help.”
- P9** “It seems that the action agent used my feedback better because I was able to get it to consistently collect the object and make it to the goal. The critique agent seemed to do what it wanted and did not take my advice.”
- P5** “The (action advice) agent responded in the right direction with a quicker time.”
- P4** “The speed and accuracy of the (advice) agent’s movements in response to my feedback. Essentially, seeing the agent do what I ‘wanted it to do’ made it seem like my feedback was used more clearly.”
- P2** “The action advice agent tended to move in the direction I had previously advised it to move, but the critique agent still moved in general directions that I had given negative feedback for previously.”

#### 6.5.4 Perceived Intelligence

The majority of participants perceived the advice agent to be more intelligent than the critique agent, even though both used the same underlying ML algorithm. Without human

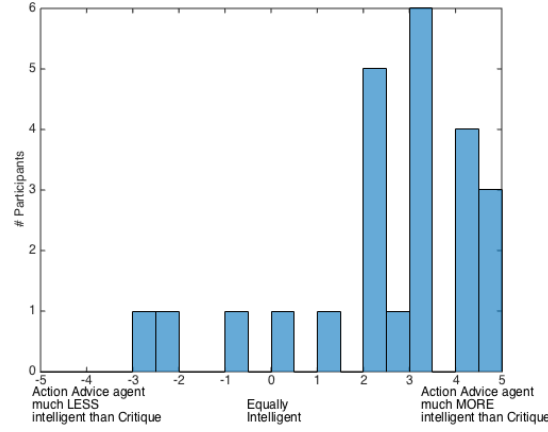


Figure 6.8: Perceived Intelligence Direct Comparison

input both agents were equally capable, but the teaching method caused the human-agent interaction to differ.

*Paired Tests:* A paired T-test was conducted in which the null hypothesis was the pairwise difference between the two paired groups had a mean equal to zero. The test found significant differences ( $t(23) = 4.5527, p \leq 0.0001$ ) of the perceived intelligence between the two agents. Participants found the action advice agent ( $\mu_a = 5.483$ ) to be more intelligent than the critique agent ( $\mu_c = 3.408$ ).

*One-sample Comparison:* A one-sample T-test was conducted in which the null hypothesis was the data came from a population with mean equal to zero. I found significant differences ( $t(23) = 5.5087, p \leq 0.0001$ ) of the perceived intelligence between the two agents. Most participants perceived the advice agent to be more intelligent than the critique agent, even though both used the same underlying ML algorithm. Both agents were equally capable, but the teaching method caused the human-agent interaction to be very different. Figure 6.8 shows the number of respondents for each value. Most participants found the action advice agent to be more intelligent than the critique agent, with 54% giving scores +3 or greater. Only 3 participants rated the critique agent as more intelligent, and none of these rated it strongly so.

At the end of the experiment, participants were given the option to write about what

made them think one agent was more intelligent than another. The main factors that made people perceive one agent as more intelligent than another were:

**Compliance with input:** whether the agent did what it was told

**Responsiveness:** how quickly the agent learned

**Effort:** the amount of input needed to train the agent

Additional factors participants noted that affected the agents' perceived intelligence levels were:

**Complexity:** the complexity of allowed human instruction

**Transparency:** whether the human understands why the agent made its choices

**Robustness and Flexibility:** the agent's ability to correct mistakes and learn alternate policies

### *Compliance*

Surprisingly, the advice agent's ability to immediately follow the human's instructions was a main reason participants gave for determining the intelligence of the agent, regardless of whether they thought the advice or critique agent was more intelligent.

Most participants said the advice agent was more intelligent because it correctly followed by the participant's input. The following are from the participants' responses:

**P22** "The Action Advice was significantly more intelligent than the Critique. It followed my comments and completed the task multiple times."

**P18** "I feel the Action Advice may be more intelligent because it took my commands well."

**P10** "Action advice seemed more intelligent because it was receptive to concise commands as opposed to the critique agent in which training with both positive and negative reinforcement got difficult."

**P19** “Action agent appeared to take direction better.”

**P23** “The Action Advice followed my instructions.”

**P9** “The action agent listened to me and did what I told it to do. With the critique agent it would do an action I told it not to do over and over again. ”

**P8** “Advice agent was able to follow the commands and get to the exit with the person. Critique agent could not follow the commands.”

**P3** “The advice agent responded with the correct results and was able to perform the tasks with minimal effort.”

The few participants who thought the critique agent was more intelligent came to this conclusion while admitting that the action advice agent performed better. They indicated that the advice agent was ‘just following orders’, not showing independent thought. The critique agent may not have performed well or made clear and immediate use of the human feedback, but that behavior was interpreted as figuring out the task on its own, like a willful or independent person who does not follow directions. The following are from the participants’ responses:

**P24** “The critique agent was technically more ‘intelligent’ simply due to the fact that it was making its own decisions. The Action Advice agent was simply controlled, but this led to a more efficient mission despite the lack of intelligence.”

**P21** “Action Advice was following commands, Critique had to ‘figure out’ to accomplish goal off my +/- feedback.”

**P13** “The action advice robot followed instructions very well. The critique robot would go in a direction multiple times despite being told ‘no.’ The critique robot did eventually learn though. As I write this, I realize I was wrong: the critique robot was more intelligent but not as good at the task.”

**P7** “I thought the critique agent was more intelligent than the action advice agent because although the action advice agent was able to get to the end successfully more, it was not as good at picking the right path without human advice.”

Note that the exploration policy of both agents without human input was identical, as directed by the Bayesian Q-learning algorithm.

These explanations bring up an interesting point of the definition of intelligence. It would be interesting to know if their opinion of intelligence would persist to other domains where the purpose of the robot is to help a human do a task instead of playing a game successfully. A robot that can successfully figure out “on its own” how to navigate a maze may seem more intelligent, but a robot that puts clothes or dishes in the wrong place multiple times after being told it was wrong may be seen as less intelligent.

### *Responsiveness*

Several participants said the advice agent was more intelligent because it was more responsive, i.e. it learned faster than the critique agent. The following are from the participants’ responses:

**P11** “I felt that the action advice agent was more intelligent because it seemed to learn faster and recover from mistakes faster.”

**P15** “The action advice agent seemed to respond to my statements faster.”

**P4** “The advice agent is more intelligent because it adapted to what I defined as the ‘desired state’ more quickly and with less instruction.”

**P20** “The action agent is able to perform correctly without input much quicker.”

**P14** “The speed of response to my input was mostly how I judged the advice agent was more intelligent.”

**P5** “The action agent responded faster when controlled.”

### *Effort*

Similar to the training speed, several participants mentioned the advice agent was more intelligent because less effort was required to teach it. The agent appeared to be able to be more autonomous with fewer instructions from the human teacher. The following are from the participants' responses:

- P3** "The Advice agent responded with the correct results and was able to perform the tasks with minimal effort."
- P2** "The action advice agent was nearly able to complete the mission autonomously by the end of training, but the critique agent still appeared to move somewhat randomly."
- P1** "As time went by, I could say less and action advice agent could still do most on its own."
- P4** "The advice agent is more intelligent because it adapted to what I defined as the "desired state" more quickly and with less instruction."

### *Robustness & Flexibility*

During the exit interview, a couple of participants mentioned that they trained the advice agent to complete the task one way, but then they retrained the advice agent to follow an alternate policy. This flexibility and retraining was not possible with the critique agent. Additionally, participants mentioned that when the advice agent made mistakes it seemed to recover from them more quickly. The following are excerpts from the participants' responses:

- P11** I felt that the action advice agent was more intelligent because it seemed to learn faster and recover from mistakes faster.



**P9** It [action agent] listened to me and did what I told it to do. With the first [critique] agent it would do an action I told it not to do over and over again. With the action agent it usually performed the actions I instructed it to do.

### *Transparency & Complexity*

At least one person (P22) mentioned that, “the Action Advice was significantly more intelligent than the Critique because I was always able to understand why it made the choices it did.” This implies that the transparency of an agent’s actions affect how intelligent the person perceives the agent to be.

I expected the complexity of the advice compared to the simpler critique to be a main factor in the perceived intelligence of the agents. While some participants did discuss that they thought the advice agent was more intelligent based on the complexity of the input, it was a less-prevalent reason compared to performance, speed, and amount of instruction. The following are excerpts from the participants’ responses:

**P12** “Complexity of Action Advice made that agent seem more intelligent.”

**P10** “Action advice seemed more intelligent because it was receptive to concise commands as opposed to the critique agent in which training with both positive and negative reinforcement got difficult. It was discouraging to give the agent negative critique when the desired command was down and it oscillated between up, left and right, due to fear of it going right (into the radiation zone) if a negative critique landed on Left.”

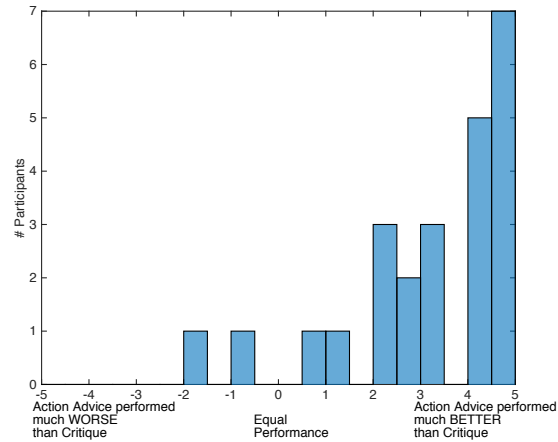


Figure 6.9: Perceived Performance Direct Comparison

#### 6.5.5 Perceived Performance

Most participants thought action advice performed better than critique. In fact, no participants found the action advice agent to perform much worse than critique, and over 60% of participants felt action advice performed much better than critique. I did not ask participants for free-form responses on perceived performance, assuming the responses to “If one agent performed the task as you intended better than the other, what made you think that?” would be redundant.

#### 6.5.6 Immediacy

Only two participants found action advice to respond slower than critique, and they had issues with the ASR software accuracy. Similar to perceived performance, I did not ask participants for free responses about immediacy, assuming tautological answers to: “If one agent responded to your input faster than another, what made you think that?” However, through the participants’ free responses to other human factors metrics, I have found the immediacy of the agent’s response to be a main factor impacting both frustration and perceived intelligence.

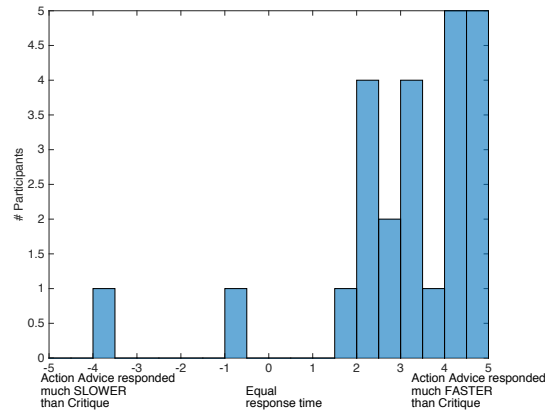


Figure 6.10: Immediacy Direct Comparison

$\kappa$	Level of Agreement
0 - 0.20	None
0.21 - 0.39	Minimal
0.40 - 0.59	Weak
0.60 - 0.79	Moderate
0.80 - 0.90	Strong
Above 0.90	Almost Perfect

Table 6.3: Interpretation of Cohen's kappa statistic

### 6.5.7 External Verification of Long Response Data Analysis

As part of the qualitative analysis of the free response answers, two researchers independently classified each of the responses for the frustration, perceived intelligence, and transparency questions. A Cohen's kappa coefficient for the inter-rater reliability was calculated for the two reviewers, with a goal of achieving a 0.60 or above, indicating moderate agreement (Table 6.3). As seen in Table 6.4, the qualitative analysis coding achieved a kappa of  $> 0.60$  in all categories, with the majority achieving  $> 0.70$ .

