ABSTRACT

Title of Dissertation: Enriching Communication
Between Humans and AI Agents

Khanh Nguyen
Doctor of Philosophy, 2022

Dissertation Directed by: Professor Hal Daumé III
Computer Science, University of Maryland

Equipping AI agents with effective, human-compatible communication capabilities is pivotal to enabling them to effectively serve and aid humans. On one hand, agents should understand humans, being able to infer intentions and extract knowledge from language utterances. On the other hand, they should also help humans understand them, conveying (un)certainties and proactively consulting humans when facing difficult situations.

This dissertation presents new training and evaluation frameworks that enrich communication between humans and AI agents. These frameworks improve two capabilities of an agent: (1) the ability to learn through natural communication with humans and (2) the ability to request and interpret information from humans during task execution. Regarding the first capability, I study the possibility and challenges of training agents with noisy human ratings. Providing humans with more expressive tools for teaching agents, I propose a framework that employs descriptive language as the teaching medium. On the second capability, I introduce new benchmarks

that evaluate an agent's ability to exchange information with humans to successfully perform indoor navigation tasks. On these benchmarks, I build agents that are capable of requesting rich, contextually useful information and show that they significantly outperform those without such capability. I conclude the dissertation with discussions on how to develop more sophisticated communication capabilities for agents.

Enriching Communication between Humans and AI Agents

by

Khanh Nguyen

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2022

Advisory Committee:
Dr. Hal Daumé III, Chair/Advisor
Dr. Yonatan Bisk, Committee Member
Dr. Abhinav Shrivastava, Committee Member
Dr. Pratap Tokekar, Committee Member
Dr. Philip Resnik, Dean's representative

Acknowledgments

First and foremost, I would like to express my deep gratitude to my advisor Hal Daumé III. Hal is an inspiring human, both in research and in regular life. During my PhD years, I made countless mistakes and put myself in various troubles. Hal never once showed even the slightest sense of frustration or disappointment on me. He has given me tremendous freedom and support so that I can comfortably pursue the topics that I like. I feel empowered when working with Hal, because he treats me as a colleague researcher and always respects my opinions and decisions. Hal is a living textbook about making life decisions. By simply mirroring the way he speaks, writes, manages professional relationships, and handles situations, I have become a much better researcher and human. Hal does so many little things that only people with the highest standards of ethics would pay attention to. The favorite lesson that I learned from him is to never set deadlines that force people to work over weekends. It may seem like an obvious thing but, for me, it is a big lesson about persistently practicing ethics and compassion.

I dedicate this dissertation to my family. My family upholds a beautiful tradition of making sacrifices to ensure that the next generation can obtain better education than the previous one. My grandparents, who are farmers, worked extremely hard so that my parents could obtain their college degrees. In their turn, my parents have relinquished many things, including the typical joy of regular parents, to grant me the best education opportunities. Being a shy boy raised in a small town in Vietnam, I never imagined that I would one day go out of the town and travel half

way around the Earth to pursue my dream. My father has given me the courage to do that. When I turned 15, he made an audacious decision to send me 180km away from my hometown to study at one of the best high schools in the country. He advised me to study computer science and following that advice has been the best decision of my life. Even until now, my father still deeply cares about my study; he got excited every time I published a new paper or had new citations. My mother is an incredibly intelligent and patient person. The older I get, the more I respect and admire her talents and life accomplishments. My life goal now is to become a person like my mother. I want to thank my younger sister for loving and respecting me even though I am rarely at home to take care of her as an older brother. I am never good at expressing my love to my family. By working hard and contributing to the knowledge of humankind, I hope I can make them proud and happy.

I want to give a warm hug to Huyền, my dear wife. There are no words that can describe her immeasurable love for me. For six years, my wife had endured loneliness in Vietnam to wait for me. When there was a chance for reunion, my wife gave up her career, learned GMAT and programming in a few months, and ventured into a new career path that she had no prior experience on. Now when we are finally together, Huyền devotes a ton of time and effort everyday to taking care of me. She has always supported my academia life and given me a lot of confidence in my research directions. Her love has made me a better man.

I want to send my deep appreciation to my research mentors, Erik Learned-Miller, Brendan O'Connor, Jordan Boyd-Graber, Debadeepta Dey, Bill Dolan, Chris Brockett, Dipendra Misra, and Yonantan Bisk. Erik gave me the first research opportunity and took great care of me during my time at UMass. Brendan led me to my first publication and handed me the very first research lessons. Jordan taught me professional research practices. Dey and Dipendra have the greatest

influence on my research directions. Bill has been extremely kind and supportive in my career. Yonatan has tremendous patience on me and is always willing to hop on a call to give me any kind of career advice that I need.

I owe a debt of gratitude to my childhood teachers, thầy Dũng, thầy Phương, thầy Hùng. They helped me discover my passions and guided me to my first academic achievements.

I also want to thank the friends whom I met during my Microsoft internships, especially Robert Huang, Yuhao Zhang, Robin Jia, Xin Wang, and Ziyu Yao. They have motivated me to work harder and achieve more. Many thanks to the CLIP students, especially my labmates Amr Sharaf, Shi Feng, Chen Zhao, Trista Cao, Anna Sotnikova, Kianté Brantley for their help and for being awesome friends. Amr literally saved my life: he carried me to the hospital when he found out my appendix was burst, and quickly informed my friends and family about my situation.

My high school friends, Phan Minh, Hồng Quang, Hy Hiếu, Thanh Vũ, Như Huân, Nguyên Lực, Hoàng Phong, Toàn Phong have inspired me with their knowledge, talents, and successes. Anh Phan Minh is a loyal and honest friend of mine and has offered me great research and life advice. I also want to thank my Vietnamese friends at UMD: Khôi Phạm, Uyên Huỳnh, Cường Nguyễn, Phụng Trương, Chương Huỳnh, Trí Vũ, Khoa Phạm for wonderful memories at Maryland.

Our CS department is super lucky to have Tom Hurst as the assistant director of graduate education and Jodie Gray as the payroll coordinator. They have been extremely nice to me and has made my graduate life extremely simple and easy.

Last but not least, many thanks to Jesse Thomason for always notifying me about my Bibtex errors :D

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Motivation

Since the inception of Artificial Intelligence (AI), researchers have imagined machines that can communicate like humans. Herbert Televox, the first humanoid robot invented in 1927 by Ron Wensley, was already equipped with the capability of speaking two simple sentences and taking actions based on human sound pitches. Alan Turing, the father of modern computer science, described his famous test of intelligence as an evaluation of an agent's ability to hold natural conversations with humans (Turing, 1950). Later benchmarks and definitions of AI (Chollet, 2019; Johnson et al., 2017; Levesque et al., 2012; Sakaguchi et al., 2020; Winograd, 1971) have consistently placed strong emphasis on matching the ways humans generate and process information.

More than a mere aspiration to mirror humans, equipping AI agents with effective, human-compatible communication capabilities serves a practically important goal: to enable these agents to effectively serve and aid humans (Marge et al., 2022). While current AI agents are showing tremendous potential to uplift the human society in various ways, in order for these agents to be adopted by humans, researchers need to endow them with features that make them helpful and safe for humans. Among many demanded features, the ability to effectively connect and establish mutual understanding with any human is one of the most important. Specifically, to be helpful for human users, an agent must understand what they want, being able to infer intentions and

extract knowledge from their utterances. On the other hand, to safely serve the users, the agent should help them predict and regulate its behaviors, by conveying its (un)certainties to them in a human-intelligible way and proactively consulting them when facing difficult choices.

Many failures of AI in research and real-world conditions have suggested agents equipped with insufficient capabilities of communicating with humans are dangerously brittle (Angwin et al., 2016; Buolamwini and Gebru, 2018; Feng et al., 2018; Hernandez, 2021; Shridhar et al., 2020; Technica, 2016; Wallace et al., 2019b). In these accounts, the power of training and modifying the agent was placed in the hands of a small group of people. The regular users, who were the main beneficiaries and risk-takers, were equipped with very limited mechanisms to harness the agent, broaden its knowledge, and neutralize the risk that it presented. The lack of communication between the agent and the human users led to both sides failing to establish mutual understanding: the agent misinterpreted what the users needed and took inapt actions; the users could not explain and anticipate the agent's behavior, thereby distrusting it.

To make AI agents more helpful and safe for regular users, the current technology needs to be changed, towards empowering human users with more control of AI agents (Amershi et al., 2014). Developing frameworks that support natural, rich, and faithful human-AI communication can accelerate progress towards this goal. Specifically, equipping AI agents with the ability to learn from natural communication with humans would allow them to be programmed by non-expert users, thereby making them more useful for those users. Meanwhile, granting the agents the ability to articulate specific uncertainties and to follow human advice would significantly reduce the risk of them committing costly mistakes.

Enhancing human-AI communication would also address the redistribution of social labor due to replacing humans with AI agents in current workflows. Substituting humans with

fully autonomous agents at a large scale would put a substantial fraction of the human workforce into temporary unemployment, which could eventually lead to social turbulence if these humans would not be quickly migrated to new occupations. Many leading scientists and policy makers agree that a sustainable path for integrating AI into our society is to set human-centered goals, focusing on creating technologies that amplify human efficiency and impact (Pretz, 2021; Riedl, 2019; Shang and Du; Shneiderman, 2020; UNESCO, 2021; Xu, 2019). The success of this strategy is predicated on whether researchers can design AI agents that can coordinate easily and successfully with regular human workers.

1.2 Current State of Human-AI Communication

Despite closing performance gaps quickly with humans in various domains, current AI agents possess limited capabilities of communicating with humans. A common paradigm for developing these agents typically involves *a dataset- or simulator-based training phase* followed by *a full-autonomy evaluation phase*. During the training phase, agents are trained on a large-scale dataset or in a simulator that can generate an infinite amount of data points. During the evaluation phase, the agents' model parameters are locked and they are tested with previously unseen tasks. The agents execute the tasks on their own, until they choose to terminate or a time or computing budget is exceeded. Finally, the agent that achieves the highest success rate is selected for deployment. In this paradigm, an agent almost never interacts directly with a human, except in certain problems where it has to converse with humans as an intrinsic requirement of the task (e.g., a chatbot). Because there are effective ways to optimize the training and evaluation objectives in this paradigm without communicating with humans, the finally selected agent usually lacks effec-

tive, human-compatible communication capabilities, including the ability to learn directly from humans through natural communication, to convey uncertainties in a human-intelligible way, or to request and interpret human advice.

Limitations of Dataset/Simulator-Based Training Frameworks While a plethora of algorithms and model architectures has been proposed, the foundational learning frameworks in AI have largely been unchanged. Since the early days, two learning frameworks, *reinforcement learning* (Sutton and Barto, 2018) and *imitation/supervised learning* (Pomerleau, 1988; Ross et al., 2011), have primarily been carrying the progress of AI. While these frameworks have powered remarkable achievements, their designs are incompatible with the human ways of teaching. Specifically, they allow communication through only highly primitive media: numerical rewards in reinforcement learning and low-level action labels imitation learning. Learning signals expressed in these media can be easily harvested or artificially simulated, allowing exposing the agent an enormously diverse set of data points. While this approach has been effective at injecting foundational knowledge, agents eventually need to be able to directly learn from human users their personal needs and preferences. In that case, communicating through rewards or low-level action labels restricts the users' ability to convey complex, abstract concepts that they can normally effectively articulate via their natural languages. Learning through narrow communication channels can be inefficient and ineffective, as the information received by the learner may contain incomplete, ambiguous signals about the teacher's intentions.

Limitations of Full-Autonomy Evaluation Frameworks Most current agents not only have restricted capability of learning directly from humans, but also are not built with incentives and skills to generate and convey information to humans. A highly essential skill that agents should

acquire is the ability to ask questions to extract relevant knowledge from humans. As the world is constantly changing, any agent would certainly face problems that are beyond its autonomous problem-solving capabilities. An agent that relied solely on its built-in knowledge would not be maximally helpful and even unsafe for serving humans. Facing a problem that it is not prepared for, the agent may perpetrate harmful decisions without warnings. Lacking communication skills, it neglect opportunities to make safer and more effective decisions by simply asking and following instructions given by human bystanders and supervisors. Unfortunately, the full-autonomy evaluation framework, which praises an agent based solely on its the ability to perform tasks on its own, is actively promoting these kinds of high-risk behavior. While improving autonomous performance of an agent is important to enhance human productivity, focusing solely on this metric may misguide the development of AI that can actually benefit humans.

Efforts to Enhance Human-AI Communication. New subfields of AI have been founded for the goal of democratizing AI and enhancing its utility for regular human users. Explainable AI (Emmert-Streib et al., 2020; Samek et al., 2019) extracts human-intelligible signals about the decision making of an agent to make it more transparent and trustworthy to humans. AI security (Amodei et al., 2016; Hendrycks et al., 2021) studies techniques for strengthening this technology against attacks from bad humans. Fair and just AI (Barocas et al., 2019; Doshi-Velez and Kim, 2017; Mehrabi et al., 2021) focuses on detecting and preventing the negative social impacts of AI on humans. Human-in-the-loop AI (Fails and Olsen Jr, 2003; Laird et al., 2017; Wang et al., 2021; Zanzotto, 2019) finds ways to incorporate humans into the development and operation of AI systems. Overall, these efforts have made AI agents significantly safer and more approachable for non-expert users. But despite these progresses, the communication bottlenecks

in the traditional learning and evaluation frameworks remain largely unaddressed. In fact, the currently dominating trend in the field is to continue exploiting the traditional frameworks to their limits, proposing benchmarks that challenge agents' autonomous decisions-making capabilities, and training agents with cheap, simple forms of learning signals to boost performance on those benchmarks (Brown et al., 2020a; Gulcehre et al., 2020; Olson et al., 2017; Rae et al., 2021; Smith et al., 2022; Thoppilan et al., 2022; Wang et al., 2019a). While this trend has produced agents that are extremely successful in their trained domains, those agents would eventually need to be able to be connected with humans to benefit them. Nevertheless, effective, human-compatible communication capabilities that are requisite for accomplishing that goal will not emerge unless the learning and evaluation criteria intentionally seek for those capabilities.

1.3 Challenges in Advancing Human-AI Communication Research

Incorporating interactions with humans into the traditional learning and evaluation frameworks is met with great challenges. The theoretical challenge is to design an end-to-end optimization framework that can accommodate the vast diversity and complexity of human language. For primitive communication media like reward or low-level action label, the learning objective is trivial to determine: maximizing the reward or minimizing the imitation gap. But for human utterances, it is unclear what is the single objective that an agent should optimize for and how to integrate diverse types of language feedback into an agent's model.

The practical challenge is to reduce cost and risk of conducting experiments with real humans. Currently, scaling up experiments with real humans to a large human population and hundreds thousands of episodes requires an immense budget that most academia research groups

cannot afford. It is also an extremely risky to allow many humans to interact directly with agents that are still under development. Hence, careful design and review processes are needed, further increasing the time and cost of performing these experiments.

1.4 Summary of Contributions

My dissertation develops solutions to the fundamental challenges in human-AI communication research. Specifically, I propose new learning and evaluation frameworks that motivate the agent to develop effective, human-compatible communication skills. In addition, I design low-cost and safe experimental procedures to accelerate empirical validation of research progress. I organize my effort into two bodies of work:

1. *Enriching human-AI communication during training*: in this body of work, I take steps towards enabling AI agents to learn via natural communication with humans. I conduct an empirical study on using reinforcement learning to improve machine learning systems with simulated noisy human ratings (Nguyen et al., 2017), and develop a novel learning framework for learning from human descriptive language (Nguyen et al., 2021);

2. *Enriching human-AI communication during task performance*: in this body of work, I focus on teaching AI agents to request and leverage human-generated information to timely improve their decisions while performing tasks. I propose evaluation settings in which agents can request information from humans, and devise algorithms for learning to decide when and what to request (Nguyen and Daumé III, 2019; Nguyen et al., 2019b, 2022).

In my work, I introduce several practical techniques for realistically simulating language-based interactions with humans. Empirically, I have shown that my proposed algorithms can teach agents

to communicate expressively and effectively in simulation and thus can fulfill tasks significantly better than agents that are equipped with more restricted communication capabilities.

1.4.1 Enriching Human-AI Communication during Training

Improving Machine Translation with Human Ratings Reinforcement learning allows a human to train an agent by assigning numerical ratings to its task executions. Compared to imitation or supervised learning, this framework does not require the human teacher to be familiar with the action space of the agent to convey learning signals. By 2016, reinforcement learning had achieved successes in video games that provide ideal conditions for these methods: a small action space and a reward function that is easy to define and compute (the game score). However, application of this framework to more challenging real-world problems remained limited. A potential use case of the framework was to leverage user ratings in online platforms to improve machine translation systems. Nevertheless, text generation problems like machine translation presents a great challenge for RL, as the action space in these problems is often gigantic (equal to the vocabulary), and it may not be sufficient to summarize various qualities of a text through a single numerical score.

I conduct a study on the benefits and limitations of using numerical ratings to improve a neural machine translation system (Nguyen et al., 2017). Emulating real-world scenarios, I generate ratings that are: (1) granular (e.g., given on a 1-to-5 discrete scale), (2) assigned to whole translations, and (3) noisy (simulating variance among humans, and inconsistency and biases within a human). These properties diverged dramatically from the ideal conditions that previous work had tested their RL algorithms on, where rewards were continuous, provided after

every word is predicted, and generated from a deterministic function. I find that performance of a then state-of-the-art RL algorithm degrades substantially when shifting from the ideal to my simulated conditions. My work has accentuated two challenges of applying RL to real-world problems: (1) modeling real-world settings (especially when rewards are from humans) and (2) the limitations of using simple numerical scores to convey complex human intentions. These aspects are often overlooked in RL research that is conducted on toy problems with simple state and action spaces. My work motivates many subsequent studies to further examine and address these challenges (e.g., work done on real human ratings at OpenAI by Stiennon et al. (2020) and EBay by Kreutzer et al. (2018a)).

Interactive Learning from Human Descriptive Language AI agents have been taught by humans primarily using highly primitive communication medium (e.g., rewards in reinforcement learning, low-level actions/labels in imitation/supervised learning). This narrow communication channel has prevented humans from efficiently transferring complex, generalizable knowledge to agents. Using natural language as the communication medium can potentially enhance agent learning efficiency while reducing human teaching effort. However, current approaches for learning from language largely convert language to rewards, inheriting the limitations of reinforcement learning and wasting rich learning signals that language utterances may contain.

I develop ILIAD: Interactive Learning from Activity Description (Nguyen et al., 2021), a learning framework where an agent learns from language descriptions of its activities. As an example, consider a robot that is asked to "*bring a mug*"; when it fails to fulfill this request and brings back a spoon, a human trainer may provide a description of what it did accomplish: "*bring a spoon*". Receiving this description helps the robot ground the meaning of its actions

and better fulfill similar tasks in the future. My work presents the first statistical-learning formulation, theoretical guarantees, and empirical results on learning purely from such feedback. Unlike the reinforcement-learning-reduction approaches, ILIAD incorporates language directly into the agent's learning and decision-making process without converting it to a reward. The framework allows the teacher to use their natural language for teaching, while not requiring the learner to have prior understanding of the language. Thus, it is suitable for non-expert users who may be unfamiliar with how their AI agents operate but want to teach them new tasks or concepts. One of our experiments teaches an agent to translate language instructions into regular expressions. Without knowledge about regular expressions and using only language descriptions, our simulated teacher trains an agent to achieve competitive success rate (88%) with another agent that learned directly from data annotated with ground-truth regular expressions. Meanwhile, an reinforcement learning baseline attains a 0% success rate in this problem. This result shows broadening the communication channel between the teacher and the learner can lead to substantial improvement in terms of both learning efficiency and effectiveness.

1.4.2 Enriching Human-AI Communication during Task Performance

Communication with humans not only improves interpretability of agents, but also enhances their safety and performance during task operations. In many scenarios, agents may be supervised or surrounded by humans that may have tremendous knowledge about the current tasks and environments. For example, while looking for an object in a house, a robot may encounter human residents that have good intuitions about where the object may be. However, the ability to leverage this human knowledge at test time (i.e. once deployed) has been largely neglected in

traditional machine learning settings, as they are primarily concerned about the agent's ability to perform tasks on its own.

I develop agents that can request and interpret human advice in a 3D photo-realistic navigation problem. The vision-language navigation problem, proposed by Anderson et al. (2018b), has attracted attention from researchers in robotics, computer vision, and NLP; however, it does not provide options for the agent to interact with humans at test time. Leveraging the infrastructure of this problem, I propose VNLA (Nguyen et al., 2019b), the first interactive navigation problem in 3D photo-realistic environments. In the follow-up work HANNA (Nguyen and Daumé III, 2019), I incorporate natural language instructions and present an algorithm that effectively solves this harder version. Specifically, to teach the agent to issue a minimal number of queries to humans, I train it to predict whether it will get lost in the future and to only request instructions when the prediction is true. My papers promote a new direction to improve the performance of an agent: enhancing its capabilities of collaborating with humans. This direction is complementary to the traditional directions of augmenting the model architecture and the learning algorithm. VNLA and HANNA have since become well-established interactive navigation tasks, frequently appearing in recent surveys on embodied multimodal learning (Gu et al., 2022; Park and Kim, 2022; Wu et al., 2021).

Taking a step further, in Nguyen et al. (2022), I formulate a general POMDP framework for teaching agents to decide not only when to request help, but also *what* to request. Previous approaches train agents to mimic human-generated questions without forming an understanding of what they intrinsically need. Inspired by how humans communicate to learn about one another's preferences, I train the agent to interact with simulated humans to discover its needs. On simulated navigation tasks, my agent effectively request and leverage human advice, achieving

up to a 7× improvement in task success rate compared to an agent that perform tasks only by itself. My work illustrates the importance of developing cognition through interaction to be able to communicate *faithfully*.

1.4.3 Simulating Language-Based Communication with Humans

Human-in-the-loop experiments are frustratingly expensive and challenging to operate. They are also hard to reproduce. The widely adopted approach of hiring crowd-workers is problematic not only because of the costs, but also because of the difficulty of assembling balanced, diverse representatives of attributes of interest (e.g., race, gender, education background), and the low standards for protecting workers against potential harms (Gray and Suri, 2019). To accelerate research progress and reduce harms to the human subjects, we should conduct human-in-the-loop experiments with agents that have attained a decent level of efficiency and safety. Bootstrapping agents to that level necessitates constructing emulators of humans that support helpful forms of interactions with the agents during their primal development. These emulators can also be used to quickly and safely evaluate the agents during prototyping.

In my papers, I introduce cost-effective experimental strategies for simulating natural-language interactions with humans. By restricting the communication to simple but useful forms of interaction, I show that imperfect simulations of humans can be used to train agents to achieve reasonable success rate in fulfilling task requests expressed in language.

In HANNA, I setup a robot navigation problem where a robot receives a high-level instruction to find an object in a simulated house and must navigate to the location of the object. Every time the robot gets lost and asks a human for directions, the human needs to generate a natural

language instruction that guides it closer to the object. For a perfect simulation of this scenario, the instruction must vary according to the robot's location and the goal location. Theoretically, I show that, by allowing the agent to make at most $O(\log N)$ queries per task, where N is the number of locations in a house, I can reduce the number of instructions needed to be crowd-sourced for the simulation from $O(N^2)$ to $O(N \log N)$. In practice, I reuse of a pre-existing dataset and simulator that were previously used to construct a non-interactive version of the navigation problem. Hence, in the end, I have constructed a highly linguistically realistic simulator with no additional human-labeling cost.

Experiments in the HARI work requires modeling a human that can answer different types of query from an agent. Specifically, the human needs to provide additional information about the agent's current, goal, and subgoal locations. I build a simulated human that provides feature sets of these locations. A feature set emulates information extracted from a human's utterance. This simulation scheme simplifies the language understanding problem and focuses on assessing the agent's capability of selecting which type of information to request. Moreover, it also enables easy control of the amount of information fed to the agent, which is helpful for constructing scenarios where the agent must request additional information to successfully accomplish tasks.

In ILIAD, the challenge is to construct a simulated teacher that generates a natural language description of any execution of an agent. I resolve the challenge by training a execution-conditioned language model using an instruction-following dataset. Given a dataset of instructions and their corresponding ground-truth executions, I train a describer model that predicts the instructions given the executions. The model is used as a teacher that is capable of generating language descriptions of agent executions. As data for training the model is scarce, the model by itself often generates imprecise or unnatural descriptions. I propose two methods for improving

the quality of the descriptions: (1) generate the descriptions pragmatically, taking into consider how it would be interpreted by the learning agent and (2) combining the language model with the instruction-following dataset, returning an instruction in the dataset as the description when the agent's execution coincides with the corresponding annotated execution.

1.5 Roadmap

The rest of the dissertation is structured as follows. I will first review traditional learning and evaluation frameworks (§ 2). Rather than presenting these frameworks from a conventional machine learning point of view, I will focus on highlighting the strengths and weaknesses of the human-agent communication models that these frameworks implement. The next two chapters (§ 3 and § 4) describe the two main bodies of my work. In the concluding chapter (§ 5), I will discuss major challenges for deploying and extending my proposed work, and my future directions to tackle those challenges.

Chapter 2: Background

The frameworks presented in this dissertation are inspired by socio-cognitive science studies that characterize the evolution of communication in the natural world (Scott-Phillips, 2014; Sperber and Wilson, 1986; Tomasello, 2005). In this chapter, I will make an attempt to distill knowledge drawn from these studies into practical principles for designing training and evaluation frameworks for AI agents.

I will first introduce a taxonomy of communication systems employed by living species in the natural world (§2.1). I will then apply this taxonomy to classify and evaluate the strengths and weaknesses of major learning frameworks (§2.2). Particularly, I will illustrate that reinforcement learning and imitation learning, the two most popular learning frameworks in AI, implement only the most primitive form of communication in the nature and thus inherit its shortcomings. The communication-system taxonomy hints at directions to augment these frameworks: that is to emulate the characteristics of more advanced natural communication systems. I will review new developments that follow this principle.

In section §2.3, I discuss how humans can provide information to enhance task performance of an agent without updating its policy. I will highlight advantages of agents that can leverage human assistance through communication over agents that lack such capability and can rely only on its own knowledge.

The last section (§ 2.4) describes a simulator that allows studying human-agent communication in visually high-fidelity environments. Many of the experiments presented in the later chapters were conducted with this simulator.

2.1 Natural Communication Systems

The natural phenomenon of communication concerns a series of exchanges of informative signals between two interlocutors. In each exchange, one interlocutor acts a *speaker*, who generates and sends signals, while the other acts as a *listener*, who receives and interprets signals. The interlocutors may swap their roles after each turn.

Communication is ubiquitous in the nature world. For any living being, whether it is a bacteria or a human, communication is the foundation of the most important life activities: learning, coordinating, attracting partners, etc. It is not a coincidence that evolution has driven living species into developing the ability to communicate. Being able to efficiently transport information across individual minds offers a significant evolutionary advantage. Thanks to this capability, members of a group are able to operate collectively as a single entity with significantly enhanced intellectual and physical power, increasing their chances of surviving and thriving in the harsh and competitive wildlife.

While each species implements a distinct system of communication that has evolved to adapt to specific environment conditions and survival strategies, more complex forms of life tend to implement more efficient communication systems. The human communication system is widely regarded as the pinnacle of all systems.

There are two important developments in the evolution of natural communication systems.

The first development is the invention of *referential communication signals* that fulfill the need of communicating about concepts or entities in the outside world (San Martín et al., 2014). This progression significantly enlarges the scope of conversations, enabling the interlocutors to perform joint activities and share knowledge about the world. The second development is the emergence of *inferential cognitive capabilities* that allow the interlocutors to leverage their knowledge about the world to extrapolate the meanings of their communication signals beyond conventional meanings (Scott-Phillips, 2014; Sperber and Wilson, 1986; Tomasello et al., 2005).

The two major developments divide natural communication systems into three classes: *dyadic*, *referential*, and *inferential*. Dyadic communication refers to communication where interlocutors only circulate information about themselves. Referential communication empowers the interlocutors with the ability to discuss third-party concepts. These two classes can be collectively referred to as *associative* communication, because the production and comprehension of communication signals are based on conventional associations of signals with meanings. Inferential communication replaces these straightforward mappings with complex reasoning processes that take into account the context in the assignment of meanings.

From Dyadic to Referential Communication Dyadic communication features conversations in which the interlocutors exchange information only about themselves and not about concepts or entities related to the external world. In these conversations, an interlocutor sends signals to stimulate or suppress certain behaviors of the other. Dyadic communication is commonly observed in early forms of life. For instance, a bacterium species named P. Aeruginosa communicates by a process called quorum sensing, where an individual releases into the external environment chemicals that can activate specific genes in other individuals when a certain level of concentration of

these chemicals is reached. Dyadic communication is also the earliest form of communication that humans develop (Beebe et al., 2016). Soon after birth, a human infant involves in face-to-face interactions with their parents and forms reactions to stimuli from them (e.g. the infant attends to and starts imitating the parents' facial expressions) (De Schuymer et al., 2011).

In referential communication, conversations are triadic, as the interlocutors are able to devise signals that are grounded to concepts or entities in the outside world. Discussions about external subjects are the building blocks of social collaboration and education. Between the first to the second year of age, humans develop referential communication skills (Carpenter et al., 1998; Matthews et al., 2012; Striano et al., 2009). At this stage, human babies establish joint attention to objects with their parents, and create signals to reference to those objects (Mundy et al., 2009). Besides humans, evidence of referential communication in other high-level forms of life have been documented. Some species of birds and primates devise different vocal sounds to warn fellows against predators (Arnold and Zuberbühler, 2006; Leavens et al., 2004; Suzuki, 2016). Cangelosi (2001) explains the main difference between the referential signals constructed by humans and those created by other animals. Human referential signals have *double references*: a word is related not only to an external concept but also to other words. Referential signals of other animals do not posses non-trivial syntactic and compositional relationships that power efficient production of new combinations.

From Associative to Inferential Communication Most non-human communication systems are associative: they rely on conventional or inborn direct mappings from signals to meanings. In other words, associative communicators do not recognize the existence of a *mind*, i.e. an non-trivial internal process that generates and interprets communication signals. They exchange

signals only to influence others' external behaviors.

The aforementioned quorum-sensing chemicals or the enemy-warning sounds are examples of associative communication signals. It is important to understand that while these signals may be linked to or activated by a world state or context, their meanings are not contextual. Specifically, the meanings of these signals have the form "*If the world is X, then do Y*". This meaning remains the same under different world states; the world state X is a part of the meaning rather than a factor that controls it.

Inferential communication allows the interlocutors to enrich their communication signals with new layers of meaning that can even be remotely related to the conventionally assigned meanings. These new layers of meaning are not indicated solely by the form of the signals but are *inferred* from the form based on a context. Consider the following conversation:

> Student: *Would you be free for a brief video call on Thursday 3PM?*
>
> Professor: *I'll be presenting a talk about human communication.*
>
> Student: *How about Friday 2PM?*
>
> Professor: *Sounds good to me!*

At a first glance, the interlocutors seem to be at cross purposes: the student sought for a confirmation (or denial) of a suggestion but the professor replied by describing what she would be doing. However, both were in fact communicating perfectly thanks to their abilities to infer the intention of the other. The professor recognized the student's intention of reserving a meeting, derived a intention of denying the suggested time, and conveyed it in a way that she believed he could recognize. As expected, the student realized the denial intention instantly and suggested

another time.

To illustrate that the meaning of the professor's first utterance (red) is contextual, we can try placing it in a new context:

> Student: *Are you a speaker at this upcoming workshop?*
>
> Professor: *I'll be presenting a talk about human communication.*
>
> Student: *Great, see you then!*

While the form of the utterance is unaltered, the new context has inverted its meaning: the utterance now signifies a confirmation rather than a denial.

In these examples, the contextual meanings of the utterance cannot be obtained by simply looking up and combining the denotations of the constituent words. These meanings seem to appear out of thin air, as if the interlocutors are capable of "reading each other's mind". This is in fact true to some degree: while inferential communicators do not have psychic mind-scanning power, they do possess certain cognitive skills that allow them to make accurate predictions about what others are thinking. For example, in the first conversation, the professor's utterance was chosen to convey her denial intention because she (accurately) postulated that the student would understand that a person would not be able to simultaneously attend a meeting and give a talk. An important pre-condition for inferential communication is the development of a mind that implements a collection of *non-trivial* algorithms to generate and update mental states given input communication signals. These mental states ultimately engender external behaviors. This starkly contrasts with associative communication, in which input signals directly drive external behaviors. Another precursor to inferential communication is the acknowledgement of an interlocu-

tor that the other interlocutor also possess a (non-trivial) mind. This assumption fundamentally changes the goal of communication: realizing that external behaviors are only manifestation of mental states, inferential communicators aim at manipulating others' mental states rather than trying to regulate their external behaviors.

How do inferential communicators manipulate others' minds without being able to directly observe them? Using innate capabilities and experience, inferential communicators can construct hypothetical models about others' minds to make predictions that inform their communicative decisions. These models are called *theories of mind* (Premack and Woodruff, 1978). While theory of mind has been largely attributed to humans, the original proposal of this concept illustrated it through experiments on chimpanzees. Recent work concludes that great apes possess theories of mind, but most evidence has suggested that their theories are not as complex as those of humans (Call and Tomasello, 2008). Theory of mind are demonstrated in humans usually through false-belief tests (Baron-Cohen et al., 1985; Wimmer and Perner, 1983), which evaluates whether an individual understands that others may have incorrect assumptions about the world. Research conducted by Gopnik and Astington (1988) shows that children develop theories of mind around the age of four to five.

The process of using theory-of-mind models to determine which communication signal to express an intention or which intention to extract from a signal is referred to as *pragmatic reasoning*. Various theories on how this process occurs have been proposed. Sperber and Wilson (1986) develop relevance theory, suggesting that pragmatic reasoning optimizes for relevance to the listener given the speaker's preferences and limited effort. Goodman and Frank (2016) present a Bayesian inference framework where pragmatic reasoning is conducted as recursive process with literal (associative) reasoning serving as the basis.

We can draw an analogy between natural communication mechanisms and variables in a programming language like Python. A communication signal m sent by a speaker can be seen as a variable. Dyadic communication corresponds to only being able to define m as one of the primitive types like `int`, `bool`, or `float`. In this case, each variable is associated with a static value and carries only a fixed number of bits of information. Referential communication is analogous to being able to define m as a pointer that references to a class object. While the pointer is made up of a small number of bits, the referenced object may convey much more information. Finally, inferential communication resembles implementing a function Interpret(m, c) to compute the value of the variable, where c is a context. Referential communication can be seen as a special case when the function is implemented as a simple look-up table. But in general, the function can execute an arbitrarily complex algorithm, allowing the message to interplay with the context in intricate ways to derive rich contextual meanings.

2.2 Human-AI Communication during Training

Learning from humans has been traditionally framed as an optimization problem, in which the goal is to find a learner's behavior policy $\hat{\pi}$ that minimizes a loss function $\mathcal{L}$ under a data distribution $\mathcal{D}$

$$\min_{\hat{\pi}} \mathcal{L}(\hat{\pi}) = \min_{\hat{\pi}} \mathbb{E}_{s\sim\mathcal{D}}[L(s, \hat{\pi}(s))] \tag{2.1}$$

This optimization view has largely shaped the focus of research. Ideas have revolved around improving the elements of the optimization problem: the policy model ($\hat{\pi}$), the optimizer ($\min_\pi$), the loss function (L), and the data ($\mathcal{D}$). While the optimization formulation serves as an the

foundation of machine learning, it does not reflect many key aspects of learning as a natural phenomenon. Nevertheless, there are important lessons that can be drawn from the evolution of natural communication and applied to developing more effective learning algorithms.

This section serves as a review of major frameworks for learning from humans. But rather than presenting them in the standard optimization view, I will focus on discussing the properties of the underlying teacher-learner communication models that these frameworks employ. Understanding a framework's underlying communication model is crucial in understanding its strengths and drawbacks. I will classify the learning frameworks into the three classes of natural communication systems introduced in the previous section. This taxonomy suggests two important aspects of learning that are not prominent in the optimization view: the characteristics of the communication signals, and the cognitive capabilities of the teacher and the learner. I will show that many successful augmentations of traditional learning frameworks can be considered as efforts to improve these aspects.

2.2.1 A General Protocol for Learning

I present a general communication protocol that serves as the skeleton for constructing learning algorithms. Most machine learning frameworks are instantiations of this protocol, varying only the communication medium used to express the teacher's feedback and the function for generating communication signals (the agent's policy and the teacher's feedback-generating function).

I view learning as a communication process between a teacher and a learner. This view implies that the roles of the teacher and the learner are equal: they both generate and interpret

communication signals from the other. They diverge only in the formats and purposes of their communication signals.

I denote by $d \in \mathcal{D}$ a communication signal sent by the teacher and by $e \in \mathcal{E}$ a message sent by the learner. A teacher's signal can be a request for execution of a task, a feedback on an execution, or any other type of information. I will use $d^\star$ to denote a task request from the teacher. I restrict the form of the learner's signals to be an *execution* of a task request. An execution consists of a sequence of contexts and actions, i.e. $e = (c_0, a_1, \cdots, c_{H-1}, a_H, c_H)$, where $c_t \in \mathcal{C}$ is an context for making decisions, a_t is an *action* belonging to the learner's action space $\mathcal{A}$, and H is the time limit for performing the task. In general, the learner can generate other communicative signals (e.g., asking questions, giving explanations or descriptions of its actions), but most learning frameworks do not model this capability. In section X, I will introduce a framework that allows the learner to send help-request signals to humans, but it operates in a non-learning setting.

To make decisions, the learner maintains a policy $\hat{\pi} : \mathcal{C} \times \mathcal{S} \to \Delta(\mathcal{A})$, where $\hat{\pi}(a \mid d^\star, c)$ is the probability of choosing action a given a task request $d^\star$ and current context c. Let $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ be a context-transition function, where $T(c' \mid c, a)$ is the probability of observing context c' if taking action a in context c. The learner generates an execution $\hat{e}$ as follows

$$\hat{e} = (c_0, a_1, \cdots, c_{H-1}, a_H, c_H) \tag{2.2}$$

$$a_t \sim \hat{\pi}(\cdot \mid c_{t-1}, d^\star) \tag{2.3}$$

$$c_t \sim T(\cdot \mid c_{t-1}, a_t) \quad \text{for } 1 \leq t \leq H \tag{2.4}$$

I denote by $P^H_{\hat{\pi}}(\hat{e} \mid d^\star, c_0)$ the probability of the learner generating an execution $\hat{e}$ for a task $d^\star$

starting from context c_0.

Learning occurs in multiple episodes, each of which operates as follows:

Algorithm 1 A general protocol for learning.

1: The learner observes an initial context c_0 and receives a task request $d^\star$ from the teacher. I assume that $c_0, d^\star$ are sampled from a task distribution $\mathfrak{T}(c_0, d^\star)$ defined by the teacher;
2: The learner generates its communication signal, an execution $\hat{e} \sim P^H_{\hat{\pi}}(\cdot \mid d^\star, c_0)$ of the request, and sends it to the teacher;
3: The teacher interprets the learner's execution, and generates its communication signal, a feedback $\hat{d}$, and sends it to the learner;
4: The learner interprets the feedback $\hat{d}$ to update its policy $\hat{\pi}$ accordingly.

Learning frameworks mainly differ in two aspects of this protocol:

1. The medium and format of the feedback $\hat{d}$;

2. The cognitive capabilities of the teacher and the learner, specifically whether they communicate in an associative or inferential manner.

2.2.2 Dyadic Learning Frameworks

Learning frameworks that bear a resemblance to dyadic communication employ low-bandwidth, non-referential signals as the teacher's feedback. These signals are issued to regulate the learner's behavior. They are interpreted by the learner via fixed rules that are hard-coded into the learning algorithm. (Policy-gradient) reinforcement learning (Haarnoja et al., 2018; Konda and Tsitsiklis, 2000; Sutton et al., 1999a) and imitation learning (Daumé et al., 2009; Pomerleau, 1988; Ross et al., 2011) manifest these characteristics.

In *reinforcement learning* (RL), the teacher's feedback has a reward function $r(c, a)$ that measures the "goodness" of action a in context c. The feedback given on a learner execution $\hat{e}$ is a sequence of scalar rewards $(r_0, r_1, \cdots, r_K)$, where each reward evaluates an action in

the execution. The number of rewards K can vary from 1 to H depending on the setting. Let $R(\hat{e}, d^\star) = \sum_{k=1}^{K} r_k$ be the total reward attained by execution $\hat{e}$ on task $d^\star$. The objective of the learner is to maximize the expected reward of its executions under a task distribution

$$\min_{\hat{\pi}} \mathcal{L}_{\text{RL}}(\hat{\pi}) = \max_{\hat{\pi}} \mathbb{E}_{(c_0, d^\star)\sim\mathfrak{T}, \hat{e}\sim P^H_{\hat{\pi}}(\cdot|c_0,d^\star)}[R(\hat{e}, d^\star)] \tag{2.5}$$

Policy-gradient RL treats rewards as associative communication signals that modulate the learner's policy. These algorithms apply the policy gradient theorem (Sutton et al., 1999a) to compute the gradient of the learning objective with respect to the learner's policy:

$$\nabla_{\hat{\pi}} \mathcal{L}_{\text{RL}}(\hat{\pi}) = \mathbb{E}_{(c_0, d^\star)\sim\mathfrak{T}, \hat{e}\sim P^H_{\pi}(\cdot|c_0,d^\star)} \left[R(\hat{e}, d^\star) \nabla_{\hat{\pi}} \log P^H_{\hat{\pi}}(\hat{e} \mid c_0, d^\star) \right] \tag{2.6}$$

and use this gradient to adjust the learner's policy via a gradient-based learning method (e.g., Adam (Kingma and Ba, 2015b)). The policy gradient theorem specifies how rewards are interpreted by policy-gradient algorithms. As seen in the above equation, the sign of the reward $R(\hat{e}, d^\star)$ dictates whether the gradient update increases or decreases the probability that the learner generates the current execution $\hat{e}$. In other words, the reward either encourages or suppresses the current behavior of the learner. Policy-gradient RL algorithms would perpetually associate the reward with this meaning, even if the teacher encoded other (even contextual or referential) meanings into the reward.

In *imitation learning* (IL), the teacher's feedback is an action demonstration $\hat{d} = e^\star \in \mathcal{E}$, which lies in the same space as that of the agent's execution $\hat{e}$. Imitation learning algorithms leverage two main types of demonstration. The first one is reference demonstration, which directly

illustrates how a task should be executed

$$e^\star := e^{\text{ref}} = (c_0^{\text{ref}}, a_1^{\text{ref}}, \cdots, a_H^{\text{ref}}, c_H^{\text{ref}}) \tag{2.7}$$

$$a_t^{\text{ref}} \sim \pi^\star(\cdot \mid c_{t-1}^{\text{ref}}, d^\star), \quad c_t^{\text{ref}} \sim T(\cdot \mid c_{t-1}^{\text{ref}}, a_t^\star), \quad \text{for } 1 \le t \le H \tag{2.8}$$

$$c_0^{\text{ref}} = c_0 \tag{2.9}$$

Here, $\pi^\star$ is the teacher's policy which the learner should imitate. Reference demonstration is the basis of offline IL algorithms like behavior cloning (Pomerleau, 1988), which trains the learner to minimize imitation loss with respect to reference demonstrations

$$\min_{\hat{\pi}} \mathcal{L}_{\text{BC}}(\hat{\pi}) = \min_{\hat{\pi}} \mathbb{E}_{(c_0, d^\star) \sim \mathfrak{T}, \hat{e} \sim P_{\pi^\star}^H(\cdot \mid c_0, d^\star)} \left[\sum_{t=1}^{H} L(\hat{p}_t, a_t^{\text{ref}}) \right] \tag{2.10}$$

where $\hat{p}_t = \hat{\pi}(\cdot \mid d^\star, c_{t-1}^\star)$, a_t^{ref} is the t-th action in the reference demonstration e^{ref} of the task $d^\star$, and L measures the discrepancy between $\hat{p}_t$ and the one-hot distribution peaked at a_t^{ref} (e.g., the cross entropy between these two distributions).

The other type of demonstration is corrective demonstration, which the teacher issues to rectify the actions that the learner have taken. This type of feedback is employed by online IL algorithms like DAgger (Ross et al., 2011) or SEARN (Daumé et al., 2009). Let $(\hat{c}_0, \cdots, \hat{c}_H)$ be the contexts in the agent's execution $\hat{e}$. A corrective demonstration features actions taken by the

reference policy in the contexts that the learner encountered

$$e^\star := e^{\text{crt}} = (c_0^{\text{crt}}, a_1^{\text{crt}}, \cdots, a_H^{\text{crt}}, c_H^{\text{crt}}) \tag{2.11}$$

$$a_t^{\text{crt}} \sim \pi^\star(\cdot \mid d, c_{t-1}^{\text{crt}}), \quad c_t^{\text{crt}} = \hat{c}_0, \quad \text{for } 1 \leq t \leq H \tag{2.12}$$

$$\hat{c}_0 = c_0 \tag{2.13}$$

In contrast to behavior cloning, DAgger minimizes the teacher-learner imitation gap on the execution distribution induced by the learner's policy rather than by the reference policy

$$\min_{\hat{\pi}} \mathcal{L}_{\text{DAgger}}(\hat{\pi}) = \min_{\hat{\pi}} \mathbb{E}_{(c_0,d^\star)\sim\mathfrak{T},e\sim P_{\hat{\pi}}^H(\cdot|c_0,d^\star)} \left[\sum_{t=1}^{H} L(\hat{p}_t, a_t^{\text{crt}}) \right] \tag{2.14}$$

In a non-sequential setting (i.e. $|e^\star| = 1$), there is no distinction between the two types of demonstration. This special case is referred to as the *supervised learning* setting, although outside the IL literature, the term may be used interchangeably with behavior cloning.

Similar to rewards, action demonstrations can also be classified as associative communication signals. A demonstration aims at forcing a certain behavior from the learner. In addition, its meaning is conventional to the learner, because it contains only actions that are in the learner's action space. A distinction between demonstrations and rewards is that providing demonstrations requires the teacher to be familiar with the learner's action space. In other words, the IL teacher must master the learner's "native language" to communicate with it. Because of this trait, IL can be expensive to deploy with humans when the cost of training human teachers to provide demonstrations is high (e.g. training a person control a robot or annotate a parse tree). On the other hand, RL uses numbers as the communication medium, which are arguably natural for the

teacher to provide. Hence, the cost of establishing the communication channel in RL is generally cheaper than in IL. However, there is no "free lunch": since rewards convey less direct learning signals than demonstrations, learning from rewards theoretically requires more interactions with the teacher than learning from demonstrations (Sun et al., 2017).