Category	Frustration	Perceived Intelligence	Transparency
Compliance with Input	0.621	0.645	1.0
Immediacy	0.792	1.0	0.771
Transparency	0.727	-	-
Complexity	0.694	-	-
Randomness	0.831	-	-
Powerlessness	0.713	-	-
Effort	-	0.633	-

Table 6.4: Cohen’s kappa for Study 3

#### 6.5.8 Objective metrics on ML Performance

*Paired tests on Interval ML Data:* Paired t-tests were conducted for the ML metrics that were interval in nature: training time and average reward. The null hypothesis was the pairwise difference between the two paired groups had a mean equal to zero. I found significant differences of the training time and reward between the two agents. Table 6.5 shows the results of each t-test.

*Paired tests on Ordinal ML Data:* Wilcoxon Signed Rank tests were conducted for the objective metrics that were ordinal in nature: average number of training inputs from the human and average number of steps to complete each episode. The null hypothesis was the difference between the pairwise samples came from a distribution with zero median. I found that there was a significant difference between advice and critique in the number of steps it took to complete a level. However, there was not a significant difference in the amount of input given by the human teacher. Table 6.6 shows the results of each test.

Table 6.5: Paired T-tests on ML Performance

Objective Metrics	Accept/Reject	$t(23)$	$p$	$\mu_{advice}$	$\mu_{critique}$
Training Time	Reject	-3.9321	7.6406e-04	38.51	99.81
Avg Reward	Reject	7.3106	3.3849e-07	-18.92	-89.57

Table 6.6: Wilcoxon Signed Rank tests on ML Performance

Objective Metrics	Accept/Reject	$Z$	$p$
Avg Number Inputs	Accept	1.1201	0.2627
Avg Number Steps	Reject	-2.9057	0.0037

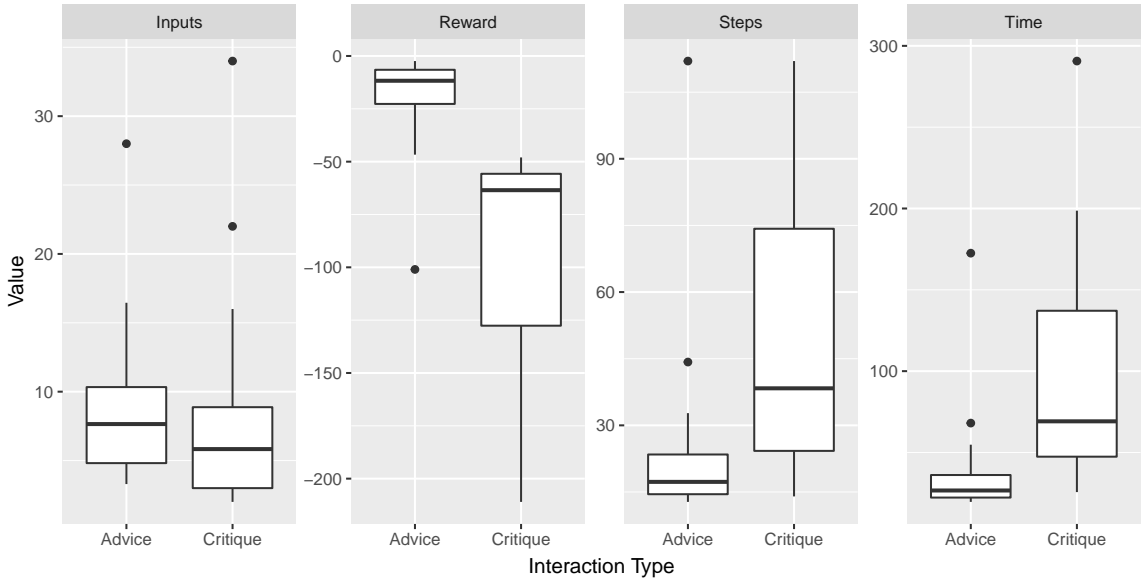


Figure 6.11: ML Performance Comparison.

*Training Time*

The training time was significantly shorter for advice than critique.

*Amount of Human Input*

The amount of input provided by the human teacher was approximately the same for advice and critique.

### *Number of Steps to Complete an Episode*

The advice agent was able to complete an episode in significantly fewer steps than critique. The critique agent spends many steps wandering around the domain, seemingly aimlessly. The median number of steps for advice was 24, while the critique agent doubled that with a median of 50.

### *Reward*

The advice agent received a significantly higher reward than critique.

#### 6.5.9 Evaluating the User's Path

Figure 6.12 shows the Radiation World domain with two specific paths. The first path is the shortest path to the goal. This is the path that will earn the highest possible reward from the reward signal, but will take the agent immediately adjacent to the radiation. The second path allows the agent to keep a thick wall between the agent and the radiation leak while only adding two steps to the total length of the shortest path. This second path that avoids radiation will earn slightly less reward (100 vs 102), which means it is a path that the underlying RL agent will never follow compared to the shortest path. However, many people would rather the agent take two steps out of the way in order to keep a wall between the agent and the radiation. For the following discussion, I will call the two paths 'Shortest' and 'AvoidRadiation'.



Figure 6.12: Radiation World with the 1) Shortest Path to Goal and 2) Path to Goal that Avoids Radiation.

To analyze whether the agents followed paths closer to the Shortest or AvoidRadiation paths, first I created a sequence of (x,y) coordinates to represent both paths. At each time step, the agent's location was compared to the sequences for both the Shortest and AvoidRadiation paths. I used Manhattan distance as a distance metric.

For example, let  $(x_{agent}, y_{agent})$  be the agent's coordinates at time step  $i$ . Let  $(x_{avoidRad}, y_{avoidRad})$  be the  $i^{th}$  element in the AvoidRadiation sequence of coordinates. Then,  $dist$  is the distance between the agent's location and the AvoidRadiation path at time step  $i$ .

$$dist = |x_{agent} - x_{avoidRad}| + |y_{agent} - y_{avoidRad}|$$

The total distance between the agent's path and the AvoidRadiation path is the distance calculation summed over the entire path. Calculating distance in this way penalizes the distance from the path as well as if the agent is in a location at the wrong time.

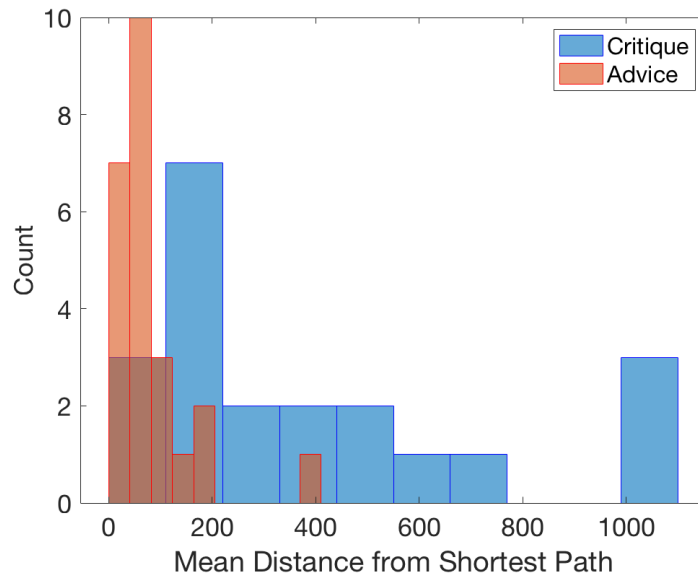


Figure 6.13: Histogram: for each participant, the mean distance (averaged across episodes) from the Shortest path.

Figure 6.13 shows a histogram that compares each participant's mean distance from the Shortest path. The critique agent tends to have a much higher average distance because the agent takes many more steps to complete the episode, particularly in early trials, compared to the action advice agent.

80% of participants using the advice agent chose a route closer to the AvoidRadiation path during the first episode. A lower 61% of critique agents followed a route closer to the AvoidRadiation path during the first training episode.

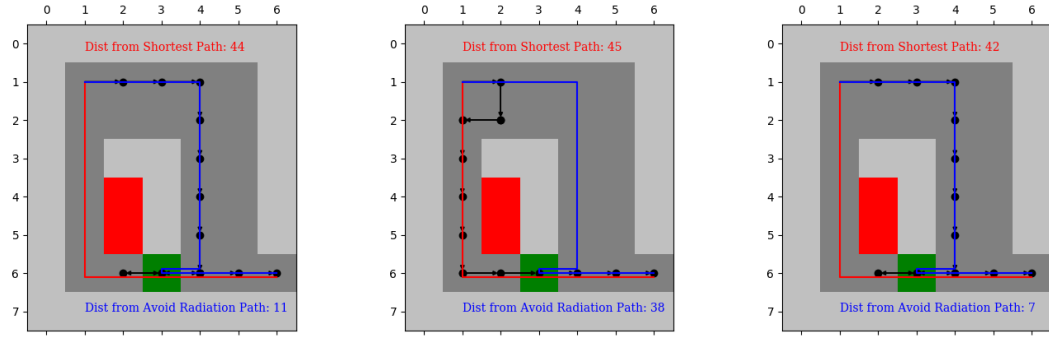


Table 6.7: Percentage of Participants who tried to 1) always follow the Shortest path, 2) always follow the AvoidRadiation path, or 3) switched between the Shortest and AvoidRadiation paths.

Path History	Critique	Advice
Every episode closest to Shortest Path	23.8	0.0
Every episode closest to AvoidRadiation Path	42.9	28.0
Switched between Closest and AvoidRadiation Paths	33.3	72.0

Table 6.7 shows the percentage of participants whose agents always followed paths closest to the Shortest path, AvoidRadiation path, or switched between the two. Close to one quarter of critique agents always followed paths closest to the Shortest path. On the other hand, not even one action advice agent always took a path closest to the Shortest path. Remember that the human does not have direct, immediate control over the critique agent. When using the action advice agent, in which the person did have direct, immediate control over the agent’s future actions, participants either had the agent always follow the AvoidRadiation path or switched between the Shortest and AvoidRadiation paths. The fact that 0.0% of participants had the agent always follow the Shortest path (that earns the highest reward according to the reward signal) is further evidence that we should create agents that can learn to complete the task as the person intends, even if it means following a slightly sub-optimal policy according to the reward function.

Several of the participants trained the advice agent to follow multiple paths. One participant said he did this to understand the agent’s limitations - in cases such as this, the path followed is less indicative of whether people would train an agent to avoid radiation in real life and more representative of the human trying to learn the agent’s capabilities. Figure 6.14 shows an example of a participant training the agent to follow drastically different paths in different episodes.



a: Episode 2: Avoid Radiation Path

b: Episode 3: Shortest Path

c: Episode 10: Avoid Radiation Path

Figure 6.14: Example of a participant training the agent to follow different paths in different episodes using the Newtonian Action Advice agent.

## 6.6 Study 3: Discussion

### 6.6.1 Perceived Intelligence: More than simply managing expectations

A robot's physical appearance is often created to manage a person's expectations (and ideally to match the capabilities) of the intelligence and level of interaction the robot is capable of. If the robot is not terribly intelligent or sophisticated, a humanoid appearance is not a good design choice - people expect the robot to have a human-level intelligence and are frustrated when the robot proves to be less intelligent than expected. The mismatch between the expected and actual intelligence causes frustration [51].

Similarly, the results here indicate that the teaching method, like morphology, impacts expectations. However, unlike morphology I suggest that the teaching method should not be downgraded when designing a robot, because it would not achieve the same result of managing expectations. With a robot's morphology, the physical appearance is chosen to match its capabilities. With teaching methods, choosing critique over action advice can cause the human to think worse of the agent regardless of its capabilities. In the case of teaching methods, the underlying ML agent is the same, but the human experience is stymied by a poor human-agent interaction.

### 6.6.2 Summary of participants' long responses

The nested Venn diagram in Figure 6.15 shows a summary of which factors impact the human teacher's frustration as well as perceptions of transparency and intelligence based on the free responses. The main factor impacting transparency was compliance with input; and these two factors along with complexity impacted both the human's frustration and perceived intelligence of the agent. Feelings of powerlessness, a probabilistic use of the user's instruction, and a sensitivity to the ASR software increased frustration. Robustness, flexibility, and effort impacted perceived intelligence.

I find it interesting that complexity does not detract from the human experience. The more complex form of input, action advice, created a better user experience by lowering frustration and creating a perception of a more intelligent agent. There is likely a point at which the input could be designed to become so complex and detailed that it becomes unwieldy and detracts from the user experience. On the other side of the complexity scale, the results show that input of a too-simplistic form can damage the human's satisfaction with the agent.

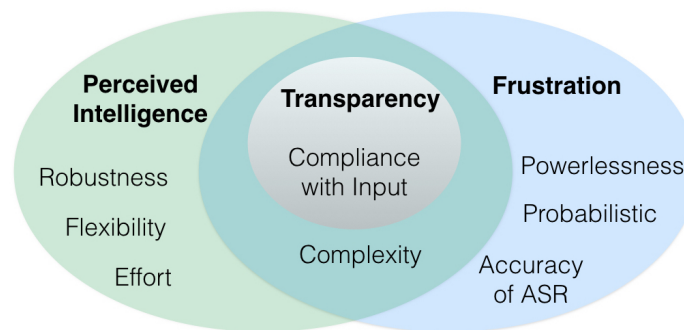


Figure 6.15: Summary of factors that impact frustration, transparency, and perceived intelligence based on participants' free responses.

### 6.6.3 Design considerations

The results of this study indicate that specific characteristics of an interaction algorithm will impact the human teacher's experience. Before researchers or designers create, modify, or choose an interaction algorithm, I suggest they consider the following characteristics to improve the human's experience of working with the agent:

1. **Instructions about future, not past.** An algorithm that uses instructions about the future (such as action advice) instead of the past fosters a positive human experience because it increases the perceived control the human has over the agent, allows for greater transparency of how the agent uses the instructions, and enables the human to immediately detect issues with the agent's compliance. Choosing a rhetorically positive teaching method will also set the tone for a more positive human-agent interaction.
2. **Compliance with Input.** If the human teacher sees the agent following instructions, the person will be less frustrated and perceive the agent to have a higher performance and intelligence.
3. **Empowerment.** An interaction algorithm that forces the agent to clearly, immediately, and consistently follow the human's instructions will decrease feelings of powerlessness, which will in turn decrease the person's frustration.
4. **Transparency.** An algorithm in which the human teacher can clearly understand how the agent used the human's instructions to choose an action will decrease frustration and increase perceived intelligence. A greater transparency can be achieved if an agent immediately complies with the instructions.
5. **Immediacy.** The human's experience will be improved if the agent immediately responds to the instructions because it creates a sort of instant gratification for the human.
6. **Deterministic interaction.** While the underlying ML algorithm will doubtlessly be probabilistic, the interaction with the human should be such that the agent follows the

instructions in a reliable, repeatable, non-probabilistic manner. A deterministic interaction with the human will decrease frustration and decrease the person's uncertainty in the agent that degrades trust.

7. **Complexity.** An algorithm that allows for more complex instructions than binary good vs. bad critique will decrease frustration and increase perceived intelligence.
8. **ASR accuracy.** When choosing ASR software, it is worth the effort to improve the accuracy in order to decrease the human's frustration. Also, while a person is more aware of the accuracy and processing duration for an advice agent, this does not mean the critique agent is better in this regard. Rather, the critique agent trades processing lag for less transparent agent response, which increases frustration and decreases perceived intelligence.
9. **Robustness and Flexibility.** The ability to correct mistakes and teach the agent alternate policies improves the human teacher's experience.

#### 6.6.4 Thoughts on using critique for IML applications

At first glance and from a ML researcher's perspective, critique is a very attractive option for using human feedback. It is simpler, binary feedback (positive vs. negative). It does not require grounding feedback to the state/action space, and would not need to be altered when switching between domains or to embodied robots. Critique can be incorporated into machine learning algorithms in several ways; it can be used directly as a reward signal or, more efficiently, as policy information in a Policy Shaping algorithm. However, Policy Shaping does not promote a positive human experience. We need a better way to connect critique to ML. Is it worth the time and effort to pursue better critique algorithms? Critique is perceived as less intelligent than action advice, and is not as human-friendly. People cannot directly control a critique agent, leading to feelings of powerlessness. It is unclear how critique is being used by the agent during training, and people become frustrated with the lack of transparency. People give instructions about the future, even if expressly told

not to [18], so it is not intuitive to use an interaction method that disallows future instructions. There is no definitive way to solve the credit attribution problem. Also, critique is inherently negative from a rhetorical perspective. It is tempting to focus on making action advice better instead of pursuing critique. In the end, it would be beneficial to use both advice and critique since people naturally switch between both, but that is beyond the scope of this work.

#### 6.6.5 Limitations

I have shown that the interaction algorithm impacts the human’s satisfaction with the agent. Specifically, the Newtonian Action Advice agent creates a better experience for the human than Policy Shaping. In terms of teaching methods, our experiment used Newtonian Action Advice as an ambassador of action advice, which can be thought of as verbal demonstrations, and Policy Shaping as a representative of critique. I cannot use the results to directly guarantee that all action advice algorithms will create a better user experience than all critique algorithms. However, after analyzing the participants’ explanations of what impacted the user experience, I identified nine design considerations that each indicate action advice is the superior teaching method and is inherently more likely to create a better experience for the human teacher than critique.

The experimental scenario was chosen to even the playing field between the advice and critique agents. A domain with a limited dimension was chosen so a critique agent could be trained in less than 15 minutes. Four primitive actions were used so the advice agent was limited to four possible inputs compared to critique’s two.

A limitation of the action advice agent is a menu of available actions must be defined by the researcher. The available natural language descriptions of the actions were grounded to the agent’s corresponding primitive actions prior to the experiment. For example, the word ‘right’ was grounded to the agent’s action that would move the agent one square to the right. The critique agent did not share this limitation.

## 6.7 Study 3: Conclusion

I have shown that the interaction algorithm can impact the human's experience with an IML agent. The human experience differed between agents and *could be measured*. The Newtonian Action Advice agent created a better experience for the human than the critique-driven Policy Shaping.

I have identified nine main characteristics that impact the human's experience with the agent, including: using human instructions about the future, compliance with input, empowerment, transparency, immediacy, a deterministic interaction, the complexity of the instructions, accuracy of the speech recognition software, and the robust and flexible nature of the interaction algorithm.

These design characteristics suggest that, in terms of teaching methods, action advice is likely to create a better experience for the human teacher than critique.

This chapter demonstrates that it is not enough to design algorithms that can theoretically use human input; we must go further and design algorithms that create a positive human experience. IML algorithms must be verified using human factors metrics such as frustration in addition to traditional ML metrics such as cumulative reward.

## 6.8 RQ Results Summary

- The interaction method (e.g. Policy Shaping critique vs. Newtonian Action Advice) impacts the human teacher's experience with the agent (perceived intelligence, perceived performance, frustration, immediacy, transparency). In fact, Newtonian Action Advice creates a better user experience than Policy Shaping critique.
  - *Validation*: Study 3.
  - *Supports thesis*: Part 1 (human experience) - this work studies how and why decisions made during the design of interaction algorithms impact the human's experience with the agent.

- The interaction method (e.g. Policy Shaping critique vs. Newtonian Action Advice) impacts the objective metrics of the teacher’s interaction with the agent (reward, training time, amount of human input provided, number of steps for the agent to complete each episode). With human teachers, NAA earned a higher reward in a smaller training time using fewer steps to complete each episode than Policy Shaping critique. Both methods used approximately equal amounts of human input.
  - *Validation:* Study 3.
  - *Supports thesis:* Part 3 (performance) - designing an RL agent (NAA) that improves the human experience with the agent did not detract from the agent’s capabilities.
- Many design decisions made during the creation of an interaction algorithm impact the human teacher’s experience, including whether instructions are about the future or past, whether the agent complies with human input, how immediately the agent uses the human input, whether the interaction with the agent is probabilistic, and the accuracy of the ASR software.
  - *Validation:* Study 3.
  - *Supports thesis:* Part 1 (human experience) - this work studies how and why decisions made during the design of interaction algorithms impact the human’s experience with the agent.
- Action advice (“Go left. Move right.”) is more rhetorically positive than critique (“Good. Bad.”)
  - *Validation:* Study 3.
  - *Supports thesis:* Part 1 (human experience) - this work explored how rhetoric is a factor that should be taken into account when choosing a teaching interaction method.



## CHAPTER 7

### STUDY 4: EFFECT OF ADVICE VARIATIONS - TEASING APART WHAT PEOPLE LIKE ABOUT ADVICE

This chapter investigates how three factors of the design of an interactive reinforcement learning agent - generalization through time, immediacy, and a time delay - impact the user's experience with the agent. We conducted a human-subject experiment in which people trained four agents with different interaction designs to play a simple game using verbal instruction. All agent variations were modified versions of the Newtonian Action Advice algorithm, an interactive reinforcement learning agent that learns from verbal advice like, "go left." The results show that a time delay between when advice is given to and followed by the agent created a poor user experience, while a probabilistic interface generated a poor to middling performance. Whether the agent generalized through time did not have a significant impact on the user experience. IML researchers should be aware of this and design algorithms that have a deterministic interface and minimize the time delay between when advice is given and used.

#### 7.1 Introduction

Our previous study in Chapter 6 analyzed human satisfaction with IML agents compared the human experience of two verbal interaction methods: advice (e.g. "right, left") and critique (e.g. "good, bad") [1]. The study found that the Newtonian Action Advice (NAA) algorithm created a better user experience than a critique Policy Shaping agent. From text responses provided by participants, the study identified several characteristics of the interaction design of IML algorithms they suspected would impact the user experience, including immediacy and a deterministic interaction. In this chapter, I have delved deeper into those design characteristics to directly study the impact of individual design decisions

in IML advice algorithms.

This chapter focuses on the question: *How do the following three factors of an agent's interaction design affect the person's experience with the agent: 1) generalization through time, 2) time delay, and 3) a probabilistic interface?* If we can isolate specific design decisions that impact the user's experience, IML researchers can design algorithms that create a positive user experience. Based on the results of Chapter 6, I hypothesized the human's experience to be improved by designing an interaction algorithm to: have a minimal delay between when human instructions are given and used by the agent; incorporate the human input in a non-probabilistic manner; and generalize the human's input through time so the human does not need to provide as much instruction.

I performed a repeated measures human-subjects experiment in which participants taught four agents how to play a game with different interaction designs. All four agent variations were based on the Newtonian Action Advice algorithm (NAA), which is an IML algorithm that enables people to teach an agent using verbal action advice (i.e. verbal demonstration) such as “move down,” and “go left.”

### 7.1.1 Research Questions

This work investigates how three factors of an agent's interaction design affect the person's experience with the agent.

- Does generalization through time impact the person's experience with the agent? If so, to what extent?
- Does a time delay between when advice is given and used impact the person's experience with the agent? If so, to what extent?
- Does a probabilistic interface impact the person's experience with the agent? If so, to what extent?

### 7.1.2 Algorithm Development

Three variations of the NAA algorithm were created for this study.

### 7.1.3 Method Summary

The research questions in this chapter were tested in Study 4, which was a human-subject experiment conducted in the Radiation World domain.

## **7.2 Algorithm Development: Newtonian Action Advice Variations**

During this study, each participant trained four agents in a different order, creating a balanced experimental design. Each agent was a variation of the NAA algorithm.