Reinforcement learning and imitation learning are the main engines behind the many feats of AI: mastering Atari games Mnih et al. (2013), Go (Silver et al., 2016, 2017), Dota 2 (Berner et al., 2019), inspiring variants of generative adversarial networks (GANs) (Creswell et al., 2018; Goodfellow et al., 2014), enabling large-scale dialog systems (Adiwardana et al., 2020; Gao et al., 2018; Ponnusamy et al., 2020), just to name a few. Despite these successes, as I have demonstrated, these frameworks employ a primitive associative communication system, limiting the learner to processing feedback expressed in low-bandwidth media. It is thus difficult and unnatural for humans, who communicate in an inferential manner, to employ these frameworks for teaching AI agents. While various limitations of IL and RL algorithms have been theoretically and empirically analyzed (Choshen et al., 2020; Dulac-Arnold et al., 2019; Kreutzer et al., 2020; Kurenkov, 2018; Ng et al., 1999; Osa et al., 2018; Rajaraman et al., 2020; Ross, 2013; Sun et al., 2017; Sutton and Barto, 2018), addressing the teacher-learner communication bottleneck requires going beyond these frameworks towards those that implement richer, more human-like forms of communication.

2.2.3 Referential Learning Frameworks

Humans have tailored their languages for communication about external concepts. Learning frameworks that employ human language as the communication medium is desirable for two

main reasons. First, they allow most humans to conveniently train AI agents, especially non-expert users who are not proficient at machine learning technologies. Second, the richness and generalizability of human languages may potentially enable agents to learn with fewer data points or interactions with humans while generalizing more robustly. Nevertheless, teaching agents using free-form human language remains a magnificent challenge in AI, due to the complexity and variety of human language, and the obscurity of the mechanisms that humans have developed for language generation and interpretation.

To formulate solvable problems, researchers have concentrated on specific scenarios and restricted types of language. A majority of language-based learning frameworks focuses on *instructive teaching*, where the teacher describes (or clarifies) the behavior they desire the learner to exhibit. In this setting, an instruction $d^\star$ is given before the learner executes a task. It could be a high-level task specification or a low-level description of a procedure that completes the task. The instruction aims to guide the learner to execute the task. An approach to learning from instructions is to compute an instruction-conditioned policy that directly translates an instruction into a sequence of actions. Estimating the instruction-conditioned policy usually requires imitation learning on a labeled dataset containing pairs of task descriptions and demonstrations. Knowledge of the instruction-conditioned policy is later distilled to an instruction-free policy, allowing the learner to accomplish tasks without guidance from the teacher at test time (Li et al., 2020c,d). Another way to leverage a demonstration dataset is to learn an execution-scoring function $R(\hat{e}, d^\star)$ that measures how good an execution $\hat{e}$ is at completing a task specified by $d^\star$. The learner can then plug this function into any RL algorithm as a reward function to estimate an instruction-free execution policy (Fu et al., 2019; Ghosh and Srivastava, 2021; Goyal et al., 2019; MacGlashan et al., 2015). This can also be viewed as inferential learning, which will be discussed

in the next section.

An interesting variant of instructive teaching is when the task's main scoring function $R(\hat{e}, d^\star)$ is given or can be estimated from demonstration data, but the teacher provide additional instructions that can be translated into supplementary scoring functions $R_i^+(e, d_i^\star)$, which are consistent with and elaborate the main scoring function (e.g., constraints, labeling rules). The supplementary functions can be used to noisily label new data that will be utilized to further fine-tune the learner's policy (Hancock et al., 2018). Alternatively, the outputs of the functions can be incorporated as a latent input of the learner's policy. Yang et al. (2021) employs an exploration policy to generate labels for learning the functions. The functions can also be inferred using an EM algorithm (if demonstration data are given) (Matuszek et al., 2010; Srivastava et al., 2017) or RL algorithm (if the main scoring function is given) (Artzi and Zettlemoyer, 2013; Clarke et al., 2010; Goldwasser and Roth, 2014), without requiring ground-truth labels.

An instruction can also be given after the learner has completed an execution, as a feedback $\hat{d}$ that suggests modifications to the execution. This setting is analogous to online IL, except that the feedback is expressed in language. Approaches mentioned in the pre-execution setting can also be deployed in this setting, reducing the learning problem to either imitation learning (Labutov et al., 2018; Wang et al., 2016b) or reinforcement learning (Fidler et al., 2017).

Apart from designating the target behavior, a post-execution feedback may also describe the behavior of the agent. I refer to settings that employ this type of feedback *descriptive teaching*. From an inferential-communication viewpoint, descriptive feedback diverges from instructive feedback in that the teacher, rather than expressing their own intentions, infers and conveys the learner's intentions. Rewards can be considered as a form of descriptive feedback. In RL, the teacher's intention is to have the learner achieve the maximum reward possible in doing a task. A

reward given at the end of an episode reveals the intention that the teacher thinks the the learner has accomplished (e.g., scoring 90 out of 100 on an exam). The learner's goal to behave so that their inferred intention aligns with the teacher's intention. In chapter §3, I will present a formal learning framework that generalizes this learning principle and is able to process descriptive feedback expressed in human language.

Attempts to learn from diverse types of language feedback has been made. Sumers et al. (2020) present a framework that can incorporate three different types of language feedback. Their approach models the teacher's reward function as a function of features extracted from a feedback utterance. Each type of feedback activates a distinct set of features. Weston (2016) introduce the problem of learning from dialogs. Besides extracting conventional learning signals such as rewards and demonstrations, the work also proposes estimating the state transition function conditioned on language utterances. Narasimhan et al. (2018) generalize in this idea to general MDP environments. Elgohary et al. (2020) collect a dataset where humans provide natural language feedback to predictions of SQL-to-text parser. The feedback contains both instructive and descriptive learning signals. The authors train a correction model via supervised learning to predict the ground-truth parses based on the feedback. They observe that incorporating the mispredicted parse as an input of the correction model boosts its performance, possibly because the presence of the parse enables the model to learn the meanings of the descriptive signals in the feedback.

2.2.4 Inferential learning Frameworks

In an inferential learning framework, the learner or the teacher constructs a hypothetical model to explain or predict the behavior of the other. Such a "theory of mind" can be considered as a form of inductive bias that can potentially accelerate knowledge transfer.

Inverse reinforcement learning (IRL; Ng et al. (2000)) is an inferential learning framework that is concerned about reconstructing the latent reward function that the teacher optimizes. This framework operates in a setting that is similar to behavior cloning, where the feedback given to the learner is a reference demonstration e^{ref}. However, instead of directly learning an execution policy, the learner estimates the latent reward function under which the reference demonstration is optimal. Concretely, Let $R(e; \theta(d^\star))$ be a reward function parameterized by a function $\theta : \mathcal{D} \to \mathbb{R}^m$ that maps a task request to a set of parameters (m is the number of parameters). In its simplest form, the learning objective of IRL is:

$$\max_{\theta} \mathbb{E}_{(c_0,d^\star)\sim\mathfrak{T},e^\star\sim P_{\pi^\star}(\cdot|c_0,d^\star)} \left[R(e^\star; \theta(d^\star))\right] \tag{2.15}$$

This objective differs from the RL's objective in two ways: (i) the searched variable is the parameters of the reward function R, not the agent's policy $\hat{\pi}$, (ii) the goal is to maximize the reward of the reference demonstration $e^\star$, not that of the agent's execution $\hat{e}$. This learning problem can be solved via feature matching (Abbeel and Ng, 2004; Syed and Schapire, 2007; Ziebart et al., 2008), max-margin optimization (Kolter et al., 2007; Ratliff et al., 2006), or Bayesian inference (Neu and Szepesvári, 2012; Ramachandran and Amir, 2007). After the reward function is determined, it can be plugged into any RL algorithm to derive an execution policy.

What are the advantages and disadvantages of IRL over behavior cloning? IRL assumes an internal structure of the teacher's communicative decision-making process. For each task $d^\star$, the corresponding reward function's parameters $\theta(d)$ can be viewed as a mental state that engenders the teacher's external behavior (the task demonstration $e^\star$). IRL postulates that maximizing the reward function R_θ is the mechanism by which the mental state causes the teacher to generate the demonstration

$$e^\star = \operatorname*{argmax}_e R(e; \theta(d^\star)) \tag{2.16}$$

With this structure, to fully determine the teacher's decision-making process, the learner only needs to estimate θ.

In contrast, behavior cloning does not assume any structure about the teacher's decision-making process and needs to learn a policy that models the entire process. Specifically, the learner has to search for a mechanism that reasonably explains how the teacher generates demonstrations. As the search space of this problem can be much larger than the space over reward functions, given the same amount of demonstrations, estimating the teacher's reward function would yield a more reliable result than estimating their policy. Consequently, a policy derived from the inferred reward function can potentially generalize more robustly than a policy learned directly from demonstrations. However, this is true only when the structure assumed by IRL is an appropriate explanation of the teacher's behavior. In practice, human teachers can behave irrationally, violating the reward-maximization assumption in Eq 2.16. In those cases, more sophisticated models of human behavior need to be adopted (Shah et al., 2019). Another drawback of IRL is that it inherits the communication bottleneck of reinforcement learning; learning a pol-

icy from rewards can be much more interaction-inefficient than learning from demonstrations.

Pragmatic reasoning (Goodman and Frank, 2016) offers alternative explanations of an agent's behavior. This framework hypothesizes that there is correlation between how an agent interprets communication signals and how they generates them. In our general protocol, suppose when the teacher observes an execution e, they can label it with a task description $\hat{d}$ sampled from a distribution $L_0(d \mid e)$ (referred to as the literal listener model). A pragmatic teacher generate reference demonstrations $d^\star$ according to the following distribution

$$P_{\pi^\star}(e \mid c_0, d^\star) := S_1(e \mid c_0, d^\star) \propto L_0(d^\star \mid e) P(e \mid c_0) \tag{2.17}$$

Here, S_1 is called the pragmatic speaker model and $P(e \mid c_0)$ is a prior over executions that represents the preferences or general knowledge of the teacher. The definition of S_1 states that the more likely the teacher labels an execution as $d^\star$, the more likely that it gives the execution as a demonstration to the learner.

Pragmatic inference is closely connected to IRL. Specifically, if we apply a maximum a posterior approximation and assume a uniform prior $P(e \mid c_0)$, the pragmatic teacher's execution distribution in Eq 2.17 can be expressed as

$$\begin{aligned} P_{\pi^\star}(e \mid c_0, d^\star) &:= \delta(e^\star_{\text{MAP}}) \\ e^\star_{\text{MAP}} &= \underset{e: c_0 \in e}{\operatorname{argmax}} L_0(d^\star \mid e) \end{aligned} \tag{2.18}$$

where $\delta(e)$ is a one-hot distribution peaked at e. The second equation is analogous to the decision-making rule assumed by IRL (Eq 2.16) if setting $R(e; \theta(d)) := L_0(d \mid e)$. However, in general,

the reward function in IRL does not have to be a probability distribution.

To recover the pragmatic teacher's policy, the learner estimates the literal speaker model L_0 from a demonstration dataset containing pairs of $(d^\star, e^\star)$, which is the same dataset given to the learner in an IRL or behavior cloning setting. After that, it can construct an execution policy according to Eq 2.17 or Eq 2.18. However, a policy constructed in this way may not be efficient for making inferences (deciding the best execution to generate). A possible solution is to follow the IRL approach, using RL to learn a new policy that efficiently maps a task description $d^\star$ to the MAP execution $e^\star = \max_e L_0(d^\star \mid e)$. An alternative approach, proposed by CITE, is to narrow the search space in Eq 2.18 by using a literal speaker to generate high-probable candidates. Let $\hat{S}_0(e \mid d, c_0)$ denote the literal speaker, which can be learned from the same demonstration dataset used for learning L_0. We can replace Eq 2.18 by the following approximation

$$e^\star = \underset{e \in \text{CAND}}{\operatorname{argmax}} L_0(d^\star \mid e) \tag{2.19}$$

where $\text{CAND} = \{e_i \mid e_i \sim S_0(\cdot \mid d^\star, c_0), 1 \leq i \leq K\}$ is a set of candidates executions and K is the number of samples drawn from S_0.

Pragmatic reasoning is typically conceptually studied in reference games (Lewis, 2008; Wittgenstein, 1995), where a speaker gives a brief description of a target object and a listener must identify it among distractors. It has been employed to enhance natural language generation and understanding (Andreas and Klein, 2016; Fried et al., 2018a,b). Kang et al. (2020) endow agents in emergent communication games with pragmatic reasoning skills and show that these capabilities improve their communication accuracy.

While the aforementioned frameworks focus on equipping the learner with theory-of-mind

capabilities, there have also been similar attempts on the teacher's side. Cooperative inverse reinforcement learning (Hadfield-Menell et al., 2016) is an extension of IRL that considers also the problem of determining a pragmatic teaching strategy for the teacher. There, the goal of the teacher is to generate demonstrations that convey the most information about the reward function, even though those demonstrations can be suboptimal in terms of maximizing the task's reward function. The work proposes the teacher implement a regularized reward-maximization objective, which penalizes visiting states that are unfamiliar to the learner. He et al. (2012) propose a coaching framework where the teacher adapts its demonstration distribution according to the capability of the learner.

2.3 Human-AI Communication during Task Performance

For many agents, the goal after learning is to be able to collaborate with humans to accomplish tasks. Communication for collaboration is concerned about about instantly and temporarily evoking certain actions from others rather than permanently changing their behavior. Thus, methods for human-AI communication during learning may not be appropriate for this scenario, as learning may take multiple rounds of interactions and model updates to alter an agent's behavior. An effective way for humans to timely influence an agent's behavior is to manipulate the input information that it uses for making decisions. This requires the agent to implement an interface that accepts a human intervention as an input. Our formulation of the agent's execution policy $\hat{\pi}(a \mid c, d^{\star})$, which takes a context and a language task request as input, satisfies this requirement. The policy can process a formulation in two forms: (1) human actions that change the current context c, and (2) human utterances that append to the task request.

This section review frameworks that enable humans coordinate and collaborate with AI agents through communication. I will first consider a basic, popular setting where a human provides only a single instruction to evoke a certain behavior from an agent. I argue that this setting limits the task performance of the agent because the instruction provided by the human may contain insufficient information or is beyond the comprehension capabilities of the agent. Enabling the agent to ask for assistance from humans can dramatically boost its task performance. I will highlight challenges in teaching an agent to request and leverage human assistance, and give a bird-eye view on principal approaches to this problem.

2.3.1 Fulfilling Human Requests without Assistance

An intelligent agent should be able to observe humans and provide timely assistance. But to provide safe and effective assistance, the agent must provide an interface for humans to communicate exactly what they want. Unlike a computer, an assistant AI is expected to perform a much more diverse and dynamic set of tasks, which usually take place in an (virtual or physical) environment. Many of these tasks are traditionally performed by humans; some may be combinations of multiple subtasks. For a regular user, it is much easier to specify these tasks through a referential and compositional language like a human language than a non-verbal or an artificial language. Request fulfilling (also known as instruction following) studies the problem of understanding and satisfying requirements specified in a human language.

Request fulfilling employs a simple yet practical communication protocol: an agent receives a language request $d^\star$ from a human user and responds with an execution $\hat{e}$ that aims to fulfill the request. This protocol requires minimal communication effort from the user, as they

can express their intentions in their natural language and do it only once before the agent executes a request. The burden of communication is placed on the agent. Since the user does not provide any additional information during its execution, the agent needs to perfectly interpret the request in order to carry it out successfully. Request-fulfilling can serve as the building block of more complex, multi-turn communication protocols like the ones introduced in the next section.

SHRDLU, proposed by Winograd (1971), is one of the earliest request-fulfilling research problems. During a session of this problem, a user can send commands such as "*pick up a big red block*" to change positions of blocks situated in a simulated 3D environment. Learning to translate natural language commands to actions has been well-studied at the intersection of NLP and robotics. Early proposals include a variety of grounded parsing models that are trained from data (Chen and Mooney, 2011; Krishnamurthy and Mitchell, 2012; Matuszek et al., 2010, 2012b, 2013; Shimizu and Haas, 2009). From a language utterance, these works construct a symbolic plan, which is a program that can be executed in an environment. More recent work follows a neural approach, learning a neural-network-based policy $\hat{\pi}(a \mid c, d)$ that makes decisions incrementally without pre-constructing a symbolic plan (Anderson et al., 2018b; Misra et al., 2017a). The policy can be learned via imitation learning or reinforcement learning. When the dynamics of the environment is given or can be approximated, a model-based or neural-symbolic hybrid approach can enhance the sample efficiency of the learning algorithm (Li et al., 2020b; Wang et al., 2018; Yi et al., 2018).

Remarkable progress in natural language processing has significantly expanded the range of applications that employ natural language interfaces. These interfaces have been deployed for web navigation (Mazumder and Riva, 2021), mobile-phone assistants (Li et al., 2020e; Liu et al., 2018; Mazumder and Riva, 2020; Shi et al., 2017; Su et al., 2017, 2018), pair-programming (Allen et al.,

2007; Azaria et al., 2016; Fast et al., 2018; Kate et al., 2005; Li et al., 2020d; Lieberman and Liu, 2006; Mihalcea et al., 2006; Pane et al., 2001), information retrieval (Zhao et al., 2022), and media editing (Shi et al., 2021). Recent years witness a rising interest in building multi-purpose request-fulfilling agents that can accomplish tasks in a variety of domains. These agents are typically large neural networks with a self-attention architecture (Vaswani et al., 2017) and are trained on massive amounts of data (Brown et al., 2020a; Chowdhery et al., 2022a; Ramesh et al., 2021). They have demonstrated surprisingly decent capabilities of performing various language tasks described only by a few examples or brief instructions.

The fundamental challenge of request-fulfilling is *grounded language learning*, i.e. acquiring the ability to decipher the meanings of human speeches in context. As mentioned in section § 2.1, the inferential nature of human communication makes the meanings of human utterances inherently contextual. Interpreting an utterance requires comprehending the intricate interplay between general knowledge about the world, the current context, and the form of the utterance. To illustrate this challenge, consider a request "*bring me a pillow*" given to a home-assistant robot. The meaning of this request cannot be derived only from the dictionary meanings of the constituent words and the linguistic structure of the request. The actual sequence of actions implied by the request varies depending on the context (e.g., the house's layout, the robot's location and functionality, the pillow's location and shape, the requester's preferences, etc.) In one context, the request could mean "*go to the bedroom and pick up the white pillow*"; in another, it may entail "*turn around and grab the watermelon-shape pillow*".

The development of grounded language learning in AI was inspired by ideas in cognitive science (Brown et al., 1989; Miller and Johnson-Laird, 1976), which postulates that a person's knowledge is inseparable from their social, cultural and physical contexts. The aforementioned

SHRDLU simulator offers one of the first platforms for computationally studying this topic. (Harnad, 1990) introduce "the symbol grounding problem" as an important subject in AI. Early grounding problems attempted to connect simple words to images (Barnard and Johnson, 2005; Barnard et al., 2003; Berg et al., 2004). Image captioning (Coyne and Sproat, 2001; Zhu et al., 2007) presented a substantially more realistic and challenging task of generating variable-length, structured language utterances from visual input. In another stream of research, the focus is on grounding language to perception and action. Researchers initially experimented on 2D environments with simplistic visual perception (Branavan et al., 2009; Chen and Mooney, 2008, 2011; Matuszek et al., 2010), and have gradually migrated to more visually and physically realistic 3D environments (Anderson et al., 2018b; Puig et al., 2018; Shridhar et al., 2020).

After large language models catalyzed tremendous performance boosts in NLP tasks (Devlin et al., 2019; Howard and Ruder, 2018; Liu et al., 2019; Peters et al., 2018), there has been substantial evidence that acquiring knowledge from only textual data deems inadequate for solving problems that require understanding and reasoning about the physical world (Habernal et al., 2018; Hill et al., 2019; Jia and Liang, 2017; McCoy et al., 2019; Wallace et al., 2019a). These proposals have reinvigorated interests in learning language interactively and in conjunction with other perception modalities. Various new research benchmarks have been developed with increased complexity and enhanced practicality, such as image captioning (Gurari et al., 2020; Lin et al., 2014), visual question answering (Antol et al., 2015), visual dialog (Das et al., 2017b), 3D scene generation from text (Chang et al., 2014, 2015a), multimodal machine translation (Elliott et al., 2016; Wang et al., 2019e), photo-realistic instruction-following (Anderson et al., 2018b; Chen et al., 2019a; Misra et al., 2017b). Recent methods for solving these tasks not only use labeled multimodal data but also leverage unlabeled data or data that comes with "free" labels (Gan

et al., 2020; Gordon et al., 2020; Li et al., 2019; Lu et al., 2019; Qi et al., 2020a; Rahman et al., 2020; Su et al., 2019; Sun et al., 2019a; Tan and Bansal, 2020; Zhou et al., 2020b). These methods learn universal grounded language representations that can be transferred across multimodal learning tasks (Hao et al., 2020; Li et al., 2020a).

2.3.2 Leveraging Assistance from Humans

Humans and agents may not always communicate perfectly with one another. Humans can under- or mis-specify their intentions, because they are not communicating cooperatively or having false expectations about others. Meanwhile, agents may have limited generalizability that hinders them from understanding novel requests given by humans. By conducting only a single round of communication, the request-fulfilling protocol does not offer opportunities for humans and agents to correct these communication mistakes. More interactive protocols that involve multiple rounds of information exchange can potentially enhance communication accuracy and thus improve performance of agents. To ensure that the productivity and satisfaction of humans is also boosted, it is essential to equip agents with effective skills as a speaker, especially the ability to timely request intervention from humans and to provide sufficient information to inform humans' supporting actions.

Building agents that can engage in multi-turn conversations with humans is not a newly emerged discipline. Automated chatbots were built since the 1960s (Weizenbaum, 1966) and have been consistently augmented. Current state-of-the-art systems are trained on massive collections of human conversations and are able to generate diverse types of questions (Adiwardana et al., 2020; Rae et al., 2021). These systems can be categorized into two types: social-bot agents and

task-oriented agents. Social-bot agents (Chen et al., 2018; Ram et al., 2018; Zhou et al., 2020a) aim to create a natural experience for a human when conversing with them. These agents may ask questions to lubricate a conversation and draw engagement from human users. On the other hand, task-oriented agents (Bordes et al., 2017; Chen et al., 2019c; Valizadeh and Parde, 2022) communicate to obtain information from humans to complete a task. These agents are directly related to the work presented in chapter §4, as they ask questions with a clear goal of enhancing their decisions on a certain task. Task-oriented dialog agents have largely been developed in purely conversational problems, where asking questions to humans is required to complete the task (Hemphill et al., 1990; Rastogi et al., 2020; Wei et al., 2018). Nevertheless, in domains where communication with humans is not an intrinsic part of the task, equipping agents with this ability has improved agent performance in novel conditions (Nguyen and Daumé III, 2019; Nguyen et al., 2019b; Tellex et al., 2014b; Wei et al., 2018). These human-in-the-loop settings are widely encountered in human-robot collaboration (Bauer et al., 2008; Chandrasekaran and Conrad, 2015; Colgate et al., 1996; Fong et al., 2003; Rosenthal and Veloso, 2012; Tellex et al., 2014b; Villani et al., 2018), and recently have also been pushed forward in the NLP community by several proposals (Menon et al., 2022; Padmakumar and He, 2021; Wang et al., 2021).

Chapter §4 focuses on a particular language-based communication protocol in which an agent generates and sends help requests to a human to receive language instructions from them. In this setting, to elicit useful information from humans while retaining trust, it is important to teach the agent to ask for help only when necessary and to generate informative requests. A popular approach to this problem is to train the agent on a collection of human-generated conversations (Adiwardana et al., 2020; Budzianowski et al., 2018; Jayannavar et al., 2020; Narayan-Chen et al., 2019b; Padmakumar et al., 2021; Qian et al., 2021; Rae et al., 2021; Thomason et al., 2019b).

While useful in many cases, naively imitating human behavior has a potential drawback: human decisions may not always be contextually optimal for agents. For example, consider the task of navigating to locations in a house. Humans are experts at object recognition, thus would rarely ask questions like "*what objects are around me?*". Nevertheless, because a robot may have imperfect perception possibly due to its limited object-recognition module, asking such questions may be useful for it in localization. In general, decisions on when and what to ask should be grounded in the current context, which importantly includes the agent's decision-making policy. A simple way to incorporate the agent's policy into these decisions is to program the agent to ask when the output uncertainty of its policy exceeds a threshold (Abbasnejad et al., 2019; Chi et al., 2020). However, this approach is unreliable when the agent is miscalibrated, i.e. its estimated probability of an event does not align with the real-world frequency of that event. Furthermore, using a fixed decision threshold for all situations is highly restrictive. Interactive learning techniques (Liu et al., 2022; Nguyen and Daumé III, 2019) address these problems, as the agent develops adaptive behavior that reacts to perception of the current situation.

Besides language-based communication, researchers have proposed other protocols for human-robot collaboration. Cha and Matarić (2016) investigate the use of extra-linguistic signals in human-robot collaboration. Kwon et al. (2018) propose a method for optimizing motion of a robot to better express its incapability. As an alternative to collecting verbal guidance, agents may also request direct physical intervention from humans (Shiomi et al., 2008; Tellex et al., 2014b). In a real-world application, drawing engagement from humans requires delicate customization and combination of various types of interaction. Factors that are often overlooked in prototypical research such as gesture, speaking tone, and appearance of an agent can markedly bolster or deter a human's willingness to help (Backhaus et al., 2018; Bajones, 2016; Bajones

et al., 2016; Cameron et al., 2015, 2016; Hüttenrauch and Severinson-Eklundh, 2006; Morales et al., 2019; Vanzo et al., 2020; Weiss et al., 2010).

2.4 Vision-Language Navigation

Experimentation on real humans and robots are often expensive, time-consuming, and even dangerous. Simulated environments provide safe and inexpensive platforms for rapidly prototyping and evaluating new ideas before deploying them in real-world scenarios. Traditionally, video-game and physics simulators have been used to benchmark reinforcement learning algorithms (Brockman et al., 2016; Kempka et al., 2016; Mnih et al., 2013; Todorov et al., 2012; Vinyals et al., 2017). Nevertheless, these environments under-represent the complexity of the real world. Moreover, most of these environments do not model interaction with humans through natural language. In NLP, there are efforts in building interactive text-based worlds (Côté et al., 2018; Fan et al., 2020; Urbanek et al., 2019) but the lack of a graphical component makes these environments not suitable for grounded language learning.

Anderson et al. (2018b) presents the Matterport3D simulator, a photo-realistic simulator that emulates first-person indoor navigation. The simulator provides 90 environments in total, featuring over 50,000 object instances and 1,600 object types. Each environment is constructed from a 3D scan of a real house collected by (Chang et al., 2017a) using a Matterport3D camera. The environment is structured as an undirected graph whose nodes correspond to locations in the house, and edges connect nearby locations that are averagely 2.5 meters apart. The agent may traverse between the nodes in the environment by following the edges. Every node is associated with a 360-degree photo capturing the view at the corresponding location. When visiting the

location, the agent is given a photo that is generated from the panoramic photo representing its current first-person view. The simulator allows customizing the field of vision of the agent. The agent may rotate its view horizontally and vertically, after which the simulator will change the current-view photo accordingly.

Based on the Matterport3D simulator, the authors propose a request-fulfilling problem called vision-language navigation (VLN). In this problem, an agent takes as input a human-language instruction that describes a path in an environment, and follows the instruction to arrive at an intended destination. To support imitation learning on this problem, the authors collected a dataset called Room-to-Room containing over 21,000 language instructions generated by human annotators.

Since its inception, VLN has attracted enormous attention from AI researchers. The problem combines the core challenges of multiple AI-related disciplines, requiring techniques for grounding language to visual perception and action. These three modalities are the main subjects of study in robotics, computer vision, and NLP, respectively. While it is not the first time that they are put together in a problem, VLN offers a degree of realism in the vision and language modalities that surpass previous proposals by a substantial margin. VLN also manifests the difficulties of a sequential decision-making problem. From the perspective of reinforcement learning researchers, the problem offers an opportunity to step up from environments with simple graphics and evaluate whether methods developed in those environments can be successfully transferred to domains with highly complex input structures. The appeal of VLN as a research benchmark is that while it presents a full package of technical challenges, the creators manage to make the problem approachable for researchers that were only interested in a subset of those challenges. Reasonable baselines can be easily constructed by applying encoder-decoder neural networks,

which is a standard deep learning architecture, with pre-computed embeddings of the location views provided by the creators. This helps researchers easily setup a working solution and start investigating solutions to aspects of the problem that they are concerned about.

Within a few years, there has been a plethora of techniques and models proposed for tackling VLN. Early improvements were gained by optimizing the inference procedure and employing RL to fine-tune models pre-trained with imitation learning. Notably, Fried et al. (2018a) apply pragmatic inference to this problem, learning a speaker model to re-rank candidate paths generated by the navigation agent. In a nutshell, the speaker model measures the compatibility between a path and a language instruction. The agent chooses to output the path that maximizes the compatibility score with input instruction estimated by the speaker model. Wang et al. (2019d) take this idea a step further, using the compatibility score as a reward to optimize iteratively with RL. More recent performance breakthroughs comes from implementing more effective architectures like BERT (Hong et al., 2020), and from pre-training on large-scale unlabeled multi-modality datasets (Guhur et al., 2021; Wang et al., 2019d). A more detailed classification of methods and trends on VLN can be found in more thorough surveys like (Gu et al., 2022; Park and Kim, 2022; Wu et al., 2021).

Chapter §4 proposes human-assisted extensions of VLN, where the navigation agent is allowed to request and leverage human instructions to recover from failure or situations with high uncertainty. To my best knowledge, these are the first interactive formulations of this problem. Following our work, there have been other variants like the CDVN dataset (Thomason et al., 2020) and the JustAsk framework (Chi et al., 2020). Outside the Matterport3D environments, there are also works that combine dialog with sequential decision-making in a high-fidelity simulator (de Vries et al., 2018; Narayan-Chen et al., 2019b; Padmakumar et al., 2021; Suhr et al., 2019).

The main distinction between my proposal and those presented in these works is that, rather than learning the agent's behavior by mimicking human behavior captured by a static dataset, I construct from the dataset simulations of humans and learn the agent's behavior by interacting with these simulations. The latter approach enables the agent to make contextually relevant decisions that are grounded in its own perception of the current situation.

Chapter 3: Enriching Human-AI Communication During Training

Intelligent agents must be able to continuously extend their capabilities in order to adapt to changing environments. Humans is an immense source of knowledge that agents can potentially leverage to enhance their capabilities. While agents can mine general knowledge from static knowledge sources like collections of texts, images and videos, humans can provide customized, refined knowledge that can be much more contextually helpful for agents. Importantly, only humans can provide information about their own requirements and preferences, which are requisite for agents to attain satisfactory performance.

To enable learning from humans, agents must employ learning algorithms that allow humans to conveniently and effectively convey knowledge to them. In this section, I explore two types of communication medium humans can use to teach agents: numerical rating and language description. Regarding the first type, I demonstrate the possibility and challenges of using reinforcement learning for improving machine translation models with human ratings (Nguyen et al. (2017); §3.1). Training with reinforcement learning results in significant performance gain, even when ratings are moderately noisy. But obtaining the best results requires pre-training the model with supervised learning, as numerical ratings alone do not contain sufficiently strong learning signals for the model. These findings motivate me to develop a framework that enable humans to deliver learning signals with their natural languages (Nguyen et al. (2021); §3.2). My framework

focuses on learning from descriptive language, which characterizes the intentions of the learner perceived the teacher. Verbally describing actions is one of the most common types of linguistic interactions that mothers conduct to distill to their babies the very first linguistic knowledge (Tamis-LeMonda et al., 2001). Teaching through descriptive feedback closely resembles the idea of back-translation in NLP (Sennrich et al., 2016), and has been studied in reinforcement learning contexts (Andrychowicz et al., 2017; Eysenbach et al., 2020; Packer et al., 2021). Taking a different perspective from previous work, my work formulates a learning framework that considers language descriptions as *communication signals* sent from the teacher, rather than as augmented data. Employing a distinct type of communication signal positions our framework as a new general statistical learning framework that is comparable with reinforcement learning and imitation learning. In fact, our framework strictly generalizes reinforcement learning, because reward can be viewed as a limited type of descriptive language.[1] Reinforcing this claim, there has been work shows that the principle for training with general descriptions presented in my work can be seamlessly adapted for training with rewards, giving rise to a new breed of reinforcement learning algorithms (Chen et al., 2021; Janner et al., 2021; Schmidhuber, 2019).

[1] If we consider the teacher's training intention as "*you [the agent] should achieve a reward of R_{max} on a task*" and let $\mathcal{S}$ be the space over all of possible teacher intentions, then the teacher's signal "*you have achieved a reward of R_{max} on a task*) essentially describes an intention in $\mathcal{S}$. Giving a reward thus represents sending a descriptive communication signal.

3.1 BANDITNMT: Reinforcement Learning for Bandit Neural Machine Translation with Simulated Human Feedback

3.1.1 Overview

Bandit structured prediction is the task of learning to solve complex joint prediction problems (like parsing or machine translation) under a very limited feedback model: a system must produce a *single* structured output (e.g., translation) and then the world reveals a *score* that measures how good or bad that output is, but provides neither a "correct" output nor feedback on any other possible output (Chang et al., 2015c; Sokolov et al., 2015). Because of the extreme sparsity of this feedback, a common experimental setup is that one pre-trains a good-but-not-great "reference" system based on whatever labeled data is available, and then seeks to improve it over time using this bandit feedback. A common motivation for this problem setting is cost. In the case of translation, bilingual "experts" can read a source sentence and a possible translation, and can much more quickly provide a rating of that translation than they can produce a full translation on their own. Furthermore, one can often collect even less expensive ratings from "non-experts" who may or may not be bilingual (Hu et al., 2014). Breaking this reliance on expensive data could unlock previously ignored languages and speed development of broad-coverage machine translation systems.

All work on bandit structured prediction we know makes an important simplifying assumption: the *score* provided by the world is *exactly* the score the system must optimize (§3.1.2). In the case of parsing, the score is attachment score; in the case of machine translation, the score is (sentence-level) BLEU. While this simplifying assumption has been incredibly useful in building

algorithms, it is highly unrealistic. Any time we want to optimize a system by collecting user feedback, we must take into account:

1. The metric we care about (e.g., expert ratings) may not correlate perfectly with the measure that the reference system was trained on (e.g., BLEU or log likelihood);
2. Human judgments might be more granular than traditional continuous metrics (e.g., thumbs up vs. thumbs down);
3. Human feedback have high *variance* (e.g., different raters might give different responses given the same system output);
4. Human feedback might be substantially *skewed* (e.g., a rater may think all system outputs are poor).

Our first contribution is a strategy to simulate expert and non-expert ratings to evaluate the robustness of bandit structured prediction algorithms in general, in a more realistic environment (§3.1.4).

We construct a family of perturbations to capture three attributes: *granularity*, *variance*, and *skew*. We apply these perturbations on automatically generated scores to simulate noisy human ratings. To make our simulated ratings as realistic as possible, we study recent human evaluation data (Graham et al., 2017) and fit models to match the noise profiles in actual human ratings (§3.1.4.2).

Our second contribution is a reinforcement learning solution to bandit structured prediction and a study of its robustness to these simulated human ratings (§3.1.3). We combine an encoder-decoder architecture of machine translation (Luong et al., 2015) with the advantage actor-critic algorithm (Mnih et al., 2016), yielding an approach that is simple to implement but works on low-resource bandit machine translation. Even with substantially restricted granularity, with

high variance feedback, or with skewed rewards, this combination improves pre-trained models (§3.1.6). In particular, under realistic settings of our noise parameters, the algorithm's online reward and final held-out accuracies do not significantly degrade from a noise-free setting.

3.1.2 Bandit Machine Translation

The bandit structured prediction problem (Chang et al., 2015c; Sokolov et al., 2015) is an extension of the contextual bandits problem (Kakade et al., 2008; Langford and Zhang, 2008c) to structured prediction. Bandit structured prediction operates over time $i = 1 \dots K$ as:

1. World reveals context $\boldsymbol{x}^{(i)}$
2. Algorithm predicts structured output $\hat{\boldsymbol{y}}^{(i)}$
3. World reveals reward $R\left(\hat{\boldsymbol{y}}^{(i)}, \boldsymbol{x}^{(i)}\right)$

We consider the problem of *learning to translate from human ratings* in a bandit structured prediction framework. In each round, a translation model receives a source sentence $\boldsymbol{x}^{(i)}$, produces a translation $\hat{\boldsymbol{y}}^{(i)}$, and receives a rating $R\left(\hat{\boldsymbol{y}}^{(i)}, \boldsymbol{x}^{(i)}\right)$ from a human that reflects the quality of the translation. We seek an algorithm that achieves high reward over K rounds (high cumulative reward). The challenge is that even though the model knows how good the translation is, it knows neither *where* its mistakes are nor *what* the "correct" translation looks like. It must balance exploration (finding new good predictions) with exploitation (producing predictions it already knows are good). This is especially difficult in a task like machine translation, where, for a twenty token sentence with a vocabulary size of $50k$, there are approximately 10^{94} possible outputs, of which the algorithm gets to test exactly one.

Despite these challenges, learning from non-expert ratings is desirable. In real-world sce-

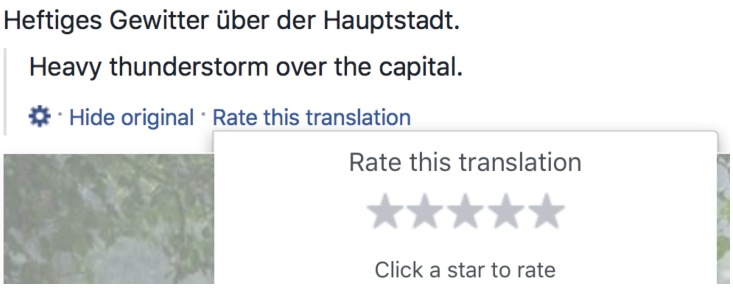


Figure 3.1: A translation rating interface provided by the Facebook social network. Users see a sentence followed by its machined-generated translation and can give ratings from 1 to 5 stars.

narios, non-expert ratings are easy to collect but other stronger forms of feedback are prohibitively expensive. Platforms that offer translations can get quick feedback "for free" from their users to improve their systems (Figure 3.1). Even in a setting in which annotators are paid, it is much less expensive to ask a bilingual speaker to provide a rating of a proposed translation than it is to pay a professional translator to produce one from scratch.

3.1.3 Effective Algorithm for Bandit Machine Translation

This section describes the neural machine translation architecture of our system (§3.1.3.1). We formulate bandit neural machine translation as a reinforcement learning problem (§3.1.3.2) and discuss why standard actor-critic algorithms struggle with this problem (§3.1.3.3). Finally, we describe a more effective training approach based on the advantage actor-critic algorithm (§3.1.3.4).

3.1.3.1 Neural machine translation

Our neural machine translation (NMT) model is an encoder-decoder that directly computes the probability of translating a target sentence $\boldsymbol{y} = (y_1, \cdots, y_m)$ from a source sentence $\boldsymbol{x}$:

$$P_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{t=1}^{m} P_{\boldsymbol{\theta}}(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x}) \tag{3.1}$$

where $P_{\boldsymbol{\theta}}(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x})$ is the probability of outputting the next word y_t at time step t given a translation prefix $\boldsymbol{y}_{<t}$ and a source sentence $\boldsymbol{x}$.

We use a neural encoder-decoder NMT architecture with global attention (Luong et al., 2015), where both the encoder and decoder are recurrent neural networks (RNN) (see § A for a more detailed description). These models are normally trained by supervised learning, but as reference translations are not available in our setting, we use reinforcement learning methods, which only require numerical feedback to function.

3.1.3.2 Bandit NMT as Reinforcement Learning

NMT generating process can be viewed as a Markov decision process on a continuous state space. The states are the hidden vectors $\boldsymbol{h}_t^{dec}$ generated by the decoder. The action space is the target language's vocabulary.

To generate a translation from a source sentence $\boldsymbol{x}$, an NMT model starts at an initial state $\boldsymbol{h}_0^{dec}$: a representation of $\boldsymbol{x}$ computed by the encoder. At time step t, the model decides the next action to take by defining a stochastic policy $P_{\boldsymbol{\theta}}(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x})$, which is directly parametrized by the parameters $\boldsymbol{\theta}$ of the model. This policy takes the current state $\boldsymbol{h}_{t-1}^{dec}$ as input and produces a

probability distribution over all actions (target vocabulary words). The next action $\hat{y}_t$ is chosen by taking $\arg\max$ or sampling from this distribution. The model computes the next state $\boldsymbol{h}_t^{dec}$ by updating the current state $\boldsymbol{h}_{t-1}^{dec}$ by the action taken $\hat{y}_t$.

The objective of bandit NMT is to find a policy that maximizes the expected reward of translations sampled from the model's policy:

$$\max_{\boldsymbol{\theta}} \mathcal{L}_{pg}(\boldsymbol{\theta}) = \max_{\boldsymbol{\theta}} \mathbb{E}_{\substack{\boldsymbol{x}\sim D_{\text{tr}} \\ \hat{\boldsymbol{y}}\sim P_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}} \Big[R(\hat{\boldsymbol{y}}, \boldsymbol{x}) \Big] \tag{3.2}$$

where D_{tr} is the training set and R is the reward function (the rater).[2] We optimize this objective function with policy gradient methods. For a fixed $\boldsymbol{x}$, the gradient of the objective in Eq 3.2 is:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \mathcal{L}_{pg}(\boldsymbol{\theta}) &= \mathbb{E}_{\hat{\boldsymbol{y}}\sim P_{\boldsymbol{\theta}}(\cdot)} \left[R(\hat{\boldsymbol{y}}) \nabla_{\boldsymbol{\theta}} \log P_{\boldsymbol{\theta}}(\hat{\boldsymbol{y}}) \right] \\ &= \mathbb{E}_{\hat{\boldsymbol{y}}\sim P_{\boldsymbol{\theta}}(\cdot)} \Big[\sum_{t=1}^{m} \sum_{\hat{y}_t} Q(\hat{\boldsymbol{y}}_{<t}, \hat{y}_t) \nabla_{\boldsymbol{\theta}} P_{\boldsymbol{\theta}}(\hat{y}_t \mid \hat{\boldsymbol{y}}_{<t}) \Big] \end{aligned} \tag{3.3}$$

where $Q(\hat{\boldsymbol{y}}_{<t}, \hat{y}_t)$ is the expected future reward of $\hat{y}_t$ given the current prefix $\hat{\boldsymbol{y}}_{<t}$, then continuing sampling from $P_{\boldsymbol{\theta}}$ to complete the translation:

$$\begin{aligned} Q(\hat{\boldsymbol{y}}_{<t}, \hat{y}_t) &= \mathbb{E}_{\hat{\boldsymbol{y}}'\sim P_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})} \Big[\tilde{R}(\hat{\boldsymbol{y}}', \boldsymbol{x}) \Big] \\ \text{with } \tilde{R}(\hat{\boldsymbol{y}}', \boldsymbol{x}) &\equiv R(\hat{\boldsymbol{y}}', \boldsymbol{x}) \mathbb{1} \left\{ \hat{\boldsymbol{y}}'_{<t} = \hat{\boldsymbol{y}}_{<t}, \hat{y}'_t = \hat{y}_t \right\} \end{aligned} \tag{3.4}$$

$\mathbb{1}\{\cdot\}$ is the indicator function, which returns 1 if the logic inside the bracket is true and returns 0 otherwise.

The gradient in Eq 3.3 requires rating all possible translations, which is not feasible in ban-

[2] Our raters are *stochastic*, but for simplicity we denote the reward as a function; it should be expected reward.

dit NMT. Naïve Monte Carlo reinforcement learning methods such as REINFORCE (Williams, 1992b) estimates Q values by sample means but yields very high variance when the action space is large, leading to training instability.

3.1.3.3 Why are actor-critic algorithms not effective for bandit NMT?

Reinforcement learning methods that rely on function approximation are preferred when tackling bandit structured prediction with a large action space because they can capture similarities between structures and generalize to unseen regions of the structure space. The actor-critic algorithm (Konda and Tsitsiklis, 1999) uses function approximation to directly model the Q function, called the *critic* model. In our early attempts on bandit NMT, we adapted the actor-critic algorithm for NMT in Bahdanau et al. (2017), which employs the algorithm in a supervised learning setting. Specifically, while an encoder-decoder critic model Q_{ω} as a substitute for the true Q function in Eq 3.3 enables taking the full sum inside the expectation (because the critic model can be queried with any state-action pair), we are unable to obtain reasonable results with this approach.

Nevertheless, insights into why this approach fails on our problem explains the effectiveness of the approach discussed in the next section. There are two properties in Bahdanau et al. (2017) that our problem lacks but are key elements for a successful actor-critic. The first is access to reference translations: while the critic model is able to observe reference translations during training in their setting, bandit NMT assumes those are never available. The second is per-step rewards: while the reward function in their setting is known and can be exploited to compute immediate rewards after taking each action, in bandit NMT, the actor-critic algorithm struggles

with credit assignment because it only receives reward when a translation is completed. Bahdanau et al. (2017) report that the algorithm degrades if rewards are delayed until the end, consistent with our observations.

With an enormous action space of bandit NMT, approximating gradients with the Q critic model induces biases and potentially drives the model to wrong optima. Values of rarely taken actions are often overestimated without an explicit constraint between Q values of actions (e.g., a sum-to-one constraint). Bahdanau et al. (2017) add an ad-hoc regularization term to the loss function to mitigate this issue and further stablizes the algorithm with a delay update scheme, but at the same time introduces extra tuning hyper-parameters.