### 7.2.1 Algorithm 1: Newtonian Action Advice, 5 Steps

The first algorithm developed for this study was the same NAA algorithm used in Section 5.2 [54]. The “5 Steps” refers to the amount of time steps a human’s advice would be followed before the NAA agent reverted back to its underlying action selection method. If a participant told the agent to move left, the agent would immediately move left for five time steps unless the participant interrupted to provide new, superseding advice.

### 7.2.2 Algorithm 2: Newtonian Action Advice, 1 Step (no generalization)

The second algorithm developed for this study differed from the first by following advice for one time step instead of five. This results in participants being required to provide advice more frequently. For example, if the participant wanted the agent to move down five steps and then right four steps, the participant would need to say, “Down, down, down, down, down, right, right, right, right.” This algorithm variation is testing whether no generalization of advice through time impacts the user’s experience. Many algorithms that learn from demonstrations treat each demonstration as a strict (state, action) pair and do

not generalize either through time or similar states.

### 7.2.3 Algorithm 3: Newtonian Action Advice, 5 Steps, probabilistic

The third algorithm developed for this study differed from the first by incorporating advice in a probabilistic manner. Each time new advice was given by the person, the agent would follow the advice with a 60% probability or choose an action based on the agent's action selection method with a 40% probability.

A certain level of unpredictability may be necessary for a fun user experience [55]. However, unpredictability can induce stress in humans [56]. A previous study found that a Policy Shaping agent that integrates the human's critique with the RL agent in a probabilistic manner resulted in a worse user experience than an NAA agent that incorporates the human's advice in a deterministic manner [1]. This Probabilistic algorithm variation is testing whether, and to what extent, using advice in a probabilistic manner impacts the user experience.

### 7.2.4 Algorithm 4: Newtonian Action Advice, 5 Steps, time delay

The fourth algorithm developed for this study differs from the first by introducing a two second time delay between when advice was provided by the participant and when the agent followed the advice. Users can adapt to frequent, expected time delays [57], and users tend to prefer delays of less than one second [58]. However, time delays can improve performance on difficult tasks [59]. This algorithm variation is testing whether, and to what extent, the immediacy of using advice impacts the user experience.

When designing an advice algorithm, a researcher may not purposefully build in an extended time delay. However, the researcher may not make an effort minimize a time delay unless studies like this one show it greatly impacts the user experience. Many IML algorithms, such as those that learn from critique, do not have a method of immediately using the human's instruction.

### 7.3 Study 4: Method

We conducted a repeated-measures human-subject experiment in which we investigated the effect of four interaction algorithm variations on the human’s satisfaction with the agent. The study collected data from 24 participants who were not experts in machine learning. The age of the participants varied from 18-65 years old.

#### 7.3.1 Radiation World Domain

This study used the same Radiation World domain setup as Study 2 (section 6.4).

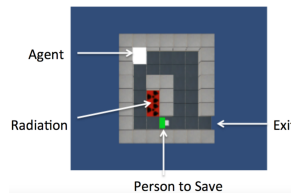


Figure 7.1: Radiation World Initial Condition

The task required participants to teach each agent to rescue a person in Radiation World, which is a game developed in the unity minecraft environment as shown in Figure 7.1 and has been used in other IML studies [60, 54, 1]. In the experimental scenario, there has been a radiation leak and an injured person is located somewhere in the grid unable to move. The agent must find and rescue the person, take him to the exit, all while avoiding the radiation. The task is complete if the agent takes the person to the exit. The task is failed if the agent enters the radiation or exits without the person. The task is repeated for several training episodes; participants were told to stop training when either the agent was performing as they intended or the participant was too frustrated to continue or wanted to stop for any reason. Consequently, the training time varied for each participant and interaction algorithm. After a participant finished training an agent, the participant completed a questionnaire concerning the experience. At the end of the experiment, participants were asked if they had any questions or additional thoughts concerning the experiment in an exit

interview.

All four agents learned from verbal instruction that can be thought of as verbal demonstrations. The verbal instructions were transcribed to text using ASR software. After language processing, the processed instructions were sent to one of the NAA interaction algorithm variations.

People were instructed to tell the agent to move in a desired direction. For example, if participants wanted the agent to move down, they should say, “down.” The only four words recognized by the action advice agents were, “up,” “down,” “left,” and “right.” The action advice agent used the Newtonian Action Advice algorithm to incorporate the advice with the Bayesian Q-learning algorithm’s action selection.

### Experimental Procedure

The procedure followed by all participants is provided below. All participants experienced the algorithm variations in a different order.

1. Greeting and Introduction
2. Instructions for agent 1. Train agent 1. Questionnaire on experience of training agent 1.
3. Instructions for agent 2. Train agent 2. Questionnaire on experience of training agent 2.
4. Instructions for agent 3. Train agent 3. Questionnaire on experience of training agent 3.
5. Instructions for agent 4. Train agent 4. Questionnaire on experience of training agent 4.
6. Free response questions about the experience of training the agents.

#### 7.3.2 Human Experience Measures

The Human Experience measures were created as a modified version of the NASA-TLX questionnaire [61]. The answers to all questions regarding the human experience were collected using an electronic questionnaire that used a sliding bar in which the selected value to the tenths decimal place was shown to the participant.

*Paired tests.* Immediately after training an agent, the participants were asked to score the intelligence of the agent on a continuous scale from [0:10]. A value of 0 indicated that the agent was not intelligent, while 10 meant very intelligent. The same scale of [0:10] was used for four additional metrics: performance, frustration, transparency, and immediacy. Values of 0 corresponded to poor performance, low frustration, non-transparent use of feedback, and a slower response time. Values of 10 meant excellent performance, high frustration, clear use of feedback, and an immediate response time, respectively.

*Explanation of Responses.* After training both agents, in item 6 of the procedure participants were given the option to explain what factors impacted their experience of working with the agents. For example, participants were asked, “If one agent was more frustrating than another, what made you feel that way?” These written responses were entirely free form with no priming or options provided by the experimenter. The purpose of these explanations is to explore *why* the interaction algorithm impacts the human’s experience. The resultant set of design characteristics that affect the person’s relationship with the ML agent allows us to analyze each of these characteristics thoroughly in future work and eventually use the characteristics to direct the design of IML algorithms.

### 7.3.3 Objective ML measures

While participants were training the agents, objective performance metrics were logged to data files. The training time and reward earned per episode were measured as continuous data. The amount of human input provided and the number of actions it took to complete an episode were measured as ordinal count data.

## **7.4 Study 4: Results and Discussion**

The algorithm variations differed across the board in terms of both human factors and objective metrics. The NAA 5 Steps and NAA 1 Step algorithms performed the best, followed by the Probabilistic variation. Overall, the Time Delay variation performed the worst.

### 7.4.1 Human Factors Metrics

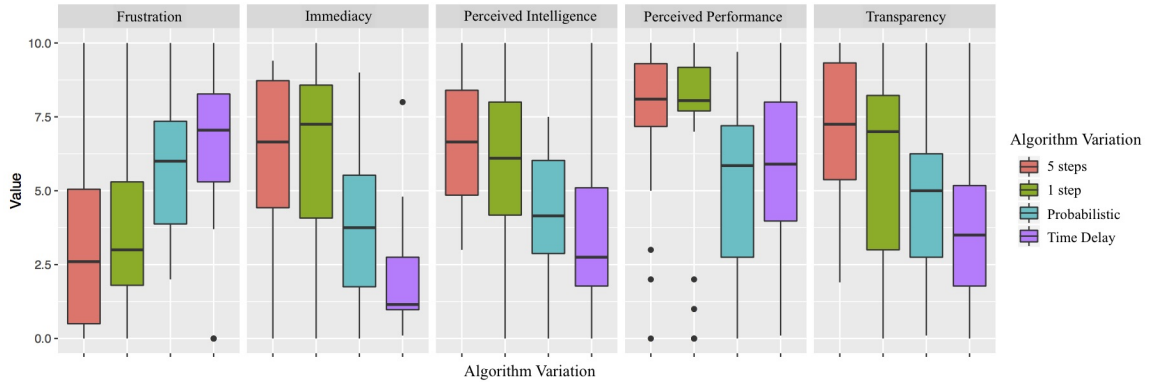


Figure 7.2: Human Factors Results for NAA Variations

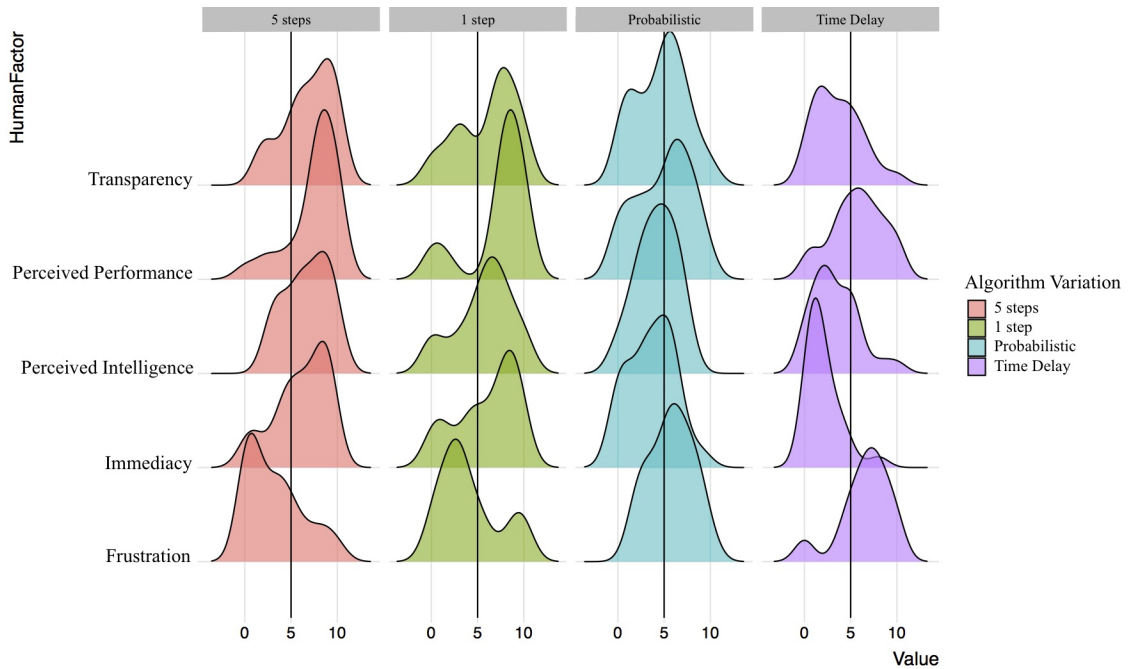


Figure 7.3: Human Factors Results for NAA Variations

Figure 7.2 shows the boxplots of all the human factors results. For all human factors metrics except frustration, higher values represent a better human experience (better performance, greater transparency, more immediate, and more intelligent). For frustration, higher values indicate a higher level of frustration, and therefore a worse human experience. The 5



Steps and 1 Step agents performed better than the Probabilistic and Time Delay variations on every human factors metric.

Figure 7.3 shows the distribution of responses for the human factors metrics. A line is drawn at the middling value of 5 for each of the human factors. Both the 5 Steps and 1 Step (no generalization) variations show distributions that are skewed toward a positive human experience: high transparency, perceived performance, intelligence, immediacy, and low frustration. The distributions for the Probabilistic variation are shifted toward a middling and poor user experience. The Time Delay distributions are skewed toward a negative user experience and are almost the opposite of the 5 Steps and 1 Step variations; the distributions show that participants thought the Time Delay variation was not transparent, not intelligent, responded slowly to advice, had middling performance, and was highly frustrating.