3.1.3.4 Advantage Actor-Critic for Bandit NMT

We follow the approach of advantage actor-critic (Mnih et al., 2016, A2C) and combine it with the neural encoder-decoder architecture. The resulting algorithm—which we call NED-A2C—approximates the gradient in Eq 3.3 by a single sample $\hat{\boldsymbol{y}} \sim P(\cdot \mid \hat{\boldsymbol{x}})$ and centers the reward $R(\hat{\boldsymbol{y}})$ using the state-specific expected future reward $V(\hat{\boldsymbol{y}}_{<t})$ to reduce variance:

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{pg}(\boldsymbol{\theta}) \approx \sum_{t=1}^{m} \bar{R}_t(\hat{\boldsymbol{y}}) \nabla_{\boldsymbol{\theta}} \log P_{\boldsymbol{\theta}}(\hat{y}_t \mid \hat{\boldsymbol{y}}_{<t}) \tag{3.5}$$

$$\text{with } \bar{R}_t(\hat{\boldsymbol{y}}) \equiv R(\hat{\boldsymbol{y}}) - V(\hat{\boldsymbol{y}}_{<t})$$

$$V(\hat{\boldsymbol{y}}_{<t}) \equiv \mathbb{E}_{\hat{y}'_t \sim P(\cdot \mid \hat{\boldsymbol{y}}_{<t})} \left[Q(\hat{\boldsymbol{y}}_{<t}, \hat{y}'_t) \right]$$

We train a separate attention-based encoder-decoder model $V_{\boldsymbol{\omega}}$ to estimate V values. This model encodes a source sentence $\boldsymbol{x}$ and decodes a sampled translation $\hat{\boldsymbol{y}}$. At time step t, it

computes $V_{\boldsymbol{\omega}}(\hat{\boldsymbol{y}}_{<t}, \boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{h}_t^{crt}$, where $\boldsymbol{h}_t^{crt}$ is the current decoder's hidden vector and $\boldsymbol{w}$ is a learned weight vector. The critic model minimizes the MSE between its estimates and the true values:

$$\mathcal{L}_{crt}(\boldsymbol{\omega}) = \mathbb{E}_{\substack{\boldsymbol{x}\sim D_{\text{tr}} \\ \hat{\boldsymbol{y}}\sim P_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}} \left[\sum_{t=1}^{m} L_t(\hat{\boldsymbol{y}}, \boldsymbol{x}) \right] \tag{3.6}$$

$$\text{with } L_t(\hat{\boldsymbol{y}}, \boldsymbol{x}) \equiv \left[V_{\boldsymbol{\omega}}(\hat{\boldsymbol{y}}_{<t}, \boldsymbol{x}) - R(\hat{\boldsymbol{y}}, \boldsymbol{x}) \right]^2 .$$

We use a gradient approximation to update $\boldsymbol{\omega}$ for a fixed $\boldsymbol{x}$ and $\hat{\boldsymbol{y}} \sim P(\cdot \mid \hat{\boldsymbol{x}})$:

$$\nabla_{\boldsymbol{\omega}} \mathcal{L}_{crt}(\boldsymbol{\omega}) \approx \sum_{t=1}^{m} \left[V_{\boldsymbol{\omega}}(\hat{\boldsymbol{y}}_{<t}) - R(\hat{\boldsymbol{y}}) \right] \nabla_{\boldsymbol{\omega}} V_{\boldsymbol{\omega}}(\hat{\boldsymbol{y}}_{<t}) \tag{3.7}$$

NED-A2C is better suited for problems with a large action space and has other advantages over actor-critic. For large action spaces, approximating gradients using the V critic model induces lower biases than using the Q critic model. As implied by its definition, the V model is robust to biases incurred by rarely taken actions since rewards of those actions are weighted by very small probabilities in the expectation. In addition, the V model has a much smaller number of parameters and thus is more sample-efficient and more stable to train than the Q model. These attractive properties were not studied in A2C's original paper (Mnih et al., 2016).

Algorithm 2 The NED-A2C algorithm for bandit NMT.

1: **for** $i = 1 \cdots K$ **do**
2: receive a source sentence $\boldsymbol{x}^{(i)}$
3: sample a translation: $\hat{\boldsymbol{y}}^{(i)} \sim P_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{x}^{(i)})$
4: receive reward $R(\hat{\boldsymbol{y}}^{(i)}, \boldsymbol{x}^{(i)})$
5: update the NMT model using Eq 3.5.
6: update the critic model using Eq 3.7.

Algorithm 2 summarizes NED-A2C for bandit NMT. For each $\boldsymbol{x}$, we draw a single sample $\hat{\boldsymbol{y}}$ from the NMT model, which is used for both estimating gradients of the NMT model and the critic model. We run this algorithm with mini-batches of $\boldsymbol{x}$ and aggregate gradients over all $\boldsymbol{x}$ in a mini-batch for each update. Although our focus is on bandit NMT, this algorithm naturally works with any bandit structured prediction problem.

3.1.4 Modeling Imperfect Ratings

Our goal is to establish the feasibility of using *real* human feedback to optimize a machine translation system, in a setting where one can collect *expert* feedback as well as a setting in which one only collects *non-expert* feedback. In all cases, we consider the expert feedback to be the "gold standard" that we wish to optimize. To establish the feasibility of driving learning from human feedback *without* doing a full, costly user study, we begin with a simulation study. The key aspects (Figure 3.2) of human feedback we capture are: (a) mismatch between training objective and feedback-maximizing objective, (b) human ratings typically are binned (§3.1.4.1), (c) individual human ratings have high variance (§ 3.1.4.2), and (d) non-expert ratings can be skewed with respect to expert ratings (§3.1.4.3).

In our simulated study, we begin by modeling gold standard human ratings using add-one-smoothed sentence-level BLEU (Chen and Cherry, 2014).[3] Our evaluation criteria, therefore, is average sentence-BLEU over the run of our algorithm. However, in any realistic scenario, human feedback will vary from its average, and so the reward that our algorithm receives will be a *perturbed* variant of sentence-BLEU. In particular, if the sentence-BLEU score is $s \in [0, 1]$, the algorithm will only observe $s' \sim \text{pert}(s)$, where pert is a perturbation distribution. Because our

[3] "Smoothing 2" in Chen and Cherry (2014). We also add one to lengths when computing the brevity penalty.

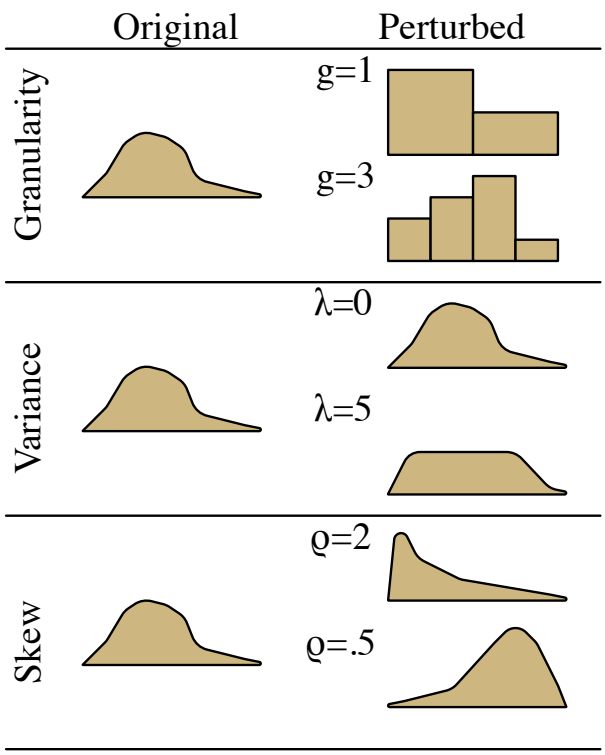


Figure 3.2: Examples of how our perturbation functions change the "true" feedback distribution (left) to ones that better capture features found in human feedback (right).

reference machine translation system is pre-trained using log-likelihood, there is already an (a) mismatch between training objective and feedback, so we focus on (b-d) below.

3.1.4.1 Humans Provide Granular Feedback

When collecting human feedback, it is often more effective to collect discrete *binned* scores. A classic example is the Likert scale for human agreement (Likert, 1932) or star ratings for product reviews. Insisting that human judges provide continuous values (or feedback at too fine a granularity) can demotivate raters without improving rating quality (Preston and Colman, 2000).

To model granular feedback, we use a simple rounding procedure. Given an integer param-

eter g for degree of granularity, we define:

$$\text{pert}^{\text{gran}}(s; g) = \frac{1}{g}\text{round}(gs) \tag{3.8}$$

This perturbation function divides the range of possible outputs into $g + 1$ bins. For example, for $g = 5$, we obtain bins $[0, 0.1)$, $[0.1, 0.3)$, $[0.3, 0.5)$, $[0.5, 0.7)$, $[0.7, 0.9)$ and $[0.9, 1.0]$. Since most sentence-BLEU scores are much closer to zero than to one, many of the larger bins are frequently vacant.

3.1.4.2 Experts Have High Variance

Human feedback has high variance around its expected value. A natural goal for a variance model of human annotators is to simulate—as closely as possible—how human raters actually perform. We use human evaluation data recently collected as part of the WMT shared task (Graham et al., 2017). The data consist of 7200 sentences multiply annotated by giving non-expert annotators on Amazon Mechanical Turk a reference sentence and a *single* system translation, and asking the raters to judge the adequacy of the translation.[4]

From these data, we treat the *average* human rating as the ground truth and consider how individual human ratings vary around that mean. To visualize these results with kernel density estimates (standard normal kernels) of the *standard deviation*. Figure 3.3 shows the mean rating (x-axis) and the deviation of the human ratings (y-axis) at each mean.[5] As expected, the standard deviation is small at the extremes and large in the middle (this is a bounded interval), with a fairly

[4]Typical machine translation evaluations evaluate pairs and ask annotators to choose which is better.

[5]A current limitation of this model is that the simulated noise is i.i.d. conditioned on the rating (homoscedastic noise). While this is a stronger and more realistic model than assuming no noise, real noise is likely heteroscedastic: dependent on the input.

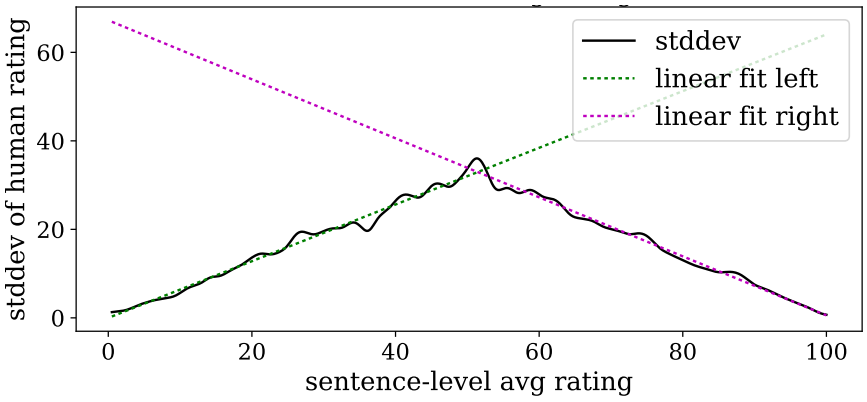


Figure 3.3: Average rating (x-axis) versus a kernel density estimate of the variance of human ratings around that mean, with linear fits. Human scores vary more around middling judgments than extreme judgments.

large range in the middle: a translation whose average score is 50 can get human evaluation scores anywhere between 20 and 80 with high probability. We use a linear approximation to define our variance-based perturbation function as a Gaussian distribution, which is parameterized by a scale λ that grows or shrinks the variances (when $\lambda = 1$ this exactly matches the variance in the plot).

$$\text{pert}^{\text{var}}(s; \lambda) = \text{Nor}\left(s, \lambda\sigma(s)^2\right) \tag{3.9}$$

$$\sigma(s) = \begin{cases} 0.64s - 0.02 & \text{if } s < 50 \\ -0.67s + 67.0 & \text{otherwise} \end{cases}$$

3.1.4.3 Non-Experts are Skewed from Experts

The preceding two noise models assume that the reward closely models the value we want to optimize (has the same mean). This may not be the case with non-expert ratings. Non-expert raters are skewed both for reinforcement learning (Loftin et al., 2014; Thomaz and Breazeal,

Table 3.1: Sentence counts in data sets.

	De-En	Zh-En
Supervised training	186K	190K
Bandit training	167K	165K
Development	7.7K	7.9K
Test	9.1K	7.4K

2008; Thomaz et al., 2006) and recommender systems (Adomavicius and Zhang, 2012; Herlocker et al., 2000), but are typically bimodal: some are harsh (typically provide very low scores, even for "okay" outputs) and some are motivational (providing high scores for "okay" outputs).

We can model both harsh and motivations raters with a simple deterministic skew perturbation function, parametrized by a scalar $\rho \in [0, \infty)$:

$$\text{pert}^{\text{skew}}(s; \rho) = s^{\rho} \tag{3.10}$$

For $\rho > 1$, the rater is harsh; for $\rho < 1$, the rater is motivational.

3.1.5 Experimental Setup

We choose two language pairs from different language families with different typological properties: German-to-English and (De-En) and Chinese-to-English (Zh-En). We use parallel transcriptions of TED talks for these pairs of languages from the machine translation track of the IWSLT 2014 and 2015 (Cettolo et al., 2012, 2014, 2015). For each language pair, we split its data into four sets for supervised training, bandit training, development and testing (Table 3.1). For English and German, we tokenize and clean sentences using Moses (Koehn et al., 2007). For Chinese, we use the Stanford Chinese word segmenter (Chang et al., 2008) to segment

sentences and tokenize. We remove all sentences with length greater than 50, resulting in an average sentence length of 18. We use IWSLT 2015 data for supervised training and development, IWSLT 2014 data for bandit training and previous years' development and evaluation data for testing.[6]

3.1.5.1 Evaluation Framework

For each task, we first use the supervised training set to pre-train a reference NMT model using supervised learning. On the same training set, we also pre-train the critic model with translations sampled from the pre-trained NMT model. Next, we enter a bandit learning mode where our models only observe the source sentences of the bandit training set. Unless specified differently, we train the NMT models with NED-A2C for one pass over the bandit training set. If a perturbation function is applied to Per-Sentence BLEU scores, it is only applied in this stage, not in the pre-training stage.

We measure the *improvement* ΔS of an evaluation metric S due to bandit training: $\Delta S = S_{A2C} - S_{ref}$, where S_{ref} is the metric computed on the reference models and S_{A2C} is the metric computed on models trained with NED-A2C. Our primary interest is *Per-Sentence* BLEU: average sentence-level BLEU of translations that are sampled and scored during the bandit learning pass. This metric represents average expert ratings, which we want to optimize for in real-world scenarios. We also measure *Heldout* BLEU: corpus-level BLEU on an unseen test set, where translations are greedily decoded by the NMT models. This shows how much our method improves translation quality, since corpus-level BLEU correlates better with human judgments than

[6]Over 95% of the bandit learning set's sentences are seen during supervised learning. Performance gain on this set mainly reflects how well a model leverages weak learning signals (ratings) to improve previously made predictions. Generalizability is measured by performance gain on the test sets, which do not overlap the training sets.

sentence-level BLEU.

Because of randomness due to both the random sampling in the model for "exploration" as well as the randomness in the reward function, we repeat each experiment five times and report the mean results with 95% confidence intervals.

3.1.5.2 Model configuration

Both the NMT model and the critic model are encoder-decoder models with global attention (Luong et al., 2015). The encoder and the decoder are unidirectional single-layer LSTMs. They have the same word embedding size and LSTM hidden size of 500. The source and target vocabulary sizes are both 50K. We do not use dropout in our experiments. We train our models by the Adam optimizer (Kingma and Ba, 2015a) with $\beta_1 = 0.9, \beta_2 = 0.999$ and a batch size of 64. For Adam's α hyperparameter, we use 10^{-3} during pre-training and 10^{-4} during bandit learning (for both the NMT model and the critic model). During pre-training, starting from the fifth pass, we decay α by a factor of 0.5 when perplexity on the development set increases. The NMT model reaches its highest corpus-level BLEU on the development set after ten passes through the supervised training data, while the critic model's training error stabilizes after five passes. The training speed is 18s/batch for supervised pre-training and 41s/batch for training with the NED-A2C algorithm.

3.1.6 Results and Analysis

In this section, we describe the results of our experiments, broken into the following questions: how NED-A2C improves reference models (§ 3.1.6.1); the effect the three perturbation

Table 3.2: Translation scores and improvements based on a single round of un-perturbed bandit feedback. Per-Sentence BLEU and Heldout BLEU are not comparable: the former is sentence-BLEU, the latter is corpus-BLEU.

	De-En			Zh-En		
	Reference	Δ_{sup}	Δ_{A2C}	Reference	Δ_{sup}	Δ_{A2C}
Fully pre-trained reference model						
Per-Sentence BLEU	38.26 ± 0.02	0.07 ± 0.05	**2.82 ± 0.03**	32.79 ± 0.01	0.36 ± 0.05	**1.08 ± 0.03**
Heldout BLEU	24.94 ± 0.00	1.48 ± 0.00	**1.82 ± 0.08**	13.73 ± 0.00	**1.18 ± 0.00**	0.86 ± 0.11
Weakly pre-trained reference model						
Per-Sentence BLEU	19.15 ± 0.01	2.94 ± 0.02	**7.07 ± 0.06**	14.77 ± 0.01	1.11 ± 0.02	**3.60 ± 0.04**
Heldout BLEU	19.63 ± 0.00	**3.94 ± 0.00**	1.61 ± 0.17	9.34 ± 0.00	**2.31 ± 0.00**	0.92 ± 0.13

functions have on the algorithm (§ 3.1.6.2); and whether the algorithm improves a corpus-level metric that corresponds well with human judgments (§3.1.6.3).

3.1.6.1 Effectiveness of NED-A2C under Un-perturbed Bandit Feedback

We evaluate our method in an ideal setting where *un-perturbed* Per-Sentence BLEU simulates ratings during both training and evaluation (Table 3.2).

Single round of feedback. In this setting, our models only observe each source sentence once and before producing its translation. On both De-En and Zh-En, NED-A2C improves Per-Sentence BLEU of reference models after only a single pass (+2.82 and +1.08 respectively).

Poor initialization. Policy gradient algorithms have difficulty improving from poor initializations, especially on problems with a large action space, because they use model-based exploration, which is ineffective when most actions have equal probabilities (Bahdanau et al., 2017; Ranzato et al., 2016). To see whether NED-A2C has this problem, we repeat the experiment with the same setup but with reference models pre-trained for only a single pass. Surprisingly, NED-

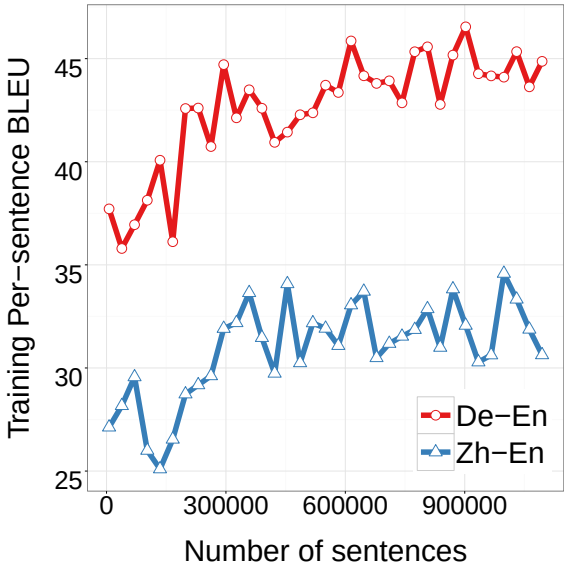


Figure 3.4: Learning curves of models trained with NED-A2C for five epochs.

A2C is highly effective at improving these poorly trained models (+7.07 on De-En and +3.60 on Zh-En in Per-Sentence BLEU).

Comparisons with supervised learning. To further demonstrate the effectiveness of NED-A2C, we compare it with training the reference models with supervised learning for a single pass on the bandit training set. Surprisingly, observing ground-truth translations barely improves the models in Per-Sentence BLEU when they are fully trained (less than +0.4 on both tasks). A possible explanation is that the models have already reached full capacity and do not benefit from more examples.[7] NED-A2C further enhances the models because it eliminates the mismatch between the supervised training objective and the evaluation objective. On weakly trained reference models, NED-A2C also significantly outperforms supervised learning (ΔPer-Sentence BLEU of NED-A2C is over three times as large as those of supervised learning).

[7]This result may vary if the domains of the supervised learning set and the bandit training set are dissimilar. Our training data are all TED talks.

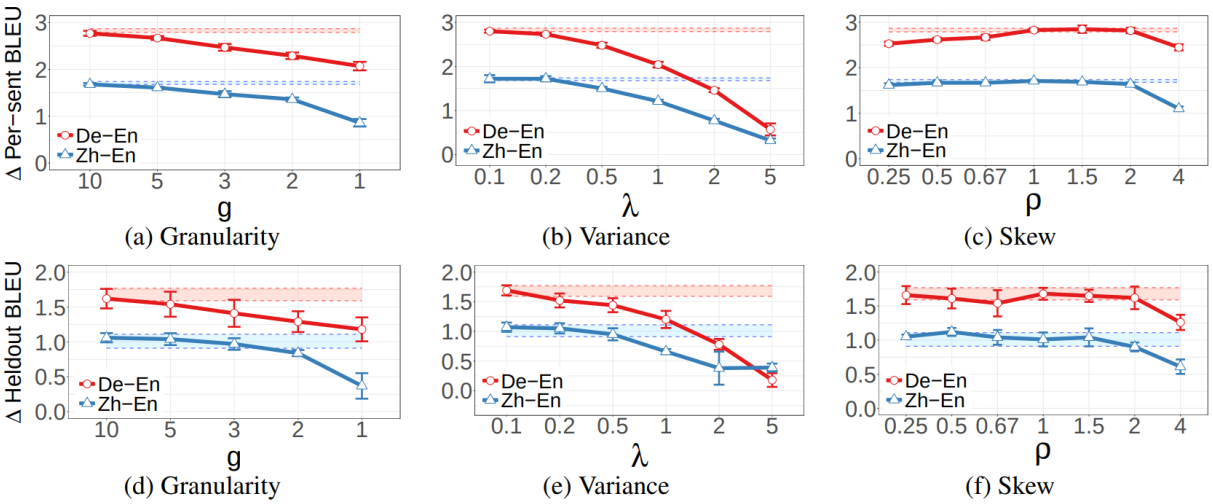


Figure 3.5: Performance gains of NMT models trained with NED-A2C in Per-Sentence BLEU (top row) and in Heldout BLEU (bottom row) under various degrees of granularity, variance, and skew of scores. Performance gains of models trained with un-perturbed scores are within the shaded regions.

Multiple rounds of feedback. We examine if NED-A2C can improve the models even further with multiple rounds of feedback.[8] With supervised learning, the models can memorize the reference translations but, in this case, the models have to be able to exploit and explore effectively. We train the models with NED-A2C for five passes and observe a much more significant ΔPer-Sentence BLEU than training for a single pass in both pairs of language (+6.73 on De-En and +4.56 on Zh-En) (Figure 3.4).

3.1.6.2 Effect of Perturbed Bandit Feedback

We apply perturbation functions defined in §3.1.4.1 to Per-Sentence BLEU scores and use the perturbed scores as rewards during bandit training (Figure 3.5).

Granular Rewards. We discretize raw Per-Sentence BLEU scores using $\text{pert}^{gran}(s; g)$ (§3.1.4.1).

[8]The ability to receive feedback on the same example multiple times might not fit all use cases though.

We vary g from one to ten (number of bins varies from two to eleven). Compared to continuous rewards, for both pairs of languages, ΔPer-Sentence BLEU is not affected with g at least five (at least six bins). As granularity decreases, ΔPer-Sentence BLEU monotonically degrades. However, even when $g = 1$ (scores are either 0 or 1), the models still improve by at least a point.

High-variance Rewards. We simulate noisy rewards using the model of human rating variance $\text{pert}^{var}(s; \lambda)$ (§ 3.1.4.2) with $\lambda \in \{0.1, 0.2, 0.5, 1, 2, 5\}$. Our models can withstand an amount of about 20% the variance in our human eval data without dropping in ΔPer-Sentence BLEU. When the amount of variance attains 100%, matching the amount of variance in the human data, ΔPer-Sentence BLEU go down by about 30% for both pairs of languages. As more variance is injected, the models degrade quickly but still improve from the pre-trained models. Variance is the most detrimental type of perturbation to NED-A2C among the three aspects of human ratings we model.

Skewed Rewards. We model skewed raters using $\text{pert}^{skew}(s; \rho)$ (§ 3.1.4.3) with $\rho \in \{0.25, 0.5, 0.67, 1, 1.5, 2, 4\}$. NED-A2C is robust to skewed scores. ΔPer-Sentence BLEU is at least 90% of unskewed scores for most skew values. Only when the scores are extremely harsh ($\rho = 4$) does ΔPer-Sentence BLEU degrade significantly (most dramatically by 35% on Zh-En). At that degree of skew, a score of 0.3 is suppressed to be less than 0.08, giving little signal for the models to learn from. On the other spectrum, the models are less sensitive to motivating scores as Per-Sentence BLEU is unaffected on Zh-En and only decreases by 7% on De-En.

3.1.6.3 Held-out Translation Quality

Our method also improves pre-trained models in Heldout BLEU, a metric that correlates with translation quality better than Per-Sentence BLEU (Table 3.2). When scores are perturbed by our rating model, we observe similar patterns as with Per-Sentence BLEU: the models are robust to most perturbations except when scores are very coarse, or very harsh, or have very high variance (Figure 3.5, second row). Supervised learning improves Heldout BLEU better, possibly because maximizing log-likelihood of reference translations correlates more strongly with maximizing Heldout BLEU of predicted translations than maximizing Per-Sentence BLEU of predicted translations.

3.1.7 Related Work and Discussion

Ratings provided by humans can be used as effective learning signals for machines. Reinforcement learning has become the *de facto* standard for incorporating this feedback across diverse tasks such as robot voice control (Tenorio-Gonzalez et al., 2010), myoelectric control (Pilarski et al., 2011), and virtual assistants (Isbell et al., 2001). Recently, this learning framework has been combined with recurrent neural networks to solve machine translation (Bahdanau et al., 2017), dialogue generation (Li et al., 2016), neural architecture search (Zoph and Le, 2017), and device placement (Mirhoseini et al., 2017). Other approaches to more general structured prediction under bandit feedback (Chang et al., 2015c; Sokolov et al., 2016a,b) show the broader efficacy of this framework. Ranzato et al. (2016) describe MIXER for training neural encoder-decoder models, which is a reinforcement learning approach closely related to ours but requires a policy-mixing strategy and only uses a linear critic model. Among work on bandit MT, ours

is closest to Kreutzer et al. (2017), which also tackle this problem using neural encoder-decoder models, but we (a) take advantage of a state-of-the-art reinforcement learning method; (b) devise a strategy to simulate noisy rewards; and (c) demonstrate the robustness of our method on noisy simulated rewards.

Our results show that bandit feedback can be an effective feedback mechanism for neural machine translation systems. This is *despite* that errors in human annotations hurt machine learning models in many NLP tasks (Snow et al., 2008). An obvious question is whether we could extend our framework to model individual annotator preferences (Passonneau and Carpenter, 2014) or learn personalized models (Mirkin et al., 2015; Rabinovich et al., 2017), and handle heteroscedastic noise (Antos et al., 2010; Kersting et al., 2007; Park, 1966). Another direction is to apply active learning techniques to reduce the sample complexity required to improve the systems or to extend to richer action spaces for problems like simultaneous translation, which requires prediction (Grissom II et al., 2014) and reordering (He et al., 2015) among other strategies to both minimize delay and effectively translate a sentence (He et al., 2016a).

3.2 ILIAD: Interactive Learning from Activity Descriptions

3.2.1 Overview

The goal of a *request-fulfilling* agent is to map a given request in a situated environment to an execution that accomplishes the intent of the request (Anderson et al., 2018b; Artzi and Zettlemoyer, 2013; Chen and Mooney, 2011; Chen et al., 2019b; Misra et al., 2017a; Nguyen and Daumé III, 2019; Nguyen et al., 2019b; Tellex et al., 2012; Winograd, 1972). Request-fulfilling agents have been typically trained using *non-verbal* interactive learning protocols such

Table 3.3: Trade-offs between the *learning* effort of the agent and the teacher in three learning protocols. Each protocol employs a different medium for the teacher to convey feedback. If a medium is not natural to the teacher (e.g., IL-style demonstration), it must learn to express feedback using that medium (*teacher communication-learning effort*). For example, in IL, to provide demonstrations, the teacher must learn to control the agent to accomplish tasks. Similarly, if a medium is not natural to the agent (e.g., human language), it needs to learn to interpret feedback (*agent communication-learning effort*). The agent also learns tasks from information decoded from feedback (*agent task-learning effort*). The qualitative claims about the "agent learning effort" column summarize our empirical findings about the learning efficiency of algorithms that implement these protocols (Table 3.4).

		Learning effort	
Protocol	Feedback medium	Teacher (communication learning)	Agent (comm. & task learning)
IL	Demonstration	Highest	Lowest
RL	Scalar reward	None	Highest
ILIAD	Description	None	Medium

as imitation learning (IL) which assumes labeled executions as feedback (Anderson et al., 2018b; Mei et al., 2016; Yao et al., 2020), or reinforcement learning (RL) which uses scalar rewards as feedback (Chaplot et al., 2018; Hermann et al., 2017a). These protocols are suitable for training agents with pre-collected datasets or in simulators, but they do not lend themselves easily to training by human teachers that only possess domain knowledge, but might not be able to precisely define the reward function, or provide direct demonstrations. To enable training by such teachers, we introduce a verbal interactive learning protocol called ILIAD: **I**nteractive **L**earning from **A**ctivity **D**escription, where feedback is limited to *descriptions of activities*, in a language that is appropriate for a given teacher (e.g., a natural language for humans).

Figure 3.6 illustrates an example of training an agent using the ILIAD protocol. Learning proceeds in episodes of interaction between a learning agent and a teacher. In each episode, the agent is presented with a request, provided in the teacher's description language, and takes a se-

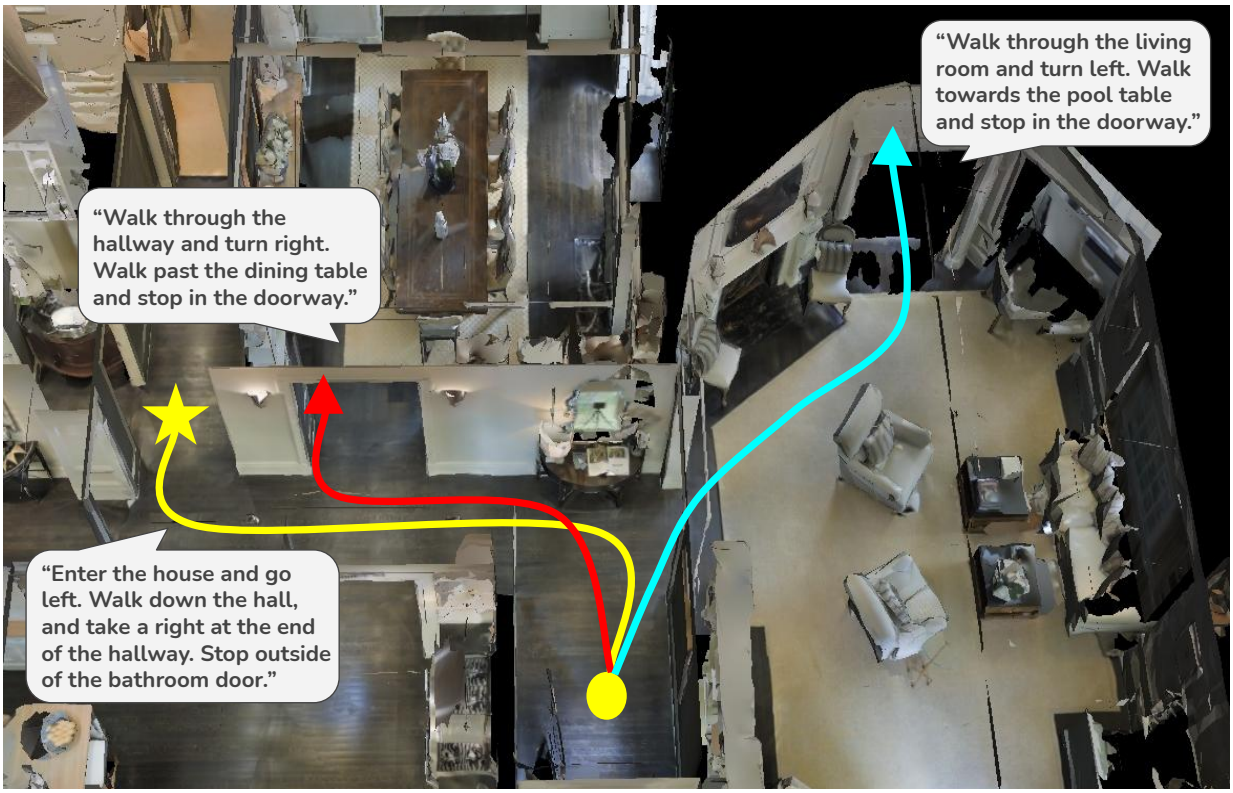


Figure 3.6: A real example of training an agent to fulfill a navigation request in a 3D environment (Anderson et al., 2018b) using ADEL, our implementation of the ILIAD protocol. The agent receives a request *“Enter the house..."* which implies the path. Initially, it wanders far from the goal because it does not understand language. Its execution () is described as *“Walk through the living room..."*. To ground language, the agent learns to generate the path conditioned on the description. After a number of interactions, its execution () is closer to the optimal path. As this process iterates, the agent learns to ground diverse descriptions to executions and can execute requests more precisely.

quence of actions in the environment to execute it. After an execution is completed, the teacher provides the agent with a description of the execution, in the same description language. The agent then uses this feedback to update its policy.

The agent receives *no other* feedback such as ground-truth demonstration (Mei et al., 2016), scalar reward (Hermann et al., 2017a), or constraint (Miryoosefi et al., 2019). Essentially, ILIAD presents a setting where task learning is enabled by *grounded* language learning: the agent improves its request-fulfilling capability by exploring the description language and learning to

ground the language to executions. This aspect distinguish ILIAD from IL or RL, where task learning is made possible by imitating actions or maximizing rewards.

The ILIAD protocol leaves two open problems: (a) *the exploration problem*: how to generate executions that elicit useful descriptions from the teacher and (b) *the grounding problem*: how to effectively ground descriptions to executions. We develop an algorithm named ADEL: **A**ctivity-**D**escription **E**xplorative **L**earner that offers practical solutions to these problems. For (a), we devise a semi-supervised execution sampling scheme that efficiently explores the description language space. For (b), we employ maximum likelihood to learn a mapping from descriptions to executions. We show that our algorithm can be viewed as density estimation, and prove its convergence in the contextual bandit setting (Langford and Zhang, 2008b), i.e., when the task horizon is 1.

Our paper does *not* argue for the primacy of one learning protocol over the others. In fact, an important point we raise is that there are multiple, possibly competing metrics for comparing learning protocols. We focus on highlighting the *complementary* advantages of ILIAD against IL and RL (Table 3.3). In all of these protocols, the agent and the teacher establish a communication channel that allows the teacher to encode feedback and send it to the agent. At one extreme, IL uses demonstration, an *agent-specific* medium, to encode feedback, thus placing the burden of establishing the communication channel entirely on the teacher. Concretely, in standard interactive IL (e.g., Ross et al., 2011), a demonstration can contain only actions in the agent's action space. Therefore, this protocol implicitly assumes that the teacher must be familiar with the agent's control interface. In practice, non-experts may have to spend substantial effort in order to learn to control an agent.[9] In these settings, the agent usually learns from relatively few demonstra-

[9]Third-person or observational IL (Stadie et al., 2017; Sun et al., 2019b) allows the teacher to demonstrate

tions because it does not have to learn to interpret feedback, and the feedback directly specifies the desired behavior. At another extreme, we have RL and ILIAD, where the teacher provides feedback via *agent-agnostic* media (reward and language, respectively). RL eliminates the agent communication-learning effort by hard-coding the semantics of scalar rewards into the learning algorithm.[10] But the trade-off of using such limited feedback is that the task-learning effort of the agent increases; state-of-the-art RL algorithms are notorious for their high sample complexity (Chaplot et al., 2018; Chevalier-Boisvert et al., 2019; Hermann et al., 2017a). By employing a natural and expressive medium like natural language, ILIAD offers a compromise between RL and IL: it can be more sample-efficient than RL while not requiring the teacher to master the agent's control interface as IL does. Overall, no protocol is superior in all metrics and the choice of protocol depends on users' preferences.

We empirically evaluate ADEL against IL and RL baselines on two tasks: vision-language navigation (Anderson et al., 2018b), and word-modification via regular expressions (Andreas et al., 2018). Our results show that ADEL significantly outperforms RL baselines in terms of both sample efficiency and quality of the learnt policies. Also, ADEL's success rate is competitive with those of the IL baselines on the navigation task and is lower by 4% on the word modification task. It takes approximately 5-9 times more training episodes than the IL baselines to reach comparable success rates, which is quite respectable considering that the algorithm has to search in an exponentially large space for the ground-truth executions whereas the IL baselines are *given* these executions. Therefore, ADEL can be a preferred algorithm whenever annotating executions with correct (agent) actions is not feasible or is substantially more expensive than describing

tasks with their action space. However, this framework is *non-interactive* because the agent imitates pre-collected demonstrations and does not interact with a teacher. We consider interactive IL (Ross et al., 2011), which is shown to be more effective than non-interactive counterparts.

[10]By design, RL algorithms understand that higher reward value implies better performance.

executions in some description language. For example, in the word-modification task, ADEL teaches the agent without requiring a teacher with knowledge about regular expressions. We believe the capability of non-experts to provide feedback will make ADEL and more generally the ILIAD protocol a strong contender in many scenarios.

3.2.2 Problem Formulation

Environment We borrow our terminology from the reinforcement learning (RL) literature (Sutton and Barto, 2018). We consider an agent acting in an environment with state space $\mathcal{S}$, action space $\mathcal{A}$, and transition function $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, where $\Delta(\mathcal{S})$ denotes the space of all probability distributions over $\mathcal{S}$. Let $\mathcal{R} = \{R : \mathcal{S} \times \mathcal{A} \to [0, 1]\}$ be a set of reward functions. A *task* in the environment is defined by a tuple $(R, s_1, d^\star)$, where $R \in \mathcal{R}$ is the task's reward function, $s_1 \in \mathcal{S}$ is the start state, and $d^\star \in \mathcal{D}$ is the task's (language) request. Here, $\mathcal{D}$ is the set of all nonempty strings generated from a finite vocabulary. The agent only has access to the start state and the task request; the reward function is only used for evaluation. For example, in robot navigation, a task is given by a start location, a task request like "*go to the kitchen*", and a reward function that measures the distance from a current location to the kitchen.

Execution Episode At the beginning of an episode, a task $q = (R, s_1, d^\star)$ is sampled from a task distribution $\mathbb{P}^\star(q)$. The agent starts in s_1 and is presented with $d^\star$ but *does not* observe R or any rewards generated by it. The agent maintains a *request-conditioned policy* $\pi_\theta : \mathcal{S} \times \mathcal{D} \to \Delta(\mathcal{A})$ with parameters θ, which takes in a state $s \in \mathcal{S}$ and a request $d \in \mathcal{D}$, and outputs a probability distribution over $\mathcal{A}$. Using this policy, it can generate an *execution* $\hat{e} = (s_1, \hat{a}_1, s_2, \cdots, s_H, \hat{a}_H)$, where H is the task horizon (the time limit), $\hat{a}_i \sim \pi_\theta(\cdot \mid s_i, d^\star)$ and $s_{i+1} \sim T(\cdot \mid s_i, \hat{a}_i)$ for

Algorithm 3 ILIAD protocol. Details of line 4 and line 6 are left to specific implementations.

1: Initialize agent policy $\pi_\theta : \mathcal{S} \times \mathcal{D} \to \Delta(\mathcal{A})$
2: **for** $n = 1, 2, \cdots, N$ **do**
3: World samples a task $q = (R, s_1, d^\star) \sim \mathbb{P}^\star(\cdot)$
4: Agent generates an execution $\hat{e}$ given s_1, $d^\star$, and π_θ
5: Teacher generates a description $\hat{d} \sim \mathbb{P}_T(\cdot \mid \hat{e})$
6: Agent uses $(d^\star, \hat{e}, \hat{d})$ to update π_θ
return π_θ

every i. Throughout the paper, we will use the notation $e \sim \mathbb{P}_\pi(\cdot \mid s_1, d)$ to denote sampling an execution e by following policy π given a start state s_1 and a request d. The objective of the agent is to find a policy π with maximum value, where we define the policy value $V(\pi)$ as:

$$V(\pi) = \mathbb{E}_{q \sim \mathbb{P}^\star(\cdot), \hat{e} \sim \mathbb{P}_\pi(\cdot \mid s_1, d^\star)} \left[\sum_{i=1}^{H} R(s_i, \hat{a}_i) \right] \tag{3.11}$$

ILIAD protocol Alg 3 describes the ILIAD protocol for training a request-fulfilling agent. It consists of a series of N training episodes. Each episode starts with sampling a task $q = (R, s_1, d^\star)$ from $\mathbb{P}^\star$. The agent then generates an execution $\hat{e}$ given s_1, $d^\star$, and its policy π_θ (line 4). The feedback mechanism in ILIAD is provided by a *teacher* that can describe executions in a description language. The teacher is modeled by a fixed distribution $\mathbb{P}_T : (\mathcal{S} \times \mathcal{A})^H \to \Delta(\mathcal{D})$, where $(\mathcal{S} \times \mathcal{A})^H$ is the space over H-step executions. After generating $\hat{e}$, the agent sends it to the teacher and receives a *description* of $\hat{e}$, which is a sample $\hat{d} \sim \mathbb{P}_T(\cdot \mid \hat{e})$ (line 5). Finally, the agent uses the triplet $(d^\star, \hat{e}, \hat{d})$ to update its policy for the next round (line 6). Crucially, the agent *never* receives any other feedback, including rewards, demonstrations, constraints, or direct knowledge of the *latent* reward function. Any algorithm implementing the ILIAD protocol has to decide how to generate executions (the exploration problem, line 4) and how to update the agent policy (the grounding problem, line 6). The protocol does not provide any constraints for these decisions.

Consistency of the teacher In order for the agent to learn to execute requests by grounding the description language, we require that the description language is similar to the request language. Formally, we define the ground-truth joint distribution over tasks and executions as follows

$$\mathbb{P}^{\star}\left(e, R, s_1, d\right) = \mathbb{P}_{\pi^{\star}}\left(e \mid s_1, d\right) \mathbb{P}^{\star}\left(R, s_1, d\right) \tag{3.12}$$

where $\pi^{\star}$ is an optimal policy that maximizes Eq 3.11. From this joint distribution, we derive the ground-truth execution-conditioned distribution over requests $\mathbb{P}^{\star}(d \mid e)$. This distribution specifies the probability that a request d can serve as a valid description of an execution e.

We expect that if the teacher's distribution $\mathbb{P}_T(d \mid e)$ is close to $\mathbb{P}^{\star}(d \mid e)$ then grounding the description language to executions will help with request fulfilling. In that case, the agent can treat a description of an execution as a request that is fulfilled by that execution. Therefore, the description-execution pairs $(\hat{d}, \hat{e})$ can be used as supervised-learning examples for the request-fulfilling problem.

The learning process can be sped up if the agent is able to exploit the compositionality of language. For example, if a request is "*turn right, walk to the kitchen*" and the agent's execution is described as "*turn right, walk to the bedroom*", the agent may not have successfully fulfilled the task but it can learn what "*turn right*" and "*walk to*" mean through the description. Later, it may learn to recognize "*kitchen*" through a description like "*go to the kitchen*" and compose that knowledge with its understanding of "*walk to*" to better execute "*walk to the kitchen*".

3.2.3 The ADEL algorithm

We frame the ILIAD problem as a density-estimation problem: *given that we can effectively draw samples from the distribution* $\mathbb{P}^\star(s_1, d)$ *and a teacher* $\mathbb{P}_T(d \mid e)$*, how do we learn a policy* π_θ *such that* $\mathbb{P}_{\pi_\theta}(e \mid s_1, d)$ *is close to* $\mathbb{P}^\star(e \mid s_1, d)$*?* Here, $\mathbb{P}^\star(e \mid s_1, d) = \mathbb{P}_{\pi^\star}(e \mid s_1, d)$ is the ground-truth request-fulfilling distribution obtained from the joint distribution defined in Eq 3.12.

If s_1 is not the start state of e, then $\mathbb{P}^\star(e \mid s_1, d) = 0$. Otherwise, by applying Bayes' rule, and noting that s_1 is included in e, we have:

$$\begin{aligned}\mathbb{P}^\star(e \mid s_1, d) &\propto \mathbb{P}^\star(e, d \mid s_1) = \mathbb{P}^\star(e \mid s_1)\mathbb{P}^\star(d \mid e, s_1),\\ &= \mathbb{P}^\star(e \mid s_1)\mathbb{P}^\star(d \mid e),\\ &\approx \mathbb{P}^\star(e \mid s_1)\mathbb{P}_T(d \mid e). \end{aligned} \tag{3.13}$$

As seen from the equation, the only missing piece required for estimating $\mathbb{P}^\star(e \mid s_1, d)$ is the marginal[11] $\mathbb{P}^\star(e \mid s_1)$. Alg 4 presents a simple method for learning an agent policy if we have access to this marginal. It is easy to show that the pairs $(\hat{e}, \hat{d})$ in the algorithm are approximately drawn from the joint distribution $\mathbb{P}^\star(e, d \mid s_1)$ and thus can be directly used to estimate the conditional $\mathbb{P}^\star(e \mid s_1, d)$.

Unfortunately, $\mathbb{P}^\star(e \mid s_1)$ is unknown in our setting. We present our main algorithm ADEL (Alg 5) which simultaneously estimates $\mathbb{P}^\star(e \mid s_1)$ and $\mathbb{P}^\star(e \mid s_1, d)$ through interactions with the teacher. In this algorithm, we assume access to an *approximate marginal* $\mathbb{P}_{\pi_\omega}(e \mid s_1)$ defined by

[11] We are largely concerned with the relationship between e and d, and so refer to the distribution $\mathbb{P}^\star(e \mid s_1)$ as the marginal and $\mathbb{P}^\star(e \mid s_1, d)$ as the conditional.

Algorithm 4 Simple algorithm for learning an agent's policy with access to the true marginal $\mathbb{P}^\star(e \mid s_1)$ and teacher $\mathbb{P}_T(d \mid e)$.