Each of the violin plots (Figures 7.4 - 7.8) show the data distribution drawn around a boxplot.

### *Frustration*

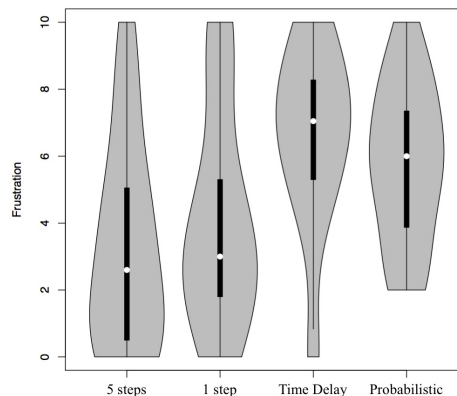


Figure 7.4: Frustration Results for NAA Variations

Figure 7.4 shows the 5 Steps and 1 Step agents resulted in the lowest frustration. The Time Delay variation was the most frustrating, followed closely by the Probabilistic agent.

A one-way repeated measures ANOVA shows that frustration is not equal across all

Table 7.1: Frustration: Repeated-Measured ANOVA

	SS	SSE	df	F	$Pr(> F)$
Intercept	2325.59	249.78	23	214.1453	3.836e-13
Algorithms	167.35	490.22	69	7.8518	0.0001397

NAA algorithm variations (Table 7.1). A post-hoc analysis of a series of paired t-tests shows the 5 Steps and 1 Step variations are equal in frustration. The Probability and Time Delay variations are equal in frustration. However, the 5 Steps and 1 Step variations are less frustrating than the Probability and Time Delay variations.

### *Perceived Performance*

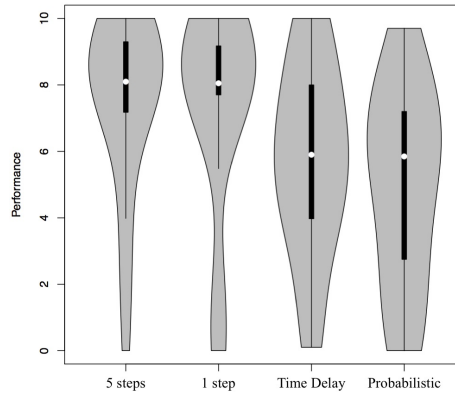


Figure 7.5: Perceived Performance Results for NAA Variations

Table 7.2: Perceived Performance: Repeated-Measured ANOVA

	SS	SSE	df	F	$Pr(> F)$
Intercept	3902.8	290.21	23	309.3039	7.814e-15
Algorithms	113.0	490.85	69	5.2971	0.002405

Figure 7.5 shows the 5 Steps and 1 Step agents were clustered around a very high

perceived performance. The Time Delay and Probabilistic variations had more variation and were overall seen as performing worse.

A one-way repeated measures ANOVA shows that perceived performance is not equal across all NAA variations (Table 7.2). A post-hoc analysis of a series of paired t-tests shows the 5 Steps and 1 Step variations are perceived to have equal performance. The Probability and Time Delay variations are equal in perceived performance. However, the 5 Steps and 1 Step variations are perceived to perform better than the Probability and Time Delay variations.

### Transparency

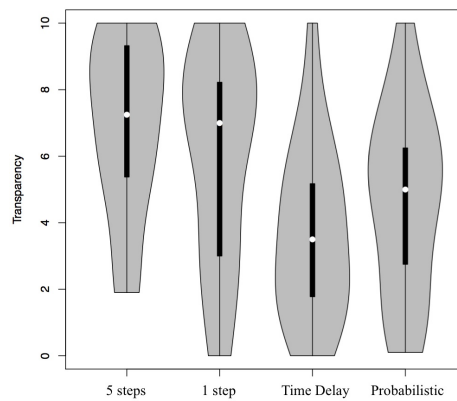


Figure 7.6: Transparency Results for NAA Variations

Table 7.3: Transparency: Repeated-Measured ANOVA

	SS	SSE	df	F	$Pr(> F)$
Intercept	2715.8	293.78	23	212.619	4.133e-13
Algorithms	151.1	423.63	69	8.204	9.584e-05

Figure 7.6 shows the 5 Steps agent was seen to be the most transparent, with no participants scoring less than 2. The 1 Step agent also scored as very transparent. The Probabilis-

tic variation was seen as middling in terms of transparency. The Time Delay variation was seen as the least transparent, with an average less than 4.

A one-way repeated measures ANOVA shows that transparency is not equal across all NAA variations (Table 7.3). A post-hoc analysis of a series of paired t-tests shows the 5 Steps and 1 Step variations are perceived to be equally transparent. The Probability and Time Delay variations are equal in transparency. However, the 5 Steps and 1 Step variations are perceived to be more transparent than the Probability and Time Delay variations.

### *Immediacy*

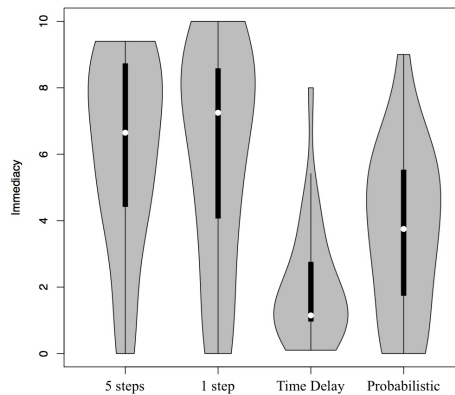


Figure 7.7: Immediacy Results for NAA Variations

Table 7.4: Immediacy: Repeated-Measured ANOVA

	SS	SSE	df	F	$Pr(> F)$
Intercept	1893.04	159.93	23	272.236	3.058e-14
Algorithms	284.45	486.21	69	13.456	5.230e-07

Figure 7.7 shows the 5 Steps and 1 Step agents were seen as implementing a human's advice very quickly. The Probabilistic variation had middling performance. The Time Delay variation was almost universally seen as responding to advice very slowly, with a mean less than 2.

A one-way repeated measures ANOVA shows that immediacy is not equal across all NAA variations (Table 7.4). The 5 Steps and 1 Step variations are perceived to respond to advice equally quickly. However, the Time Delay variation is not equal to the Probability variation, which is in turn not equal to the 5 Steps algorithm. People found the Time Delay variation to respond the slowest to input, followed by the Probability variation. The 5 Steps and 1 Step algorithms responded the fastest.

### *Perceived Intelligence*

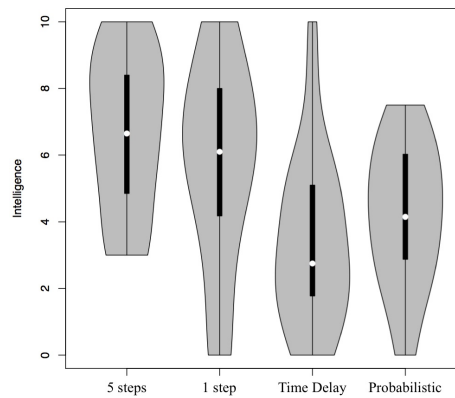


Figure 7.8: Perceived Intelligence Results for NAA Variations

Table 7.5: Perceived Intelligence: Repeated-Measured ANOVA

	SS	SSE	df	F	$Pr(> F)$
Intercept	2379.05	190.0	23	287.987	1.679e-14
Algorithms	161.12	398.3	69	9.304	3.023e-05

Figure 7.8 shows the 5 Steps agent had the highest mean score of perceived intelligence, with no participants responding less than a score of 3, followed closely by the 1 Step agent. The Probabilist variation had a higher mean than the Time Delay agent, but the Probabilistic agent had no scores greater than 7.5.

A one-way repeated measures ANOVA shows that perceived intelligence is not equal across all NAA variations (Table 7.5). A post-hoc analysis shows the 5 Steps and 1 Step variations are equal in perceived intelligence. The Probability and Time Delay variations are equal in perceived intelligence. However, the 5 Steps and 1 Step variations are perceived to be more intelligent than the Probability and Time Delay variations.

## 7.4.2 Objective Metrics

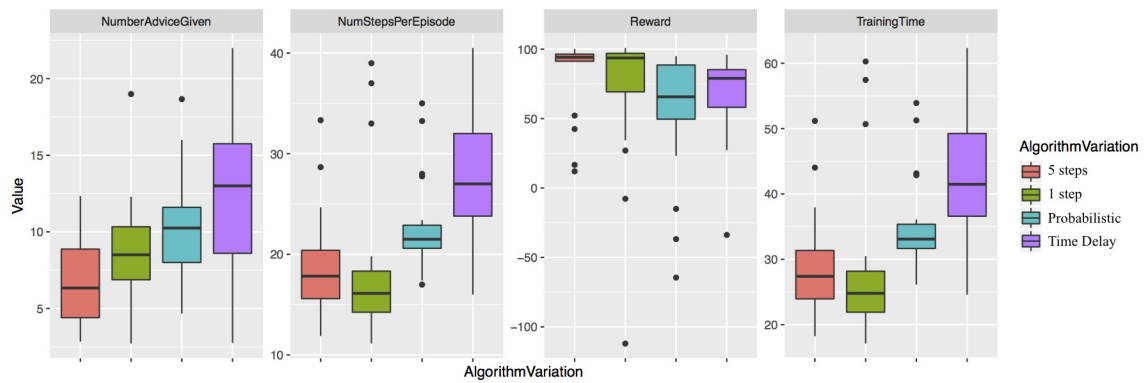


Figure 7.9: Objective Metric Results for average values of NAA Variations

The 5 Steps and 1 Step agents performed approximately equally in terms of objective metrics; they earned a higher reward in less time than the Time Delay and Probabilistic variations. People provided less advice for the 5 Steps algorithm than the others.

### Avg Training Time

Table 7.6: Training Time: Repeated-Measured ANOVA

	SS	SSE	df	F	$Pr(> F)$
Intercept	96617	2595.9	20	744.383	2.2e-16
Algorithms	2993	4163.8	60	14.375	3.633e-07

In Figure 7.9, the 1 Step agent has the lowest mean training time, followed closely by the 5 Steps agent. The Time Delay algorithm took the longest for people to train.

A one-way repeated measures ANOVA shows that training time is not equal across all NAA variations (Table 7.6). A post-hoc analysis of a series of paired t-tests shows the 5 Steps and 1 Step variations are equal in training time. The Probability variation has a longer training time than the 5 Steps and 1 Step variations. The Time Delay variation has the longest training time.

#### *Avg Amount of Advice Given by Human*

Table 7.7: Amount of Advice: Repeated-Measured ANOVA

	SS	SSE	df	F	$Pr(> F)$
Intercept	7796.3	457.17	20	341.068	4.916e-14
Algorithms	340.2	687.12	60	9.903	2.135e-05

In Figure 7.9, the 5 Steps agent required the least amount of advice from the human during training. Compared to the 5 Steps algorithm, people gave more advice to the 1 Step agent; this is expected because the lack of generalization through time in the 1 Step agent means the person needs to repeat themselves. The Probabilistic variation was given more advice than the 1 Step agent. The Time Delay variation was given the most advice out of all the variations; this is interesting because the Time Delay agent can theoretically be trained using the same amount of advice as the 5 Steps algorithm, but people either did not realize a time delay existed or refused to wait for their advice to be enacted and kept providing advice.

A one-way repeated measures ANOVA shows that amount of advice given by the human is not equal across all NAA variations (Table 7.7). In fact, a post-hoc analysis shows none of the variations resulted in an equal amount of advice. People provided the least

amount of input while training the 5 Steps variation, followed by the 1 Step, Probability, and Time Delay variations.