$\mathcal{B} = \emptyset$
for $i = 1, 2, \cdots, N$ **do**
    World samples a task $q = (R, s_1, d^\star) \sim \mathbb{P}^\star(\cdot)$
    Sample $(\hat{e}, \hat{d})$ as follows: $\hat{e} \sim \mathbb{P}^\star(\cdot \mid s_1), \hat{d} \sim \mathbb{P}_T(\cdot \mid \hat{e})$
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(\hat{e}, \hat{d})\}$
Train a policy $\pi_\theta(a \mid s, d)$ via maximum log-likelihood:
    $\max_\theta \sum_{(\hat{e},\hat{d})\in\mathcal{B}} \sum_{(s,\hat{a}_s)\in\hat{e}} \log \pi_\theta(\hat{a}_s \mid s, \hat{d})$
    where $\hat{a}_s$ is the action taken by the agent in state s
return π_θ

an *explorative policy* $\pi_\omega(a \mid s)$. This policy can be learned from a dataset of unlabeled executions or be defined as a program that synthesizes executions. In many applications, reasonable unlabeled executions can be cheaply constructed using knowledge about the structure of the execution. For example, in robot navigation, valid executions are collision-free and non-looping; in semantic parsing, predicted parses should follow the syntax of the semantic language.

After constructing the approximate marginal $\mathbb{P}_{\pi_\omega}(e \mid s_1)$, we could substitute it for the true marginal in Alg 4. However, using a fixed approximation of the marginal may lead to sample inefficiency when there is a mismatch between the approximate marginal and the true marginal. For example, in the robot navigation example, if most human requests specify the kitchen as the destination, the agent should focus on generating executions that end in the kitchen to obtain descriptions that are similar to those requests. If instead, a uniform approximate marginal is used to generate executions, the agent obtains a lot of irrelevant descriptions.

ADEL minimizes potential marginal mismatch by iteratively using the estimate of the marginal $\mathbb{P}^\star(e \mid s_1)$ to improve the estimate of the conditional $\mathbb{P}^\star(e \mid s_1, d)$ and vice versa. Initially, we set $\mathbb{P}_{\pi_\omega}(e \mid s_1)$ as the marginal over executions. In each episode, we *mix* this distribution with $\mathbb{P}_{\pi_\theta}(e \mid s_1, d)$, the current estimate of the conditional, to obtain an improved estimate of the

Algorithm 5 ADEL: our implementation of the ILIAD protocol.

1: **Input**: teacher $\mathbb{P}_T(d|e)$, approximate marginal $\mathbb{P}_{\pi_\omega}(e|s_1)$, mixing weight $\lambda \in [0,1]$, annealing rate $\beta \in (0,1)$
2: Initialize $\pi_\theta : \mathcal{S} \times \mathcal{D} \to \Delta(\mathcal{A})$ and $\mathcal{B} = \emptyset$
3: **for** $n = 1, 2, \cdots, N$ **do**
4: World samples a task $q = (R, s_1, d^\star) \sim \mathbb{P}^\star(\cdot)$
5: Agent generates $\hat{e} \sim \tilde{\mathbb{P}}(\cdot \mid s_1, d^\star)$ (see Eq 3.14)
6: Teacher generates a description $\hat{d} \sim \mathbb{P}_T(\cdot \mid \hat{e})$
7: $\mathcal{B} \leftarrow \mathcal{B} \cup (\hat{e}, \hat{d})$
8: Update agent policy:

$$\theta \leftarrow \max_{\theta'} \sum_{(\hat{e},\hat{d}) \in \mathcal{B}} \sum_{(s,\hat{a}_s) \in \hat{e}} \log \pi_{\theta'}(\hat{a}_s \mid s, \hat{d})$$

where $\hat{a}_s$ is the action taken by the agent in state s
9: Anneal mixing weight: $\lambda \leftarrow \lambda \cdot \beta$
10: **return** π_θ

marginal (line 5). Formally, given a start state s_1 and a request $d^\star$, we sample an execution $\hat{e}$ from the following distribution:

$$\tilde{\mathbb{P}}(\cdot \mid s_1, d^\star) \triangleq \lambda \mathbb{P}_{\pi_\omega}(\cdot \mid s_1) + (1-\lambda)\mathbb{P}_{\pi_\theta}(\cdot \mid s_1, d^\star) \tag{3.14}$$

where $\lambda \in [0,1]$ is a mixing weight that is annealed to zero over the course of training. Each component of the mixture in Eq 3.14 is essential in different learning stages. Mixing with $\mathbb{P}_{\pi_\omega}$ accelerates convergence at the early stage of learning. Later, when π_θ improves, $\mathbb{P}_{\pi_\theta}$ skews $\tilde{\mathbb{P}}$ towards executions whose descriptions are closer to the requests, closing the gap with $\mathbb{P}^\star(e \mid s_1)$. In line 6-8, similar to Alg 4, we leverage the (improved) marginal estimate and the teacher to draw samples $(\hat{e}, \hat{d})$ and use them to re-estimate $\mathbb{P}_{\pi_\theta}$.

Theoretical Analysis. We analyze an epoch-based variant of ADEL and show that under certain assumptions, it converges to a near-optimal policy. In this variant, we run the algorithm in epochs,

where the agent policy is only updated at the end of an epoch. In each epoch, we collect a fresh batch of examples $\{(\hat{e}, \hat{d})\}$ as in ADEL (line 4-7), and use them to perform a batch update (line 8). We provide a sketch of our theoretical results here and defer the full details to §B.

We consider the case of $H = 1$ where an execution $e = (s_1, a)$ consists of the start state s_1 and a single action a taken by the agent. This setting while restrictive captures the non-trivial class of contextual bandit problems (Langford and Zhang, 2008b). Sequential decision-making problems where the agent makes decisions solely based on the start state can be reduced to this setting by treating a sequence of decisions as a single action (Kreutzer et al., 2017; Nguyen et al., 2017). We focus on the convergence of the iterations of epochs, and assume that the maximum likelihood estimation problem in each epoch can be solved optimally. We also ablate the teacher learning difficulty by assuming access to a perfectly consistent teacher, i.e., $\mathbb{P}_T(d \mid e) = \mathbb{P}^\star(d \mid e)$.

We make two crucial assumptions. Firstly, we make a standard realizability assumption to ensure that our policy class is expressive enough to accommodate the optimal solution of the maximum likelihood estimation. Secondly, we assume that for every start state s_1, the teacher distribution's matrix $\mathbb{P}^\star(d \mid e_{s_1})$ over descriptions and executions e_{s_1} starting with s_1, has a non-zero minimum singular value $\sigma_{min}(s_1)$. Intuitively, this assumption implies that descriptions are rich enough to help in deciphering actions. Under these assumptions, we prove the following result:

Theorem 1 (Main Result)**.** *Let $\mathbb{P}_n(e \mid s_1)$ be the marginal distribution in the n^{th} epoch. Then for*

any $t \in \mathbb{N}$ *and any start state* s_1 *we have:*

$$\|\mathbb{P}^\star(e \mid s_1) - \frac{1}{t}\sum_{n=1}^{t}\mathbb{P}_n(e \mid s_1)\|_2 \leq \frac{1}{\sigma_{min}(s_1)}\sqrt{\frac{2\ln|\mathcal{A}|}{t}}.$$

Theorem 1 shows that the running average of the estimated marginal distribution converges to the true marginal distribution. The error bound depends logarithmically on the size of action space, and therefore, suitable for problems with exponentially large action space. As argued before, access to the true marginal can be used to easily learn a near-optimal policy. For brevity, we defer the proof and other details to §B. Hence, our results show that under certain conditions, we can expect convergence to the optimal policy. We leave the question of sample complexity and addressing more general settings for future work.

3.2.4 Experimental Setup

In this section, we present a general method for simulating an execution-describing teacher using a pre-collected dataset (§3.2.4.1). Then we describe setups of the two problems we conduct experiments on: vision-language navigation (§3.2.4.2) and word modification (§3.2.4.3). Details about the data, the model architecture, training hyperparameters, and how the teacher is simulated in each problem are in the Appendix.

We emphasize that the ILIAD protocol or the ADEL algorithm do *not* propose learning a teacher. Similar to IL and RL, ILIAD operates with a fixed, black-box teacher that is given in the environment. Our experiments specifically simulate *human* teachers that train request-fulfilling agents by talking to them (using descriptions). We use labeled executions only to learn approximate models of human teachers.

3.2.4.1 Simulating Teachers

ILIAD assumes access to a teacher $\mathbb{P}_T(d \mid e)$ that can describe agent executions in a description language. For our experimental purposes, employing human teachers is expensive and irreproducible, thus we simulate them using pre-collected datasets. We assume availability of a dataset $\mathcal{B}_{\text{sim}} = \{(\mathcal{D}_n^\star, e_n^\star)\}_{n=1}^N$, where $\mathcal{D}_n^\star = \{d_n^{\star(j)}\}_{j=1}^M$ contains M human-generated requests that are fulfilled by execution $e_n^\star$. Each of the two experimented problems is accompanied by data that is partitioned into training/validation/test splits. We use the training split as $\mathcal{B}_{\text{sim}}$ and use the other two splits for validation and testing, respectively. Our agents do *not* have direct access to $\mathcal{B}_{\text{sim}}$. From an agent's perspective, it communicates with a black-box teacher that can return descriptions of its executions; it does not know how the teacher is implemented.

Each ILIAD episode (Alg 3) requires providing a request $d^\star$ at the beginning and a description $\hat{d}$ of an execution $\hat{e}$. The request $d^\star$ is chosen by first uniformly randomly selecting an example $(\mathcal{D}_n^\star, e_n^\star)$ from $\mathcal{B}_{\text{sim}}$, and then uniformly sampling a request $d_n^{\star(j)}$ from $\mathcal{D}_n^\star$. The description $\hat{d}$ is generated as follows. We first gather all the pairs $(d_n^{\star(j)}, e_n^\star)$ from $\mathcal{B}_{\text{sim}}$ and train an RNN-based conditional language model $\tilde{\mathbb{P}}_T(d \mid e)$ via standard maximum log-likelihood. We can then generate a description of an execution $\hat{e}$ by greedily decoding[12] this model conditioned on $\hat{e}$: $\hat{d}_{\text{greedy}} = \texttt{greedy}\big(\tilde{\mathbb{P}}_T(\cdot \mid \hat{e})\big)$. However, given limited training data, this model may not generate sufficiently high-quality descriptions. We apply two techniques to improve the quality of the descriptions. First, we provide the agent with the human-generated requests in $\mathcal{B}_{\text{sim}}$ when the executions are near optimal. Let $\texttt{perf}\,(\hat{e}, e^\star)$[13] be a performance metric that evaluates an

[12]Greedily decoding an RNN-based model refers to stepwise choosing the highest-probability class of the output softmax. In this case, the classes are words in the description vocabulary.

[13]The metric `perf` is only used in simulating the teachers and is not necessarily the same as the reward function R.

agent's execution $\hat{e}$ against a ground-truth $e^\star$ (higher is better). An execution $\hat{e}$ is near optimal if $\texttt{perf}\left(\hat{e}, e^\star\right) \geq \tau$, where τ is a constant threshold. Second, we apply pragmatic inference (Andreas and Klein, 2016; Fried et al., 2018a), leveraging the fact that the teacher has access to the environment's simulator and can simulate executions of descriptions. The final description given to the agent is

$$\hat{d} \sim \begin{cases} \texttt{Unif}\left(\mathcal{D}_n^\star\right) & \text{if } \texttt{perf}\left(\hat{e}, e_n^\star\right) \geq \tau, \\ \texttt{Unif}\left(\mathcal{D}_{\text{prag}} \cup \{\emptyset\}\right) & \text{otherwise} \end{cases} \tag{3.15}$$

where $\texttt{Unif}(\mathcal{D})$ is a uniform distribution over elements of $\mathcal{D}$, $e_n^\star$ is the ground-truth execution associated with $\mathcal{D}_n^\star$, $\mathcal{D}_{\text{prag}}$ contains descriptions generated using pragmatic inference (which we will describe next), and $\emptyset$ is the empty string.

Improved Descriptions with Pragmatic Inference. Pragmatic inference emulates the teacher's ability to mentally simulate task execution. Suppose the teacher has its own execution policy $\pi_T(a \mid s, d)$, which is learned using the pairs $\left(e_n^\star, d_n^{\star(j)}\right)$ of $\mathcal{B}_{\text{sim}}$, and access to a simulator of the environment. A *pragmatic* execution-describing teacher is defined as $\mathbb{P}_T^{\text{prag}}(d \mid e) \propto \mathbb{P}_{\pi_T}(e \mid s_1, d)$. For this teacher, the more likely that a request d causes it to generate an execution e, the more likely that it describes e as d.

In our problems, constructing the pragmatic teacher's distribution explicitly is not feasible because we would have to compute a normalizing constant that sums over all possible descriptions. Instead, we follow Andreas et al. (2018), generating a set of candidate descriptions and using $\mathbb{P}_{\pi_T}(e \mid s_1, d)$ to re-rank those candidates. Concretely, for every execution $\hat{e}$ where $\texttt{perf}\left(\hat{e}, e_n^\star\right) < \tau$, we use the learned language model $\tilde{\mathbb{P}}_T$ to generate a set of candidate descrip-

tions $\mathcal{D}_{\text{cand}} = \{\hat{d}_{\text{greedy}}\} \cup \{\hat{d}^{(k)}_{\text{sample}}\}_{k=1}^{K}$. This set consists of the greedily decoded description $\hat{d}_{\text{greedy}} = \texttt{greedy}\left(\tilde{\mathbb{P}}_T(\cdot \mid \hat{e})\right)$ and K descriptions $\hat{d}^{(k)}_{\text{sample}} \sim \tilde{\mathbb{P}}_T(\cdot \mid \hat{e})$. To construct $\mathcal{D}_{\text{prag}}$, we select descriptions in $\mathcal{D}_{\text{cand}}$ from which π_T generates executions that are similar enough to $\hat{e}$:

$$\mathcal{D}_{\text{prag}} = \left\{d \mid d \in \mathcal{D}_{\text{cand}} \wedge \texttt{perf}\left(e^d, \hat{e}\right) \geq \tau\right\} \tag{3.16}$$

where $e^d = \texttt{greedy}\left(\mathbb{P}_{\pi_T}(\cdot \mid s_1, d)\right)$ and s_1 is the start state of $\hat{e}$.

3.2.4.2 Vision-Language Navigation (NAV)

Problem and Environment. An agent executes natural language requests (given in English) by navigating to locations in environments that photo-realistically emulate residential buildings (Anderson et al., 2018b). The agent successfully fulfills a request if its final location is within three meters of the intended goal location. Navigation in an environment is framed as traversing in a graph where each node represents a location and each edge connects two nearby unobstructed locations. A state s of an agent represents its location and the direction it is facing. In the beginning, the agent starts in state s_1 and receives a navigation request $d^\star$. At every time step, the agent is not given the true state s but only receives an observation o, which is a real-world RGB image capturing the panoramic view at its current location.

Agent Policy. The agent maintains a policy $\pi_\theta\left(a \mid o, d\right)$ that takes in a current observation o and a request d, and outputs an action $a \in V_{\text{adj}}$, where V_{adj} denotes the set of locations that are adjacent to the agent's current location according to the environment graph. A special `<stop>` action is taken when the agent wants to terminate an episode or when it has taken H actions.

Simulated Teacher. We simulate a teacher that does not know how to control the navigation agent and thus cannot provide demonstrations. However, the teacher can verbally describe navigation paths taken by the agent. We follow §3.2.4.1, constructing a teacher $\mathbb{P}_T(d \mid e)$ that outputs language descriptions given executions $e = (o_1, a_1, \cdots, o_H)$.

3.2.4.3 Word Modification (REGEX)

Problem. A human gives an agent a natural language request (in English) $d^\star$ asking it to modify the characters of a word w^{inp}. The agent must execute the request and outputs a word $\hat{w}^{\text{out}}$. It successfully fulfills the request if $\hat{w}^{\text{out}}$ exactly matches the expected output w^{out}. For example, given an input word *embolden* and a request "*replace all n with c*", the expected output word is *emboldec*. We train an agent that solves this problem via a semantic parsing approach. Given w^{inp} and $d^\star$, the agent generates a regular expression $\hat{a}_{1:H} = (\hat{a}_1, \cdots, \hat{a}_H)$, which is a sequence of characters. It then uses a regular expression compiler to apply the regular expression onto the input word to produce an output word $\hat{w}^{\text{out}} = \texttt{compile}\left(w^{\text{inp}}, \hat{a}_{1:H}\right)$.

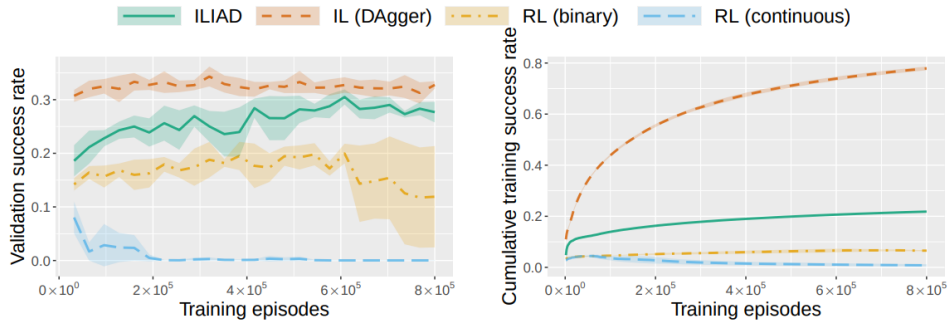
Agent Policy and Environment. The agent maintains a policy $\pi_\theta\left(a \mid s, d\right)$ that takes in a state s and a request d, and outputs a distribution over characters $a \in V_{\text{regex}}$, where V_{regex} is the regular expression (character) vocabulary. A special `<stop>` action is taken when the agent wants to stop generating the regular expression or when the regular expression exceeds the length limit H. We set the initial state $s_1 = \left(w^{\text{inp}}, \emptyset\right)$, where $\emptyset$ is the empty string. A next state is determined as

follows

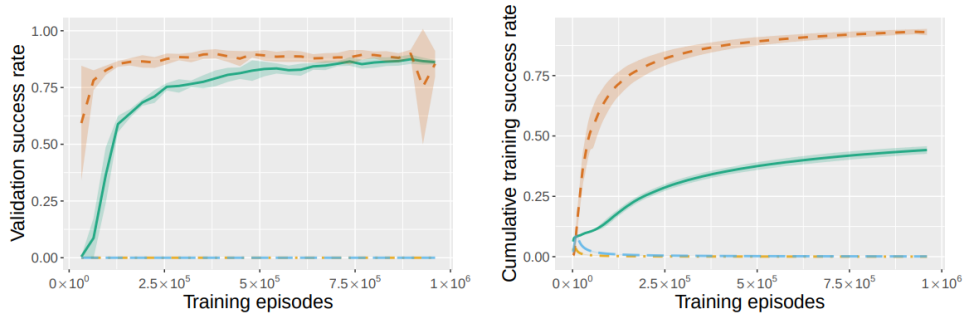
$$s_{t+1} = \begin{cases} (\hat{w}^{\text{out}}, \hat{a}_{1:t}) & \text{if } \hat{a}_t = \texttt{<stop>}, \\ \left(w^{\text{inp}}, \hat{a}_{1:t}\right) & \text{otherwise} \end{cases} \tag{3.17}$$

where $\hat{w}^{\text{out}} = \texttt{compile}\left(w^{\text{inp}}, \hat{a}_{1:t}\right)$.

Simulated Teacher. We simulate a teacher that does not have knowledge about regular expressions. Hence, instead of receiving full executions, which include regular expressions $\hat{a}_{1:H}$ predicted by the agent, the teacher generates descriptions given only pairs $(w_j^{\text{inp}}, \hat{w}_j^{\text{out}})$ of an input word w_j^{inp} and the corresponding output generated by the agent $\hat{w}_j^{\text{out}}$. In addition, to reduce ambiguity, the teacher requires multiple word pairs generate a description. This issue is better illustrated in the following example. Suppose the agent generates the pair *embolden* $\rightarrow$ *emboldec* by predicting a regular expression that corresponds to the description "*replace all n with c*". However, because the teacher does not observe the agent's regular expression (and cannot understand the expression even if it does), it can also describe the pair as "*replace the last letter with c*". Giving such a description to the agent would be problematic because the description does not correspond to the predicted regular expression. Observing multiple word pairs increases the chance that the teacher's description matches the agent's regular expression (e.g. adding *now* $\rightarrow$ *cow* help clarify that "*replace all n with c*" should be generated). In the end, the teacher is a model $P_T\big(d \;\big|\; \{w_j^{\text{inp}}, \hat{w}_j^{\text{out}}\}_{j=1}^{J}\big)$ that takes as input J word pairs. To generate J word pairs, in every episode, in addition to the episode's input word, we sample $J-1$ more words from the dictionary and execute the episode's request on the J words. We do *not* use any regular expression data in constructing the teacher. To train the teacher's policy π_T for pragmatic inference



(a) NAV



(b) REGEX

Figure 3.7: Validation success rate (average held-out return) and cumulative training success rate (average training return) over the course of training. For each algorithm, we report means and standard deviations over five runs with different random seeds.

(§3.2.4.1), we use a dataset that consists of tuples $\left(\mathcal{D}_n^\star, (w_n^{\text{inp}}, w_n^{\text{out}})\right)$ which are not annotated with ground-truth regular expressions. π_T directly generates an output word instead of predicting a regular expression like the agent policy π_θ.

3.2.4.4 Baselines and Evaluation Metrics

We compare interactive learning settings that employ different teaching media:

- *Learning from activity description* (ILIAD): the teacher returns a language description $\hat{d}$.
- *Imitation learning* (IL): the teacher demonstrates the correct actions in the states that the agent visited, returning $e^\star = (s_1, a_1^\star, \cdots, a_H^\star, s_H)$, where s_i are the states in the agent's

execution and $a_i^\star$ are the optimal actions in those states.

- *Reinforcement learning* (RL): the teacher provides a scalar reward that evaluates the agent's execution. We consider a special case when rewards are provided only at the end of an episode. Because such feedback is cheap to collect (e.g., star ratings) (Kreutzer et al., 2018b, 2020; Nguyen et al., 2017), this setting is suitable for large-scale applications. We experiment with both binary reward that indicates task success, and continuous reward that measures normalized distance to the goal (see §C.3).

We use ADEL in the ILIAD setting, DAgger (Ross et al., 2011) in IL, and REINFORCE[14] (Williams, 1992a) in RL. We report the *success rates* of these algorithms, which are the fractions of held-out (validation or test) examples on which the agent successfully fulfills its requests. All agents are initialized with random parameters.

3.2.5 Results

We compare the learning algorithms on not only success rate, but also the effort expended by the teacher. While task success rate is straightforward to compute, teacher effort is hard to quantify because it depends on many factors: the type of knowledge required to teach a task, the cognitive and physical ability of a teacher, etc. For example, in REGEX, providing demonstrations in forms of regular expressions may be easy for a computer science student, but could be challenging for someone who is unfamiliar with programming. In NAV, controlling a robot may not be viable for an individual with motor impairment, whereas generating language descriptions

[14]We use a moving-average baseline to reduce variance. We also experimented with A2C (Mnih et al., 2016) but it was less stable in this sparse-reward setting. At the time this paper was written, we were not aware of any work that successfully trained agents using RL without supervised-learning bootstrapping in the two problems we experimented on.

Table 3.4: Main results. We report means and standard deviations of success rates (%) over five runs with different random seeds. RL-Binary and RL-Cont refer to the RL settings with binary and continuous rewards, respectively. Sample complexity is the number of training episodes (or number of teacher responses) required to reach a validation success rate of at least c. Note that the teaching efforts are not comparable across the learning settings: providing a demonstration can be more or less tedious than providing a language description depending on various characteristics of the teacher. Hence, even though ADEL requires more episodes to reach the same performance as DAgger, we do not draw any conclusions about the primacy of one algorithm over the other in terms of teaching effort.

				Sample complexity ↓		
Learning setting	Algorithm	Val success rate (%) ↑	Test success rate (%) ↑	# Demonstrations	# Rewards	# Descriptions
Vision-language navigation				(c = 30.0%)		
IL	DAgger	35.6 ± 1.35	32.0 ± 1.63	45K ± 26K	-	-
RL-Binary	REINFORCE	22.4 ± 1.15	20.5 ± 0.58	-	$+\infty$	-
RL-Cont	REINFORCE	11.1 ± 2.19	11.3 ± 1.25	-	$+\infty$	-
ILIAD	ADEL	32.2 ± 0.97	31.9 ± 0.76	-	-	406K ± 31K
Word modification				(c = 85.0%)		
IL	DAgger	92.5 ± 0.53	93.0 ± 0.37	118K ± 16K	-	-
RL-Binary	REINFORCE	0.0 ± 0.00	0.0 ± 0.00	-	$+\infty$	-
RL-Cont	REINFORCE	0.0 ± 0.00	0.0 ± 0.00	-	$+\infty$	-
ILIAD	ADEL	88.1 ± 1.60	89.0 ± 1.30	-	-	573K ± 116K

may infeasible for someone with a verbal-communication disorder. Because it is not possible cover all teacher demographics, our goal is to *quantitatively* compare the learning algorithms on learning effectiveness and efficiency, and *qualitatively* compare them on the teacher effort to learn to express feedback using the protocol's communication medium. Our overall findings (Table 3.3) highlight the strengths and weaknesses of each learning algorithm and can potentially aid practitioners in selecting algorithms that best suit their applications.

Main results. Our main results are in Table 3.4. Overall, results in both problems match our expectations. The IL baseline achieves the highest success rates (on average, 35.6% on NAV and 92.5% on REGEX). This framework is most effective because the feedback directly specifies ground-truth actions. The RL baseline is unable to reach competitive success rates. Especially, in REGEX, the RL agent cannot learn the syntax of the regular expressions and completely fails

Table 3.5: Effects of mixing execution policies in ADEL.

Mixing weight	Val success rate (%) ↑	Sample complexity ↓
Vision-language navigation		
$\lambda = 0$ (no marginal)	0.0	$+\infty$
$\lambda = 1$	29.4	$+\infty$
$\lambda = 0.5$ (final)	32.0	384K
Word modification		
$\lambda = 0$ (no marginal)	0.2	$+\infty$
$\lambda = 1$	55.7	$+\infty$
$\lambda = 0.5$ (final)	88.0	608K

at test time. This shows that the reward feedback is not sufficiently informative to guide the agent to explore efficiently in this problem. ADEL's success rates are slightly lower than those of IL (3-4% lower than) but are substantially higher than those of RL (+9.8% on NAV and +88.1% on REGEX compared to the best RL results).

To measure learning efficiency, we report the number of training episodes required to reach a substantially high success rate (30% for NAV and 85% for REGEX). We observe that all algorithms require hundreds of thousands of episodes to attain those success rates. The RL agents cannot learn effectively even after collecting more than 1M responses from the teachers. ADEL attains reasonable success rates using 5-9 times more responses than IL. This is a decent efficiency considering that ADEL needs to find the ground-truth executions in exponentially large search spaces, while IL directly communicates these executions to the agents. As ADEL lacks access to ground-truth executions, its average training returns are 2-4 times lower than those of IL (Figure 3.7).

Ablation. We study the effects of mixing with the approximate marginal ($\mathbb{P}_{\pi_\omega}$) in ADEL (Table 3.5). First of all, we observe that learning cannot take off without using the approximate

marginal (λ = 0). On the other hand, using only the approximate marginal to generate executions ($\lambda = 1$) degrades performance, in terms of both success rate and sample efficiency. This effect is more visible on REGEX where the success rate drops by 33% (compared to a 3% drop in NAV), indicating that the gap between the approximate marginal and the true marginal is larger in REGEX than in NAV. This matches our expectation as the set of unlabeled executions that we generate to learn π_ω in REGEX covers a smaller portion of the problem's execution space than that in NAV. Finally, mixing the approximate marginal and the agent-estimated conditional ($\lambda = 0.5$) gives the best results.

3.2.6 Related Work

Learning from Language Feedback. Frameworks for learning from language-based communication have been previously proposed. Common approaches include: reduction to reinforcement learning (Fu et al., 2019; Goldwasser and Roth, 2014; Goyal et al., 2019; Ling and Fidler, 2017; MacGlashan et al., 2015; Sumers et al., 2020), learning to ground language to actions (Bisk et al., 2016; Chen and Mooney, 2011; Li et al., 2017, 2020c,d; Liu et al., 2016; Misra et al., 2014; Wang et al., 2016a), or devising EM-based algorithms to parse language into logical forms (Labutov et al., 2018; Matuszek et al., 2012a). The first approach may discard useful learning signals from language feedback and inherits the limitations of RL algorithms. The second requires extra effort from the teacher to provide demonstrations. The third approach has to bootstrap the language parser with labeled executions. ADEL enables learning from a specific type of language feedback (language description) without reducing it to reward, requiring demonstrations, or assuming access to labeled executions.

Description Feedback in Reinforcement Learning. Recently, several papers have proposed using language description feedback in the context of reinforcement learning (Chan et al., 2019; Cideron et al., 2020; Colas et al., 2020; Jiang et al., 2019). These frameworks can be viewed as extensions of hindsight experience replay (HER; Andrychowicz et al., 2017) to language goal generation. While the teacher in ILIAD can be considered as a language goal generator, an important distinction between ILIAD and these frameworks is that ILIAD models a completely *reward-free* setting. Unlike in HER, the agent in ILIAD does not have access to a reward function that it can use to compute the reward of any tuple of state, action, and goal. With the feedback coming solely from language descriptions, ILIAD is designed so that task learning relies only on extracting information from language. Moreover, unlike reward, the description language in ILIAD does not contain information that *explicitly* encourages or discourages actions of the agent. The formalism and theoretical studies of ILIAD presented in this work are based on a probabilistic formalism and do not involve reward maximization.

Description Feedback in Vision-Language Navigation. Several papers (Fried et al., 2018b; Tan et al., 2019) apply back-translation to vision-language navigation (Anderson et al., 2018b). While also operating with an output-to-input translator, back-translation is a single-round, offline process, whereas ILIAD is an iterative, online process. Zhou and Small (2021a) study a test-time scenario that is similar to ILIAD but requires labeled demonstrations to learn the execution describer and to initialize the agent. The teacher in ILIAD is more general: it can be automated (i.e., learned from labeled data), but it can also be a human. Our experiments emulate applications where non-expert humans teach agents new tasks by only giving them verbal feedback. We use labeled demonstrations to simulate human teachers, but it is part of the experimental setup, not

part of our proposed protocol and algorithm. Our agent does not have access to labeled demonstrations; it is initialized with *random parameters* and is trained with only language-description feedback. Last but not the least, we provide theoretical guarantees for ADEL, while these works only present empirical studies. **Connection to Pragmatic Reasoning.** Another related line of research is work on the rational speech act (RSA) or pragmatic reasoning (Andreas and Klein, 2016; Fried et al., 2018a; Golland et al., 2010; Goodman and Frank, 2016; Grice, 1975; Monroe and Potts, 2015), which is also concerned with transferring information via language. It is important to point out that RSA is a mental *reasoning* model whereas ILIAD is an *interactive* protocol. In RSA, a speaker (or a listener) constructs a pragmatic message-encoding (or decoding) scheme by building an internal model of a listener (or a speaker). Importantly, during that process, one agent *never* interacts with the other. In contrast, the ILIAD agent learns through interaction with a teacher. In addition, RSA focuses on encoding (or decoding) a single message while ILIAD defines a process consisting of multiple rounds of message exchanging. We employ pragmatic inference to improve the quality of the simulated teachers but in our context, the technique is used to set up the experiments and is not concerned about communication between the teacher and the agent.

Connection to Emergent Language. Finally, our work also fundamentally differs from work on (RL-based) emergent language (Das et al., 2017a; Evtimova et al., 2018; Foerster et al., 2016; Havrylov and Titov, 2017; Kottur et al., 2017; Lazaridou et al., 2017) in that we assume the teacher speaks a fixed, well-formed language, whereas in these works the teacher begins with no language capability and learns a language over the course of training.

3.2.7 Conclusion

The communication protocol of a learning framework places natural boundaries on the learning efficiency of any algorithm that instantiates the framework. In this work, we illustrate the benefits of designing learning algorithms based on a natural, descriptive communication medium like human language. Employing such expressive protocols leads to ample room for improving learning algorithms. Exploiting compositionality of language to improve sample efficiency, and learning with diverse types of feedback are interesting areas of future work. Extending the theoretical analyses of ADEL to more general settings is also an exciting open problem.

Chapter 4: Enriching Human-AI Communication During Task Performance

It is conceptually unrealistic to expect AI agents to always successfully perform assigned tasks on their own. The physical and virtual worlds are constantly changing, while human minds also persistently update their preferences, desires, and beliefs. These variations would eventually put an agent in situations that are beyond its realm of wisdom (e.g., baking a new type of cake, navigating in a newly renovated house, assembling novel objects, etc.). The techniques presented in the previous chapter can help agents learn from past experiences to improve its future performances. However, while the agent is performing a task, it can also leverage assistance from humans to immediately resolve current difficult situations, without having to make invasive changes to its model parameters.

This section focuses on teaching agents the ability to request and interpret information from human bystanders or supervisors to enhance their decisions in situ. I propose interactive settings where the agent is evaluated on its ability to complete the assigned task using both its task-solving capabilities and information-requesting capabilities. These settings contrast with the traditional full-autonomy setting in which only the task-solving capabilities are evaluated.

In my construction of the agent policy $\hat{\pi}(a \mid c, d^\star)$, a human can influence the agent's decision in two ways: (1) taking actions that modify the task request $d^\star$, and (2) taking actions that change the current context c. My VNLA and HANNA frameworks (Nguyen and Daumé III,

2019; Nguyen et al., 2019b) model the second type of intervention, while the HARI framework (Nguyen et al., 2022) captures both types, allowing the agent to convey more diverse request intentions.

To reduce assisting effort from humans, the agent should be intelligent about choosing when and what to ask. In HANNA, I propose solutions to the *when* problem based on teaching the agent to request help only when it is highly uncertain or anticipates that it cannot make further progress. The HARI framework presents a rigorous POMDP-based formulation that integrates both the *when* and *what* problems, enabling solving them simultaneously via reinforcement learning.

4.1 VNLA: Vision-Based Navigation with Language-Based Assistance

4.1.1 Overview

Rich photorealistic simulators are finding increasing use as research testbeds and precursors to real-world embodied agents such as self-driving cars and drones (Dosovitskiy et al., 2017; Kempka et al., 2016; Shah et al., 2018). Recently, growing interest in grounded visual navigation from natural language is facilitated by the development of more realistic and complex simulation environments (Anderson et al., 2018b; Brodeur et al., 2017; Kolve et al., 2017; Puig et al., 2018; Savva et al., 2017; Wu et al., 2018; Xia et al., 2018) in place of simple toy environments (Bisk et al., 2016; Branavan et al., 2012, 2009; Chen and Mooney, 2011; MacMahon et al., 2006). Several variants of this task have been proposed. In Anderson et al. (2018b), agents learn to execute natural language instructions crowd-sourced from humans. Das et al. (2018) train agents to navigate to answer questions about objects in the environment. de Vries et al. (2018) present a scenario where a guide and a tourist chat to direct the tourist to a destination.

In this paper, we present Vision-based Navigation with Language-based Assistance (VNLA), a grounded vision-language task that models a practical scenario: a mobile agent, equipped with monocular visual perception, is requested via language to assist humans with finding objects in indoor environments. A realistic setup of this scenario must (a) not assume the requester knows how to accomplish a task before requesting it, and (b) provide additional assistance to the agent when it is completely lost and can no longer make progress on a task. To accomplish (a), instead of using detailed step-by-step instructions, we request tasks through high-level instructions that only describe end-goals (e.g. “Find a pillow in one of the bedrooms"). To fulfill (b), we introduce into the environment an advisor who is present at all times to assist the agent upon request with low-level language subgoals such as “Go forward three steps, turn left". In VNLA, therefore, the agent must (a) ground the object referred with the initial end-goal in raw visual inputs, (b) sense when it is lost and use an assigned budget for requesting help, and (c) execute language subgoals to make progress.

VNLA motivates a novel imitation learning setting that we term Imitation Learning with Indirect Intervention (I3L). In conventional Imitation Learning (IL), a learning agent learns to mimic a teacher, who is only available at training time, by querying the teacher’s demonstrations on situations the agent encounters. I3L extends this framework in two ways. First, an advisor is present in the environment to assist the agent not only during training time but also at test time. Second, the advisor assists the agent not by directly making decisions on the agent’s behalf, but by modifying the environment to influence its decisions. I3L models assistance via language subgoals by treating the subgoals as extra information added to the environment.

4.1.2 The Practical Problem: Visual Navigation with Human Assistance

The goal of the VNLA problem is to train an agent, with vision as the perception modality, that can navigate indoors to find objects by requesting and executing language instructions from humans. The agent is able to "see" the environment via a monocular camera capturing its first-person view as an RGB image. It is also capable of executing language instructions and requesting additional help when in need. The camera image stream and language instructions are the only external input signals provided; the agent is not given a map of the environments or its location (e.g. via GPS or indoor localization techniques).

The agent starts at a random location s_1 in an indoor environment. A *requester* assigns it an object-finding task by sending a high-level *request* $d^\star$, namely to locate an object in a particular room (e.g., "Find a cup in one of the bathrooms."). The task is always feasible: there is always an object instance in the environment that satisfies the request. The agent is considered to have fulfilled the request if it stops at a location within ϵ meters along the shortest path to an instance of the desired object. Here ϵ is the *success radius*, a task-specific hyperparameter. During execution, the agent may get lost and become unable to progress. We enable the agent to automatically sense when this happens and signal an *assistant* for help.[1] Upon receiving a help request from the agent, the assistant responds with a language *instruction*. The instruction is a sub-task that is significantly easier to accomplish than the requested task. In VNLA, we consider instructions that *describe the next k optimal actions* (e.g. "Go forward two steps, look right."). We assume that strictly following the instruction helps the agent make progress. An example of a VNLA task

[1]For simplicity, we assume the assistant has perfect knowledge of the environment, the agent, and the task. In general, as the assistant's main task is to help the agent, perfect knowledge is not necessary. The assistant needs to only possess advantages over the agent (e.g., human-level common sense or reasoning ability, greater experience at indoor navigation, etc.).

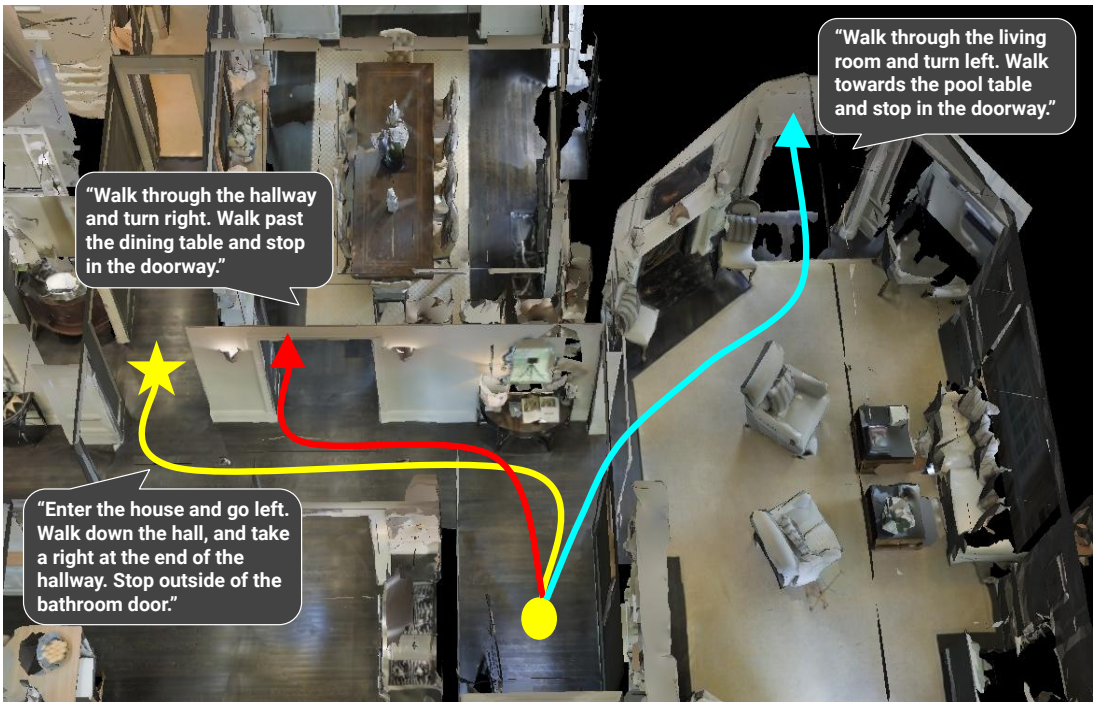


Figure 4.1: An example VNLA task. (a) A bird-eye view of the environment annotated with the agent's path. The agent observes the environment only through a first-person view. (b) A requester (wearing a hat) asks the agent to "find a towel in the kitchen". Two towels (pink circle) are in front of the agent but the room is labeled as a "bathroom". The agent ignores them without being given the room label. (c) The agent escapes the bathroom but runs into an unfamiliar region. Sensing that it is lost, the agent signals the assistant (with mustache) for help. The assistant responds with an "easier" low-level instruction "turn 60 degrees right, go forward, turn left". (d) After executing the instruction, the agent is closer to the kitchen but is still confused. It thus requests help one more time. After making this request, the agent has exhausted its request budget and can only rely on its own. (e) Executing the second instruction helps the agent see the target towel (cyan circle). It successfully walks to the goal without further assistance. A video demo is at https://youtu.be/Vp6C29qTKQ0.

is illustrated in Figure 4.1.

By specifying the agent's task with a high-level request, our setup does *not* assume the requester knows how to accomplish the task before requesting it. This aspect, along with the agent-assistant interaction, distinguishes our setup from instruction-following setups (Anderson et al., 2018b; Blukis et al., 2018; Chen and Mooney, 2011; Chen et al., 2019a; Misra et al., 2018b, 2014), in which the requester provides the agent with detailed sequential steps to execute a task only at the beginning.

4.1.3 The Theoretical Problem: Imitation Learning with Indirect Intervention

Motivated by the VNLA problem, we introduce Imitation Learning with Indirect Intervention (I3L), which models scenarios where a learning agent is monitored by a more qualified assistant (e.g., a human) and receives help through an imperfect communication channel (e.g., language).

Assistant Conventional Imitation Learning (IL) settings (Chang et al., 2015b; Daumé et al., 2009; Ross and Bagnell, 2010, 2014; Ross et al., 2011; Sharaf and Daumé III, 2017; Sun et al., 2017) involve interaction between a learning agent and a teacher: the agent learns by querying and imitating demonstrations of the teacher. In I3L, in addition to interacting with a teacher, the agent also receives guidance from an *assistant*. Unlike the teacher, who only interacts with the agent at training time, the assistant assists the agent during both training and test time. The role of the teacher in this setting is to train the agent to perform tasks by interacting with the assistant.

Intervention The assistant directs the agent to take a sequence of actions through an *intervention*, which can be direct or indirect. Interventions are direct when the assistant overwrites the agent's decisions with its own. By definition, direct interventions are always executed perfectly, i.e. the agent always takes actions the assistant wants it to take. When interventions are indirect, the assistant does not "take over" the agent but instead modifies the environment to influence its decisions. To utilize indirect interventions, the agent must learn to *interpret* them, by mapping them from signals in the environment to sequences of actions in its action space. The direct/indirect distinction is illustrated more tangibly in a physical agent such as a self-driving car. Turning off automatic driving mode and taking control of the steering wheel constitutes a direct

intervention, while issuing a verbal command to stop the car represents an indirect intervention (the command is treated as new information added to the environment).

Formulation We formulate VNLA as a I3L problem where the indirect interventions have the forms of language instructions. We consider the interactive learning setup defined in § 3.1.3.2 with state space $\mathcal{S}$ and language request space $\mathcal{D}$. However, instead of implementing a single policy, the agent maintains two policies: a main policy $\hat{\pi}_{\text{main}} : \mathcal{S} \times \mathcal{D} \to \Delta(\mathcal{A}_{\text{main}})$ for making decisions on the main task, and a help-requesting policy $\hat{\pi}_{\text{ask}} : \mathcal{S} \times \mathcal{D} \to \Delta(\mathcal{A}_{\text{ask}})$ for deciding when the assistant should intervene. Here, $\mathcal{A}_{\text{main}}$ is the action space defined by the environment, and $\mathcal{A}_{\text{ask}} = \{\texttt{request}, \texttt{do\_nothing}\}$ contains actions for interacting with the assistant. We also assume existence of teacher policies $\pi^{\star}_{\text{main}} : \mathcal{S} \times \mathcal{D} \to \Delta(\mathcal{A}_{\text{main}})$ and $\pi^{\star}_{\text{ask}} : \mathcal{S} \times \mathcal{D} \to \Delta(\mathcal{A}_{\text{ask}})$, and an assistant $\Phi : \mathcal{S} \times \mathcal{D} \to \mathcal{D}$ that generates language instructions. Teacher policies are only available during training, while the assistant is always present. Having a policy $\hat{\pi}_{\text{ask}}$ that actively decides when to ask for help reduces efforts of the assistant to monitor the agent. However, this does not prevent the assistant from actively intervening when necessary, because it is able to control $\hat{\pi}_{\text{ask}}$'s decisions by modifying the environment appropriately.

Let d_i^+ be the request executed by the agent in time step i. Initially, $d_0^+ = d^\star$. Also, let $\hat{a}_i^{\text{ask}}$ be the decision of the help-requesting policy in time step i. If $\hat{a}_i^{\text{ask}} = \texttt{request}$, the agent notifies the assistant that it needs help. The assistant then outputs an instruction $\hat{d}_i$ that directs the agent to take a sequence of actions. In this work, we consider the case when the instruction guides the

agent to take the next k reference actions $\left(a^\star_i, a^\star_{i+1}, \cdots, a^\star_{i+k-1}\right)$ suggested by the teacher

$$a^\star_{i+j} \sim \pi^*_{\text{main}}(\cdot \mid s_{i+j}, d^\star), \tag{4.1}$$

$$s_{i+j+1} \sim T\left(\cdot \mid s_{i+j}, a^\star_{i+j}\right) \quad 0 \leq j < k$$

The agent then incorporates the received instruction with the original request to form a new request: $d^+_i = d^\star \oplus \hat{d}_i$, where the operator $\oplus$ defines how the assisting instruction is incorporated. For example, in my experiments, the agent simply overwrites the request with the instruction. A special action $\texttt{stop} \in \mathcal{A}_{\text{main}}$ is taken when the agent decides that it has completed the current request. At that time, if $d^+_i \neq d^\star$ the agent resumes executing the main request, setting $d^+_{i+1} = d^\star$. Otherwise, if $d^+_i = d^\star$, the agent terminates the episode. The follow algorithm demonstrates how the agent interacts with the assistant during an episode at test time

Algorithm 6 Imitation Learning with Language-Based Indirect Intervention (test time).

Agent receives request $d^\star$ and start state s_1.
Initialize $d^+_1 = d^\star$
for $i = 1, 2, \cdots, H$ **do**
    Agent chooses help-request action: $\hat{a}^{\text{ask}}_i \sim \hat{\pi}_{\text{ask}}(\cdot \mid s_i, d^+_i)$
    if $\hat{a}^{\text{ask}}_i = \texttt{request}$ **then**
        Agent requests instruction from assistant: $\hat{d}_i = \Phi(s_i, d^\star)$
        Agent incorporates instruction with request: $d^+_{i+1} = d^\star \oplus \hat{d}_i$
        Agent remains in the same state: $s_{i+1} = s_i$
    else
        Agent chooses environment action: $\hat{a}^{\text{main}}_i \sim \hat{\pi}_{\text{main}}(\cdot \mid s_i, d^+_i)$
        if $\hat{a}^{\text{main}}_i = \texttt{stop}$ **then**
            if $d^+_i \neq d^\star$ **then**
                Agent resumes executing main request: $d^+_{i+1} = d^\star$
                Set $s_{i+1} = s_i$
            else
                break
        else
            Set $d^+_{i+1} = d^+_i$
            Agent transitions to a new state: $s_{i+1} \sim T(\cdot \mid s_i, \hat{a}^{\text{main}}_i)$

We denote by $P_{\hat{\pi}_{\text{main}},\hat{\pi}_{\text{ask}}}(e \mid s_1, d^\star)$ the distribution of executions generated using Alg 6. Similar to standard DAgger, we define the training objective in I3L as minimizing expected imitation loss on the agent-induced execution distribution:

$$\min_{\pi_{\text{main}},\pi_{\text{ask}}} \mathbb{E}_{(s_1,d)\sim P^\star(\cdot),\hat{e}\sim P_{\pi_{\text{main}},\pi_{\text{ask}}}(\cdot|s_1,d^\star)} \left[\mathcal{L}\left(\hat{e}, \pi_{\text{main}}, \pi_{\text{ask}}\right)\right] \tag{4.2}$$

$$\mathcal{L}(\hat{e}, \pi_{\text{main}}, \pi_{\text{main}}) = \sum_{i=1}^{H} L\left(s_i, a_i^{\star\text{main}}, \pi_{\text{main}}\right) + L\left(s_i, a_i^{\star\text{ask}}, \pi_{\text{ask}}\right) \tag{4.3}$$

where s_i are states in $\hat{e}$, $a_i^{\star\text{main}} \sim \pi^\star_{\text{main}}(\cdot \mid s_i, d^\star)$, $a_i^{\star\text{ask}} \sim \pi^\star_{\text{ask}}(\cdot \mid s_i, d^\star)$, and $L(s, a^\star, \pi)$ computes the imitation loss between policy π and reference action $a^\star$ on state s.

4.2 HANNA: Visual Navigation with Natural Multimodal Assistance

4.2.1 Overview

HANNA stands for "Help! Automatic Natural Navigation Assistance." The motivation for developing HANNA is to simulate natural human assistance that are:

- derived from *interpersonal* interaction (a lost tourist asks a local for directions);
- *reactive* to the situation of the receiver, based on the assistant's knowledge (the local may guide the tourist to the goal, or may redirect them to a different source of assistance);
- delivered via a *multimodal* communication channel (the local uses a combination of language, images, maps, gestures, etc.).

Similar to VNLA, the agent in HANNA is able to request and interpret additional instructions. However, HANNA models a setting in which a human is not always available to help, but rather that human assistants are scattered throughout the environment and provide help upon

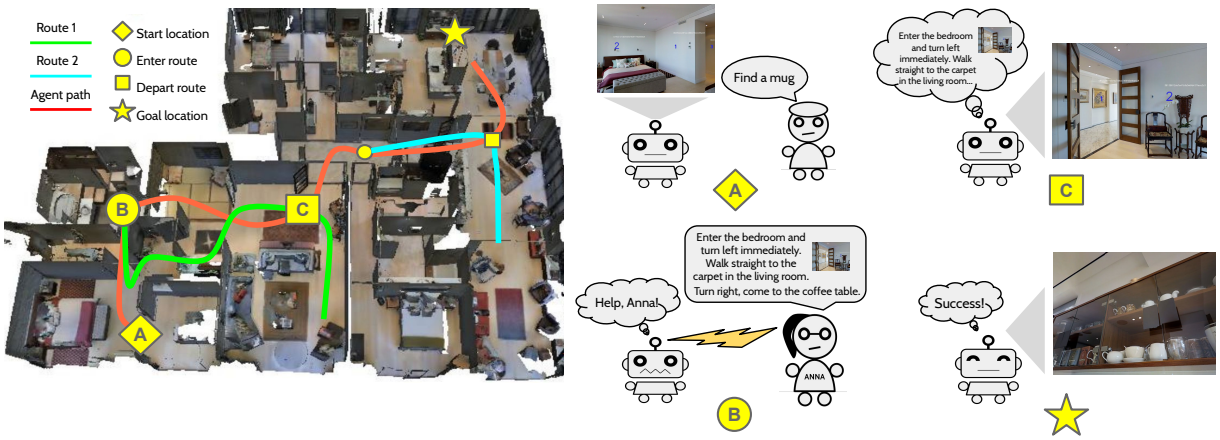


Figure 4.2: An example HANNA task. Initially, the agent stands in the bedroom at A and is requested to "find a mug." The agent begins, but gets lost somewhere in the bathroom. It gets to the start location of route (B) to request help from a nearby human. Upon request, the human gives the agent a *language instruction* that guides the agent to a target location, and an image of the view at that location. The agents follows the instruction and arrives at C, where it observes a match between the target image and the current view, thus decides to depart route. After that, it resumes the main task of finding a mug. From this point, the agent gets lost one more time and has to query another human for another instruction, which helps it follow route and enter the kitchen. The agent successfully fulfills the task it finally stops within ϵ meters of an instance of the requested object (☆). Here, the human instructive feedback is simulated using two pre-collected language-assisted routes (and).

request (modeling the *interpersonal* aspect). The assistants are *not* omniscient: they are only familiar with certain regions of the environment and, upon request, provide *subtasks*, expressed in language and images (modeling the *multimodal* aspect), for getting *closer* to the goal, not necessarily for fully completing the task (modeling the *reactive* aspect). Table 4.1 compares HANNA with other photo-realistic navigation problems.

4.2.2 The HANNA simulator

Problem Similar to VNLA, HANNA also simulates a scenario where a *human requester* asks a mobile *agent* via language to find an object in an indoor environment. The task request is only a high-level command $d^\star$ ("find [object(s)]"), modeling the general case when the requester does not need know how to accomplish a task when requesting it. We assume the task is always

Table 4.1: Comparing HANNA with other photo-realistic navigation problems. VLN (Anderson et al., 2018b) does not allow agent to request help. CVDN (Thomason et al., 2019b) contains natural conversations in which a human assistant aids another human in navigation tasks but offers limited language interaction simulation, as language assistance is not available when the agent deviates from the collected trajectories and tasks. VNLA (Nguyen et al., 2019a) models an advisor who is always present to help but speaks simple, templated language. HANNA simulates human assistants that provide natural language-and-vision instructions that adapt to the agent's current position and goal.

Problem	Request assistance	Natural multimodal instructions	Simulated humans
VLN	✗	✗	✗
CVDN	✓	✗	✗
VNLA	✓	✗	✓
HANNA (this work)	✓	✓	✓

feasible: there is at least an instance of the requested object in the environment.

Figure 4.2, to which references in this section will be made, illustrates an example where the agent is asked to "find a mug." The agent starts at a random location s_1 (), is given a task request, and is allotted a budget of H time steps to complete the task. The agent succeeds the if its final location is within $\epsilon_{\text{success}}$ meters of the location of any instance of the requested object (). The agent is not given any sensors that help determine its location or the object's location and must navigate only with a monocular camera that captures its first-person view as an RGB image (e.g., image in the upper right of Figure 4.2).

The only source of help the agent can leverage in the environment is *human assistants*, who are present at both training and evaluation time. The assistants are not aware of the agent unless it enters their *zones of attention*, which include all locations within ϵ_{attn} meters of their locations. When the agent is in one of these zones, it has an option to request help from the corresponding assistant. The assistant helps the agent by giving a *subtask*, described by a natural language instruction that guides the agent to a specific location, and an image of the view at that location.

In my example, at B, the assistant says *"Enter the bedroom and turn left immediately. Walk straight to the carpet in the living room. Turn right, come to the coffee table."* and provides an image of the destination in the living room. Executing the subtask may not fulfill the main task, but is guaranteed to get the agent to a location closer to a goal than where it was before (e.g., C).

Photo-realistic Navigation Simulator HANNA uses the Matterport3D simulator (Anderson et al., 2018b; Chang et al., 2017b) to photo-realistically emulate a first-person view while navigating in indoor environments. HANNA features 68 Matterport3D environments, each of which is a residential building consisting of multiple rooms and floors. Navigation is modeled as traversing an undirected graph $G = (V, E)$, where each location corresponds to a node $v \in V$ with 3D-coordinates $\boldsymbol{x}_v$, and edges are weighted by their lengths (in meters).

The state of the agent is fully determined by its pose $\tau = (v, \psi, \omega)$, where v is its location, $\psi \in (0, 2\pi]$ is its heading (horizontal camera angle), and $\omega \in \left[-\frac{\pi}{6}, \frac{\pi}{6}\right]$ is its elevation (vertical camera angle). The agent does not know v, and the angles are constrained to multiples of $\frac{\pi}{6}$. In each step, the agent can either stay at its current location, or it can rotate toward and go to a location adjacent to it in the graph[2]. Every time the agent moves (and thus changes pose), the simulator recalculates the image to reflect the new view.

Automatic Natural Navigation Assistants (ANNA) ANNA is a simulation of human assistants who do not necessarily know themselves how to optimally accomplish the agent's goal: they are only familiar with scenes along certain paths in the environment, and thus give advice to help the agent make partial progress. Specifically, the assistance from ANNA is modeled by a set of *language-assisted routes* $\Xi = \{\xi_1, \xi_2, \ldots, \xi_{|R|}\}$. Each route $\xi = (\psi^\xi, \omega^\xi, p^\xi, d^\xi)$ is defined

[2]We use the "panoramic action space" (Fried et al., 2018b).

by initial camera angles $(\psi^{\xi}, \omega^{\xi})$, a path p^{ξ} in the environment graph, and a natural language instruction d^{ξ}. A route becomes *enterable* when its start location is adjacent to and within ϵ_{attn} meters of the agent's location. When the agent enters a route, it first adjusts its camera angles to $(\psi^{\xi}, \omega^{\xi})$, then attempts to interpret the language instructions l^{ξ} to traverse along p^{ξ}. At any time, the agent can *depart* the route by stopping following d^{ξ}. An example of a route in Figure 4.2 is the combination of the initial camera angles at B, the path ⤳, and the language instruction *"Enter the bedroom and turn left immediately..."*

The set of all routes starting from a location simulates a *human assistant* who can recall scenes along these routes' paths. The *zone of attention* of the simulated human is the set of all locations from which the agent can enter one of the routes; when the agent is in this zone, it may ask the human for help. Upon receiving a help request, the human selects a route $\xi^{\star}$ for the agent to enter (e.g., ⤳), and a location v_{depart} on the route where it wants the agent to depart (e.g., C). It then replies the agent with a *multimedia message* $(d^{\xi^{\star}}, \mathcal{I}^{v_{\text{depart}}})$, where $d^{\xi^{\star}}$ is the selected route's language instruction, and $\mathcal{I}^{v_{\text{depart}}}$ is an image of the panoramic view at the departure location. The message describes a *subtask* which requires the agent to follow the direction described by $d^{\xi^{\star}}$ and to stop if it reaches the location referenced by $\mathcal{I}^{v_{\text{depart}}}$. The route $\xi^{\star}$ and the departure node v_{depart} are selected to get the agent as close to a goal location as possible. Concretely, let Ξ_{curr} be the set of all routes associated with the requested human. The selected route minimizes the distance to the goal locations among all routes in Ξ_{curr}:

$$r^{\star} = \underset{\xi \in \Xi_{\text{curr}}}{\operatorname{argmin}} \ \bar{\delta}\left(\xi, V_{\text{goal}}\right) \tag{4.4}$$

$$\text{where } \bar{\delta}\left(\xi, V_{\text{goal}}\right) \stackrel{\text{def}}{=} \min_{g \in V_{\text{goal}}, v \in p^{\xi}} \delta\left(g, v\right) \tag{4.5}$$

$\delta(.,.)$ returns the (shortest-path) distance between two locations, and V_{goal} is the set of all goal locations. The departure location minimizes the distance to the goal locations among all locations on the selected route:

$$v_{\text{depart}} = \underset{g \in V_{\text{goal}}, v \in p^{\xi^\star}}{\operatorname{argmin}} \; \delta\left(g, v\right) \tag{4.6}$$

When the agent chooses to depart the route (not necessarily at the departure node), the human further assists it by providing $\mathcal{I}^{g^\star}$, an image of the panoramic view at the goal location closest to the departure node:

$$g^\star = \underset{g \in V_{\text{goal}}}{\operatorname{argmin}} \; \delta\left(g, v_{\text{depart}}\right) \tag{4.7}$$

The way the agent leverages ANNA to accomplish tasks is analogous to how humans travel using public transportation systems (e.g., bus, subway). For example, passengers of a subway system utilize fractions of pre-constructed routes to make progress toward a destination. They execute travel plans consisting of multiple subtasks, each of which requires entering a start stop, following a route (typically described by its name and last stop), and exiting at a departure stop (e.g., *"Enter the Penn Station, hop on the Red line in the direction toward the South Ferry, get off at the World Trade Center"*). Occasionally, users walk short distances (at a loIr speed) to switch routes. Our setup follows the same principle, but instead of having physical vehicles and railways, we employ low-level language-and-vision instructions as the "high-speed means" to accelerate travel.

Constructing ANNA route system Given a photo-realistic simulator, the primary cost for constructing the HANNA problem comes from crowd-sourcing the natural language instructions. Ide-

ally, we want to collect sufficient instructions to simulate humans in any location in the environment. Let $N = |V|$ be the number of locations in the environment. Since each simulated human is familiar with at most N locations, in the worst case, we need to collect $O(N^2)$ instructions to connect all location pairs. However, we theoretically prove that, assuming the agent executes instructions perfectly, it is possible to guide the agent between any location pair by collecting only $O(N \log N)$ instructions. The key idea is using $O(\log N)$ instructions instead of a single instruction to connect each location pair, and reusing an instruction for multiple routes.

Lemma 2. *(proof in Appendix A) To guide the agent between any two locations using $O(\log N)$ instructions, we need to collect instructions for $O(N \log N)$ location pairs.*

Proof. Consider a spanning tree of the environment graph that is rooted at v_r. For each node v, let $d(v)$ be the distance from v to v_r. For $i = 0 \dots \lfloor \log_2 d(v) \rfloor - 1$, collect an instruction for the path from v to its 2^i-th ancestor, and an instruction for the reverse path. In total, no more than $2 \cdot N \log_2 N = O(N \log N)$ instructions are collected.

A language-assisted path is a path that is associated with a natural language instruction. It is easy to show that, under this construction, $O(\log N)$ language-assisted paths are sufficient to traverse between v and any ancestor of v as follows. For any two arbitrary nodes u and v, let w be their least common ancestor. To traverse from u to v, we first go from u to w, then from w to v. The total number of language-assisted path is still $O(\log N)$. □

In our experiments, we leverage the pre-existing Room-to-room dataset (Anderson et al., 2018b) to construct the route system. This dataset contains 21,567 natural language instructions crowd-sourced from humans and is originally intended to be used for the Vision-Language Navigation task (such as those in Figure 4.2), where an agent executes a language instruction to go to a

Algorithm 7 Task episode at test time, given agent help-request policy $\hat{\pi}_{\text{ask}}$ and navigation policy $\hat{\pi}_{\text{main}}$

Agent receives task request $d^\star$ and start state s_1
Initialize the agent mode: $m \leftarrow$ `main_task`
Initialize the language instruction: $d_1^+ \leftarrow d^\star$
Initialize the target image: $I_1^{\text{tgt}} \leftarrow$ `None`
for $i = 1 \dots H$ **do**
    Let o_i be the current observation and $\tau_i = (v_i, \psi_i, \omega_i)$ the current pose
    Agent makes a help-request decision $\hat{a}_i^{\text{ask}} \sim \hat{\pi}_{\text{ask}}\left(\cdot \mid o_i, d_i^+, I_i^{\text{tgt}}\right)$
    if $\hat{a}_i^{\text{ask}} =$ `request` **then**
        Set mode: $m \leftarrow$ `sub_task`
        Agent requests help: $\left(\xi, I^{\text{depart}}, I^{g^\star}\right) \leftarrow \Psi(s_i, d^\star)$, where $g^\star$ is defined in Eq 4.7
        Set the language instruction: $d_{i+1}^+ \leftarrow d^\xi$
        Set the target image: $I_{i+1}^{\text{tgt}} \leftarrow I^{\text{depart}}$
        Set the navigation action:
        $\hat{a}_t^{\text{main}} \leftarrow (p_0^\xi, \psi^\xi - \psi_i, \omega^\xi - \omega_i)$,
        where p_0^ξ is the start location of route ξ
    else
        Agent chooses navigation action: $\hat{a}_i^{\text{main}} \sim \hat{\pi}_{\text{main}}\left(\cdot \mid o_i, d_i^+, I_i^{\text{tgt}}\right)$
        if $\hat{a}_i^{\text{main}} =$ `stop` **then**
            if $m =$ `main_task` **then**
                break
            else
                Set mode: $m \leftarrow$ `main_task`
                Set the language instruction: $d_{i+1}^+ \leftarrow d^\star$
                Set the target image: $I_{i+1}^{\text{tgt}} \leftarrow I^{g^\star}$
                Set navigation action: $\hat{a}_i^{\text{main}} \leftarrow (v_i, 0, 0)$
    Agent executes $\hat{a}_i^{\text{main}}$ to go to the next location: $s_{i+1} \sim T(\cdot \mid s_i, \hat{a}_i^{\text{main}})$

location. We exclude instructions of the test split and their corresponding environments because ground-truth paths are not given. We use (on average) 211 routes to connect (on overage) 125 locations per environment. Even though the routes are selected randomly in the original dataset, my experiments show that they are sufficient for completing the tasks (assuming perfect assistance interpretation).

4.2.3 Simultaneously Learning to Navigate and Request Help

Agent Policies. Let s be a fully-observed state that contains ground-truth information about the environment and the agent (e.g., object locations, environment graph, agent parameters, etc.). Let o_s be the corresponding observation given to the agent, which only encodes the current view, the current task, and extra information that the agent keeps track of (e.g., time, action history, etc.).

The agent maintains two stochastic policies: a navigation policy $\hat{\pi}_{\text{main}}$ and a help-request policy $\hat{\pi}_{\text{ask}}$. Each policy takes in an observation o, a language request d, and an image I and outputs a probability distribution over its action space. Navigation actions are tuples $(v, \Delta\psi, \Delta\omega)$, where v is a next location that is adjacent to the current location and $(\Delta\psi, \Delta\omega)$ is the camera angle change. A special `stop` action is added to the set of navigation actions to signal that the agent wants to terminate the main task or a subtask (by departing a route). The action space of the help-request policy contains two actions: `request` and `do_nothing`. The `request` action is only available when the agent is in a zone of attention. Alg 7 describes the effects of these actions during a task episode.

Imitation Learning Objective The agent is trained with DAgger (Ross and Bagnell, 2010) to mimic behaviors suggested by a navigation teacher $\pi^{\star}_{\text{main}}$ and a help-request teacher $\pi^{\star}_{\text{ask}}$, who have access to the fully-observed states. In general, DAgger (Ross et al., 2011) finds a policy $\hat{\pi}$ that minimizes the expected total imitation loss with respect decisions of a teacher policy $\pi^{\star}$ under the agent-induced execution distribution:

$$\min_{\pi} \mathbb{E}_{(s_1,d)\sim P^{\star}(\cdot),\hat{e}\sim P_{\pi}(\cdot|s_1,d)} \left[\sum_{i=1}^{H} L(s_i, a_i^{\star}, \pi) \right] \tag{4.8}$$

where $P^\star$ is the task distribution, $P_\pi(e \mid s_1, d^\star)$ is the (conditional) execution distribution induced by policy π, s_i are states of $\hat{e}$, $a_i^\star \sim \pi^\star(\cdot \mid s_i, d^\star)$, and L computes the imitation loss. We frame the HANNA problem as an instance of *Imitation Learning with Indirect Intervention* (I3L) (§ 4.1.3). Under this framework, assistance is viewed as augmenting the current environment with new information. Interpreting the assistance is cast as finding the optimal acting policy in the augmented environment. Formally, given teacher policies $\pi^\star_{\text{main}}$ and $\pi^\star_{\text{ask}}$, I3L searches for policies that optimize:

$$\min_{\pi_{\text{main}}, \pi_{\text{ask}}} \mathbb{E}_{s_1, d \sim P^\star(\cdot), \hat{e} \sim P_{\pi_{\text{main}}, \pi_{\text{ask}}}(\cdot \mid s_1, d^\star)} \left[\mathcal{L}\left(\hat{e}, \pi_{\text{main}}, \pi_{\text{ask}}\right) \right] \tag{4.9}$$

$$\mathcal{L}(\hat{e}, \pi_{\text{main}}, \pi_{\text{main}}) = \sum_{i=1}^{H} L_{\text{main}}\left(s_i, a_i^{\star\text{main}}, \pi_{\text{main}}\right) + L_{\text{ask}}\left(s_i, a_i^{\star\text{ask}}, \pi_{\text{ask}}\right) \tag{4.10}$$

where $a_i^{\star\text{main}} \sim \pi^\star_{\text{main}}(\cdot \mid s_i, d^\star)$ and $a_i^{\star\text{ask}} \sim \pi^\star_{\text{ask}}(\cdot \mid s_i, d^\star)$.

A common choice for the loss function L is the agent-estimated negative log likelihood of the reference action:

$$L_{\text{NL}}(s, a^\star, \pi) \stackrel{\text{def}}{=} -\log \pi(a^\star \mid o_s) \tag{4.11}$$

We introduce novel loss functions that enforce more complex behaviors than simply mimicking reference actions.

Reference Actions The navigation teacher suggests a reference action $a^{\text{main}\star}$ that takes the agent to the next location on the shortest path from its location to the target location. Here, the target location refers to the nearest goal location (if no target image is available), or the location referenced by the target image (provided by ANNA). If the agent is already at the target location,

$a^{\text{main}\star} = \texttt{stop}$. To decide whether the agent should request help, the help-request teacher verifies the following conditions:

1. `lost`: the agent will not get (strictly) closer to the target location in the future;
2. `uncertain_wong`: the entropy[3] of the navigation action distribution is greater than or equal to a threshold γ, and the highest-probability predicted navigation action is *not* suggested by the navigation teacher;
3. `never_asked`: the agent previously never requested help at the current location;

If condition (1) or (2), and condition (3) are satisfied, we set $a^{\text{ask}\star} = \texttt{request}$; otherwise, $a^{\text{ask}\star} = \texttt{do\_nothing}$.

Curiosity-Encouraging Navigation Teacher In addition to a reference action, the navigation teacher returns $A^{\text{main}\otimes}$, the set of all non-reference actions that the agent took at the current location while executing the same language instruction:

$$A_t^{\text{main}\otimes} = \big\{ a \in A^{\text{main}} : \exists t' < t,\ \ v_t = v_{t'}, \\ l_t = l_{t'},\ \ a = a_{t'}^{\text{main}} \neq a_{t'}^{\text{main}\star} \big\} \tag{4.12}$$

where A^{main} is the navigation action space.

We devise a *curiosity-encouraging* loss $\mathcal{L}_{\text{curious}}$, which minimizes the log likelihoods of actions in $A^{\text{main}\otimes}$. This loss prevents the agent from repeating past mistakes and motivates it to

[3] Precisely, we use *efficiency*, or entropy of base $|A^{\text{main}}| = 37$, where A^{main} is the navigation action space.

explore untried actions. The navigation loss is:

$$L_{\text{main}}(s, a^{\text{main}\star}, \hat{\pi}_{\text{main}}) = \overbrace{-\log \hat{\pi}_{\text{main}}(a^{\text{main}\star} \mid o_s)}^{\text{negative log-likelihood loss}} + \alpha \underbrace{\frac{1}{|A^{\text{main}\otimes}|} \sum_{a \in A^{\text{main}\otimes}} \log \hat{\pi}_{\text{main}}(a \mid o_s)}_{\text{curiosity-encouraging loss}} \tag{4.13}$$

where $\alpha \in [0, \infty)$ is a weight hyperparameter.

Retrospective Interpretable Help-Request Teacher In deciding whether the agent should ask for help, the help-request teacher must consider the agent's future situations. Standard imitation learning algorithms (e.g., DAgger) employ an *online* mode of interaction which queries the teacher at every time step. This mode of interaction is not suitable for my problem: the teacher must be able to predict the agent's future actions if it is queried when the episode is not finished. To overcome this challenge, we introduce a more efficient *retrospective* mode of interaction, which waits until the agent completes an episode and queries the teacher for reference actions for *all* time steps at once. With this approach, because the future actions at each time step are now fully observed, they can be taken into consideration when computing the reference action.

To help the agent better justify its help-request decisions, we train a *reason classifier* Ψ to predict which conditions are satisfied. To train this classifier, the teacher provides a reason vector $\rho^\star \in \{0, 1\}^3$, where $\rho^\star_i = 1$ indicates that the i-th condition is met. We formulate this prediction problem as multi-label binary classification and employ a binary logistic loss for each condition. Learning to predict the conditions helps the agent make more accurate and interpretable

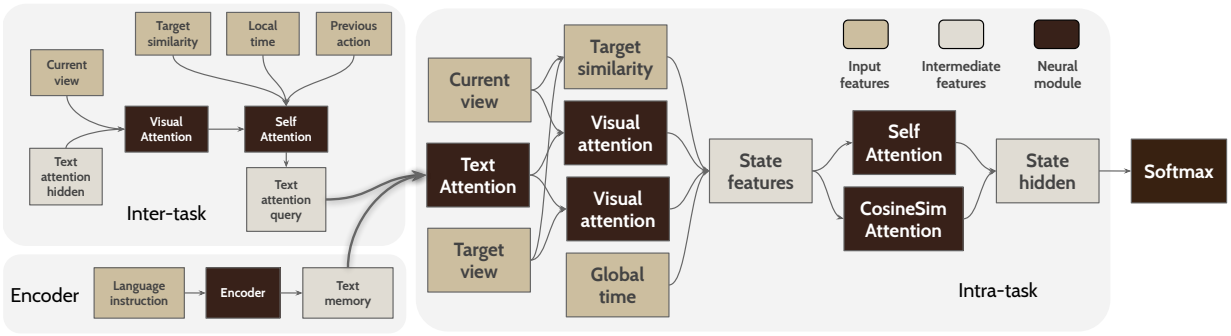


Figure 4.3: Our hierarchical recurrent model architecture (the navigation network). The help-request network is mostly similar except that the navigation action distribution is fed as an input to compute the "state features".

decisions. The help-request loss is:

$$\mathcal{L}_{\text{ask}}(s, a^{\text{main}\star}, \hat{\pi}_{\text{ask}}) = \overbrace{-\log \hat{\pi}_{\text{ask}}(a^{\text{ask}\star} \mid o_s)}^{\text{negative log-likelihood loss}} \underbrace{-\frac{1}{3}\sum_{i=1}^{3}[\rho_i^\star \log \hat{\rho}_i + (1-\rho_i^\star)\log(1-\hat{\rho}_i)]}_{\text{reason-classification loss}} \tag{4.14}$$

where $\left(a^{\text{ask}\star}, \rho^\star\right) = \pi^\star_{\text{ask}}(s, d^\star)$, and $\hat{\rho} = \Psi(o_s)$ is the agent-estimated likelihoods of the conditions.

4.2.4 Model Architecture

We model the navigation policy and the help-request policy as two separate neural networks. The two networks have similar architectures, which consists of three main components: the *text-encoding* component, the *inter-task* component, and the *intra-task* component (Figure 4.2). We use self-attention instead of recurrent neural networks to better capture long-term dependency, and develop novel cosine-similarity attention and ResNet-based time-encoding. Detail on the computations in each module is in the Appendix of (Nguyen and Daumé III, 2019).

The text-encoding component computes a text memory M^{text}, which stores the hidden representation of the current language instruction. The inter-task module computes a vector h_t^{inter} representing the state of the current task's execution. During the episode, every time the current task is altered (due to the agent requesting help or departing a route), the agent re-encodes the new language instruction to generate a new text memory and resets the inter-task state to a zero vector. The intra-task module computes a vector h_t^{intra} representing the state of the entire episode. To compute this state, we first calculate $\bar{h}_t^{\text{intra}}$, a tentative current state, and $\tilde{h}_t^{\text{intra}}$, a weighted combination of the past states at nearly identical situations. h_t^{intra} is computed as:

$$h_t^{\text{intra}} = \bar{h}_t^{\text{intra}} - \beta \cdot \tilde{h}_t^{\text{intra}} \tag{4.15}$$

$$\beta = \sigma(W_{\text{gate}} \cdot [\bar{h}_t^{\text{intra}}; \tilde{h}_t^{\text{intra}}]) \tag{4.16}$$

Eq 4.15 creates an context-sensitive dissimilarity between the current state and the past states at nearly identical situations. The scale vector β determines how large the dissimilarity is based on the inputs. This formulation incorporates past related information into the current state, thus enables the agent to optimize the curiosity-encouraging loss effectively. Finally, h_t^{intra} is passed through a softmax layer to produce an action distribution.

4.2.5 Experimental Setup

Dataset We generate a dataset of object-finding tasks in the HANNA environments to train and evaluate our agent. Table 4.2 summarizes the dataset split. Our dataset features 289 object types; the language instruction vocabulary contains 2,332 words. The numbers of locations on the

Table 4.2: Data split.

Split	Environments	Tasks	ANNA Instructions
Train	51	82,484	8,586
Val SeenEnv	51	5,001	4,287
Val UnseenAll	7	5,017	2,103
Test SeenEnv	51	5,004	4,287
Test Unseen	10	5,012	2,331

shortest paths to the requested objects are restricted to be between 5 and 15. With an average edge length of 2.25 meters, the agent has to travel about 9 to 32 meters to reach its goals. We evaluate the agent in environments that are seen during training (SEENENV), and in environments that are not seen (UNSEENALL). Even in the case of SEENENV, the tasks and the ANNA language instructions given during evaluation were never given in the same environments during training.

Hyperparameters See Appendix §D.3.

Baselines and Skylines We compare our agent against the following *non-learning* agents:

- SHORTEST: uses the navigation teacher policy to make decisions (this is a skyline);
- RANDOMWALK: randomly chooses a navigation action at every time step;
- FORWARD10: navigates to the next location closest to the center of the current view to advance for 10 time steps.

We compare our learned help-request policy with the following *heuristics*:

- NOASK: does not request help;
- RANDOMASK: randomly chooses to request help with a probabilty of 0.2, which is the average help-request ratio of our learned agent;
- ASKEVERY5: requests help as soon as walking at least 5 time steps.

Table 4.3: Results on test splits. The agent with "perfect assistance interpretation" uses the teacher navigation policy ($\pi^{\star}_{\text{nav}}$) to make decisions when executing a subtask from ANNA. Results of our final system are in bold.

Agent	SEENENV				UNSEENALL			
	SR ↑ (%)	SPL ↑ (%)	Nav. ↓ Err. (m)	Requests/ task ↓	SR ↑ (%)	SPL ↑ (%)	Nav. ↓ Err. (m)	Requests/ task ↓
Non-learning agents								
RANDOMWALK	0.54	0.33	15.38	0.0	0.46	0.23	15.34	0.0
FORWARD10	5.98	4.19	14.61	0.0	6.36	4.78	13.81	0.0
Learning agents								
No assistance	17.21	13.76	11.48	0.0	8.10	4.23	13.22	0.0
Learn to interpret assistance (ours)	**86.67**	**63.27**	**1.63**	**2.7**	**45.43**	**25.02**	**8.04**	**4.4**
Skylines								
SHORTEST	100.00	100.00	0.00	0.0	100.00	100.00	0.00	0.0
Perfect assistance interpretation	90.19	68.76	1.16	2.4	79.01	54.86	2.65	3.0

Evaluation metrics Our main metrics are: *success rate* (SR), the fraction of examples on which the agent successfully solves the task; *navigation error*, the average (shortest-path) distance between the agent's final location and the nearest goal from that location; and *SPL* (Anderson et al., 2018a), which weights task success rate by travel distance as follows:

$$\text{SPL} = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{L_i}{\max(P_i, L_i)} \tag{4.17}$$

where N is the number of tasks, S_i indicates whether task i is successful, P_i is the agent's travel distance, and L_i is the shortest-path distance to the goal nearest to the agent's final location. Agents are evaluated with a batch size of 1.

Table 4.4: Success rates (%) of agents on test splits with different types of assistance.

Assistance type	SEENENV	UNSEENALL
Target image only	84.29	29.91
+ Language instruction	**86.67**	**45.43**

4.2.6 Results

Main results From Table 4.3, we see that my problem is challenging: simple heuristic-based baselines such as RANDOMWALK and FORWARD10 attain success rates less than 7%. An agent that learns to accomplish tasks without additional assistance from ANNA succeeds only 17.21% of the time on TEST SEENENV, and 8.10% on TEST UNSEENALL. Leveraging help from ANNA dramatically boosts the success rate by 71.16% on TEST SEENENV and by 39.35% on TEST UNSEENALL over not requesting help. Given the small size of our dataset (e.g., the agent has fewer than 9,000 subtask instructions to learn from), it is encouraging that our agent is successful in nearly half of its tasks. On average, the agent takes paths that are 1.38 and 1.86 times longer than the optimal paths on TEST SEENENV and TEST UNSEENALL, respectively. In unseen environments, it issues on average twice as many requests to as it does in seen environments. To understand how well the agent interprets the ANNA instructions, we also provide results where our agent uses the optimal navigation policy to make decisions while executing subtasks. The large gaps on TEST SEENENV indicate there is still much room for improvement in the future, purely in learning to execute language instructions.

Does understanding language improve generalizability? Our agent is assisted with both language and visual instructions; similar to Thomason et al. (2019a), we disentangle the useful-

Table 4.5: Success rates (%) of different help-request policies on test splits.

	SEENENV		UNSEENALL	
$\hat{\pi}_{\text{ask}}$	SR ↑ (%)	Requests/ task ↓	SR ↑ (%)	Requests/ task ↓
NOASK	17.21	0.0	8.10	0.0
RANDOMASK	78.40	4.1	35.97	6.6
ASKEVERY5	86.63	3.5	33.62	7.1
Learned (ours)	**86.67**	**2.7**	**45.43**	**4.4**

ness two these two modes of assistance. As seen in Table 4.4, the improvement from language on TEST UNSEENALL (+15.17%) is substantially more than that on TEST SEENENV (+3.42%), largely the agent can simply memorize the seen environments. This confirms that understanding language-based assistance effectively enhances the agent's capability of accomplishing tasks in novel environments.

Is learning to request help effective? Table 4.5 compares our learned help-request policies with baselines. We find that ASKEVERY5 provides a surprisingly strong baseline for this problem, leading to an improvement of +26.32% over not requesting help on TEST UNSEENALL. Nevertheless, our learned policy, with the ability to predict the future and access to the agent's uncertainty, outperforms all baselines by at least 10.40% in success rate on TEST UNSEENALL, while making less help requests. The small gap between the learned policy and ASKEVERY5 on TEST UNSEENALL is expected because, on this split, the performance is mostly determined by the model's memorizing capability and is mostly insensitive to the help-request strategy.

Is proposed model architecture effective? We implement an LSTM-based encoder-decoder model that is based on the architecture proposed by (Wang et al., 2019c). To incorporate the target image, we add an attention layer that uses the image's vector set as the attention mem-

Table 4.6: Results on TEST UNSEENALL of our model, trained with and without curiosity-encouraging loss, and an LSTM-based encoder-decoder model (both models have about 15M parameters). "Navigation mistake repeat" is the fraction of time steps on which the agent repeats a non-optimal navigation action at a previously visited location while executing the same task. "Help-request repeat" is the fraction of help requests made at a previously visited location while executing the same task.

Model	SR ↑ (%)	Nav. mistake ↓ repeat (%)	Help-request ↓ repeat (%)
LSTM-ENCDEC	19.25	31.09	49.37
Our model ($\alpha = 0$)	41.92	25.20	39.85
Our model ($\alpha = 1$)	**45.43**	**20.53**	**10.80**

ory. We train this model with imitation learning using the standard negative log likelihood loss (Eq 4.11), without the curiosity-encouraging and reason-prediction losses. As seen in Table 4.6, our hierarchical recurrent model outperforms this model by a large margin on TEST UNSEENALL (+28.2%).

Does the proposed imitation learning algorithm achieve its goals? The curiosity-encouraging training objective is proposed to prevent the agent from making the same mistakes at previously encountered situations. Table 4.6 shows that training with the curiosity-encouraging objective reduces the chance of the agent looping and making the same decisions repeatedly. As a result, its success rate is greatly boosted (+4.33% on TEST UNSEENALL) over no curiosity-encouraging.

4.2.7 Conclusion

In this work, we present a photo-realistic simulator that mimics primary characteristics of real-life human assistance. We develop effective imitation learning techniques for learning to request and interpret the simulated assistance, coupled with a hierarchical neural network model for representing subtasks. Future work aims to provide more natural, linguistically realistic inter-

action between the agent and humans (e.g., providing the agent the ability ask a natural *question* rather than just signal for help), and to establish a theoretical framework for modeling human assistance. We are also exploring ways to deploy and evaluate our methods on real-world platforms.

4.3 HARI: A Framework for Learning to Request Rich and Contextually Useful Information from Humans

4.3.1 Overview

Machine learning has largely focused on creating agents that can solve problems on their own. Despite much progress, these autonomous agents struggle to operate in unfamiliar environments and fulfill new requirements from users (Eykholt et al., 2018; Goodfellow et al., 2015; Jia and Liang, 2017; Qi et al., 2020b; Shridhar et al., 2020). As autonomous agents are built to perform tasks by themselves, they are typically endowed with limited capabilities of communicating with humans during task execution. This design significantly constrains the performance of these agents, as human bystanders or supervisors can provide useful information that improves the agents' decisions. Uncommunicative autonomous agents are also highly unsafe. They do not consult humans on difficult decisions and can potentially commit catastrophic mistakes without warnings.

The reliability of autonomous agents can be dramatically enhanced if they are equipped with communication skills for leveraging knowledge of humans in the same environment (Nguyen and Daumé III, 2019; Nguyen et al., 2019b; Rosenthal et al., 2010; Tellex et al., 2014a; Thomason et al., 2020). In many cases, a task that is initially out of reach of an agent can be elaborated or

re-formulated into a task that the agent is familiar with. Through effective communication, the agent can help a human revise the task specification into a form that it can easily interpret and accomplish. As a motivating example, consider a home-assistant robot assigned a task of finding a newly bought *rice cooker* in a house, which it has never heard of and therefore does not know how to proceed. The robot, however, knows how to navigate to familiar landmarks in the house. In this case, the robot can ask a human it encounters a question like "*where is the rice cooker?*", expecting an instruction like "*find it in the kitchen, near the sink*", which it may know how to execute. By communicating with the human, the robot can convert the initially impossible task into a more feasible one. On the other hand, by giving the right information to the agent, a human can lift its task performance without needing to collect extra data and acquire expertise to upgrade its model.

We present HARI: **H**uman-**A**ssisted **R**einforced **I**nteraction, a general, POMDP-based framework that allows agents to effectively request and interpret information from humans. We identify and provide solutions to two fundamental problems: the *speaker* problem and the *listener* problem. The speaker problem concerns teaching an agent to elicit information from humans that can improve its decision making. Inspired the intention-based theory of human communication (Scott-Phillips, 2014; Sperber and Wilson, 1986; Tomasello et al., 2005), we equip a learning agent with a set of general information-seeking intentions that are helpful in solving any POMDP problem. At every time step, the agent can express to a human an intention of requesting additional information about (i) its current state, (ii) the goal state which it needs to reach, or (iii) a new subgoal state which, if reached, helps it make progress on the main goal. The agent learns to decide what information to request through reinforcement learning, by learning how much each type of information enhances its performance in a given situation. The agent's request decisions are thus grounded in its own perception of its (un)certainties about the current situation and are

optimized to be maximally contextually useful to it.

Our approach contrasts with methods to train agents to ask questions by mimicking those asked by humans (De Vries et al., 2017; Labutov et al., 2015; Liu et al., 2020; Mostafazadeh et al., 2016; Rao and Daumé III, 2018; Thomason et al., 2020). While effective in some situations, this approach has a potential significant drawback: questions asked by humans may not always be contextually useful for agents. For example, humans are experts in recognizing objects, thus rarely ask questions like "*what objects are next to me*?" Meanwhile, such questions may be helpful for robot localization, as robots may have imperfect visual perception in unfamiliar environments. Rather than focusing on naturalness, we aim to endow an agent with the cognitive capability of determining which type of information would be contextually useful to itself. Such capability requires an understanding of the casual effects of the agent's decisions on its future performance, which is exactly the knowledge acquired through reinforcement learning.

In addition to being able to *ask* for useful information, the agent must be able to incorporate the information it receives into its decision-making process: the *listener* problem. Humans can offer various types of assistance and may express them using diverse media (e.g., language, gesture, image). HARI implements a flexible communication protocol that is defined by the agent's task-solving policy: information coming from the human is given as state descriptions that the policy can take as input and map into actions. Hence, the space of information that the human can convey is as rich as the input space of the policy. Our design takes advantage of the flexibility of constructing an agent policy: (i) the policy can be further trained to interpret new information from humans, and (ii) it can leverage modern neural network architectures to be able to encode diverse forms of information. This contrasts with existing approaches based on reinforcement learning (RL) or imitation learning (IL) (Judah et al., 2010, 2014; Knox and Stone, 2009; Torrey

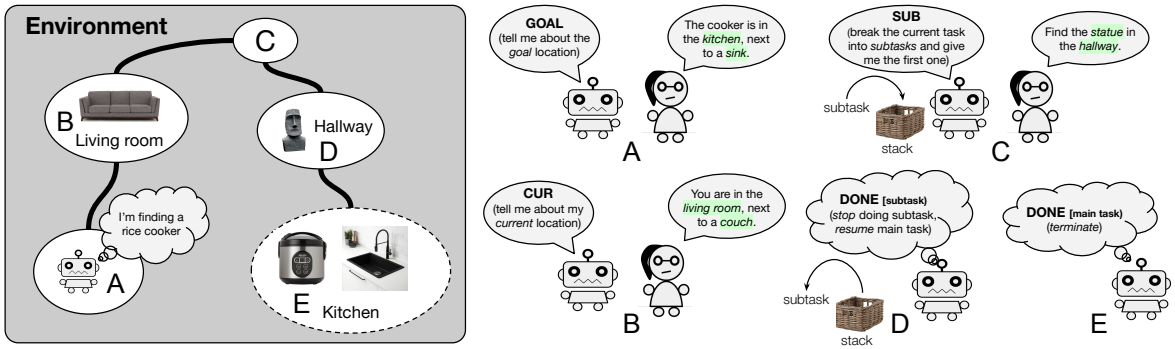


Figure 4.4: An illustration of the HARI framework in a navigation task. The agent can only observe part of an environment and is asked to find a rice cooker. An assistant communicates with the agent and can provide information about the environment and the task. Initially (A) the agent may request information about the goal, but may not know where it currently is. For example, at location B, due to limited perception, it does not recognize that it is next to a couch in a living room. It can obtain such information from the assistant. If the current task becomes too difficult (like at location C), the agent can ask the assistant for a simpler subtask that helps it make progress toward the goal. When the agent receives a subtask, it pushes the subtask to the top of a goal stack, and when it decides to stop executing a subtask, it pops the subtask from the stack (e.g., at location D). At location E, the agent empties the stack and terminates its execution.

and Taylor, 2013), in which feedback to the agent is limited to scalar feedback of rewards (in RL) or actions (in IL).

To evaluate our framework, we simulate a human-assisted navigation problem where an agent can request information about the environment from a human. Our agent learns an *intention policy* to decide at each step whether it wants to request additional information, and, if so, which type of information it wants to obtain. On tasks in previously unseen environments, the ability to ask for help improves the agent's success rate by 7$\times$ compared to performing tasks on its own. This human-assisted agent even outperforms an agent that has perfect perception and goal descriptions in unseen environments, thanks to the ability to simplify difficult tasks via requesting subgoals.

4.3.2 Preliminaries

We consider an environment defined by a partially observed Markov decision process (POMDP) $E = (\mathcal{S}, \mathcal{A}, T, c, \mathcal{D}, \rho)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, cost function $c : (\mathcal{S} \times \mathcal{S}) \times \mathcal{A} \to \mathbb{R}$, description space $\mathcal{D}$, and description function $\rho : \mathcal{S} \to \Delta(\mathcal{D})$, where $\Delta(\mathcal{Y})$ denotes the set of probability distributions over $\mathcal{Y}$.[4] We refer to this as the *execution environment*, where the agent performs tasks—e.g., a navigation environment.