#### *Avg Reward*

Table 7.8: Earned Avg Reward: Repeated-Measured ANOVA

	SS	SSE	df	F	$Pr(> F)$
Intercept	391159	43853	20	178.3938	2.004e-11
Algorithms	8152	80719	60	2.0198	0.1207

In Figure 7.9, the 5 Steps agent earned the highest reward with the least variation, followed by the 1 Step (no generalization) algorithm. The Time Delay had the third highest average reward per episode. The Probabilistic variation earned the lowest average reward.

A one-way repeated measures ANOVA shows that average reward per episode is statistically equal across all NAA variations (Table 7.8). Most algorithms are compared using the reward earned by the agent. The human factors results show that the user experience is very different for the variations, even if the average reward is statistically equal.

#### *Avg Number of Steps Agent took to Complete Each Episode*

Table 7.9: Number of Steps: Repeated-Measured ANOVA

	SS	SSE	df	F	$Pr(> F)$
Intercept	40815	1084.8	20	752.478	2.2e-16
Algorithms	1268	1747.0	60	14.514	3.225e-07

In Figure 7.9, the 1 Step agent has the lowest mean number of steps to complete each episode, followed closely by the 5 Steps agent. The Time Delay algorithm took the greatest number of steps to complete each episode with the greatest variation. This data distribution



is very similar to training time since each game step takes approximately the same amount of time to execute.

A one-way repeated measures ANOVA shows that average amount of steps the agent took to complete each level is not equal across all NAA variations (Table 7.9). A post-hoc analysis shows the 5 Steps and 1 Step variations are equal in steps to completion. The Probability variation has a larger average steps to completion than the 5 Steps a 1 Step variations. The Time Delay variation has the largest average steps to completion.

### 7.4.3 Open-Ended Responses from Participants

At the end of the experiment, participants were given a chance to provide text responses about their experiences interacting with the agents. According to the responses, several aspects of the agent's interaction impacted the participants' perceived intelligence and transparency as well as frustration with the agent. There were many commonalities across the participants' responses; we have organized and tallied these responses in Table 7.10.

The four aspects of the agent's interaction that were most often cited by participants as impacting frustration, intelligence, and transparency were: 1) compliance with input (whether the agent followed the person's advice), 2) immediacy (whether the agent immediately followed the person's advice), 3) amount of instruction the person had to provide, and 4) randomness (whether the agent was perceived to act in a random or deterministic manner).

The question prompts for the long responses are listed below. The participant quotes given in the following sections are answers to these question prompts.

- If one agent was more frustrating than another, what made you feel that way?
- If one agent was more intelligent than another, what made you think that?
- If one agent used your feedback in a clearer way than another, what made you think that?

Table 7.10: Percentage of participants who mentioned in their long responses certain features that contributed to the user experience.

Feature	% Mentioned Feature Impacted HF		
	Frustration	Intelligence	Transparency
Compliance	71	54	23
Immediacy	38	54	50
Less Instruction		29	27
Randomness	21		23
Improvement through time		17	
Memory		9	14
Had to repeat myself	13		
Effort	9		
Frustration		4	
Transparency	4		

As part of the qualitative analysis of the free response answers, two researchers independently classified each of the responses for the frustration, perceived intelligence, and transparency questions. A Cohen’s kappa coefficient for the inter-rater reliability was calculated for the two reviewers, with a goal of achieving a 0.60 or above, indicating moderate agreement (Table 6.3). As seen in Table 7.11, the qualitative analysis coding achieved a kappa of  $> 0.60$  in all categories, with the majority achieving  $> 0.70$ . A few of the less common categories were condensed for this analysis.

#### *Compliance with input*

Compliance with input, i.e. whether the agent followed the person’s advice, was mentioned by approximately 3/4 of participants to lower frustration, over half the participants to increase perceived intelligence, and 1/4 of participants to increase transparency.

**P9, Frustration** “Frustration came from not knowing if the agent was actually taking my input and changing the way it was behaving because of it. (ie.

Category	Frustration	Perceived Intelligence	Transparency
Compliance with Input	0.647	0.625	0.776
Immediacy	1.0	0.917	0.814
Less Instruction	-	-	0.699
Randomness	0.903	-	0.818
Agent Learns / Memory	-	0.834	0.744
Effort	0.882	-	-

Table 7.11: Cohen's kappa for Study 4

talking to a wall))”

**P22, Intelligence** “ I judged its intelligence on how well it mimicked what I told it to do.”

**P1, Transparency** “It went the way I told it to quickly, and I could anticipate when to give an effective command (timing easy to ascertain).”

### *Immediacy*

An agent that immediately followed a participant's advice was mentioned by approximately half of participants as improving the perceived intelligence and transparency of the agent, and it decreased the frustration of almost 2/5 of participants.

**P15, Frustration** “The delay between my instruction and when they performed the action.”

**P5, Frustration** “It was very slow in responding to commands and/or ignored commands.”

**P9, Intelligence** “A more intelligent agent seemed to respond quicker.”

**P19, Transparency** “The ones that responded quickly seemed to use feedback most clearly.”

### *Amount of instruction*

Almost 30% of participants mentioned that they thought an agent was more intelligent and transparent if the agent was able to learn from less advice given by the human teacher.

**P12, Intelligence** “Much less direction required to get it on the correct path.”

**P15, Intelligence** “One agent required me to give it instruction every move which seemed less intelligent.”

**24, Transparency** “One agent responded with less reinforcing instructions.”

### *Randomness*

A little over 20% of the participants wrote that the agent acting randomly caused the person to be more frustrated and think of the agent as less transparent.

**P8, Frustration** “If the agent seemed to be guessing random directions to go if it didn’t know, rather than just staying the last direction I had given it, it was frustrating.”

**P19, Transparency** “The ones that didn’t seem to try random things on their own when I wasn’t talking seemed to use feedback most clearly.”

**P7, Transparency** “When I could predict what telling it something would make it do, or if I could figure out what I’d need to tell it to get it to do what I wanted it to.”

**P5, Transparency** “If I could predict the agents next move even if it wasn’t responding to me accurately.”

### *Improvement through time*

Whether the agent’s performance improved through time was mentioned as a contributing factor by 1/5 of participants for how intelligent the agent was perceived to be. One participant thought one agent was more intelligent than others if it showed “improvement over

repeated tries,” while another rated intelligence based on, “if the agent began to predict where I wanted it to move.”

### *Memory*

If the agent remembered the advice, some participants mentioned that contributed to perceived intelligence and transparency.

**P22, Intelligence** “If one remembers what to do between two loops better than the other, I thought that it was a more intelligent agent.”

**P20, Transparency** “The agent would use the exact advice given from prior runs to achieve the goal.”

### *Other factors*

One participant wrote, “Repeating the same instruction multiple times that the agent was not registering made some agents more frustrating.”

Some other factors mentioned by participants were the effort it took to train the agent, as well as frustration and transparency. The lack of transparency of some agents increased a user’s frustration, which in turn decreased how intelligent the human perceived the agent to be.

#### 7.4.4 Design recommendations

For similar domains, we recommend IML advice agents should be designed with the following characteristics:

- Minimize the time delay between when advice is given by the person and used by the agent.
- Create an interaction that is deterministic from the human’s perspective rather than probabilistic.

- While the generalization of advice through the next 5 time steps rather than using advice for only one time step did not have a significant effect in terms of human factors in this case, it did enable participants to train an agent with less advice. It is possible for generalizing advice through time to decrease the amount of effort and instruction required by the human to train the agent, as well as decrease frustration by enabling people to repeat themselves less. However, as generalizing advice through time was not shown to significantly impact the user experience, we recommend prioritizing the creation of a minimal time delay and deterministic interaction.

## 7.5 Study 4: Conclusion

This chapter delved deeper to understand which aspects of the action advice interaction people liked. Specifically, this chapter focuses on the question: *How do the following three factors of an agent's interaction design affect the person's experience with the agent: 1) generalization through time, 2) time delay, and 3) a probabilistic interface?* The Time Delay and Probabilistic interfaces had a significant, detrimental effect on the human experience. We found the agent that generalized advice through five steps immediately in the future and the agent that had no generalization through time performed the best, both in terms of human factors and objective metrics. While both the Time Delay and Probabilistic agents created a worse experience, the Time Delay had the most negative impact on both the user experience and objective metrics.

In order to create a transparent user interaction with low frustration, high perceived intelligence, and high perceived performance, we recommend creating interfaces that have a deterministic interaction with the person and a minimal time delay between when advice is given to and used by the agent. To decrease training time, train with less advice, train with fewer steps to task completion, and create an 'instant gratification' interaction that is perceived to respond immediately to a human's instruction, it is especially important to minimize the time delay between when advice is provided and used, but it is also

recommended to have a deterministic interface.

If you consider an agent to be ‘liked’ by a person if it scores well on human factors metrics, then the results of this study show that the human-agent team performs worse on algorithms people like the least. The agents that scored the best on the human factors metrics - the 5 Steps and 1 Step (no generalization) agents - also scored the best on objective machine learning metrics.

## 7.6 RQ Results Summary

- Generalization through time did not have a significant impact on the user experience in the RadWorld domain.
  - *Validation:* Study 4.
  - *Supports thesis:* Part 1 (human experience) - While generalizing through 5 time steps allowed people to provide significantly less advice than when no generalization was used, this was not a significant factor for the human experience, as opposed to a time delay or a probabilistic interface. IML algorithm designers should focus on other factors.
- A time delay between when advice is given and used had a significant detrimental impact on the person’s experience with the agent.
  - *Validation:* Study 4.
  - *Supports thesis:* Part 1 (human experience) - IML designers should minimize the time delay between when advice is given by the person and used by the agent.
- A probabilistic interface had a significant detrimental impact on the person’s experience with the agent.
  - *Validation:* Study 4.

- *Supports thesis:* Part 1 (human experience) - IML researchers should create an interaction that is deterministic from the human's perspective rather than probabilistic.



## CHAPTER 8

### CONCLUDING DISCUSSION

Interactive Machine Learning is a relatively new field, and how the nature of the interaction impacts the user's experience is not widely studied. However, algorithms and methods developed in this dissertation bring us closer to the creation of IML algorithms that can be taught by end-users with no specialized training in a natural, intuitive way using verbal instruction.

This dissertation makes the following contributions to the field of Interactive Machine Learning: (1) *design recommendations for IML algorithms* to allow researchers to create algorithms with a positive human-agent interaction; (2) two new *IML algorithms* to foster a pleasant user-experience; (3) a 3-step *design and verification process for IML algorithms using human factors*; and (4) new methods for the *application of NLP tools to IML*.

#### 8.1 Design Recommendations for IML algorithms

IML algorithms have historically been designed to improve objective performance and efficiency metrics, while largely ignoring the human experience. Unfortunately, this leads to algorithms that create a poor experience for the human teacher. To address this, I studied IML algorithms from a human factors perspective to better understand how the interaction and nature of the ML algorithm can affect a person's experience with the agent. My goal is to inform IML designers of the ramifications of their interaction design decisions.

I have identified nine main characteristics in an IML algorithm that impact the human's experience with the agent, including: using human instructions about the future, compliance with input, empowerment, transparency, immediacy, a deterministic interaction, the complexity of the instructions, accuracy of the speech recognition software, and the robust and flexible nature of the interaction algorithm.

A subset of my design recommendations to align with these characteristics include:

- *Learning from Non-Interactive Explanations.* Create agents that allow people to provide explanations without state information. To improve the quality of the person’s explanation, the explanation format should be strictly structured or entirely free-form; a partially-structured explanation format can impose a high cognitive load leading to an explanation of poor quality.
- *Learning from Interactive Advice.* In order to create a transparent user interaction with low frustration, high perceived intelligence, and high perceived performance, we recommend creating interfaces that have a deterministic interaction with the person and a minimal time delay between when advice is given to and used by the agent. To decrease training time, train with less advice, train with fewer steps to task completion, and create an ‘instant gratification’ interaction that is perceived to respond immediately to a human’s instruction, it is especially important to minimize the time delay between when advice is provided and used.