A (goal-reaching) *task* is a tuple (s_1, g_1, d_1^g) where s_1 is the start state, g_1 is the goal state, and d_1^g is a limited description of g_1.[5] Initially, a task (s_1, g_1, d_1^g) is sampled from a distribution $\mathfrak{T}$. An agent starts in s_1 and is given the goal description d_1^g. It must reach the goal state g_1 within H steps. Let g_t and d_t^g be the goal state and goal description executed at time step t, respectively. In a standard POMDP, $g_t = g_1$ and $d_t^g = d_1^g$ for $1 \leq t \leq H$. Later, we will enable the agent to set new goals via communication with humans.

At time t, the agent does not know its state s_t but only receives a state description $d_t^s \sim \rho(s_t)$. Given d_t^s and d_t^g, the agent makes a decision $a_t \in \mathcal{A}$, transitions to the next state $s_{t+1} \sim T(s_t, a_t)$, and incurs an *operation cost* $c_t \triangleq c(s_t, g_t, a_t)$. When the agent takes the a_{done} action to terminate its execution, it receives a *task error* $c(s_t, g_t, a_{\text{done}})$ that indicates how far it is from actually completing the current task. The agent's goal is to reach g_1 with minimum total cost $C(\tau) = \sum_{t=1}^{H} c_t$, where $\tau = (s_1, d_1^s, a_1, \ldots, s_H, d_H^s)$ is an execution of the task.

We denote by b_t^s a belief state—a representation of the history $(d_1^s, a_1, \ldots, d_t^s)$—and by $\mathcal{B}$

[4] We say "description" in lieu of "observation" to emphasize (i) the description can come in varied modalities (e.g., image or text), and (ii) it can be obtained via perception as well as communication.

[5] Our notion of "goal state" refers not only to physical goals (e.g. a location in a house), but also to abstract states (e.g. a level of house cleanliness).

the set of all belief states. The agent maintains an *execution policy* $\hat{\pi} : \mathcal{B} \times \mathcal{D} \to \Delta(\mathcal{A})$. The learning objective is to estimate an execution policy that minimizes the expected total cost of performing tasks:

$$\min_{\pi} \mathbb{E}_{(s_1, g_1, d_1^g) \sim \mathfrak{T}, \tau \sim P_\pi(\cdot | s_1, d_1^g)} [C(\tau)] \tag{4.18}$$

where $P_\pi(\cdot \mid s_1, d_1^g)$ is the distribution over executions generated by a policy π given start state s_1 and goal description d_1^g. In a standard POMDP, an agent performs tasks on its own, without asking for any external assistance.

4.3.3 Leveraging Assistance via Communication

For an agent to accomplish tasks beyond its autonomous capabilities, we introduce an *assistant*, who can provide rich information about the environment and the task. We then describe how the agent requests and incorporates information from the assistant. Figure 4.4 illustrates an example communication between the agent and the assistant on an example object-finding task. Our formulation is inspired by the intention-based theory of human communication (Scott-Phillips, 2014; Sperber and Wilson, 1986; Tomasello et al., 2005), which characterizes communication as the expression and recognition of intentions in context. In this section, we draw connections between the theory and elements of POMDP learning, which allows us to derive reinforcement learning algorithms to teach the agent to make intentional requests.

Assistant. We assume an ever-present assistant who knows the agent's current state s_t, the goal state g_t, and the optimal policy $\pi^\star$. Their first capability is to provide a description of a state, defined by a function $\rho_A : \mathcal{S} \times \mathcal{D} \to \Delta(\mathcal{D})$ where $\rho_A(d' \mid s, d)$ specifies the probability of giving

d' to describe state s given a current description d, which can be empty. The second capability is to propose a subgoal of a current goal, specified by a function $\omega_A : \mathcal{S} \times \mathcal{S} \to \Delta(\mathcal{S})$, where $\omega_A(g' \mid s, g)$ indicates the probability of proposing g' as a subgoal given a current state s and a goal state g.

Common ground. Common ground represents mutual knowledge between the interlocutors and is a prerequisite for communication (Clark and Brennan, 1991; Stalnaker, 2002). In our context, knowledge of the agent is contained in its execution policy $\hat{\pi}$, while knowledge of the assistant is given by $\pi^\star$. When $\hat{\pi}$ and $\pi^\star$ maps to the same action distribution given an input (b^s, d^g), we say that the input belongs to the common ground of the agent and the assistant. Substantial common ground is needed for effective communication to take place. Thus, we assume that execution policy has been pre-trained to a certain level of performance so that there exists a non-empty subset of inputs on which the outputs of $\hat{\pi}$ and $\pi^\star$ closely match.

4.3.3.1 The Listener Problem: Incorporating Rich Information Provided by the Assistant

Upon receiving a request from the agent, the assistant replies with a state description $d \in \mathcal{D}$. The generality of our notion of state description (see §4.3.2) means that the assistant can provide information in any medium and format, so long as it is compatible with the input interface of $\hat{\pi}$. This reflects that only information interpretable by the agent is useful.

Concretely, the assistant can provide a new current-state description d^s_{t+1}, which the agent appends to its history to compute a new belief state b^s_{t+1} (e.g., using a recurrent neural network). The assistant can also provide a new goal description d^g_{t+1}, in which case the agent simply replaces

the current goal description d_t^g with the new one. This formulation allows the human-agent communication protocol to be augmented in two ways: (i) enlarging the common ground by training $\hat{\pi}$ to interpret new state descriptions or (ii) designing the agent model architecture to support new types of input information. In comparison, frameworks that implement a reinforcement learning- or imitation learning-based communication protocol (e.g., Knox and Stone, 2009; Torrey and Taylor, 2013) allow humans to only give advice using low-bandwidth media like rewards or primitive actions.

4.3.3.2 The Speaker Problem: Requesting Contextually Useful Information

Asking Questions as a Cognitive Capability. Asking questions is a cognitive process motivated by a person's self-recognition of knowledge deficits or common ground mismatches (Graesser et al., 1992). The intention-based theory of human communication suggest that a question, like other communicative acts, should convey an information-seeking intention that is grounded in the speaking context. In the case of question generation, the speaker's decision-making policy should be included in the context because, ultimately, a speaker asks a question to make better decisions.

Approaches that teach an agent to mirror questions asked by humans (e.g., De Vries et al., 2017), do not consider the agent's policy as part of the context for generating the question. The questions selected by humans may not be useful for the learning agent. To address this issue, we endow the agent with the cognitive capability of anticipating how various types of information would affect its future performance. The agent learns this capability using reinforcement learning, via *interacting* with the assistant and the environment, rather than imitating human behaviors.

Information-Seeking Intentions. While humans possess capabilities that help them come up with questions, it is unclear how to model those capabilities. Some approaches (e.g., Mostafazadeh et al., 2016; Rao and Daumé III, 2018) rely on pre-composed questions, which are domain-specific. We instead endow the agent with a set of intentions that correspond to speech acts in embodied dialogue Thomason et al. (2020). They are relevant to solving general POMDPs, while being agnostic to the problem domain and the implementation of the execution policy:

1. CUR: requests a new description of the current state s_t and receives $d^s_{t+1} \sim \rho_A(\cdot \mid s_t, d^s_t)$;
2. GOAL: requests a new description of the current goal g_t and receives $d^g_{t+1} \sim \rho_A(\cdot \mid g_t, d^g_t)$;
3. SUB: requests a description of a subgoal and receives $d^g_{t+1} \sim \rho_A(\cdot \mid g_{t+1}, \emptyset)$ where the subgoal $g_{t+1} \sim \omega_A(\cdot \mid s_t, g_t)$ and $\emptyset$ is an empty description.

We leave constructing more specific intentions (e.g., asking about a specific input feature) to future work.

Intention Selection. To decide which intention to invoke, the agent learns an intention policy ψ_θ, which is itself a function of the execution policy $\hat{\pi}$. The intention policy's action space consists of five intentions: $\bar{\mathcal{A}} = \{\text{CUR}, \text{GOAL}, \text{SUB}, \text{DO}, \text{DONE}\}$. The first three actions convey the three intentions defined previously. The remaining two actions are used for traversing in the environment:

4. DO: executes the most-probable action $a^{\text{do}}_t \triangleq \text{argmax}_{a \in \mathcal{A}} \hat{\pi}(a \mid b^s_t, d^g_t)$. The agent transitions to a new state $s_{t+1} \sim T(s_t, a^{\text{do}}_t)$;
5. DONE: decides that the current goal g_t has been reached. If g_t is the main goal ($g_t = g_1$), the episode ends. If g_t is a subgoal ($g_t \neq g_1$), the agent may choose a new goal to follow.

In our implementation, we feed the hidden features and the output distribution of $\hat{\pi}$ as inputs to ψ_θ so the agent can use its execution policy in choosing its intentions. Later we will specify the input space of the interaction policy and how the next subgoal is chosen (upon DONE), as these details depend on how the agent implements its goal memory. The next section introduces an instantiation where the agent uses a stack data structure to manage the goals it receives.

4.3.4 Learning When and What to Ask

In this section, we formalize the HARI framework for learning to select information-seeking intentions. We construct the POMDP environment that the intention policy acts in, referred to as the *intention environment* (§ 4.3.4.1) to distinguish with the environment that the execution policy acts in. Our construction employs a *goal stack* to manage multiple levels of (sub)goals (§ 4.3.4.2). A goal stack stores all the (sub)goals the agent has been assigned but has not yet decided to terminate. It is updated in every step depending on the selected intention. In §4.3.4.4, we design a cost function that trades off between taking few actions and completing tasks.

4.3.4.1 Intention Environment

Given an execution environment $E = (\mathcal{S}, \mathcal{A}, T, c, \mathcal{D}, \rho)$, the intention environment is a POMDP $\bar{E} = (\bar{\mathcal{S}}, \bar{\mathcal{A}}, \bar{T}, \bar{c}, \bar{\mathcal{D}}, \bar{\rho})$:

- **State space** $\bar{\mathcal{S}} = \mathcal{S} \times \mathcal{D} \times \mathcal{G}_L$, where $\mathcal{G}_L$ is the set of all goal stacks of at most L elements. Each state $\bar{s} = (s, d^s, G) \in \bar{\mathcal{S}}$ is a tuple of an execution state s, description d^s, and goal stack G. Each element in the goal stack G is a tuple of a goal state g and description d^g;
- **Action space** $\bar{\mathcal{A}} = \{\text{CUR}, \text{GOAL}, \text{SUB}, \text{DO}, \text{DONE}\}$;

- **State-transition** function $\bar{T} = T_s \cdot T_G$ where $T_s : \mathcal{S} \times \mathcal{D} \times \bar{\mathcal{A}} \to \Delta(\mathcal{S} \times \mathcal{D})$ and $T_G : \mathcal{G}_L \times \bar{\mathcal{A}} \to \Delta(\mathcal{G}_L)$;
- **Cost function** $\bar{c} : (\mathcal{S} \times \mathcal{G}_L) \times \bar{\mathcal{A}} \to \mathbb{R}$, defined in §4.3.4.4 to trade off operation cost and task error;
- **Description space** $\bar{\mathcal{D}} = \mathcal{D} \times \mathcal{G}_L^d$ where $\mathcal{G}_L^d$ is the set of all goal-description stacks of size L. The agent cannot access the environment's goal stack G, which contains true goal states, but only observe descriptions in G. We call this partial stack a *goal-description stack*, denoted by G^d;
- **Description function** $\bar{\rho} : \bar{\mathcal{S}} \to \bar{\mathcal{D}}$, where $\bar{\rho}(\bar{s}) = \bar{\rho}(s, d^s, G) = (d^s, G^d)$. Unlike in the standard POMDP formulation, this description function is deterministic.

A belief states $\bar{b}_t$ of the intention environment summarizes a history $(\bar{s}_1, \bar{a}_1, \cdots, \bar{s}_t)$. We define the intention policy as $\psi_\theta : \bar{\mathcal{B}} \to \Delta(\mathcal{A})$, where $\bar{\mathcal{B}}$ is the set of all belief states.

4.3.4.2 Goal Stack

The goal stack is a list of tasks that the agent has not declared complete. The initial stack $G_1 = \{(g_1, d_1^g)\}$ contains the main goal and its description. At time t, the agent executes the goal at the top of the current stack, i.e. $g_t = G_t.\text{top}()$. Only the GOAL, SUB, and DONE actions alter the stack. GOAL replaces the top goal description of G_t with the new description d_{t+1}^g from the assistant. SUB, which is only available when the stack is not full, pushes a new subtask (g_{t+1}, d_{t+1}^g) to G_t. DONE pops the top element from the stack. The goal-stack transition function is $T_G(G_{t+1} \mid G_t, \bar{a}_t) = \mathbb{1}\{G_{t+1} = G_t.\texttt{update}(\bar{a}_t)\}$ where $\mathbb{1}\{.\}$ is an indicator and $G_t.\text{update}(a)$ is the updated stack after action a is taken.

4.3.4.3 State Transition

The transition function T_s is factored into two terms:

$$T_s(s_{t+1}, d^s_{t+1} \mid s_t, d^s_t, \bar{a}_t) = \underbrace{P(s_{t+1} \mid s_t, \bar{a}_t)}_{\triangleq p_{\text{state}}(s_{t+1})} \cdot \underbrace{P(d^s_{t+1} \mid s_{t+1}, d^s_t, \bar{a}_t)}_{\triangleq p_{\text{desp}}(d^s_{t+1})} \tag{4.19}$$

Only the DO action changes the execution state, with: $p_{\text{state}}(s_{t+1}) = T\left(s_{t+1} \mid s_t, a^{\text{do}}_t\right)$, where a^{do}_t is the action chosen by $\hat{\pi}$ and T is the execution environment's state transition function. The current-state description is altered when the agent requests a new current-state description (CUR), where $p_{\text{desp}}(d^s_{t+1}) = \rho_A(d^s_{t+1} \mid s_{t+1}, d^s_t)$, or moves (DO), where $p_{\text{desp}}(d^s_{t+1}) = \rho(d^s_{t+1} \mid s_{t+1})$ (here, ρ is the description function of the execution environment).

4.3.4.4 Cost Function

The intention policy needs to trade-off between two types of cost: the cost of operation (making information requests and traversing in the environment), and the cost of not completing a task (task error). The two types of cost are in conflict: the agent may lower its task error if it is willing to suffer a larger operation cost by making more requests to the assistant.

We employ a simplified model where all types of cost are non-negative real numbers of the same unit. Making a request of type a is assigned a constant cost γ_a. The cost of taking the DO action is $c(s_t, a^{\text{do}}_t)$, the cost of executing the a^{do}_t action in the environment. Calling DONE to terminate execution of the main goal g_1 incurs a task error $c(s_t, a_{\text{done}})$. We exclude the task errors of executing subgoals because the intention policy is only evaluated on reaching the main goal.

The cost function is concretely defined as follows

$$\bar{c}(s_t, G_t, \bar{a}_t) = \begin{cases} c(s_t, g_t, a_t^{\text{do}}) & \text{if } \bar{a}_t = \textsc{Do}, \\ \gamma_{\bar{a}_t} & \text{if } \bar{a}_t \notin \{\textsc{Do}, \textsc{Done}\}, \\ c(s_t, g_t, a_{\text{done}}) & \text{if } \bar{a}_t = \textsc{Done}, |G_t| = 1, \\ 0 & \text{if } \bar{a}_t = \textsc{Done}, |G_t| > 1 \end{cases}$$

The magnitudes of the costs naturally specify a trade-off between operation cost and task error. For example, setting the task errors much larger than the other costs indicates that completing tasks is prioritized over taking few actions.

To facilitate learning, we apply reward shaping (Ng et al., 1999), augmenting a cost $c(s, G, a)$ with $\Phi(s, g)$, the task error received if the agent terminated in s. The cost received at time step t is $\tilde{c}_t \triangleq \bar{c}_t + \Phi(s_{t+1}, g_{t+1}) - \Phi(s_t, g_t)$. We assume that after the agent terminates the main goal, it transitions to a special terminal state $s_{\text{term}} \in \mathcal{S}$ and remains there. We set $\Phi(s_{\text{term}}, \texttt{None}) = 0$, where $g_t = \texttt{None}$ signals that the episode has ended. The cumulative cost of an execution under the new cost function is

$$\sum_{t=1}^{H} \tilde{c}_t = \sum_{t=1}^{H} \Big[\bar{c}_t + \Phi(s_{t+1}, g_{t+1}) - \Phi(s_t, g_t)\Big] \tag{4.20}$$

$$= \sum_{t=1}^{H} \bar{c}_t - \Phi(s_1, g_1) \tag{4.21}$$

Since $\Phi(s_1, g_1)$ does not depend on the action taken in s_1, minimizing the new cumulative cost does not change the optimal policy for the task (s_1, g_1).

4.3.5 Experimental Setup: Modeling Human-Assisted Navigation

Problem. We apply HARI to modeling a human-assisted navigation (HAN) problem, in which a human requests an agent to find an object in an indoor environment. Each task request asks the agent to go to a room of type r and find an object of type o (e.g., find a mug in a kitchen). The agent shares its current view with the human (e.g. via an app). We assume that the human is familiar with the environment and can recognize the agent's location. Before issuing a task request, the human imagines a goal location (not revealed to the agent). We are interested in evaluating success in *goal-finding*, i.e. whether the agent can arrive at the human's intended goal location. Even though there could be multiple locations that match a request, the agent only succeeds if it arrives exactly at the chosen goal location.

Environment. We construct the execution environments using the house layout graphs provided by the Matterport3D simulator (Anderson et al., 2018b). Each graph is generated from a 3D model of a house where each node is a location in the house and each edge connects two nearby unobstructed locations. At any time, the agent is at a node of a graph. Its action space $\mathcal{A}$ consists of traversing to any of the nodes that are adjacent to its current node.

Scenario. We simulate the scenario where the agent is pre-trained in simulated environments and then deployed in real-world environments. Due to mismatches between the pre-training and real-world environments, the capability of the agent degrades at deployment time. It may not reliably recognize objects, the room it is currently in, or the intent of a request. However, a simulated human assistant is available to provide additional information about the environment and the task, which helps the agent relate its current task to the ones it has learned to fulfill during

pre-training.

Subgoals. If the agent requests a subgoal, the assistant describes a new goal roughly halfway to its current goal (but limited to a maximum distance l_{max}). Specifically, let p be the shortest path from the agent's current state s_t to the current goal g_t, and p_i be the i-th node on the path ($0 \leq i < |p|$). The subgoal location is chosen as p_k where $k = \min(\lfloor |p|/2 \rfloor, l_{\text{max}})$, where l_{max} is a pre-defined constant.

State Description. We employ a discrete bag-of-features representation for state descriptions.[6] A bag-of-feature description (d^s or d^g) emulates the information that the agent has extracted from a raw input that it perceives (e.g., an image, a language sentence). Concretely, we assume that when the agent sees a view, it detects the room and nearby objects. Similarly, when it receives a task request or a human response to its request, it identifies information about rooms, objects, and actions in the response. Working with this discrete input representation allows us to easily simulate various types and amounts of information given to the agent.

Specifically, we model three types of input features: the name of a room, information about an object (name, distance and direction relative to a location), and the description of a navigation action (travel distance and direction). The human can supply these types of information to assist the agent. We simulate two settings of descriptions: *dense* and *sparse*. A dense description is a variable-length lists containing the following features: the current room's name and features of at most 20 objects within five meters of a viewpoint. Sparse descriptions represent imperfect perception of the agent in real-world environments. A sparse description is derived by first

[6]While our representation of state descriptions simplifies the object/room detection problem for the agent, it does not necessarily make the navigation problem easier than with image input, as images may contain information that is not captured by our representation (e.g., object shapes and colors, visualization of paths).

constructing a dense description and then removing the features of objects that are not in the top 100 most frequent (out of ∼1K objects). The sparse description of the current location (d^s) does not contain the room name, but that of the goal location (d^g) does. As each description is a sequence of feature vectors, whose length varies depend on the location, we use a Transformer model (Vaswani et al., 2017) to be able to encode such inputs into continuous feature vectors. Details about the feature representation and the model architecture are provided in the Appendix.

Experimental Procedure. We conduct our experiments in three phases. In the *pre-training* phase, the agent learns an execution policy $\hat{\pi}$ with access to only dense descriptions. This emulates training in a simulator that supplies rich information to the agent.

In the *training* phase, the agent is only given sparse descriptions of its current and goal locations. This mimics the degradation of the agent's perception about the environment and the task when deployed in real-world conditions. In this phase, the agent can request dense descriptions from the human. Upon a request for information about a (goal or current) location, the human gives a dense description with room and object features of that location. However, when the agent requests a subgoal (SUB) *and* is adjacent to the subgoal that the human wants to direct it to, the human simply gives a description of the next optimal navigation action to get to that location. We use advantage actor-critic (Mnih et al., 2016) to learn an intention policy ψ_θ that determines which type of information to request. The intention policy is now trained in environment graphs that are previously seen as well as unseen during the pre-training phase.

Finally, in the *evaluation* phase, the agent is tested on three conditions: seen environment and target object type but starting from a new room (UNSEENSTR), seen environment but new target object type (UNSEENOBJ), and unseen environment (UNSEENENV). The execution policy

Table 4.7: Test success rates and the average number of different types of actions taken by the agent (on all task types).

Agent	Success Rate % ↑ Unseen Start	Unseen Object	Unseen Env.	Avg. number of actions ↓ CUR	GOAL	SUB	DO
Rule-based intention policy ψ_θ (when to call DONE is decided by the execution policy $\hat{\pi}$)							
(d^s: current-state description, d^g: goal description)							
No assistance (always DO until DONE)	43.4	16.4	3.0	-	-	-	13.1
Dense d^g (GOAL then always DO until DONE)	67.2	56.6	9.7	-	1.0	-	12.6
Dense d^s (always CUR then DO until DONE)	77.9	30.6	4.1	12.0	-	-	12.0
Dense d^g and d^s (GOAL then always CUR then DO until DONE)	97.8	81.7	9.4	11.0	1.0	-	11.0
Random + rules to match with # of actions of learned-RL ψ_θ	78.8	68.5	12.7	2.0	1.0	1.7	11.3
learned-RL intention policy ψ_θ							
With pre-trained navigation policy $\hat{\pi}$ (ours)	85.8	78.2	19.8	2.1	1.0	1.7	11.1
With uncooperative assistant (change a request randomly to CUR, GOAL or SUB)	81.1	71.2	16.1	2.7	2.7	1.5	9.6
With perfect navigation policy on sub-goals (skyline)	94.3	95.1	92.6	0.0	0.0	6.3	7.3

$\hat{\pi}$ is fixed during the training and evaluation phases. We created 82,104 examples for pre-training, 65,133 for training, and approximately 2,000 for each validation or test set. Additional details about the training procedure and dataset are included in the Appendix §E.

4.3.6 Results

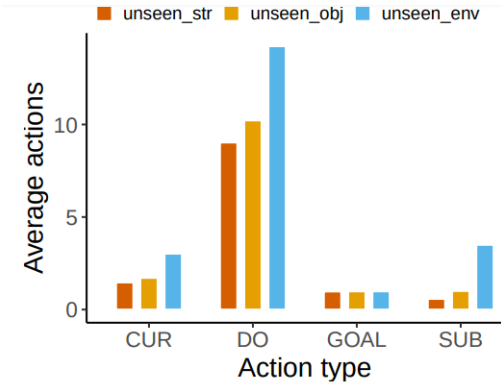
Settings. In our main experiments, we set: the cost of taking a CUR, GOAL, SUB, or DO action to be 0.01 (we will consider other settings subsequently), the cost of calling DONE to terminate the main goal (i.e. task error) equal the (unweighted) length of the shortest-path from the agent's location to the goal, and the goal stack's size (L) to be 2.

We compare our learned-RL intention policy with several rule-based *baselines*. Their descriptions are given in Table 4.7. We additionally construct a strong baseline that first takes the GOAL action (to clarify the human's intent) and then selects actions in such a way to match the distribution of actions taken by our RL-trained policy. In particular, this policy is constrained to

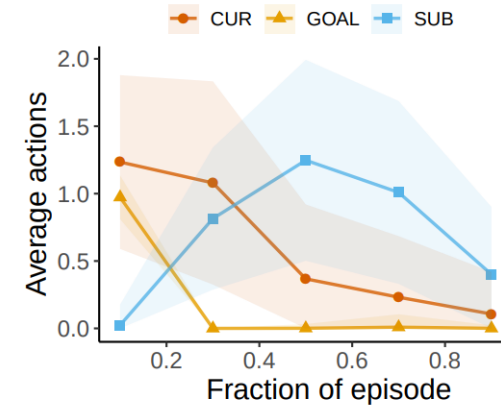
take at most $\lfloor X_a \rfloor + y$ actions of type a, where $y \sim \text{Bernoulli}(X_a - \lfloor X_a \rfloor)$ and X_a is a constant tuned on the validation set so that the policy has the same average count of each action as the learned-RL policy. To prevent early termination, we enforce that the rule-based policy cannot take more DONE actions than SUB actions unless its SUB action's budget is exhausted. When there is no constraint on taking an action, the policy chooses a random action in the set of available actions. Our learned-RL policy can only outperform this policy by being able to determine contextually useful information to request, which is exactly the capability we wish to evaluate. We also construct a *skyline* where the intention policy is also learned by RL but with an execution policy that always fulfill subgoals perfectly.

In the problem we construct, different types of information are useful in different contexts (Table 4.7). Comparing the rule-based baselines reveals the benefits of making each type of request. Overall, we observe that information about the current state is helpful on only tasks in seen environments (UNSEENSTR and UNSEENOBJ). Information about the goal greatly improves performance of the agent in unseen environments (dense-d^g outperforms no-assistance by 6.4% in UNSEENENV). Dense-d^g outperforms dense-d^s in UNSEENOBJ but underperforms in UNSEENSTR, showing that goal information is more useful than current-state information in finding new objects but less useful in finding seen ones. The two types of information is complementary: dense-d^g-and-d^s improves success rate versus dense-d^g and dense-d^s in most evaluation conditions.

We expect subgoal information to be valuable mostly in unseen environments. This is confirmed by the superior performance of our learned-RL policy, which can request subgoals, over the rule-based baselines, which do not have this capability, in those environments.

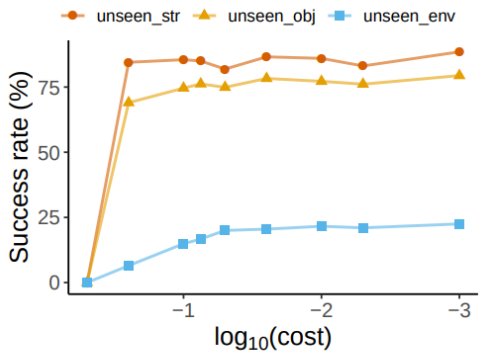


(a) Subgoals are requested much more in unseen environments.

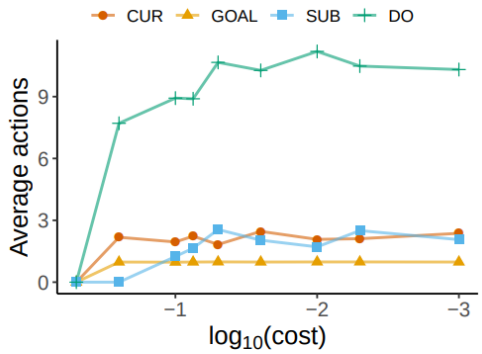


(b) Subgoals are requested in the middle, goal information at the beginning.

Figure 4.5: Analyzing the behavior of the learned-RL intention policy (on validation environments).



(a) Effect of cost on success.



(b) Effect of cost on actions.

Figure 4.6: Analyzing the effect of simultaneously varying the cost of the CUR, GOAL, SUB, DO actions (on validation environments), thus trading off success rate versus number of actions taken.

Our learned-RL policy learns the advantages of each type of information (Table 4.7 & Figure 4.5a). Aided by our learned-RL policy, the agent observes a substantial $\sim 2\times$ increase in success rate on UNSEENSTR, $\sim 5\times$ on UNSEENOBJ, and $\sim 7\times$ on UNSEENENV, compared to when performing tasks without assistance. This result indicates that the assistance-requesting skills are increasingly more helpful as the tasks become more unfamiliar. Figure 4.5a shows the request pattern of the policy in each evaluation condition. Our policy learns that it is not useful to make more than one GOAL request. It relies increasingly on CUR and SUB requests in more

difficult conditions (UNSEENOBJ and UNSEENENV). The policy makes about 3.5× more SUB requests in UNSEENENV than in UNSEENOBJ. Specifically, with the capability of requesting subgoals, the agent impressively doubles the success rate of the dense-d^g-and-d^s policy in unseen environments, which *always* has access to dense descriptions in these environments. Moreover, our policy does not have to bother the assistant in every time step: only $1/4$ of its actions are requests for information.

Our learned-RL policy selects contextually useful requests (Table 4.7, bottom half & Figure 4.5b). The policy is significantly more effective than the rule-based baseline that uses the same number of average actions per episode (+7.1% on UNSEENENV). Note that we enforce rules so that this baseline only differs from our learned-RL policy in where it places CUR and SUB requests. Thus, our policy has gained advantages by making these requests in more appropriate situations. To further showcase the importance of being able to obtain the right type of information, we use RL to train a policy with an *uncooperative* assistant, who disregards the agent's request intention and instead replies to an intention randomly selected from $\{\text{CUR}, \text{GOAL}, \text{SUB}\}$. As expected, performance of the agent drops in all evaluation conditions. This shows that our agent has effectively leveraged the cooperative assistant by making useful requests.

We also observe that the agent adapts its request strategy through the course of an episode. As observed in Figure 4.5b. the GOAL action, if taken, is always taken only once and immediately in the first step. The number of CUR actions gradually decreases over time. The agent makes most SUB requests in the middle of an episode, after its has attempted but failed to accomplish the main goals. We observe similar patterns on the other two validation sets.

Finally, results obtained by the skyline shows that further improving performance of the

Table 4.8: Success rates and numbers of actions taken with different stack sizes (on validation). Larger stack sizes significantly aid success rates in unseen environments, but not in seen environments.

Stack size	Success rate (%) ↑ Unseen Start	Unseen Obj.	Unseen Env.	Average # of actions ↓ CUR	GOAL	SUB	DO
1 (no subgoals)	92.2	78.4	12.5	5.1	1.9	0.0	10.7
2	86.9	77.6	21.6	2.1	1.0	1.7	11.2
3	83.2	78.6	33.5	1.3	1.0	5.0	8.2

execution policy on short-distance goals would effectively enhance the agent's performance on long-distance goals.

Effects of Varying Action Cost (Figure 4.6). We assign the same cost to each CUR, GOAL, SUB, or DO action. Figure 4.6a demonstrates the effects of changing this cost on the success rate of the agent. An equal cost of 0.5 makes it too costly to take any action, inducing a policy that always calls DONE in the first step and thus fails on all tasks. Overall, the success rate of the agent rises as we reduce the action cost. The increase in success rate is most visible in UNSEENENV and least visible in UNSEENSTR. Similar patterns are observed with various time limits ($H = 10, 20, 30$). Figure 4.6b provides more insights. As the action cost decreases, we observe a growth in the number of SUB and DO actions taken by the intention policy. Meanwhile, the numbers of CUR and GOAL actions are mostly static. Since requesting subgoals is more helpful in unseen environments, the increase in the number of SUB actions leads the more visible boost in success rate on UNSEENENV tasks.

Recursively Requesting Subgoals of Subgoals (Table 4.8). In Table 4.8, we test the functionality of our framework with a stack size 3, allowing the agent to request subgoals of subgoals. As expected, success rate on UNSEENENV is boosted significantly (+11.9% compared to using a

stack of size 2). Success rate on UNSEENOBJ is largely unchanged; we find that the agent makes more SUB requests on those tasks (averagely 4.5 requests per episode compared to 1.0 request made when the stack size is 2), but doing so does not further enhance performance. The agent makes less CUR requests, possibly in order to offset the cost of making more SUB requests, as we keep the action cost the same in these experiments. Due to this behavior, success rate on UNSEENSTR declines with larger stack sizes, as information about the current state is more valuable for these tasks than subgoals. These results show that the critic model overestimates the V values in states where SUB actions are taken, leading to the agent learning to request more subgoals than needed. This suggests that the our critic model is not expressive enough to encode different stack configurations.

4.3.7 Related Work

Knowledge Transfer in Reinforcement Learning. Various frameworks have been proposed to model information transfer between humans and agents (Da Silva and Costa, 2019). A basic human-agent communication protocol is instruction-following (Alomari et al., 2017; Anderson et al., 2018b; Bisk et al., 2016; Chen and Mooney, 2011; Misra et al., 2018a; Yu et al., 2018), where agents execute a natural language request. For multiple-round communication, Torrey and Taylor (2013) introduce the action-advising framework where a learner decides situations where it needs to request reference actions from a teacher. This can be viewed as active learning (Angluin, 1988; Cohn et al., 1994) in an RL setting. The request policy can be constructed based on uncertainty measurements (Culotta and McCallum, 2005; Da Silva et al., 2020) or learned via RL (Fang et al., 2017). An agent-to-agent variant poses the problem of learning a teaching policy

in addition to the request policy (Da Silva et al., 2017; Omidshafiei et al., 2019; Zimmer et al., 2014). Overall, this literature uses IL-based communication protocols that assume the teacher shares a common action space with the learner and provides actions in this space in response to the agent's requests. Others employ standard RL communication protocols, where the human conveys knowledge through numerical scores or categorical feedback (Christiano et al., 2017; Griffith et al., 2013; Judah et al., 2010; Knox and Stone, 2009; Lee et al., 2021; Peng et al., 2016). In Maclin and Shavlik (1996), humans advise the agent with domain-specific language, specifying rules to incorporate into the agent's model.

Recent frameworks (Jiang et al., 2019; Kim et al., 2020; Nguyen and Daumé III, 2019; Nguyen et al., 2019b) allow the teacher to specify high-level goals instead of low-level actions or rewards. HARI generalizes these, allowing specification of subgoal information and descriptions of the current and goal states. Importantly, the framework enables the agent to convey specific intentions rather than passively receiving instructions or calling for generic help.

Information expressed in human language has been incorporated into RL frameworks to guide task execution (Hermann et al., 2017b) or assist with learning (Ammanabrolu et al., 2020)—see Luketina et al. (2019) for a thorough survey. Language-based inverse RL frameworks infer the reward function from language utterances (Akakzia et al., 2021; Fu et al., 2019; Sumers et al., 2020; Zhou and Small, 2021b). In contrast, we investigate human-agent communication *during* task execution—the agent does *not* update its task-executing policy. Similar to language-conditioned RL, we directly incorporate feedback as input to the agent's policy. This approach also bears a resemblance to work on learning from language description (Chan et al., 2019; Cideron et al., 2020; Colas et al., 2020; Jiang et al., 2019; Nguyen et al., 2021), except that the agent does not learn from the response and can incorporate more diverse feedback. Our set-

ting is closely related to He et al. (2013), where an agent selects a subset of useful input features, but we remove the assumption that features arrive in a fixed order.

Our work requires solving a hierarchical reinforcement learning problem (Chane-Sane et al., 2021; Kulkarni et al., 2016; Le et al., 2018; Sutton et al., 1999b). While standard formulations mostly consider two levels of goal, our stack-based implementation offers a convenient way to construct deeper goal hierarchies.

Task-Oriented Dialog and Natural Language Questions. HARI models a task-oriented dialog problem. Most problem settings require the agent to compose questions to gather information from humans (Das et al., 2017b; De Vries et al., 2017; de Vries et al., 2018; Narayan-Chen et al., 2019a; Padmakumar et al., 2022; Shi et al., 2022; Thomason et al., 2020). A related line of work is generating language explanations of model decisions (Camburu et al., 2018; Hendricks et al., 2016; Rajani et al., 2019). The dominant approach in these spaces is to mimic pre-collected human utterances. However, naively mirroring human behavior cannot enable agents to understand the limits of their knowledge. We take a different approach, teaching the agent to understand its intrinsic needs through interaction. Teaching agents to act and communicate with intentions is understudied (Karimpanal, 2021). Notable work in this space proposes a computational theory-of-mind for agents to enable them to reason about their own mental states and those of others (Andreas and Klein, 2016; Bara et al., 2021; Fried et al., 2018a; Kang et al., 2020; Roman Roman et al., 2020; Zhu et al., 2021). Our work is not concerned about designing theory-of-mind models, instead concentrating on highlighting the importance of interaction in the learning of intentional behavior.

4.3.8 Conclusion

Our framework can theoretically capture rich human-agent communication, and even communication with other knowledge sources like search engines or language models (Liu et al., 2022). An important empirical question is how well it generalizes to more realistic interactions and environments (e.g., (Shridhar et al., 2020)). Especially when deploying with real humans, using human simulations to pre-train the agent can help reducing human effort. Large language models (Brown et al., 2020b; Chowdhery et al., 2022b; Rae et al., 2021) can be leveraged for constructing such simulators. Towards generating more specific, natural questions, methods for generating faithful explanations (Kumar and Talukdar, 2020; Madsen et al., 2021), measuring feature importance (Das and Rad, 2020; Koh and Liang, 2017; Zeiler and Fergus, 2014), or learning the casual structure of black-box policies (Geiger et al., 2021) are relevant to investigate. Another exciting direction is to enable the agent to learn from the obtained information. We expect that, similar to curriculum-based learning (Wang et al., 2019b), the communication protocol of our framework would guide the agent to request and learn increasingly more difficult tasks over time.

Chapter 5: Conclusion and Future Directions

My dissertation has demonstrated the feasibility and benefits of enriching communication between humans and AI agents. I have proposed new frameworks for training and evaluating AI agents and have shown that these frameworks can serve as strong foundations for developing agents that are more helpful and safe for humans. The proposed frameworks represent tangible progress in tackling the two fundamental challenges in human-AI communication research mentioned in section § 1.3. Concretely, addressing the limitations of current machine learning frameworks, the ILIAD framework offers a new theoretical principle for learning from a specific type of language expressions that traditional frameworks cannot handle. Similarly, the HARI framework relaxes constraints on the form of human-generated information that an AI agent can process. Tackling the practical challenges in conducting human-AI communication research, I have introduced methods for constructing human simulations for training and evaluating my proposed algorithms. My approach is not meant to replace experiments with real humans, but to reduce the cost and risk of those experiments by quickly and cheaply bootstrapping the agents to a sufficient level of competence.

5.1 Limitations of Current Work

While my frameworks have broadened the communication channel between humans and agents, they consider simplified scenarios where the human speaks a restricted type of language and their intentions can be perfectly identified. For example, in ILIAD, the feedback is defaulted to describing the agent's activities and has to be drawn from the same distribution as the task request. Similarly, HARI assumes the human responds perfectly to the agent's requests, neglecting the problem of intention inference. Nevertheless, in real-world scenarios, humans may convey diverse intentions to transfer their knowledge and those intentions may not be straightforward to recognize. For example, suppose a person asks a robot to "*make tea*". When receiving the tea from the robot, the person says "*I don't like hot tea. Bring me some ice*!" This utterance conveys at least three types of feedback: (a) what is made is hot tea (descriptive feedback), (b) the assigned task is only partially fulfilled (evaluative feedback), (c) the agent should have brought iced tea (corrective/instructive feedback). Developing a unifying statistical learning framework that can incorporate all of these types of human intentions is necessary for enabling learning via natural conversations with humans.

While my agents can learn from rich, natural types of interactions with humans, they are highly inefficient, requiring thousands of interactions to achieve respectable performance. Their inefficiency can be attributed to the fact that they are not built in with fundamental knowledge about the world, human behavior, and natural language. Humans leverage such knowledge to enrich the meanings of their utterances. Without similar knowledge, the agents cannot effectively recognize human intentions and learn quickly from them. Models that are trained on vast amounts of data have demonstrated remarkable commonsense knowledge (Brown et al., 2020a;

Chowdhery et al., 2022a; Ramesh et al., 2021). However, a major challenge for employing these models is to adapt them for communication in specific contexts and environments.

Finally, while training and evaluating with human simulators is a promising direction to accelerate research, the main challenge of this approach is to construct faithful emulations of human traits. The simulators that I construct in this dissertation are only rudimentary model of real-life humans. To prioritize the naturalness of the language, I have restricted the simulators to being able to convey only a small set of intentions. I have also assumed that they are perfectly cooperative and rational. To guide the improvement of more expressive and faithful simulations, it is essential to characterize and estimate the divergence between these simulators and real humans, and the effect of the divergence on the generalizability of the agents to real-world scenarios. Empirically, the divergence can be quantified by replacing the simulators with real humans in the experiments. Theoretically, we can contrast the characteristics of the simulators with the characteristics of humans that are well-documented in cognitive science and psycho-linguistics. Frameworks developed in these fields can offer formal principles for enhancing the simulators.

In the next sections, I describe two specific problems that I believe will motivate solutions to the aforementioned limitations.

5.2 Learning from More Expressive Teachers

To design agents that can learn directly from humans, we should begin with constructing theoretical frameworks that employ realistic models of humans. There are two aspects about the human ways of teaching that are not adequately captured in current learning frameworks. The first aspect is that humans can teach in diverse ways. They can convey a variety of intentions and

use miscellaneous types of communication medium to express those intentions. For example, for tasks that are difficult to express in words, humans can demonstrate through actions. On the other hand, for tasks that are too tedious to demonstrate, humans can leverage the generalizability and compositionality of language to efficiently specify them. Most current frameworks employ teacher models that can only express a single intention using a single communication medium.

The second under-represented aspect is that human communication aims at manipulating mental states rather than external behaviors. In reinforcement and imitation learning, the teacher is a black box that generates learning signals; the underlying process for generating the signals is not specified in the formulation. Inverse reinforcement learning models a teacher that is driven by an internal reward function R_θ, which is defined by a mental-state vector θ. However, the teacher still communicates the mental-state vector to the learner through only action demonstrations. Extending this framework to modeling communication through multiple types of communication medium and intention is an exciting direction.

5.3 Learning to Be Aware of and Expressively Convey Specific Needs

Learning to ask questions to obtain contextually useful for completing a task is a problem whose solutions can significantly advance the communication and cognitive capabilities of AI agents. This problem differs from many current AI benchmarks in that the agent cannot simply imitate human behaviors to derive effective decisions, as a question must be derived from an agent's own perception of its knowledge deficits in the current context. In other words, given a situation, humans simply know what type of information that would be most useful for an agent. As a result, simply training an agent on large collections of human-generated data is not suf-

ficient. To ask good questions, the agent needs to develop cognitive perception about its own needs and be able to convey the needs in a human-intelligible way. The HARI work have shown that knowledge about intrinsic needs can be acquired through interaction with simulated humans. Nevertheless, the specificity of the agent questions and the realism of the human simulator in the work remains limited. My future work will aim to explore ways to improve these aspects. To generate more specific questions, I will look for ways to combine the interaction-based approach with the imitation-based approach. A promising approach is to first extract a set of general intentions from a dataset of human-generated questions and then use interactive learning to teach the agent to determine the most useful intention to convey in a given context. To augment realism of the human simulator, more powerful knowledge sources such as a search engine, a large language models can be leveraged (Liu et al., 2022).

Appendices

A Neural Machine Translation

Our neural machine translation (NMT) model consists of an encoder and a decoder, each of which is a recurrent neural network (RNN). We closely follow Luong et al. (2015) for the structure of our model. It directly models the posterior distribution $P_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x})$ of translating a source sentence $\boldsymbol{x} = (x_1, \cdots, x_n)$ to a target sentence $\boldsymbol{y} = (y_1, \cdots, y_m)$:

$$P_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{t=1}^{m} P_{\boldsymbol{\theta}}(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x}) \tag{A.1}$$

where $\boldsymbol{y}_{<t}$ are all tokens in the target sentence prior to y_t.

Each local disitribution $P_{\boldsymbol{\theta}}(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x})$ is modeled as a multinomial distribution over the target language's vocabulary. We compute this distribution by applying a linear transformation followed by a softmax function on the decoder's output vector $\boldsymbol{h}_t^{dec}$:

$$P_{\boldsymbol{\theta}}(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x}) = \text{softmax}(\boldsymbol{W}_s \, \boldsymbol{h}_t^{dec}) \tag{A.2}$$

$$\boldsymbol{h}_t^{dec} = \tanh(\boldsymbol{W}_o[\tilde{\boldsymbol{h}}_t^{dec}; \boldsymbol{c}_t]) \tag{A.3}$$

$$\boldsymbol{c}_t = \text{attend}(\tilde{\boldsymbol{h}}_{1:n}^{enc}, \tilde{\boldsymbol{h}}_t^{dec}) \tag{A.4}$$

where $[\cdot;\cdot]$ is the concatenation of two vectors, $\mathrm{attend}(\cdot,\cdot)$ is an attention mechanism, $\tilde{\boldsymbol{h}}^{enc}_{1:n}$ are all encoder's hidden vectors and $\tilde{\boldsymbol{h}}^{dec}_{t}$ is the decoder's hidden vector at time step t. We use the "general" global attention in Luong et al. (2015).

During training, the encoder first encodes $\boldsymbol{x}$ to a continuous vector $\Phi(\boldsymbol{x})$, which is used as the initial hidden vector for the decoder. In our paper, $\Phi(\boldsymbol{x})$ simply returns the last hidden vector of the encoder. The decoder performs RNN updates to produce a sequence of hidden vectors:

$$\begin{aligned}\tilde{\boldsymbol{h}}^{dec}_{0} &= \Phi(\boldsymbol{x}) \\ \tilde{\boldsymbol{h}}^{dec}_{t} &= f_{\boldsymbol{\theta}}\left(\tilde{\boldsymbol{h}}^{dec}_{t-1}, \left[\boldsymbol{h}^{dec}_{t-1}; e(y_t)\right]\right)\end{aligned} \tag{A.5}$$

where $e(.)$ is a word embedding lookup function and y_t is the ground-truth token at time step t. Feeding the output vector $\boldsymbol{h}^{dec}_{t-1}$ to the next step is known as "input feeding".

At prediction time, the ground-truth token y_t in Eq A.5 is replaced by the model's own prediction $\hat{y}_t$:

$$\hat{y}_t = \arg\max_{y} P_{\boldsymbol{\theta}}(y \mid \hat{\boldsymbol{y}}_{<t}, \boldsymbol{x}) \tag{A.6}$$

In a supervised learning framework, an NMT model is typically trained under the maximum log-likelihood objective:

$$\max_{\boldsymbol{\theta}} \mathcal{L}_{sup}(\boldsymbol{\theta}) = \max_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D_{\text{tr}}}\left[\log P_{\boldsymbol{\theta}}\left(\boldsymbol{y} \mid \boldsymbol{x}\right)\right] \tag{A.7}$$

where D_{tr} is the training set. However, this learning framework is not applicable to bandit learning since ground-truth translations are not available.