The results of the final study show that the human-agent team performs worse on algorithms people like the least, which reinforces a central focus of my thesis that the human experience is an important aspect of algorithmic design.

## **8.2 IML algorithms**

Because the IML field has not employed user-focused design, the existing IML algorithms do not necessarily create a positive human-agent interaction. To address this, I created two IML algorithms that foster a positive experience for the person. The first algorithm, Object-focused advice, allows an agent to learn from *a priori* explanations without state information. The second algorithm, Newtonian Action Advice, enables a person to provide advice about the future that is immediately and transparently implemented by the agent,

can be easily overwritten in the future to correct mistakes, and creates a deterministic interaction with the person. I isolated and analyzed individual design characteristics of the Newtonian Action Advice agent to determine the extent that each had on the human experience.

### **8.3 Design and Verification Process for IML algorithms using Human Factors**

The IML field evaluates algorithms using oracles (i.e. simulated feedback) that measure objective metrics such as training time and cumulative reward. While this is necessary to test the theoretical efficacy of an algorithm, it turns a blind eye to the human experience. For example, an oracle will never get frustrated with an agent. Similarly, IML algorithms are designed to optimize objective performance and efficiency metrics rather than improve the human-agent interaction. This is a bottleneck that can delay the transition of IML algorithms from academic to commercial applications because if people have a poor experience with the agent, they will not want to use the agent regardless of the efficiency of an algorithm's theoretical learning curves.

To remedy this, I proposed a 3-step design and verification method in which researchers: 1) design for a positive human experience, 2) test the algorithm with oracles to establish theoretical efficiency and behavior, and 3) verify the algorithm with a human-subject experiment measuring the human experience.

Think of control systems. Control systems usually contain a controller and observer. An observer estimates values of the state space like position and velocity so the controller knows where the agent is and what it has to do to reach a goal. If a control system does not observe part of the state space, like velocity, the controller may push the system to move dangerously fast or mathematically diverge to infinity and “blow up”. IML verification that only uses oracle simulations is like a control system that is not fully-observed; the human experience is “blowing up” because it is never observed by researchers, and so algorithms are not designed to correct for the human experience. To correct this, we need to observe the

human experience by performing human-subject experiments that analyze human factors such as frustration. Then, we need to use feedback from the human participants to influence the design of algorithms for a better human experience.

#### **8.4 Application of NLP tools to IML**

Using verbal instruction as an interaction medium is an attractive option because speech is a near-universal interface that non-experts do not need to be trained to use. However, a verbal audio signal is noisy, complex, varies between people, and depends on a time history and context. Researchers have developed tools to analyze speech signals, but much of NLP has yet to be incorporated into IML algorithms. I have developed three methods of applying NLP tools to IML in order to improve the human experience.

1. In order to allow human teachers to speak more naturally and not focus on whether they are providing advice or warnings, I developed a method of using sentiment analysis to filter natural language instructions into advice of ‘what to do’ and warnings of ‘what not to do’. The automatic classification of advice and warnings can also decrease human error.
2. To enable people to provide critique without being restricted to a limited set of words, I created a method of using sentiment analysis to classify natural language critique as positive and negative. This creates a more natural and intuitive interface for the human.
3. To allow the agent to gauge the human’s frustration interactively by analyzing the speech signal rather than collecting subjective responses from questionnaires at the conclusion of a task, I created a supervised learning model that uses prosodic features as an objective metric of frustration. A future area of study would be to determine an efficient method of incorporating the human’s frustration into the IML agent in real time.

## 8.5 Summary

Verbal communication is a rich medium that has been underutilized in teaching Reinforcement Learning algorithms. This dissertation demonstrated that 1) the nature of the language instruction used by a human teacher to train an RL algorithm affects a person's experience and satisfaction of teaching an agent; 2) the information people verbally provide was efficiently communicated to the agent; and 3) the RL agents taught using verbal instruction achieved a reasonable level of performance.

Building on this work, we can continue toward the goal of creating intelligent agents that can easily be taught by individuals with no specialized training using intuitive teaching methods.

# **Appendices**

## APPENDIX A

### STUDY 3 QUESTIONNAIRE - (NAA) ADVICE AGENT

Immediately after training the Newtonian Action Advice agent in Study 3, participants filled out the following questionnaire.

Intelligence: How intelligent do you think the action advice agent is?

Not intelligent 0 1 2 3 4 5 6 7 8 9 10 Very intelligent

Intelligence: Action Advice



Frustration: How insecure, discouraged, irritated, stressed, and annoyed were you when training the action advice agent?

Very Low 0 1 2 3 4 5 6 7 8 9 10 Very High

Frustration: Action Advice



Clarity: Was it clear how the agent used your action advice?

Not clear 0 1 2 3 4 5 6 7 8 9 10 Very clear

Clarity: Action Advice



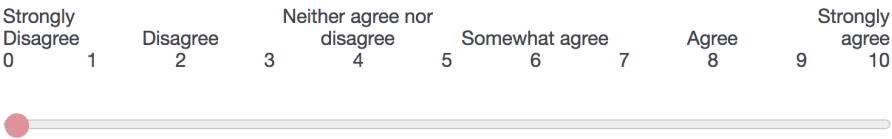
Immediacy: How quickly did the agent respond to your action advice?

Very slowly 0 1 2 3 4 5 6 7 8 9 10 Very quickly

Immediacy: Action Advice



In the end, the agent performed the task as you intended.





**APPENDIX B**

**STUDY 3 QUESTIONNAIRE - (POLICY SHAPING) CRITIQUE AGENT**

Immediately after training the Policy Shaping critique agent in Study 3, participants filled out the following questionnaire.

Intelligence: How intelligent do you think the critique agent is?

Not intelligent                      Very intelligent  
0           1           2           3           4           5           6           7           8           9           10

Intelligence: Critique



Frustration: How insecure, discouraged, irritated, stressed, and annoyed were you when training the critique agent?

Very Low                      Very High  
0           1           2           3           4           5           6           7           8           9           10

Frustration: Critique



Clarity: Was it clear how the agent used your critique?

Not clear                      Very clear  
0           1           2           3           4           5           6           7           8           9           10

Clarity: Critique



Immediacy: How quickly did the agent respond to your critique?

Very slowly                      Very quickly  
0           1           2           3           4           5           6           7           8           9           10

Immediacy: Critique



In the end, the agent performed the task as you intended.

Strongly Disagree				Neither agree nor disagree		Somewhat agree		Agree		Strongly agree
0	1	2	3	4	5	6	7	8	9	10

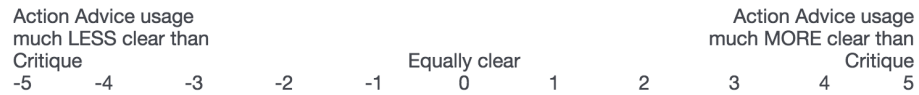
Click to write Choice 1



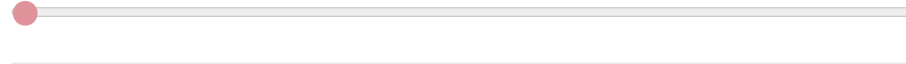
## STUDY 3 QUESTIONNAIRE - AFTER TRAINING BOTH AGENTS

---

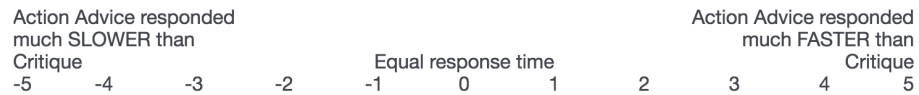
Clarity: Was it more clear how the agent used your action advice or critique?



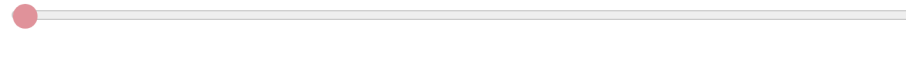
Clarity: Action advice vs. Critique



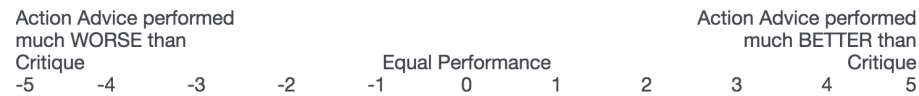
Immediacy: Did the Action Advice agent respond to your input faster than the Critique agent?



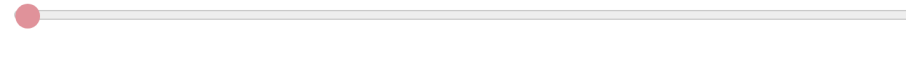
Immediacy: Action advice vs. Critique



In the end, did the Action Advice agent perform the task as you intended better than the Critique agent?



Performance: Action advice vs. Critique



If one agent was more intelligent than another, what made you think that?

If one agent was more frustrating than another, what made you feel that way?

If one agent used your feedback in a clearer way than another, what made you think that?

Is there anything else you'd like to comment on concerning the experience of working with these agents or the experiment in general?

## APPENDIX D

### STUDY 4 QUESTIONNAIRE - (NAA) ADVICE AGENT

Immediately after training each agent in Study 3, participants filled out the following questionnaire.

Intelligence: How intelligent do you think the action advice agent is?

Not intelligent 0 1 2 3 4 5 6 7 8 9 10 Very intelligent

Intelligence: Action Advice



Frustration: How insecure, discouraged, irritated, stressed, and annoyed were you when training the action advice agent?

Very Low 0 1 2 3 4 5 6 7 8 9 10 Very High

Frustration: Action Advice



Clarity: Was it clear how the agent used your action advice?

Not clear 0 1 2 3 4 5 6 7 8 9 10 Very clear

Clarity: Action Advice



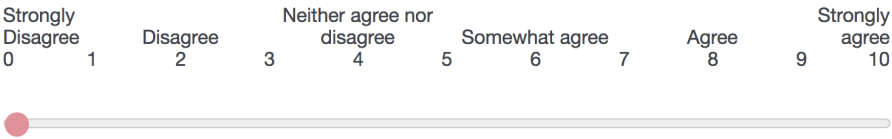
Immediacy: How quickly did the agent respond to your action advice?

Very slowly 0 1 2 3 4 5 6 7 8 9 10 Very quickly

Immediacy: Action Advice



In the end, the agent performed the task as you intended.



---

Is there anything you'd like to comment on concerning the experience of working with this agent?

## APPENDIX E

### STUDY 4 QUESTIONNAIRE - AFTER TRAINING ALL FOUR AGENTS

At the end of the experiment after training all four agents and filling out the associated questionnaires in Study 4, participants filled out the following questionnaire.

If one agent was more intelligent than another, what made you think that?

If one agent was more frustrating than another, what made you feel that way?

If one agent used your feedback in a clearer way than another, what made you think that?

Is there anything else you'd like to comment on concerning the experience of working with these agents or the experiment in general?