Table B.1: List of common notations and their definitions.

Notation	Definition
$\Delta(\mathcal{U})$	Space of all distributions over a set $\mathcal{U}$
$\texttt{unf}(\mathcal{U})$	Denotes the uniform distribution over a set $\mathcal{U}$
$\|.\|_p$	p-norm
$\mathrm{KL}Q_1(x \mid y)Q_2(x \mid y)$	KL-divergence between two distributions $Q_1(\cdot \mid y)$ and $Q_2(\cdot \mid y)$ over a countable set $\mathcal{X}$. Formally, $\mathrm{KL}Q_1(x \mid y)Q_2(x \mid y) = \sum_{x \in \mathcal{X}} Q_1(x \mid y) \ln \frac{Q_1(x\mid y)}{Q_2(x\mid y)}$.
$\texttt{supp}\, Q(x)$	Support of a distribution $Q \in \Delta(\mathcal{X})$. Formally, $\texttt{supp}Q(x) = \{x \in \mathcal{X} \mid Q(x) > 0\}$.
$\mathbb{N}$	Set of natural numbers
$\mathcal{S}$	State space
s	A single state in $\mathcal{S}$
$\mathcal{A}$	Finite action space
a	a single action in $\mathcal{A}$
$\mathcal{D}$	Set of all possible descriptions and requests
d	A single description or request
$T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$	Transition function with $T(s' \mid s, a)$ denoting the probability of transitioning to state s' given state s and action a.
$\mathcal{R}$	Family of reward functions
$R : \mathcal{S} \times \mathcal{A} \to [0, 1]$	Reward function with $R(s, a)$ denoting the reward for taking action a in state s
H	Horizon of the problem denoting the number of actions in a single episode.
e	An execution $e = (s_1, a_1, s_2, \cdots, s_H, a_H)$ describing states and actions in an episode.
$q = (R, d, s_1)$	A single task comprising of reward function R, request d and start state s_1
$\mathbb{P}^\star(q)$	Task distribution defined by the world
$\mathbb{P}^\star(e, R, s, d)$	Joint distribution over executions and task (see Equation 3.12).
$\mathbb{P}_T(d \mid e)$	Teacher model denoting distribution over descriptions d for a given execution e.
Θ	Set of all parameters of agent's policy.
θ	Parameters of agent's policy. Belongs to the set Θ.
$\pi_\theta(a \mid s, d)$	Agent's policy denoting the probability of action a given state s, description d, and parameters θ.

B Theoretical Analysis of ADEL

In this section, we provide a theoretical justification for an epoch-version of ADEL for the case of $H = 1$. We provide a list of notations in Table B.1 on page 157. We prove consistency results showing ADEL learns a near-optimal policy, and we also derive the convergence rate under the assumption that we perform maximum likelihood estimation optimally and the teacher is consistent. We call a teacher model $\mathbb{P}_T(d \mid e)$ to be consistent if for every execution e and description d we have $\mathbb{P}_T(d \mid e) = \mathbb{P}^\star(d \mid e)$. Recall that the conditional distribution $\mathbb{P}^\star(d \mid e)$ is

derived from the joint distribution defined in Equation 3.12. We will use superscript * to denote all probability distributions that are derived from this joint distribution.

We start by writing the epoch-version of ADEL in Algorithm 8 for an arbitrary value of H. The epoch version of ADEL runs an outer loop of epochs (line 3-10). The agent model is updated only at the end of an epoch. In the inner loop (line 5-9), the agent samples a batch using the teacher model and the agent model. This is used to update the model at the end of the epoch.

At the start of the n^{th} epoch, our sampling scheme in line 6-9 defines a procedure to sample $(\hat{e}, \hat{d})$ from a distribution D_n that remains fixed over this whole epoch. To define D_n, we first define $\mathbb{P}_n(e) = \mathbb{E}_{(R,d,s_1)\sim\mathbb{P}^\star(q)}\left[\mathbb{P}_n(e \mid s_1, d)\right]$ where we use the shorthand $\mathbb{P}_n(e \mid s_1, d)$ to refer to $\mathbb{P}_{\pi_{\theta_n}}(e \mid s_1, d)$. Note that $\hat{e} \sim \mathbb{P}_n(e)$ in line 7. As $\hat{d} \sim \mathbb{P}^\star(d \mid \hat{e})$, therefore, we arrive at the following form of D_n:

$$D_n(\hat{e}, \hat{d}) = \mathbb{P}^\star(\hat{d} \mid \hat{e})\mathbb{P}_n(\hat{e}). \tag{B.1}$$

We will derive our theoretical guarantees for $H = 1$. This setting is known as the contextual bandit setting Langford and Zhang (2008a), and while simpler than general reinforcement learning setting, it captures a large non-trivial class of problems. In this case, an execution $e = [s_1, a_1]$ can be described by the start state s_1 and a single action $a_1 \in \mathcal{A}$ taken by the agent. Since there is a single state and action in any execution, therefore, for cleaner notations we will drop the subscript and simply write s, a instead of s_1, a_1. For convenience, we also define a few extra notations. Firstly, we define the marginal distribution $D_n(s, \hat{d}) = \sum_{a' \in \mathcal{A}} D_n([s, a'], d)$. Secondly, let $\mathbb{P}^\star(s)$ be the marginal distribution over start state s given by $\mathbb{E}_{(R,d,s_1)\sim\mathbb{P}^\star(q)}[\mathbf{1}\{s_1 = s\}]$. We state some useful relations between these probability distributions in the next lemma.

Algorithm 8 EPOCHADEL: Epoch Version of ADEL. We assume the teacher is consistent, i.e., $\mathbb{P}_T(d \mid e) = \mathbb{P}^\star(d \mid e)$ for every (d, e).

Input: teacher model $\mathbb{P}^\star(d \mid e)$ and task distribution model $\mathbb{P}^\star(q)$.

Initialize agent policy $\pi_{\theta_1} : \mathcal{S} \times \mathcal{D} \to \texttt{unf}(\mathcal{A})$

for $n = 1, 2, \cdots, N$ **do**

  $\mathcal{B} = \emptyset$

  for $m = 1, 2, \cdots, M$ **do**

    World samples $q = (R, d^\star, s_1) \sim \mathbb{P}^\star(\cdot)$

    Agent generates $\hat{e} \sim \mathbb{P}_{\pi_{\theta_n}}(\cdot \mid s_1, d^\star)$

    Teacher generates description $\hat{d} \sim \mathbb{P}^\star(\cdot \mid \hat{e})$

    $\mathcal{B} \leftarrow \mathcal{B} \cup \left\{ \left(\hat{e}, \hat{d} \right) \right\}$

  Update agent policy using batch updates:

$$\theta_{n+1} \leftarrow \arg\max_{\theta' \in \Theta} \sum_{(\hat{e}, \hat{d}) \in \mathcal{B}} \sum_{(s, a_s) \in \hat{e}} \log \pi_{\theta'}(a_s \mid s, \hat{d})$$

  where a_s is the action taken by the agent in state s in execution $\hat{e}$.

return π_θ

Lemma 3. *For any $n \in \mathbb{N}$, we have:*

$$\mathbb{P}_n(e := [s, a]) = \mathbb{P}^\star(s)\mathbb{P}_n(a \mid s), \text{ where } \mathbb{P}_n(a \mid s) := \sum_d \mathbb{P}^\star(d \mid s)\mathbb{P}_n(a \mid s, d). \tag{B.2}$$

Proof. We first compute the marginal distribution $\sum_{a' \in \mathcal{A}} \mathbb{P}_n(e' := [s, a'])$ over s:

$$\sum_{a' \in \mathcal{A}} \mathbb{P}_n(e' := [s, a']) = \sum_{a' \in \mathcal{A}} \sum_{R,d} \mathbb{P}^\star(R, d, s) \mathbb{P}_n(a' \mid s, d) = \sum_{R,d} \mathbb{P}^\star(R, d, s) = \mathbb{P}^\star(s).$$

Next we compute the conditional distribution $\mathbb{P}_n(a \mid s)$ as shown:

$$\begin{aligned} \mathbb{P}_n(a \mid s) = \frac{\mathbb{P}_n([s, a])}{\sum_{a' \in \mathcal{A}} \mathbb{P}_n([s, a'])} &= \sum_{R,d} \frac{\mathbb{P}^\star(R, d, s) \mathbb{P}_n(a \mid s, d)}{\mathbb{P}^\star(s)} \\ &= \sum_{d} \frac{\mathbb{P}^\star(s, d) \mathbb{P}_n(a \mid s, d)}{\mathbb{P}^\star(s)} = \sum_{d} \mathbb{P}^\star(d \mid s) \mathbb{P}_n(a \mid s, d). \end{aligned}$$

This also proves $\mathbb{P}_n([s, a]) = \mathbb{P}^\star(s) \mathbb{P}_n(a \mid s)$. □

For $H = 1$, the update equation in line 10 solves the following optimization equation:

$$\max_{\theta' \in \Theta} J_n(\theta') \qquad \text{where} \quad J_n(\theta) := \sum_{(\hat{e} := [s,a], \hat{d}) \in \mathcal{B}} \ln \pi_{\theta'}(a \mid s, \hat{d}). \tag{B.3}$$

Here $J_n(\theta)$ is the empirical objective whose expectation over draws of batches is given by:

$$\mathbb{E}[J_n(\theta)] = \mathbb{E}_{(\hat{e}=[s,a],d) \sim D_n} \left[\ln \pi_\theta(a \mid s, d) \right].$$

As this is negative of the cross entropy loss, the Bayes optimal value would be achieved for $\pi_\theta(a \mid s, d) = D_n(a \mid s, d)$ for all $a \in \mathcal{A}$ and every $(s, d) \in \texttt{supp} D_n(s, d)$. We next state the form of this Bayes optimal model and then state our key realizability assumption.

Lemma 4. *Fix $n \in \mathbb{N}$. For every $(s, d) \in \texttt{supp} D_n(s, d)$ the value of the Bayes optimal model*

$D_n(a \mid s, d)$ at the end of the n^{th} epoch is given by:

$$D_n(a \mid s, d) = \frac{\mathbb{P}^\star(d \mid [s, a])\mathbb{P}_n(a \mid s)}{\sum_{a' \in \mathcal{A}} \mathbb{P}^\star(d \mid [s, a'])\mathbb{P}_n(a' \mid s)}.$$

Proof. The Bayes optimal model is given by $D_n(a \mid s, d)$ for every $(s, d) \in \texttt{supp} D_n(s, d)$. We compute this using Bayes' theorem.

$$\begin{aligned} D_n(a \mid s, d) = \frac{D_n([s, a], d)}{\sum_{a' \in \mathcal{A}} D_n([s, a'], d)} &= \frac{\mathbb{P}^\star(d \mid [s, a])\mathbb{P}_n([s, a])}{\sum_{a' \in \mathcal{A}} \mathbb{P}^\star(d \mid [s, a'])\mathbb{P}_n([s, a'])} \\ &= \frac{\mathbb{P}^\star(d \mid [s, a])\mathbb{P}_n(a \mid s)}{\sum_{a' \in \mathcal{A}} \mathbb{P}^\star(d \mid [s, a'])\mathbb{P}_n(a' \mid s)}. \end{aligned}$$

The last equality above uses Lemma 3. □

In order to learn the Bayes optimal model, we need our policy class to be expressive enough to contain this model. We formally state this *realizability* assumption below.

Assumption 1 (Realizability)**.** *For every $\theta \in \Theta$, there exists $\theta' \in \Theta$ such that for every start state s, description d we have:*

$$\forall a \in \mathcal{A}, \quad \pi_{\theta'}(a \mid s, d) = \frac{\mathbb{P}^\star(d \mid [s, a])Q_\theta(a \mid s)}{\sum_{a' \in \mathcal{A}} \mathbb{P}^\star(d \mid [s, a'])Q_\theta(a' \mid s)}, \tag{B.4}$$

$$\textit{where} \quad Q_\theta(a \mid s) = \sum_{d'} \mathbb{P}^\star(d' \mid s)\pi_\theta(a \mid s, d').$$

We can use the realizability assumption along with convergence guarantees for log-loss to state the following result:

Theorem 5 (Theorem 21 of Agarwal et al. (2020))**.** *Fix $m \in \mathbb{N}$ and $\delta \in (0, 1)$. Let $\{(d^{(i)}, e^{(i)} = [s^{(i)}, a^{(i)}]\}_{i=1}^m$ be i.i.d draws from $D_n(e, d)$ and let θ_{n+1} be the solution to the optimization problem*

in line 10 of the n^{th} epoch of EPOCHADEL. *Then with probability at least $1-\delta$ we have:*

$$\mathbb{E}_{s,d\sim D_n}\left[\|D_n(a \mid s,d) - \mathbb{P}_{\pi_{\theta_{n+1}}}(a \mid s,d)\|_1\right] \leq C\sqrt{\frac{1}{m}\ln {|\Theta|}/{\delta}}, \tag{B.5}$$

where $C > 0$ is a universal constant.

Please see Agarwal et al. (2020) for a proof. Lemma 5 implies that assuming realizability, as $M \to \infty$, our learned solution converges to the Bayes optimal model pointwise on the support over $D_n(s,d)$. Since we are only interested in consistency, we will assume $M \to \infty$ and assume $\mathbb{P}_{n+1}(a \mid s,d) = D_n(a \mid s,d)$ for every $(s,d) \in \texttt{supp}D_n(s,d)$. We will refer to this as optimally performing the maximum likelihood estimation at n^{th} epoch. If the learned policy is given by $\mathbb{P}_{n+1}(a \mid s,d) = D_n(a \mid s,d)$, then the next Lemma states the relationship between the marginal distribution $\mathbb{P}_{n+1}(a \mid s)$ for the next time epoch and marginal $\mathbb{P}_n(a \mid s)$ for this epoch.

Lemma 6 (Inductive Relation Between Marginals)**.** *For any $n \in \mathbb{N}$, if we optimally perform the maximum likelihood estimation at the n^{th} epoch of* EPOCHADEL*, then for all start states s, the marginal distribution $\mathbb{P}_{n+1}(a \mid s)$ for the $(n+1)^{th}$ epoch is given by:*

$$\mathbb{P}_{n+1}(a \mid s) = \sum_d \frac{\mathbb{P}^\star(d \mid [s,a])\mathbb{P}_n(a \mid s)\mathbb{P}^\star(d \mid s)}{\sum_{a'\in\mathcal{A}}\mathbb{P}^\star(d \mid [s,a'])\mathbb{P}_n(a' \mid s)}.$$

Proof. The proof is completed as follows:

$$\mathbb{P}_{n+1}(a \mid s) = \sum_d \mathbb{P}^\star(d \mid s)\mathbb{P}_{n+1}(a \mid s,d) = \sum_d \frac{\mathbb{P}^\star(d \mid [s,a])\mathbb{P}_n(a \mid s)\mathbb{P}^\star(d \mid s)}{\sum_{a'\in\mathcal{A}}\mathbb{P}^\star(d \mid [s,a'])\mathbb{P}_n(a' \mid s)},$$

where the first step uses Lemma 3 and the second step uses $\mathbb{P}_{n+1}(a \mid s,d) = D_n(a \mid s,d)$

(optimally solving maximum likelihood) and the form of D_n from Lemma 4. □

B.1 Proof of Convergence for Marginal Distribution

Our previous analysis associates a probability distribution $\mathbb{P}_n(a \mid s, d)$ and $\mathbb{P}_n(a \mid s)$ with the n^{th} epoch of EPOCHADEL. For any $n \in \mathbb{N}$, the n^{th} epoch of EPOCHADEL can be viewed as a transformation of $\mathbb{P}_n(a \mid s, d) \mapsto \mathbb{P}_{n+1}(a \mid s, d)$ and $\mathbb{P}_n(a \mid s) \mapsto \mathbb{P}_{n+1}(a \mid s)$. In this section, we show that under certain conditions, the running average of the marginal distributions $\mathbb{P}_n(a \mid d)$ converges to the optimal marginal distribution $\mathbb{P}^\star(a \mid d)$. We then discuss how this can be used to learn the optimal policy $\mathbb{P}^\star(a \mid s, d)$.

We use a potential function approach to measure the progress of each epoch. Specifically, we will use KL-divergence as our choice of potential function. The next lemma bounds the change in potential after a single iteration.

Lemma 7. *[Potential Difference Lemma] For any $n \in \mathbb{N}$ and start state s, we define the following distribution over descriptions $\mathbb{P}_n(d \mid s) := \sum_{a' \in \mathcal{A}} \mathbb{P}^\star(d \mid [s, a])\mathbb{P}_n(a \mid s)$. Then for every start state s we have:*

$$\mathrm{KL}\mathbb{P}^\star(a \mid s)\mathbb{P}_{n+1}(a \mid s) - \mathrm{KL}\mathbb{P}^\star(a \mid s)\mathbb{P}_n(a \mid s) \leq -\mathrm{KL}\mathbb{P}^\star(d \mid s)\mathbb{P}_n(d \mid s).$$

Proof. The change in potential from the start of n^{th} epoch to its end is given by:

$$\mathrm{KL}\mathbb{P}^\star(a \mid s)\mathbb{P}_{n+1}(a \mid s) - \mathrm{KL}\mathbb{P}^\star(a \mid s)\mathbb{P}_n(a \mid s) = -\sum_{a \in \mathcal{A}} \mathbb{P}^\star(a \mid s) \ln\left(\frac{\mathbb{P}_{n+1}(a \mid s)}{\mathbb{P}_n(a \mid s)}\right) \quad \text{(B.6)}$$

Using Lemma 6 and the definition of $\mathbb{P}_n(d \mid s)$ we get:

$$\frac{\mathbb{P}_{n+1}(a \mid s)}{\mathbb{P}_n(a \mid s)} = \sum_d \frac{\mathbb{P}^\star(d \mid [s,a])\mathbb{P}^\star(d \mid s)}{\sum_{a' \in \mathcal{A}} \mathbb{P}^\star(d \mid [s,a'])\mathbb{P}_n(a' \mid s)} = \sum_d \mathbb{P}^\star(d \mid [s,a]) \frac{\mathbb{P}^\star(d \mid s)}{\mathbb{P}_n(d \mid s)}.$$

Taking logarithms and applying Jensen's inequality gives:

$$\ln\left(\frac{\mathbb{P}_{n+1}(a \mid s)}{\mathbb{P}_n(a \mid s)}\right) = \ln\left(\sum_d \mathbb{P}^\star(d \mid [s,a]) \frac{\mathbb{P}^\star(d \mid s)}{\mathbb{P}_n(d \mid s)}\right) \geq \sum_d \mathbb{P}^\star(d \mid [s,a]) \ln\left(\frac{\mathbb{P}^\star(d \mid s)}{\mathbb{P}_n(d \mid s)}\right). \tag{B.7}$$

Taking expectations of both sides with respect to $\mathbb{P}^\star(a \mid s)$ gives us:

$$\begin{aligned}
\sum_a \mathbb{P}^\star(a \mid s) \ln\left(\frac{\mathbb{P}_{n+1}(a \mid s)}{\mathbb{P}_n(a \mid s)}\right) &\geq \sum_a \sum_d \mathbb{P}^\star(a \mid s)\mathbb{P}^\star(d \mid [s,a]) \ln\left(\frac{\mathbb{P}^\star(d \mid s)}{\mathbb{P}_n(d \mid s)}\right) \\
&= \sum_d \mathbb{P}^\star(d \mid s) \ln\left(\frac{\mathbb{P}^\star(d \mid s)}{\mathbb{P}_n(d \mid s)}\right) \\
&= \mathrm{KL}\mathbb{P}^\star(d \mid s)\mathbb{P}_n(d \mid s)
\end{aligned}$$

where the last step uses the definition of $\mathbb{P}_n(d \mid s)$. The proof is completed by combining the above result with Equation B.6. □

The $\mathbb{P}_s$ matrix. For a fixed start state s, we define $\mathbb{P}_s$ as the matrix whose entries are $\mathbb{P}^\star(d \mid [s,a])$. The columns of this matrix range over actions, and the rows range over descriptions. We denote the minimum singular value of the description matrix $\mathbb{P}_s$ by $\sigma_{min}(s)$.

We state our next assumption that the minimum singular value of $\mathbb{P}_s$ matrix is non-zero.

Assumption 2 (Minimum Singular Value is Non-Zero)**.** *For every start state s, we assume $\sigma_{min}(s) > 0$.*

Intuitively, this assumption states that there is enough information in the descriptions for the agent to decipher probabilities over actions from learning probabilities over descriptions. More formally, we are trying to decipher $\mathbb{P}^\star(a \mid s)$ using access to two distributions: $\mathbb{P}^\star(d \mid s)$ which generates the initial requests, and the teacher model $\mathbb{P}^\star(d \mid [s, a])$ which is used to describe an execution $e = [s, a]$. This can result in an underspecified problem. The only constraints these two distributions place on $\mathbb{P}^\star(a \mid s)$ is that $\sum_{a \in \mathcal{A}} \mathbb{P}^\star(d \mid [s, a])\mathbb{P}^\star(a \mid s) = \mathbb{P}^\star(d \mid s)$. This means all we know is that $\mathbb{P}^\star(a \mid s)$ belongs to the following set of solutions of the previous linear systems of equation:

$$\left\{ Q(a \mid s) \mid \sum_{a \in \mathcal{A}} \mathbb{P}^\star(d \mid [s, a])Q(a \mid s) = \mathbb{P}^\star(d \mid s)\ \forall d, \qquad Q(a \mid s) \text{ is a distribution} \right\}.$$

As $\mathbb{P}^\star(a \mid s)$ belongs to this set hence this set is nonempty. However, if we also assume that $\sigma_{min}(s) > 0$ then the above set has a unique solution. Recall that singular values are square root of eigenvalues of $\mathbb{P}_s^\top \mathbb{P}_s$, and so $\sigma_{min}(s) > 0$ implies that the matrix $\mathbb{P}_s^\top \mathbb{P}_s$ is invertible. [1] This means, we can find the unique solution of the linear systems of equation by multiplying both sides by $(\mathbb{P}_s^\top \mathbb{P}_s)^{-1}\mathbb{P}_s^\top$. Hence, Assumption 2 makes it possible for us to find $\mathbb{P}^\star(a \mid s)$ using just the information we have. Note that we cannot solve the linear system of equations directly since the description space and action space can be extremely large. Hence, we use an oracle based solution via reduction to supervised learning.

The next theorem shows that the running average of learned probabilities $\mathbb{P}_n(a \mid s)$ converges to the optimal marginal distribution $\mathbb{P}^\star(a \mid s)$ at a rate determined by the inverse square root of the number of epochs of ADEL, the minimum singular value of the matrix $\mathbb{P}_s$, and the

[1] Recall that a matrix of the form $A^\top A$ always have non-negative eigenvalues.

KL-divergence between optimal marginal and initial value.

Theorem 8. *[Rate of Convergence for Marginal] For any $t \in \mathbb{N}$ we have:*

$$\|\mathbb{P}^\star(a \mid s) - \frac{1}{t}\sum_{n=1}^{t}\mathbb{P}_n(a \mid s)\|_2 \leq \frac{1}{\sigma_{min}(s)}\sqrt{\frac{2}{t}\mathrm{KL}\mathbb{P}^\star(a \mid s)\mathbb{P}_1(a \mid s)},$$

and if $\mathbb{P}_1(a \mid s, d)$ is a uniform distribution for every s and d, then

$$\|\mathbb{P}^\star(a \mid s) - \frac{1}{t}\sum_{n=1}^{t}\mathbb{P}_n(a \mid s)\|_2 \leq \frac{1}{\sigma_{min}(s)}\sqrt{\frac{2\ln|\mathcal{A}|}{t}}.$$

Proof. We start with Lemma 7 and bound the right hand side as shown:

$$\begin{aligned}
\mathrm{KL}\mathbb{P}^\star(a \mid s)\mathbb{P}_{n+1}(a \mid s) - \mathrm{KL}\mathbb{P}^\star(a \mid s)\mathbb{P}_n(a \mid s) &\leq -\mathrm{KL}\mathbb{P}^\star(d \mid s)\mathbb{P}_n(d \mid s) \\
&\leq -\frac{1}{2}\|\mathbb{P}^\star(d \mid s) - \mathbb{P}_n(d \mid s)\|_1^2, \\
&\leq -\frac{1}{2}\|\mathbb{P}^\star(d \mid s) - \mathbb{P}_n(d \mid s)^2\|_2^2 \\
&= -\frac{1}{2}\|\mathbb{P}_s\left\{\mathbb{P}^\star(a \mid s) - \mathbb{P}_n(a \mid s)\right\}\|_2^2, \\
&\leq -\frac{1}{2}\sigma_{min}(s)^2\|\mathbb{P}^\star(a \mid s) - \mathbb{P}_n(a \mid s)\|_2^2,
\end{aligned}$$

where the second step uses Pinsker's inequality. The third step uses the property of p-norms, specifically, $\|\nu\|_2 \leq \|\nu\|_1$ for all ν. The fourth step, uses the definition of $\mathbb{P}^\star(d \mid s) = \sum_{a' \in \mathcal{A}}\mathbb{P}(d \mid s, a')\mathbb{P}^\star(a' \mid s))$ and $\mathbb{P}_n(d \mid s) = \sum_{a' \in \mathcal{A}}\mathbb{P}(d \mid s, a')\mathbb{P}_n(a' \mid s)$. We interpret the notation $\mathbb{P}^\star(a \mid s)$ as a vector over actions whose value is the probability $\mathbb{P}^\star(a \mid s)$. Therefore, $\mathbb{P}_s\mathbb{P}^\star(a \mid s)$ represents a matrix-vector multiplication. Finally, the last step, uses $\|Ax\|_2 \geq \sigma_{min}(A)\|x\|_2$ for any vector x and matrix A of compatible shape such that Ax is de-

fined, where $\sigma_{min}(A)$ is the smallest singular value of A.

Summing over n from $n = 1$ to t and rearranging the terms we get:

$$\mathrm{KL}\mathbb{P}^\star(a \mid s)\mathbb{P}_{t+1}(a \mid s) \leq \mathrm{KL}\mathbb{P}^\star(a \mid s)\mathbb{P}_1(a \mid s) - \frac{1}{2}\sigma_{min}(s)^2 \sum_{n=1}^{t} \|\mathbb{P}^\star(a \mid s) - \mathbb{P}_n(a \mid s)\|_2^2.$$

As the left hand-side is positive we get:

$$\sum_{n=1}^{t} \|\mathbb{P}^\star(a \mid s) - \mathbb{P}_n(a \mid s)\|_2^2 \leq \frac{2}{\sigma_{min}(s)^2}\mathrm{KL}\mathbb{P}^\star(a \mid s)\mathbb{P}_1(a \mid s).$$

Dividing by t and applying Jensen's inequality (specifically, $\mathbb{E}[X^2] \geq \mathbb{E}[|X|]^2$) we get:

$$\frac{1}{t}\sum_{n=1}^{t} \|\mathbb{P}^\star(e) - \mathbb{P}_n(e)\|_2 \leq \frac{1}{\sigma_{min}(s)}\sqrt{\frac{2}{t}\mathrm{KL}\mathbb{P}^\star(a \mid s)\mathbb{P}_1(a \mid s)} \tag{B.8}$$

Using the triangle inequality, the left hand side can be bounded as:

$$\frac{1}{t}\sum_{n=1}^{t} \|\mathbb{P}^\star(a \mid s) - \mathbb{P}_n(a \mid s)\|_2 \geq \|\mathbb{P}^\star(a \mid s) - \frac{1}{t}\sum_{n=1}^{t} \mathbb{P}_n(a \mid s)\|_2 \tag{B.9}$$

Combining the previous two equations proves the main result. Finally, note that if $\mathbb{P}_1(a \mid s, d) = {}^1/_{|\mathcal{A}|}$ for every value of s, d, and a, then $\mathbb{P}_1(a \mid s)$ is also a uniform distribution over actions. The initial KL-divergence is then bounded by $\ln |\mathcal{A}|$ as shown below:

$$\mathrm{KL}\mathbb{P}^\star(a \mid s)\mathbb{P}_1(a \mid s) = -\sum_{a \in \mathcal{A}} \mathbb{P}^\star(a \mid s) \ln \frac{1}{|\mathcal{A}|} + \sum_{a \in \mathcal{A}} \mathbb{P}^\star(a \mid s) \ln \mathbb{P}^\star(a \mid s) \leq \ln |\mathcal{A}|,$$

where the second step uses the fact that entropy of a distribution is non-negative. This completes

the proof. □

B.2 Proof of Convergence to Near-Optimal Policy

Finally, we discuss how to learn $\mathbb{P}^\star(a \mid s, d)$ once we learn $\mathbb{P}^\star(a \mid s)$. Since we only derive convergence of running average of $\mathbb{P}_n(a \mid s)$ to $\mathbb{P}^\star(a \mid s)$, therefore, we cannot expect $\mathbb{P}_n(a \mid s, d)$ to converge to $\mathbb{P}^\star(a \mid s, d)$. Instead, we will show that if we perform line 4-10 in Algorithm 8 using the running average of policies, then the learned Bayes optimal policy will converge to the near-optimal policy. The simplest way to accomplish this with Algorithm 8 is to perform the block of code in line 4-10 twice, once when taking actions according to $\mathbb{P}_n(a \mid s, d)$, and once when taking actions according to running average policy $\tilde{\mathbb{P}}_n(a \mid s, d) = \frac{1}{n}\sum_{t=1}^{n} \tilde{\mathbb{P}}_t(a \mid s, d)$. This will give us two Bayes optimal policy in 10 one each for the current policy $\mathbb{P}_n(a \mid s, d)$ and the running average policy $\tilde{\mathbb{P}_n}(a \mid s, d)$. We use the former for roll-in in the future and the latter for evaluation on held-out test set.

For convenience, we first define an operator that denotes mapping of one agent policy to another.

W operator. Let $\mathbb{P}(a \mid s, d)$ be an agent policy used to generate data in any epoch of EPOCHADEL (line 5-9). We define the W operator as the mapping to the Bayes optimal policy for the optimization problem solved by EPOCHADEL in line 10 which we denote by $(W\mathbb{P})$. Under the realizability assumption (Assumption 1), the agent learns the $W\mathbb{P}$ policy when $M \to \infty$. Using Lemma 3 and Lemma 4, we can verify that:

$$(W\mathbb{P})(a \mid s, d) = \frac{\mathbb{P}^\star(d \mid [s, a])\mathbb{P}(a \mid s)}{\sum_{a' \in \mathcal{A}} \mathbb{P}^\star(d \mid [s, a'])\mathbb{P}(a' \mid s)}, \quad \text{where} \quad \mathbb{P}(a \mid s) = \sum_{d} \mathbb{P}^\star(d \mid s)\mathbb{P}(a \mid s, d).$$

We first show that our operator is smooth around $\mathbb{P}^\star(a \mid s)$.

Lemma 9 (Smoothness of W)**.** *For any start state s and description $d \in \texttt{supp}\ \mathbb{P}^\star(d \mid s)$, there exists a finite constant K_s such that:*

$$\|W\mathbb{P}(a \mid s, d) - W\mathbb{P}^\star(a \mid s, d)\|_1 \leq K_s \|\mathbb{P}(a \mid s) - \mathbb{P}^\star(a \mid s)\|_1.$$

Proof. We define $\mathbb{P}(d \mid s) = \sum_{a' \in \mathcal{A}} \mathbb{P}^\star(d \mid s, a')\mathbb{P}(a' \mid s)$. Then from the definition of operator W we have:

$$\begin{aligned}
&|W\mathbb{P}(a \mid s, d) - W\mathbb{P}^\star(a \mid s, d)|_1 \\
&= \sum_{a \in \mathcal{A}} \left| \frac{\mathbb{P}^\star(d \mid [s, a])\mathbb{P}(a \mid s)}{\mathbb{P}(d \mid s)} - \frac{\mathbb{P}^\star(d \mid [s, a])\mathbb{P}^\star(a \mid s)}{\mathbb{P}^\star(d \mid s)} \right| \\
&= \sum_{a \in \mathcal{A}} \mathbb{P}^\star(d \mid [s, a]) \frac{|\mathbb{P}(a \mid s)\mathbb{P}^\star(d \mid s) - \mathbb{P}^\star(a \mid s)\mathbb{P}(d \mid s)|}{\mathbb{P}(d \mid s)\mathbb{P}^\star(d \mid s)} \\
&\leq \sum_{a \in \mathcal{A}} \mathbb{P}^\star(d \mid [s, a])\mathbb{P}(a \mid s) \frac{|\mathbb{P}^\star(d \mid s) - \mathbb{P}(d \mid s)|}{\mathbb{P}(d \mid s)\mathbb{P}^\star(d \mid s)} + \sum_{a \in \mathcal{A}} \mathbb{P}^\star(d \mid [s, a]) \frac{|\mathbb{P}(a \mid s) - \mathbb{P}^\star(a \mid s)|}{\mathbb{P}^\star(d \mid s)} \\
&= \frac{|\mathbb{P}^\star(d \mid s) - \mathbb{P}(d \mid s)|}{\mathbb{P}^\star(d \mid s)} + \sum_{a \in \mathcal{A}} \mathbb{P}^\star(d \mid [s, a]) \frac{|\mathbb{P}(a \mid s) - \mathbb{P}^\star(a \mid s)|}{\mathbb{P}^\star(d \mid s)} \\
&\leq 2 \sum_{a \in \mathcal{A}} \mathbb{P}^\star(d \mid [s, a]) \frac{|\mathbb{P}(a \mid s) - \mathbb{P}^\star(a \mid s)|}{\mathbb{P}^\star(d \mid s)}, \qquad \text{(using the definition of } \mathbb{P}(d \mid s)) \\
&\leq \frac{2}{\mathbb{P}^\star(d \mid s)} \|\mathbb{P}(a \mid s) - \mathbb{P}^\star(a \mid s)\|_1.
\end{aligned}$$

Note that the policy will only be called on a given pair of (s, d) if and only if $\mathbb{P}^\star(d \mid s) > 0$, hence, the constant is bounded. We define $K_s = \max_d \frac{2}{\mathbb{P}^\star(d|s)}$ where maximum is taken over all descriptions $d \in \texttt{supp}\ \mathbb{P}^\star(d \mid s)$. □

Theorem 10 (Convergence to Near Optimal Policy)**.** *Fix $t \in \mathbb{N}$, and let $\tilde{\mathbb{P}}_t(a \mid s, d) = \frac{1}{t} \sum_{n=1}^{t} \mathbb{P}_n(a \mid$*

$s, d)$ *be the average of the agent's policy across epochs. Then for every start state* s *and description* $d \in \texttt{supp}\, \mathbb{P}^\star(d \mid s)$ *we have:*

$$\lim_{t\to\infty} (W\tilde{\mathbb{P}}_t)(a \mid s, d) = \mathbb{P}^\star(a \mid s, d).$$

Proof. Let $\tilde{\mathbb{P}}_t(a \mid s) = \sum_d \mathbb{P}^\star(d \mid s)\tilde{\mathbb{P}}_t(a \mid s, d)$. Then it is easy to see that $\tilde{\mathbb{P}}_t(a \mid s) = \frac{1}{t}\sum_{n=1}^{t} \mathbb{P}_n(a \mid s)$. From Theorem 8 we have $\lim_{t\to\infty} \|\tilde{\mathbb{P}}_t(a \mid s) - \mathbb{P}^\star(a \mid s)\|_2 = 0$. As $\mathcal{A}$ is finite dimensional, therefore, $\|\cdot\|_2$ and $\|\cdot\|_1$ are equivalent, i.e., convergence in one also implies convergence in the other. This implies, $\lim_{t\to\infty} \|\tilde{\mathbb{P}}_t(a \mid s) - \mathbb{P}^\star(a \mid s)\|_1 = 0$.

From Lemma 9 we have:

$$\lim_{t\to\infty} \|(W\tilde{\mathbb{P}}_t)(a \mid s, d) - (W\mathbb{P}^\star)(a \mid s, d)\|_1 \le K_s \lim_{t\to\infty} \|\tilde{\mathbb{P}}_t(a \mid s) - \mathbb{P}^\star(a \mid s)\|_1 = 0.$$

This shows $\lim_{t\to\infty}(W\tilde{\mathbb{P}}_t)(a \mid s, d) = (W\mathbb{P}^\star)(a \mid s, d)$. Lastly, we show that the optimal policy $\mathbb{P}^\star(a \mid s, d)$ is a fixed point of W:

$$\begin{aligned}(W\mathbb{P}^\star)(a \mid s, d) = \frac{\mathbb{P}^\star(d \mid s, a)\mathbb{P}^\star(a \mid s)}{\sum_{a'\in\mathcal{A}} \mathbb{P}^\star(d \mid s, a')\mathbb{P}^\star(a' \mid s)} &= \frac{\mathbb{P}^\star(d, a \mid s)}{\sum_{a'\in\mathcal{A}} \mathbb{P}^\star(d, a' \mid s)} \\ &= \frac{\mathbb{P}^\star(d, a \mid s)}{\mathbb{P}^\star(d \mid s)} = \mathbb{P}^\star(a \mid s, d).\end{aligned}$$

This completes the proof. □

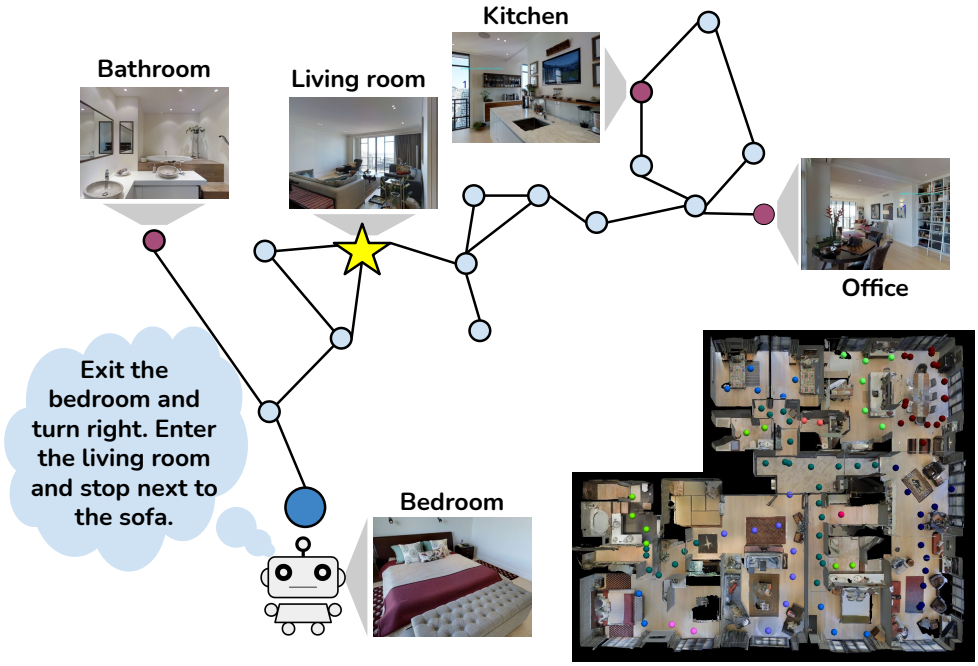


Figure C.1: *Vision-language navigation* (NAV): a (robot) agent fulfills a navigational natural-language request in a photo-realistic simulated house. Locations in the house are connected as a graph. In each time step, the agent receives a photo of the panoramic view at its current location (due to space limit, here we only show part of a view). Given the view and the language request, the agent chooses an adjacent location to go to. On average, each house has about 117 locations.

C Experimental Setup of ILIAD

C.1 Problem settings

Figure C.1 and Figure C.2 illustrate the two problems that we conduct experiments on.

C.1.1 Vision-Language Navigation

Environment Simulator and Data. We use the Matterport3D simulator and the Room-to-Room dataset[2] developed by Anderson et al. (2018b). The simulator photo-realistically emulates

[2] https://github.com/peteanderson80/Matterport3DSimulator/blob/master/tasks/R2R/data/download.sh

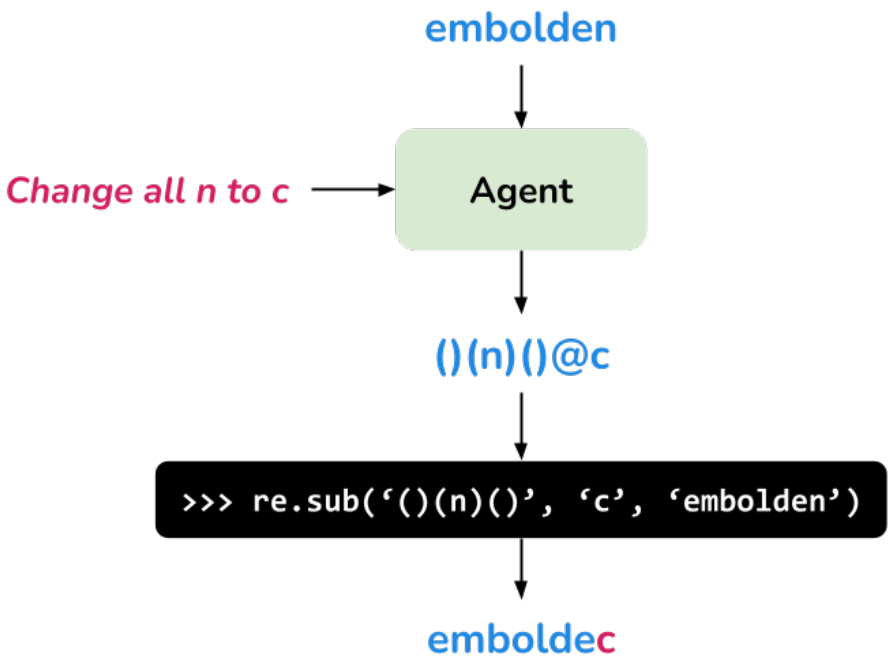


Figure C.2: *Word modification* (REGEX): an agent is given an input word and a natural-language request that asks it to modify the word. The agent outputs a regular expression that follows our specific syntax. The regular expression is executed by the Python's `re.sub()` method to generate an output word.

the first-person view of a person walking in a house. The dataset contains tuples of human-generated English navigation requests annotated with ground-truth paths in the environments. To evaluate on the test set, the authors require submitting predictions to an evaluation site[3], which limits the number of submissions to five. As our goal is not to establish state-of-the-art results on this task, but to compare performance of multiple learning frameworks, we re-split the data into 4,315 simulation, 2,100 validation, and 2,349 test data points. The simulation split, which is used to simulate the teacher, contains three requests per data point (i.e. $|\mathcal{D}_n^\star| = 3$). The validation and test splits each contains only one request per data point. On average, each request includes 2.5 sentences and 26 words. The word vocabulary size is 904 and the average number of optimal

[3] https://eval.ai/web/challenges/challenge-page/97/overview

actions required to reach the goal is 6.

Simulated Teacher. We use SDTW (Magalhaes et al., 2019) as the `perf` metric and set the threshold $\tau = 0.5$. The SDTW metric re-weights success rate by the shortest (order-preserving) alignment distance between a predicted path and a ground-truth path, offering more fine-grained evaluation of navigation paths.

Approximate marginal $P_{\pi_\omega}(e \mid s_1)$. The approximate marginal is a function that takes in a start location s_1 and randomly samples a shortest path on the environment graph that starts at s_1 and has (unweighted) length between 2 and 6.

C.1.2 Word Modification

Regular Expression Compiler. We use Python 3.7's `re.sub(pattern, replace, string)` method as the regular expression compiler. The method replaces every substring of `string` that matches a regular expression `pattern` with the string `replace`. A regular expression predicted by our agent $\hat{a}_{1:H}$ has the form "`pattern@replace`", where `pattern` and `replace` are strings and `@` is the at-sign character. For example, given the word *embolden* and the request "*replace all n with c*", the agent should ideally generate the regular expression "`()(n)()@c`". We then split the regular expression by the character `@` into a string `pattern` = "`()(n)()`" and a string `replace` = "*c*". We execute the Python's command `re.sub('()(n)()', 'c', 'embolden')` to obtain the output word *emboldec*.

Data. We use the data collected by Andreas et al. (2018). The authors presented crowd-workers with pairs of input and output words where the output words are generated by applying regular

expressions onto the input words. Workers are asked to write English requests that describe the change from the input words to the output words. From the human-generated requests, the authors extracted 1,917 request templates. For example, a template has the form *add an* `AFTER` *to the start of words beginning with* `BEFORE`, where `AFTER` and `BEFORE` can be replaced with latin characters to form a request. Each request template is annotated with a regular expression template that it describes. Since the original dataset is not designed to evaluate generalization to previously unseen request templates, we modified the script provided by the authors to generate a new dataset where the simulation and evaluation requests are generated from disjoint sets of request templates. We select 110 regular expressions templates that are each annotated with more than one request template. Then, we further remove pairs of regular expression and request templates that are mistakenly paired. We end up with 1111 request templates describing these 110 regular expression templates. We use these templates to generate tuples of requests and regular expressions. In the end, our dataset consists of 114,503 simulation, 6,429 validation, and 6,429 test data points. The request templates in the simulation, validation, and test sets are disjoint.

Simulated Teacher. We extend the performance metric `perf` in §3.2.4.1 to evaluating multiple executions. Concretely, given executions $\{w_j^{\text{inp}}, \hat{w}_j^{\text{out}}\}_{j=1}^{J}$, the metric counts how many pairs where the predicted output word matches the ground-truth: $\sum_{j=1}^{J} \mathbb{1}\left\{\hat{w}_j^{\text{out}} = w_j^{\text{out}}\right\}$. We set the threshold $\tau = J$.

Approximate marginal $P_{\pi_\omega}(e \mid s_1)$. The approximate marginal is a uniform distribution over a dataset of (unlabeled) regular expressions. These regular expressions are generated using the code provided by Andreas et al. (2018).[4]

[4] https://github.com/jacobandreas/l3/blob/master/data/re2/generate.py

Algorithm 9 ADEL: Learning from Activity Describers via Semi-Supervised Exploration (experimental version).