## REFERENCES

- [1] S. Krening and K. M. Feigh, “Interaction algorithm effect on human experience with reinforcement learning,” *Transactions on Human-Robot Interaction*, vol. 1, no. 1, 2018.
- [2] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 1. MIT press Cambridge, 1998, vol. 1.
- [3] B. F. Skinner, “The behavior of organisms: An experimental analysis.,” 1938.
- [4] B. F. Skinner, “Selection by consequences,” *Science*, vol. 213, no. 4507, pp. 501–504, 1981.
- [5] C. Diuk, A. Cohen, and M. L. Littman, “An object-oriented representation for efficient reinforcement learning,” in *Proceedings of the 25th international conference on Machine learning*, ACM, 2008, pp. 240–247.
- [6] 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, 2015.
- [7] L. C. Cobo, C. L. Isbell, and A. L. Thomaz, “Object focused q-learning for autonomous agents,” in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 1061–1068.
- [8] R. Dearden, N. Friedman, and S. Russell, “Bayesian q-learning,” in *AAAI/IAAI*, 1998, pp. 761–768.
- [9] H. Sahni, B. Harrison, K. Subramanian, T. Cederborg, C. Isbell, and A. Thomaz, “Policy shaping in domains with multiple optimal policies,” in *Proceedings of the 2016 International Conference on AAMAS*, International Foundation for AAMAS, 2016, pp. 1455–1456.
- [10] K. Subramanian, C. L. Isbell Jr, and A. L. Thomaz, “Exploration from demonstration for interactive reinforcement learning,” in *Proceedings of the 2016 International Conference on AAMAS*, International Foundation for AAMAS, 2016, pp. 447–456.
- [11] S. Chernova and A. L. Thomaz, “Robot learning from human teachers,” *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 8, no. 3, pp. 1–121, 2014.



- [12] C. Isbell, C. R. Shelton, M. Kearns, S. Singh, and P. Stone, “A social reinforcement learning agent,” in *Proceedings of the fifth international conference on Autonomous agents*, ACM, 2001, pp. 377–384.
- [13] S. Griffith, K. Subramanian, J. Scholz, C. Isbell, and A. L. Thomaz, “Policy shaping: Integrating human feedback with reinforcement learning,” in *Advances in Neural Information Processing Systems*, 2013, pp. 2625–2633.
- [14] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, “A survey of robot learning from demonstration,” *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.
- [15] P. Abbeel and A. Y. Ng, “Apprenticeship learning via inverse reinforcement learning,” in *Proceedings of the twenty-first international conference on Machine learning*, ACM, 2004, p. 1.
- [16] C. L. Isbell, M. Kearns, S. Singh, C. R. Shelton, P. Stone, and D. Kormann, “Cobot in lambdamoo: An adaptive social statistics agent,” *Autonomous Agents and Multi-Agent Systems*, vol. 13, no. 3, pp. 327–354, 2006.
- [17] W. B. Knox and P. Stone, “Combining manual feedback with subsequent mdp reward signals for reinforcement learning,” in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1-Volume 1*, International Foundation for Autonomous Agents and Multiagent Systems, 2010, pp. 5–12.
- [18] A. L. Thomaz and C. Breazeal, “Teachable robots: Understanding human teaching behavior to build more effective robot learners,” *Artificial Intelligence*, vol. 172, no. 6-7, pp. 716–737, 2008.
- [19] T. Cederborg, I. Grover, C. L. Isbell, and A. L. Thomaz, “Policy shaping with human teachers,” in *IJCAI*, 2015, pp. 3366–3372.
- [20] 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 *Proceedings of the National Conference on Artificial intelligence*, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, vol. 20, 2005, p. 819.
- [21] G. Kuhlmann, P. Stone, R. Mooney, and J. Shavlik, “Guiding a reinforcement learner with natural language advice: Initial results in robocup soccer,” in *The AAAI-2004 workshop on supervisory control of learning and adaptive systems*, 2004.
- [22] M. Joshi, R. Khobragade, S. Sarda, U. Deshpande, and S. Mohan, “Object-oriented representation and hierarchical reinforcement learning in infinite mario,” in *Tools*

with Artificial Intelligence (ICTAI), 2012 IEEE 24th International Conference on, IEEE, vol. 1, 2012, pp. 1076–1081.

- [23] B. D. Argall, B. Browning, and M. Veloso, “Learning robot motion control with demonstration and advice-operators,” in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, IEEE, 2008, pp. 399–404.
- [24] C. Meriçli, S. D. Klee, J. Paparian, and M. Veloso, “An interactive approach for situated task specification through verbal instructions,” in *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 1069–1076.
- [25] M. S. Sivamurugan and B. Ravindran, “Instructing a reinforcement learner.,” in *FLAIRS Conference*, 2012.
- [26] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. J. Teller, and N. Roy, “Understanding natural language commands for robotic navigation and mobile manipulation.,” in *AAAI*, vol. 1, 2011, p. 2.
- [27] D. Huggins-Daines, M. Kumar, A. Chan, A. W. Black, M. Ravishankar, and A. I. Rudnicky, “Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices,” in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, IEEE, vol. 1, 2006, pp. I–I.
- [28] B. Pang and L. Lee, “Opinion mining and sentiment analysis,” *Foundations and trends in information retrieval*, vol. 2, no. 1-2, pp. 1–135, 2008.
- [29] S. Krening, B. Harrison, K. M. Feigh, C. L. Isbell, M. Riedl, and A. Thomaz, “Learning from explanations using sentiment and advice in rl,” *IEEE Transactions on Cognitive and Developmental Systems*, vol. 9, no. 1, pp. 44–55, 2017.
- [30] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, “The Stanford CoreNLP natural language processing toolkit,” in *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2014, pp. 55–60.
- [31] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, “Recursive deep models for semantic compositionality over a sentiment treebank,” in *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, Citeseer, vol. 1631, 2013, p. 1642.
- [32] B. Pang and L. Lee, “Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales,” in *Proceedings of the 43rd Annual Meeting*

on Association for Computational Linguistics, Association for Computational Linguistics, 2005, pp. 115–124.

- [33] J. Thomason, S. Zhang, R. J. Mooney, and P. Stone, “Learning to interpret natural language commands through human-robot dialog,” in *IJCAI*, 2015, pp. 1923–1929.
- [34] J. F. Cohn, T. S. Kruez, I. Matthews, Y. Yang, M. H. Nguyen, M. T. Padilla, F. Zhou, and F. De la Torre, “Detecting depression from facial actions and vocal prosody,” in *Affective Computing and Intelligent Interaction and Workshops, 2009. ACII 2009. 3rd International Conference on*, IEEE, 2009, pp. 1–7.
- [35] E. S. Kim, D. Leyzberg, K. M. Tsui, and B. Scassellati, “How people talk when teaching a robot,” in *Human-Robot Interaction (HRI), 2009 4th ACM/IEEE International Conference on*, IEEE, 2009, pp. 23–30.
- [36] J. Thomason, H. V. Nguyen, and D. Litman, “Prosodic entrainment and tutoring dialogue success,” in *International Conference on Artificial Intelligence in Education*, Springer, 2013, pp. 750–753.
- [37] P. Boersma *et al.*, “Praat, a system for doing phonetics by computer,” *Glott international*, vol. 5, 2002.
- [38] I. I. for Speech and Hearing. (2010). Frequently asked questions in acoustics of speech and hearing, (visited on 10/25/2018).
- [39] S. Wood and SW-Phonetics. (2018). What are formants? (Visited on 10/25/2018).
- [40] J. Ang, R. Dhillon, A. Krupski, E. Shriberg, and A. Stolcke, “Prosody-based automatic detection of annoyance and frustration in human-computer dialog,” in *Seventh International Conference on Spoken Language Processing*, 2002.
- [41] R. Cowie, E. Douglas-Cowie, N. Tsapatsoulis, G. Votsis, S. Kollias, W. Fellenz, and J. G. Taylor, “Emotion recognition in human-computer interaction,” *IEEE Signal processing magazine*, vol. 18, no. 1, pp. 32–80, 2001.
- [42] D. Ververidis and C. Kotropoulos, “Emotional speech recognition: Resources, features, and methods,” *Speech communication*, vol. 48, no. 9, pp. 1162–1181, 2006.
- [43] S. Krening, B. Harrison, K. M. Feigh, C. Isbell, and A. Thomaz, “Object-focused advice in reinforcement learning,” in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 1447–1448.
- [44] J. Togelius, S. Karakovskiy, and R. Baumgarten, “The 2009 mario ai competition,” in *Evolutionary Computation (CEC), 2010 IEEE Congress on*, IEEE, 2010, pp. 1–8.

- [45] J. Heinrichs, *Thank you for arguing: What aristotle, lincoln, and homer simpson can teach us about the art of persuasion*. Three Rivers Press (CA), 2017.
- [46] C. Bartneck, T. Kanda, O. Mubin, and A. Al Mahmud, “Does the design of a robot influence its animacy and perceived intelligence?” *International Journal of Social Robotics*, vol. 1, no. 2, pp. 195–204, 2009.
- [47] T. Phillips, “Put your money where your mouth is: The effects of southern vs. standard accent on perceptions of speakers,” *Social Sciences*, pp. 53–56, 2010.
- [48] T. Rakić, M. C. Steffens, and A. Mummendey, “Blinded by the accent! the minor role of looks in ethnic categorization,” *Journal of Personality and Social Psychology*, vol. 100, no. 1, p. 16, 2011.
- [49] C. Bartneck, D. Kulić, E. Croft, and S. Zoghbi, “Measurement instruments for the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots,” *International journal of social robotics*, vol. 1, no. 1, pp. 71–81, 2009.
- [50] A. L. Thomaz, G. Hoffman, and C. Breazeal, “Reinforcement learning with human teachers: Understanding how people want to teach robots,” in *Robot and Human Interactive Communication, 2006. ROMAN 2006. The 15th IEEE International Symposium on*, IEEE, 2006, pp. 352–357.
- [51] T. Fong, I. Nourbakhsh, and K. Dautenhahn, “A survey of socially interactive robots,” *Robotics and autonomous systems*, vol. 42, no. 3, pp. 143–166, 2003.
- [52] B. R. Duffy, “Anthropomorphism and the social robot,” *Robotics and autonomous systems*, vol. 42, no. 3, pp. 177–190, 2003.
- [53] C. F. DiSalvo, F. Gemperle, J. Forlizzi, and S. Kiesler, “All robots are not created equal: The design and perception of humanoid robot heads,” in *Proceedings of the 4th conference on Designing interactive systems: Processes, practices, methods, and techniques*, ACM, 2002, pp. 321–326.
- [54] S. Krening, “Newtonian action advice: Integrating human verbal instruction with reinforcement learning,” arXiv, 2018, arXiv:1804.05821.
- [55] G. Davenport, L. E. Holmquist, and M. Thomas, “Fun: A condition of creative research,” *IEEE MultiMedia*, no. 3, pp. 10–15, 1998.
- [56] N. Moraveji and C. Soesanto, “Towards stress-less user interfaces: 10 design heuristics based on the psychophysiology of stress,” in *CHI’12 extended abstracts on Human factors in computing systems*, ACM, 2012, pp. 1643–1648.

- [57] C. Kohrs, N. Angenstein, and A. Brechmann, “Delays in human-computer interaction and their effects on brain activity,” *PloS one*, vol. 11, no. 1, e0146250, 2016.
- [58] B. Shneiderman, “Response time and display rate in human performance with computers,” *ACM Computing Surveys (CSUR)*, vol. 16, no. 3, pp. 265–285, 1984.
- [59] M. T. Stokes, “Time in human-computer interaction: Performance as a function of delay type, delay duration, and task difficulty,” PhD thesis, Texas Tech University, 1990.
- [60] S. Krening and K. M. Feigh, “Characteristics that influence perceived intelligence in ai design,” in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 2018.
- [61] S. G. Hart, “Nasa-task load index (nasa-tlx); 20 years later,” in *Proceedings of the human factors and ergonomics society annual meeting*, Sage Publications Sage CA: Los Angeles, CA, vol. 50, 2006, pp. 904–908.

## **VITA**

Samantha Krening received her B.S. and M.S. degrees in Aerospace Engineering from the University of Colorado at Boulder, where she emphasized in astrodynamics and control. She then worked for NASA's Jet Propulsion Laboratory for three years. During her last two years at JPL, Samantha was a Guidance & Control Engineer on the Attitude and Articulation Control System team for the Cassini spacecraft orbiting Saturn. This dissertation is in support of her Ph.D. in Robotics from the Georgia Institute of Technology, where she has been part of the Cognitive Engineering Center.