Input: teacher model $\mathbb{P}_T(d \mid e)$, approximate marginal $\mathbb{P}_{\pi_\omega}(e \mid s_1)$, mixing weight $\lambda \in [0, 1]$
Initialize policy $\pi_\theta : \mathcal{S} \times \mathcal{D} \to \Delta(\mathcal{A})$
Initialize policy $\pi_\beta : \mathcal{S} \times \mathcal{D} \to \Delta(\mathcal{A})$
for $n = 1, 2, \cdots, N$ **do**
    Word samples $q = (R, s_1, d^\star) \sim \mathbb{P}^\star(\cdot)$
    Agent generates $\hat{e} \sim \mathbb{P}_{\pi_\beta}(\cdot \mid s_1, d^\star)$
    Teacher generates $\hat{d} \sim \mathbb{P}_T(\cdot \mid \hat{e})$
    Agent samples $\tilde{e} \sim \mathbb{P}_{\pi_\omega}(\cdot \mid s_1)$
    Compute losses:

$$\mathcal{L}(\theta) = \sum_{(s,\hat{a}_s)\in\hat{e}} \log \pi_\theta(\hat{a}_s \mid s, \hat{d})$$

$$\mathcal{L}(\beta) = \lambda \sum_{(s,\tilde{a}_s)\in\tilde{e}} \log \pi_\beta(\tilde{a}_s \mid s, \hat{d}) + (1-\lambda) \sum_{(s,\hat{a}_s)\in\hat{e}} \log \pi_\beta(\hat{a}_s \mid s, \hat{d})$$

    Compute gradients $\nabla\mathcal{L}(\theta)$ and $\nabla\mathcal{L}(\beta)$
    Use gradient descent to update θ and β with $\nabla\mathcal{L}(\theta)$ and $\nabla\mathcal{L}(\beta)$, respectively
return $\pi : s, d \mapsto \operatorname{argmax}_a \pi_\theta(a \mid s, d)$

C.2 Practical Implementation of ADEL

In our experiments, we employ the following implementation of ADEL (Alg 9), which learns a policy π_β such that $\mathbb{P}_{\pi_\beta}(e \mid s_1, d)$ approximates the mixture $\tilde{\mathbb{P}}(e \mid s_1, d)$ in Alg 5. In each episode, we sample an execution $\hat{e}$ using the policy π_β. Then, similar to Alg 5, we ask the teacher $\mathbb{P}_T$ for a description of $\hat{e}$ and the use the pair $(\hat{e}, \hat{d})$ to update the agent policy π_θ. To ensure that $\mathbb{P}_{\pi_\beta}$ approximates $\tilde{\mathbb{P}}$, we draw a sample $\tilde{e}$ from the approximate marginal $\mathbb{P}_{\pi_\omega}(e \mid s_1)$ and update π_β using a λ-weighted loss of the log-likelihoods of the two data points $(\tilde{e}, \hat{d})$ and $(\hat{e}, \hat{d})$. We only use $(\hat{e}, \hat{d})$ to update the agent policy π_θ.

An alternative (naive) implementation of sampling from the mixture $\tilde{\mathbb{P}}$ is to first choose a policy between π_ω (with probability λ) and π_θ (with probability $1 - \lambda$), and then use this policy to generate an execution. Compared to this approach, our implementation has two advantages:

Table C.1: Effects of annealing the mixing weight λ. When annealed, the mixing weight is updated as $\lambda \leftarrow \max(\lambda_{\text{min}}, \lambda \cdot \beta)$, where the annealing rate $\beta = 0.5$ and the minimum mixing rate $\lambda_{\text{min}} = 0.1$. Initially, λ is set to be 0.5. All results are on validation data. Sample complexity is the number of training episodes required to reach a success rate of at least c ($c = 30\%$ in NAV, and $c = 85\%$ in REGEX).

	NAV		REGEX	
Anneal λ every L steps	Success rate (%) ↑	Sample complexity ↓	Success rate (%) ↑	Sample complexity ↓
L = 2000	31.4	304K	87.7	368K
L = 5000	32.5	384K	86.4	448K
No annealing (final)	32.0	384K	88.0	608K

1. Sampling from the mixture is simpler: instead of choosing between π_θ and π_ω, we always use π_β to generate executions;

2. More importantly, samples are more diverse: in the naive approach, the samples are either completely request-agnostic (if generated by π_ω) or completely request-guided (if generated by π_θ). As a machine learning-based model that learns from a mixture of data generated by π_ω and π_θ, the policy π_β can generalize and generate executions that are partially request-agnostic.

Effects of the Annealing Mixing Weight. We do not anneal the mixing weight λ in our experiments. Table C.1 shows the effects of annealing the mixing weight with various settings. We find that annealing improves the sample complexity of the agents, i.e. they reach a substantially high success rate in less training episodes. But overall, not annealing yields slightly higher final success rates.

C.3 Training details

Reinforcement learning's continuous reward. In REGEX, the continuous reward function is

$$\frac{|w^{\text{out}}| - \text{editdistance}\left(\hat{w}^{\text{out}}, w^{\text{out}}\right)}{|w^{\text{out}}|} \tag{C.1}$$

where w^{out} is the ground-truth output word, $\hat{w}^{\text{out}}$ is the predicted output word, editdistance(.,.) is the string edit distance computed by the Python's editdistance module.

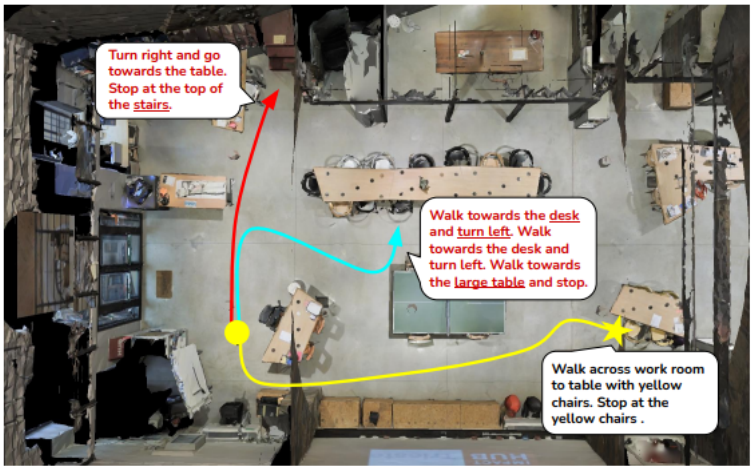
In NAV, the continuous reward function is

$$\frac{\text{shortest}\left(s_1, s_g\right) - \text{shortest}\left(s_H, s_g\right)}{\text{shortest}\left(s_1, s_g\right)} \tag{C.2}$$

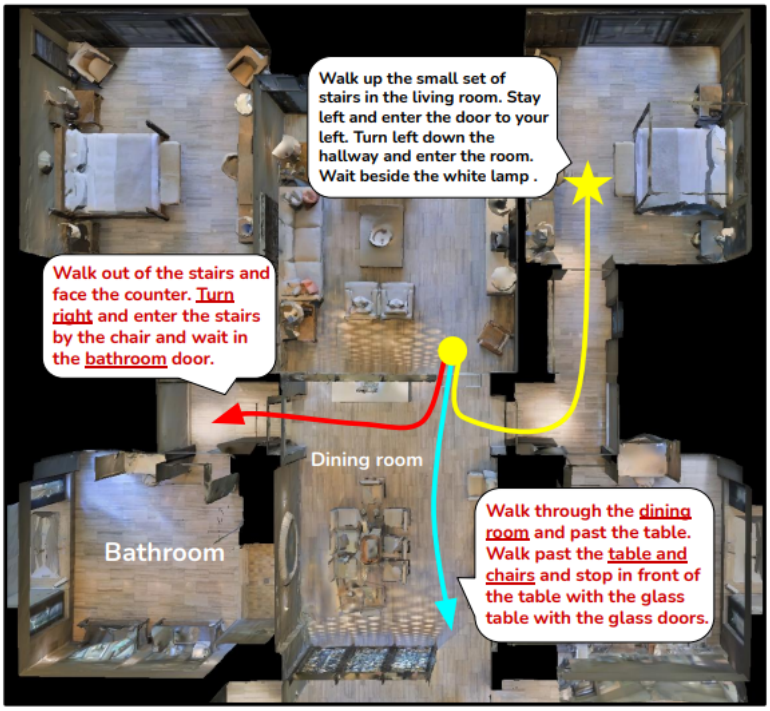
where s_1 is the start location, s_g is the goal location, s_H is the agent's final location, and shortest$(.,.)$ is the shortest-path distance between two locations (according to the environment's navigation graph).

Model architecture. Figure C.4 and Figure C.5 illustrate the architectures of the models that we train in the two problems, respectively. For each problem, we describe the architectures of the student policy $\pi_\theta(a \mid s, d)$ and the teacher's language model $\tilde{\mathbb{P}}(d \mid e)$. All models are encoder-decoder models but the NAV models use Transformer as the recurrent module while REGEX models use LSTM.

Hyperparameters. Model and training hyperparameters are provided in Table C.2. Each model is trained on a single NVIDIA V100 GPU, GTX 1080, or Titan X. Training with the ADEL algorithm takes about 19 hours for NAV and 14 hours for REGEX on a machine with an Intel

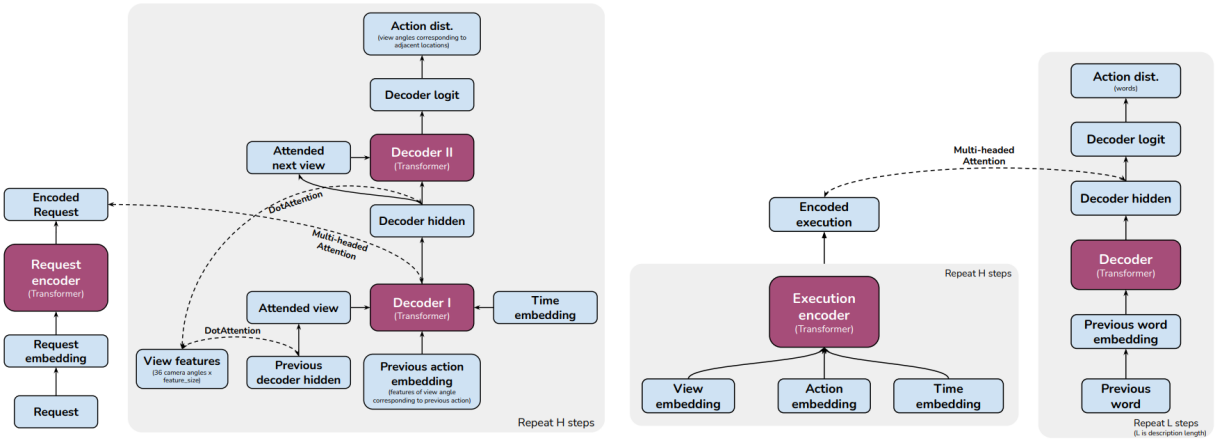


(a)



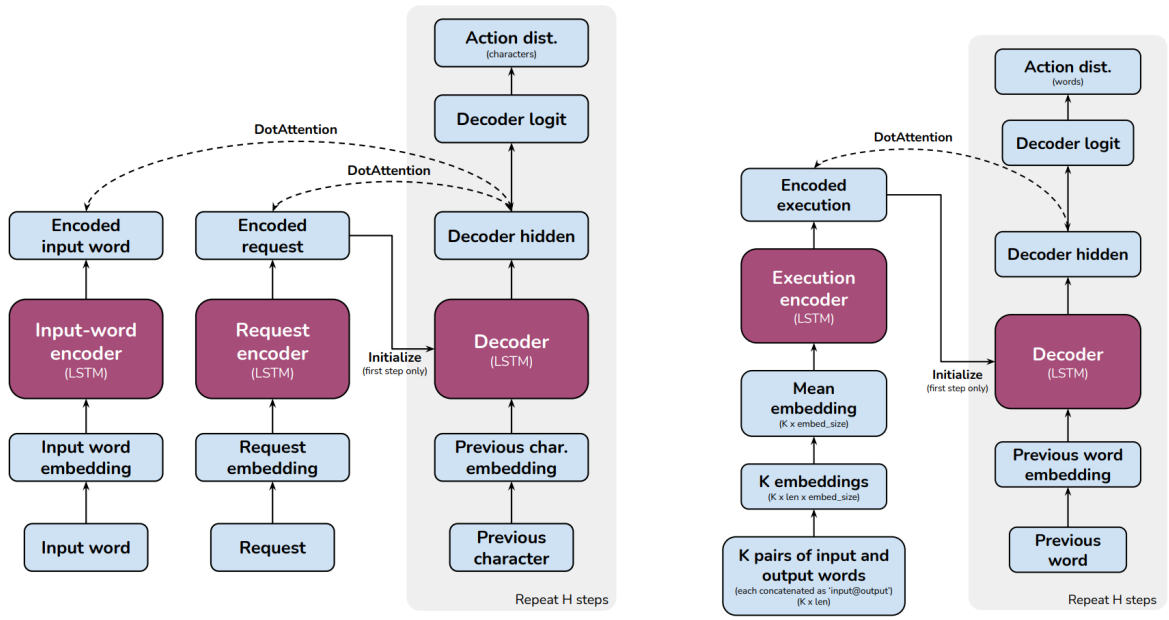
(b)

Figure C.3: Qualitative examples in the NAV problem. The **black texts** (no underlines) are the initial requests $d^\star$ generated by humans. The paths are the ground-truth paths implied by the requests. and are some paths are taken by the agent during training. Here, we only show two paths per example. The **red texts** are descriptions $\hat{d}$ generated by the teacher's learned (conditional) language model $\tilde{\mathbb{P}}(d \mid e)$. We show the bird-eye views of the environments for better visualization, but the agent only has access to the first-person panoramic views at its locations.



(a) Student model (b) Teacher model

Figure C.4: Student and teacher models in NAV.



(a) Student model (b) Teacher model

Figure C.5: Student and teacher models in REGEX.

i7-4790K 4.00GHz CPU and a Titan X GPU.

Table C.2: Hyperparameters for training with the ADEL algorithm.

Hyperparameter	NAV	REGEX
Student policy π_θ and Teacher's describer model $\tilde{P}_T$		
Base architecture	Transformer	LSTM
Hidden size	256	512
Number of hidden layers (of each encoder or decoder)	1	1
Request word embedding size	256	128
Character embedding size (for the input and output words)	-	32
Time embedding size	256	-
Attention heads	8	1
Observation feature size	2048	-
Teacher simulation		
`perf` metric	STDW (Magalhaes et al., 2019)	full-match success rate
Sample size for pragmatic inference ($\lvert\mathcal{D}_{\text{cand}}\rvert$)	5	10
Threshold (τ)	0.5	$J = 5$
Training		
Time horizon (H)	10	40
Batch size	32	32
Learning rate	10^{-4}	10^{-3}
Optimizer	Adam	Adam
Number of training iterations	25K	30K
Mixing weight (λ, no annealing)	0.5	0.5

Table C.3: Qualitative examples in the REGEX problem. We show pairs of input and output words and how the teacher's language model $\tilde{\mathbb{P}}(d \mid e)$ describes the modifications applied to the input words.

Input word	Output word	Description generated by $\tilde{\mathbb{P}}(d \mid e)$
attendant	xjtendxjt	replace [a] and the letter that follows it with an [x j]
disclaims	esclaims	if the word does not begin with a vowel , replace the first two letters with [e]
inculpating	incuxlpating	for any instance of [l] add a [x] before the [l]
flanneling	glanneling	change the first letter of the word to [g]
dhoti	jhoti	replaced beginning of word with [j]
stuccoing	ostuccoing	all words get a letter [o] put in front
reappearances	reappearanced	if the word ends with a consonant , change the consonant to [d]
bigots	vyivyovyvy	replace each consonant with a [v y]

C.4 Qualitative examples

Figure C.3 and Table C.3 show the qualitative examples in the NAV and REGEX problems, respectively.

D Experimental Setup of HANNA

D.1 Features

At time step t, the model makes use of the following features:

- I_t^{cur}: set of visual feature vectors representing the current panoramic view;
- I_t^{tgt}: set of visual feature vectors representing the target location's panoramic view;
- a_{t-1}: embedding of the last action;
- δ_t: a vector representing how similar the target view is to the current view;
- η_t^{loc}: local-time embedding encoding the number of time steps since the last time the inter-task module was reset;
- η_t^{glob}: global-time embedding encoding the number time steps since the beginning of the episode;
- P_t^{nav}: the navigation action distribution output by $\hat{\pi}_{\text{nav}}$. Each action corresponding to a view angle is mapped to a static index to ensure that the order of the actions is independent of the view angle. This feature is only used by the help-request network.

Visual features. To embed the first-person view images, we use the visual feature vectors provided Anderson et al. (2018b), which are extracted from a ResNet-152 (He et al., 2016b) pretrained on ImageNet (Russakovsky et al., 2015). Following the Speaker-Follower model (Fried et al., 2018b), at time step t, we provide the agent with a feature set I_t^{cur} representing the current panoramic view. The set consists of visual feature vectors that represent all 36 possible first-person view angles (12 headings $\times$ 3 elevations). Similarly, the panoramic view at the target location is given by a feature set I_t^{tgt}. Each next location is associated with a view angle whose

center is closest to it (in angular distance). The embedding of a navigation action $(v, \Delta\psi, \Delta\omega)$ is constructed by concatenating the feature vector of the corresponding view and an orientation feature vector $[\sin\Delta\psi; \cos\Delta\psi; \sin\Delta\omega; \sin\Delta\omega]$ where $\Delta\psi$ and $\Delta\omega$ are the camera angle change needed to shift from the current view to the view associated with v. The `stop` action is mapped to a zero vector. The action embedding set E_t^{nav} contains embeddings of all navigation actions at time step t.

Target similarity features. The vector δ represents the similarity between I^{cur} and I^{tgt}. To compute this vector, we first construct a matrix C, where $C_{i,j}$ is the cosine similarity between I_i^{cur} and I_j^{tgt}. δ_t is then obtained by taking the maximum values of the rows of C. The intuition is, for each current view angle, we find the most similar target view angle. If the current view perfectly matches with the target view, δ_t is a vector of ones.

Time features. The local-time embedding is computed by a residual neural network (He et al., 2016b) that learns a time-incrementing operator

$$\eta_t^{\text{loc}} = \textsc{IncTime}\left(\eta_{t-1}^{\text{loc}}\right) \tag{D.1}$$

where $\textsc{IncTime}(x) = \textsc{LayerNorm}(x + \textsc{ReLu}(\textsc{Linear}(x)))$. The global-time embedding is computed similarly but also incorporates the current local-time

$$\eta_t^{\text{glob}} = \textsc{IncTime}\left(\left[\eta_t^{\text{loc}}; \eta_{t-1}^{\text{glob}}\right]\right) \tag{D.2}$$

The linear layers of the local and global time modules do not share parameters. Our learned time features generalizes to previously unseen numbers of steps. They allow us to evaluate the agent

on longer episodes than during training, significantly reducing training cost. We use the sinusoid encoding (Vaswani et al., 2017) for the text-encoding module, and the ResNet-based encoding for the decoder modules. In our preliminary experiments, we also experimented using the sinusoid encoding in all modules but doing so significantly degraded success rate.

D.2 Model

Modules. The building blocks of our architecture are the Transformer modules (Vaswani et al., 2017)

- TRANSENCODER(l) is a Transformer-style encoder, which generates a set of feature vectors representing an input sequence l;
- MULTIATTEND(q, K, V) is a multi-headed attention layer that takes as input a query vector q, a set of key vectors V, and a set of value vectors V. The output is added a residual connection (He et al., 2016b) and is layer-normalized (Lei Ba et al., 2016).
- FFN(x) is a Transformer-style feed-forward layer, which consists of a regular feed-forward layer with a hidden layer and the RELU activation function, followed by a residual connection and layer normalization. The hidden size of the feed-forward layer is four times larger than the input/output size.

  In addition, we devise the following new attention modules
- SELFATTEND(q, K, V) implements a multi-headed attention layer internally. It calculates an output $h =$ MULTIATTEND(q, K, V). After that, the input q and output h are appended to K and V, respectively. This module is different from the Transformer's self-attention in that the keys and values are distinct.

- SIMATTEND(q, K, V) computes a weighted value $h = \sum_i \tilde{a}_i V_i$ where each weight $\tilde{a}_i$ is defined as

$$\tilde{a}_i = \mathbb{I}\{a_i > 0.9\} \frac{a_i}{\sum_j a_j} \tag{D.3}$$

$$a_i = \text{COSINESIMILARITY}(q, K_i) \tag{D.4}$$

where COSINESIMILARITY$(.,.)$ returns the cosine similarity between two vectors, and $\mathbb{I}\{.\}$ is an indicator function. Intuitively, this module finds keys that are nearly identical to the query and returns the weighted average of values corresponding to those keys. We use this module to obtain a representation of related past, which is crucial to enforcing curiosity-encouraging training.

We now describe the navigation network in detail. For notation brevity, we omit the $^{\text{nav}}$ superscripts in all variables.

Text-encoding component. The agent maintains a text memory M^{text}, which stores the hidden representation of the current language instruction l_t. At the beginning of time, the agent encodes the task request to generate an initial text memory. During time step t, if the current task is altered (due to the agent requesting help or departing a route), the agent encodes the new language instruction to generate a new text memory

$$M^{\text{text}} = \text{TRANSENCODER}(l_t) \tag{D.5}$$

$$\text{if } l_t \neq l_{t-1} \text{ or } t = 0$$

Inter-task component. The inter-task module computes a vector h_t^{inter} representing the state of the current task's execution. This state and the local time embedding are reset to zero vectors every time the agent switches task. Otherwise, a new state is computed as follows

$$h_t^{\text{inter}} = \textsc{SelfAttend}(q_t^{\text{inter}}, M_{\text{in}}^{\text{inter}}, M_{\text{out}}^{\text{inter}}) \tag{D.6}$$

$$q_t^{\text{inter}} = W_{\text{inter}}[c_t^{\text{inter}}; a_{t-1}; \delta_t] + \eta_t^{\text{loc}} \tag{D.7}$$

$$c_t^{\text{inter}} = \textsc{DotAttend}(h_{t-1}^{\text{inter}}, I_t^{\text{cur}}) \tag{D.8}$$

where $M_{\text{in}}^{\text{inter}} = \{q_0^{\text{inter}}, \cdots, q_{t-1}^{\text{inter}}\}$ and $M_{\text{in}}^{\text{inter}} = \{h_0^{\text{inter}}, \cdots, h_{t-1}^{\text{inter}}\}$ are the input and output inter-task memories, and $\textsc{DotAttend}(q, M)$ is the dot-product attention (Luong et al., 2015).

Next, h_t^{inter} is used to select which part of the language instruction should be interpreted in this step

$$c_t^{\text{text}} = \text{FFN}\left(\tilde{c}^{\text{text}}\right) \tag{D.9}$$

$$\tilde{c}^{\text{text}} = \textsc{MultiAttend}(h_t^{\text{inter}}, M_t^{\text{text}}, M_t^{\text{text}}) \tag{D.10}$$

Finally, c_t^{text} is used to select which areas of the current view and target view the agent should focus on

$$c_t^{\text{cur}} = \textsc{DotAttend}(c_t^{\text{text}}, I_t^{\text{cur}}) \tag{D.11}$$

$$c_t^{\text{tgt}} = \textsc{DotAttend}(c_t^{\text{text}}, I_t^{\text{tgt}}) \tag{D.12}$$

Intra-task component. The inter-task module computes a vector h_t^{intra} representing the state of

the entire episode. To compute this state, we first calculate $\bar{h}_t^{\text{intra}}$, a tentative current state, and $\tilde{h}_t^{\text{intra}}$, a weighted combination of the past states at nearly identical situations

$$\bar{h}_t^{\text{intra}} = \text{FFN}\left(\textsc{SelfAttend}(q_t^{\text{intra}}, M_{\text{in}}^{\text{intra}}, M_{\text{out}}^{\text{intra}})\right) \tag{D.13}$$

$$\tilde{h}_t^{\text{intra}} = \textsc{SimAttend}(q_t^{\text{intra}}, M_{\text{in}}^{\text{intra}}, M_{\text{out}}^{\text{intra}}) \tag{D.14}$$

$$q_t^{\text{intra}} = W_{\text{intra}}\left[c_t^{\text{text}}; c_t^{\text{cur}}; c_t^{\text{tgt}}; \delta_t\right] + \eta_t^{\text{glob}} \tag{D.15}$$

where $M_{\text{in}}^{\text{intra}} = \{q_0^{\text{intra}}, \cdots, q_{t-1}^{\text{intra}}\}$ and $M_{\text{in}}^{\text{intra}} = \{h_0^{\text{intra}}, \cdots, h_{t-1}^{\text{intra}}\}$ are the input and output intra-task memories. The final state is determined by subtracting a scaled version of $\tilde{h}_t^{\text{intra}}$ from $\bar{h}_t^{\text{intra}}$

$$h_t^{\text{intra}} = \bar{h}_t^{\text{intra}} - \beta \cdot \tilde{h}_t^{\text{intra}} \tag{D.16}$$

$$\beta = \sigma(W_{\text{gate}} \cdot [\bar{h}_t^{\text{intra}}; \tilde{h}_t^{\text{intra}}]) \tag{D.17}$$

Finally, the navigation action distribution is computed as follows

$$P_t^{\text{nav}} = \textsc{Softmax}(W^{\text{out}}\, h_t^{\text{intra}}\, W^{\text{act}}\, E_t^{\text{nav}})) \tag{D.18}$$

where E_t^{nav} is a matrix containing embeddings of the navigation actions. The computations in the help-request network is almost identical except that (a) the navigation action distribution P_t^{nav} is fed as an extra input to the intra-task components (D.15), and (b) the help-request and reason

Table D.1: Hyperparameters. (*) Training collapses when using $\alpha = 1$ to train the agent with the help-request policy baselines (NOASK, RANDOMASK, ASKEVERY5). Instead, we use $\alpha = 0.5$ in those experiments.

Hyperparameter	Value
Common	
Hidden size	256
Navigation action embedding size	256
Help-requesting action embedding size	128
Word embedding size	256
Number of self-attention heads	8
Number of instruction-attention heads	8
ResNet-extracted visual feature size	2048
Help-request teacher	
Uncertainty threshold (γ)	0.25
Training	
Optimizer	Adam
Number of training iterations	3×10^4
Learning rate	10^{-4}
Training batch size	32
Dropout ratio	0.3
Training time steps	20
Maximum instruction length	50
Curiosity-encouraging weight (α)	$1.0^{(*)}$
Evaluation	
Success radius ($\epsilon_{\text{success}}$)	2 meters
Attention radius (ϵ_{attn})	2 meters
Evaluation time steps	50
Evaluation batch size	32

distributions are calculated as follows

$$P_t^{\text{ask}} = \textsc{Softmax}(E_t^{\text{ask}} E_t^{\text{reason}} h_t^{\text{ask,intra}}) \tag{D.19}$$

$$P_t^{\text{reason}} = \textsc{Softmax}(E_t^{\text{reason}} h_t^{\text{ask,intra}}) \tag{D.20}$$

where E_t^{ask} contains embeddings of the help-request actions and E_t^{reason} contains embeddings of the reason labels.

Table D.2: Ablation study on our proposed techniques. Models are evaluated on the validation splits and with a batch size of 1. Best results in each column are in bold.

Agent	VAL SEENENV				VAL UNSEENALL			
	SR ↑ (%)	SPL ↑ (%)	Nav. ↓ Err. (m)	Requests/ task ↓	SR ↑ (%)	SPL ↑ (%)	Nav. ↓ Err. (m)	Requests/ task ↓
Final agent	86.06	62.49	1.48	2.7	45.39	23.33	8.21	**4.5**
No inter-task reset	83.62	60.44	1.44	3.2	40.60	19.89	8.26	7.3
No condition prediction	69.51	44.87	3.11	4.7	**45.72**	**23.35**	**6.56**	8.1
No cosine-similarity attention ($\beta = \mathbf{0}$)	80.44	56.63	1.89	3.7	38.69	19.86	8.81	8.5
No curiosity-encouraging loss ($\alpha = 0$)	**86.66**	**69.66**	**1.28**	**2.0**	39.92	21.33	8.66	7.7

D.3 Hyperparameters

Table D.1 summarizes all training and evaluation hyperparameters. Training our agent takes approximately 9 hours on a single Titan Xp GPU.

D.4 Analysis

Ablation Study Table D.2 shows an ablation study on the techniques we propose in this paper. We observe the performance on VAL SEENENV of the model trained without the curiosity-encouraging loss ($\alpha = 0$) is slightly higher than that of our final model, indicating that training with the curiosity-encouraging loss slightly hurts memorization. This is expected because the model has relatively small size but has to devote part of its parameters to learn the curiosity-encouraging behavior. On the other hand, optimizing with the curiosity-encouraging loss helps our agent generalize better to unseen environments. Predicting help-request conditions produces contrasting effects on the agent in seen and unseen environments, boosting performance in VAL SEENENV while slightly degrading performance in VAL UNSEENALL. We investigate the help-request patterns and find that, in both types of environments, the agent makes significantly more

requests when not learning to predict the conditions. Specifically, the no-condition-prediction agent meets the `uncertain_wrong` condition considerably more often (+5.2% on VAL SEENENV and +6.4% on VAL UNSEENALL), and also requests help at a previously visited location while executing the same task more frequently (+7.22% on VAL SEENENV and +8.59% on VAL UNSEENALL). This phenomenon highlights a challenge in training the navigation and training the help-request policies jointly: as the navigation policy gets better, there are less positive examples (i.e., situations where the agent needs help) for the help-request policy to learn from. In this specific case, learning a navigation policy that is less accurate in seen environments is beneficial to the agent when it encounters unseen environments because such a navigation policy creates more positive examples for training the help-request. Devising learning strategies that learns both efficient navigation and help-request policies is an exciting future direction.

Help-request Behavior on TEST UNSEENALL Our final agent requests about 18% of the total number of time steps (Figure D.1(a)). Overall, it learns a conservative help-request policy (Figure D.1(b)). Figure D.2 shows how accurate our agent predicts the help-request conditions. The agent achieves high precision scores in predicting the `lost` and `uncertain_wrong` conditions (76.9% and 86.6%), but achieves lower recalls in all conditions (less than 50%). Surprisingly, it predicts the `already_asked` condition less accurate than the other two, even though this condition is intuitively fairly simple to realize.

E Experimental Setup of HARI

Training Algorithms. We pre-train the execution policy $\hat{\pi}$ with DAgger (Ross et al., 2011), minimizing the cross entropy between its action distribution with that of a shortest-path oracle

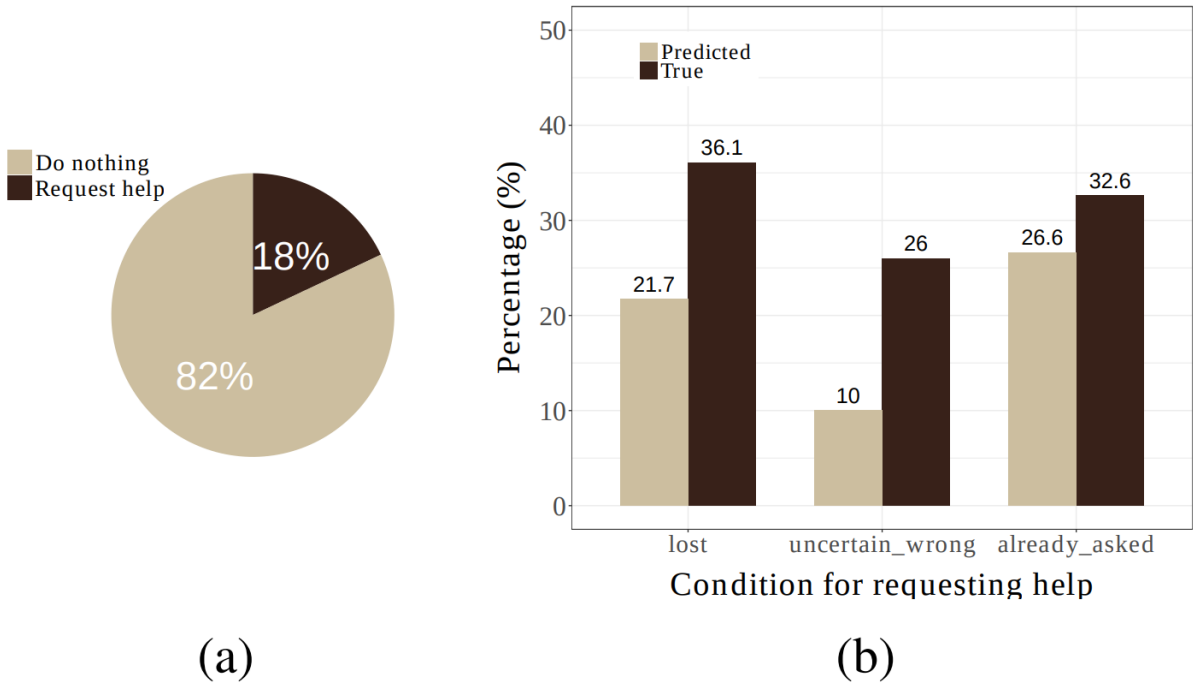


Figure D.1: Help-request behavior on TEST UNSEENALL: (a) fraction of time steps where the agent requests help and (b) predicted and true condition distributions. The `already_asked` condition is the negation of the `never_asked` condition.

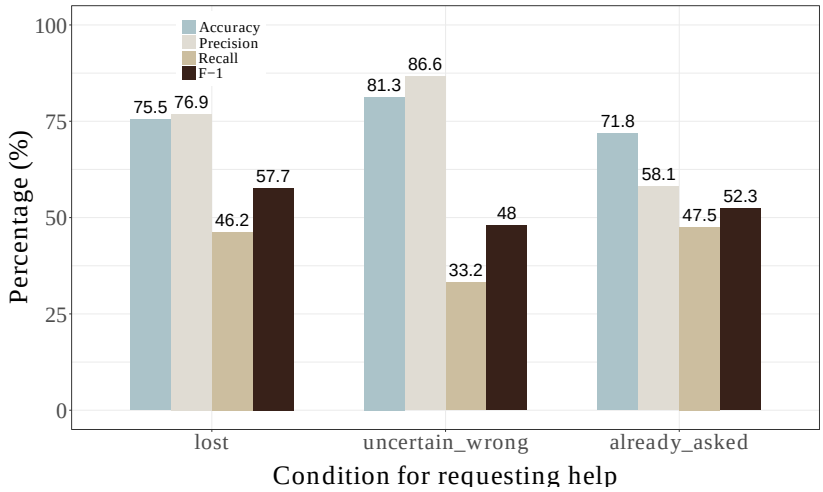


Figure D.2: Accuracy, precision, recall, and F-1 scores in predicting the help-request conditions on TEST UNSEENALL. The `already_asked` condition is the negation of the `never_asked` condition.

(which is a one-hot distribution with all probability concentrated on the optimal action).

We use advantage actor-critic (Mnih et al., 2016) to train the intention policy ψ_θ. This method simultaneously estimates an actor policy $\psi_\theta : \bar{\mathcal{B}} \to \Delta(\bar{\mathcal{A}})$ and a critic function $V_\phi : \bar{\mathcal{B}} \to \mathbb{R}$. Given an execution $\bar{\tau} = (\bar{s}_1, \bar{a}_1, \bar{c}_1 \cdots, \bar{s}_H)$, the gradients with respect to the actor and critic are

$$\nabla_\theta \mathcal{L}_{\text{actor}} = \sum_{t=1}^{H} \left(V_\phi(\bar{b}_t^v) - C_t\right) \nabla_\theta \log \psi_\theta(\bar{a}_t \mid \bar{b}_t^a) \tag{E.1}$$

$$\nabla_\phi \mathcal{L}_{\text{critic}} = \sum_{t=1}^{H} \left(V_\phi(\bar{b}_t^v) - C_t\right) \nabla_\phi V_\phi(\bar{b}_t^v) \tag{E.2}$$

where $C_t = \sum_{j=t}^{H} c_j$, $\bar{b}_t^a$ is a belief state that summarizes the partial execution $\bar{\tau}_{1:t}$ for the actor, and $\bar{b}_t^v$ is a belief state for the critic.

Model Architecture. We adapt the V&L BERT architecture (Hong et al., 2020) for modeling the execution policy $\hat{\pi}$. Our model has two components: an encoder and a decoder; both are implemented as Transformer models (Vaswani et al., 2017). The encoder takes as input a description d_t^s or d_t^g and generates a sequence of hidden vectors. In every step, the decoder takes as input the previous hidden vector b_{t-1}^s, the sequence of vectors representing d_t^s, and the sequence of vectors representing d_t^g. It then performs self-attention on these vectors to compute the current hidden vector b_t^s and a probability distribution over navigation actions p_t.

The intention policy ψ_θ (the actor) is an LSTM-based recurrent neural network. The input of this model is the execution policy's model outputs, b_t^s and p_t, and the embedding of the previously taken action $\bar{a}_{t-1}$. The critic model also has a similar architecture but outputs a real number (the V value) rather than an action distribution. When training the intention policy, we always fix

the parameters of the execution policy. We find it necessary to pre-train the critic before training it jointly with the actor.

Representation of State Descriptions. The representation of each object, room, or action is computed as follows. Let f^{name}, f^{horz}, f^{vert}, f^{dist}, and f^{type} are the features of an object f, consisting of its name, horizontal angle, vertical angle, distance, and type (a type is either `Object`, `Room`, or `Action`; in this case, the type is `Object`). For simplicity, we discretize real-valued features, resulting in 12 horizontal angles (corresponding to $\pi/6 \cdot k, 0 \leq k < 12$), 3 vertical angles (corresponding to $\pi/6 \cdot k, -1 \leq k \leq 1$), and 5 distance values (we round down a real-valued distance to the nearest integer). We then lookup the embedding of each feature from an embedding table and sum all the embeddings into a single vector that represents the corresponding object. For a room, f^{horz}, f^{vert} f^{dist} are zeroes. For an action, f^{name} is either `ActionStop` for the stop action a_{done} or `ActionGo` otherwise.

During the pre-training phase, we randomly drop features in d_t^s and d_t^g so that the execution policy is familiar with making decisions under sparse information. Concretely, we refer to all features of an object, room or action as a *feature set*. For d_t^s, let M be the number of objects in a description. We uniformly randomly keep m feature sets among the $M + 1$ feature sets of d_t^s (the plus one is the room's feature set), where $m \sim \text{Uniform}(\min(5, M + 1), M + 1)$.

For d_t^s, we have two cases. If g_1 is not adjacent or equals to s_1, we uniformly randomly alternate between giving a dense and a sparse description. In this case, the sparse description contains the features of the target object and the goal room's name. Otherwise, with a probability of ⅓, we give either (a) a dense description (b) a (sparse) description that contains the target object's features and the goal room's name, or (c) a (sparse) description that describes the next

Table E.1: Dataset statistics.

Split	Number of examples
Pre-training	82,104
Pre-training validation	3,000
Training	65,133
Validation UNSEENSTR	1,901
Validation UNSEENOBJ	1,912
Validation UNSEENENV	1,967
Test UNSEENSTR	1,653
Test UNSEENOBJ	1,913
Test UNSEENENV	1,777

ground-truth action.

We pre-train the execution policy on various path lengths (ranging from 1 to 10 graph nodes) so that it learns to accomplish both long-distance main goals and short-distance subgoals.

Data. Table E.1 summarizes the data splits. From a total of 72 environments provided by the Matterport3D dataset, we use 36 environments for pre-training, 18 as unseen environments for training, 7 for validation UNSEENENV, and 11 for test UNSEENENV. We use a vocabulary of size 1738, which includes object and room names, and special tokens representing the distance and direction values. The length of a navigation path ranges from 5 to 10 graph nodes.

Hyperparameters See Table E.2.

Table E.2: Hyperparameters.

Hyperparameter Name	Value
Environment	
Max. subgoal distance (l_{max})	3 nodes
Max. stack size (L)	2
Max. object distance for d_t^s	5 meters
Max. object distance for d_t^g	3 meters
Max. number of objects (M_{max})	20
Cost of taking each CUR, GOAL, SUB, DO action	0.01
Operation policy $\hat{\pi}$	
Hidden size	256
Number of hidden layers	2
Attention dropout probability	0.1
Hidden dropout probability	0.1
Number of attention heads	8
Optimizer	Adam
Learning rate	10^{-4}
Batch size	32
Number of training iterations	10^5
Max. number of time steps (H)	15
Interaction policy ψ_θ	
Hidden size	512
Number of hidden layers	1
Entropy regularization weight	0.001
Optimizer	Adam
Learning rate	10^{-5}
Batch size	32
Number of critic pre-training iterations	5×10^3
Number of training iterations	5×10^4
Max. number of time steps (H)	30
Max. number of time steps for executing a subgoal	$3\times$ shortest distance to the subgoal

Bibliography

Ehsan Abbasnejad, Qi Wu, Qinfeng Shi, and Anton van den Hengel. What's to know? uncertainty as a guide to asking goal-oriented questions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4155–4164, 2019.

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

Daniel Adiwardana, Minh-Thang Luong, David R So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, et al. Towards a human-like open-domain chatbot. *arXiv preprint arXiv:2001.09977*, 2020.

Gediminas Adomavicius and Jingjing Zhang. Impact of data characteristics on recommender systems performance. *ACM Transactions on Management Information Systems (TMIS)*, 3(1): 3, 2012.

Alekh Agarwal, Sham Kakade, Akshay Krishnamurthy, and Wen Sun. Flambe: Structural complexity and representation learning of low rank mdps. In *Proceedings of Advances in Neural Information Processing Systems*, 2020.

Ahmed Akakzia, Cédric Colas, Pierre-Yves Oudeyer, Mohamed Chetouani, and Olivier Sigaud. Grounding language to autonomously-acquired skills via goal generation, 2021.

James Allen, Nathanael Chambers, George Ferguson, Lucian Galescu, Hyuckchul Jung, Mary Swift, and William Taysom. Plow: A collaborative task learning agent. In *AAAI*, volume 7, pages 1514–1519, 2007.

Muhannad Alomari, Paul Duckworth, David Hogg, and Anthony Cohn. Natural language acquisition and grounding for embodied robotic systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. Power to the people: The role of humans in interactive machine learning. *Ai Magazine*, 35(4):105–120, 2014.

Prithviraj Ammanabrolu, Ethan Tien, Matthew Hausknecht, and Mark O Riedl. How to avoid being eaten by a grue: Structured exploration strategies for textual worlds. *arXiv preprint arXiv:2006.07409*, 2020.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018a.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018b.

Jacob Andreas and Dan Klein. Reasoning about pragmatics with neural listeners and speakers. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1125.

Jacob Andreas, Dan Klein, and Sergey Levine. Learning with latent language. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2166–2179, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1197.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Dana Angluin. Queries and concept learning. *Machine learning*, 2(4):319–342, 1988.

Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. In *Ethics of Data and Analytics*, pages 254–264. Auerbach Publications, 2016.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.

András Antos, Varun Grover, and Csaba Szepesvári. Active learning in heteroscedastic noise. *Theoretical Computer Science*, 411(29-30):2712–2728, 2010.

Kate Arnold and Klaus Zuberbühler. Semantic combinations in primate calls. *Nature*, 441(7091): 303–303, 2006.

Yoav Artzi and Luke Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62, 2013.

Amos Azaria, Jayant Krishnamurthy, and Tom M Mitchell. Instructable intelligent personal agent. In *AAAI*, volume 4, 2016.

Nils Backhaus, Patricia H Rosen, Andrea Scheidig, Horst-Michael Gross, and Sascha Wischniewski. Somebody help me, please?!" interaction design framework for needy mobile service robots. In *2018 IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, pages 54–61. IEEE, 2018.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. In *International Conference on Learning Representations (ICLR)*, 2017.

Markus Bajones. Enabling long-term human-robot interaction through adaptive behavior coordination. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 597–598. IEEE, 2016.

Markus Bajones, Astrid Weiss, and Markus Vincze. Help, anyone? a user study for modeling robotic behavior to mitigate malfunctions with the help of the user. *arXiv preprint arXiv:1606.02547*, 2016.

Cristian-Paul Bara, Sky CH-Wang, and Joyce Chai. MindCraft: Theory of mind modeling for situated dialogue in collaborative tasks. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1112–1125, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.85.

Kobus Barnard and Matthew Johnson. Word sense disambiguation with pictures. *Artificial Intelligence*, 167(1-2):13–30, 2005.

Kobus Barnard, Pinar Duygulu, David Forsyth, Nando de Freitas, David M Blei, and Michael I Jordan. Matching words and pictures. *Journal of machine learning research*, 3(Feb):1107–1135, 2003.

Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning*. fairmlbook.org, 2019. http://www.fairmlbook.org.

Simon Baron-Cohen, Alan M. Leslie, and Uta Frith. Does the autistic child have a "theory of mind" ? *Cognition*, 21(1):37–46, 1985. ISSN 0010-0277. doi: https://doi.org/10.1016/0010-0277(85)90022-8.

Andrea Bauer, Dirk Wollherr, and Martin Buss. Human–robot collaboration: a survey. *International Journal of Humanoid Robotics*, 5(01):47–66, 2008.

Beatrice Beebe, Daniel Messinger, Lorraine E Bahrick, Amy Margolis, Karen A Buck, and Henian Chen. A systems view of mother–infant face-to-face communication. *Developmental psychology*, 52(4):556, 2016.

Tamara L Berg, Alexander C Berg, Jaety Edwards, Michael Maire, Ryan White, Yee-Whye Teh, Erik Learned-Miller, and David A Forsyth. Names and faces in the news. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II. IEEE, 2004.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Y. Bisk, D. Yuret, and D. Marcu. Natural language communication with robots. In *North American Association for Computational Linguistics (NAACL)*, 2016.

Valts Blukis, Dipendra Misra, Ross A Knepper, and Yoav Artzi. Mapping navigation instructions to continuous control actions with position-visitation prediction. In *Conference on Robot Learning*, pages 505–518, 2018.

Antoine Bordes, Y-Lan Boureau, and Jason Weston. Learning end-to-end goal-oriented dialog. In *Proceedings of the International Conference on Learning Representations*, 2017.

S. Branavan, N. Kushman, T. Lei, and R. Barzilay. Learning high-level planning from text. In *Association for Computational Linguistics (ACL)*, pages 126–135, 2012.

Satchuthananthavale RK Branavan, Harr Chen, Luke S Zettlemoyer, and Regina Barzilay. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 82–90. Association for Computational Linguistics, 2009.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Simon Brodeur, Ethan Perez, Ankesh Anand, Florian Golemo, Luca Celotti, Florian Strub, Jean Rouat, Hugo Larochelle, and Aaron Courville. Home: A household multimodal environment. In *Proceedings of Advances in Neural Information Processing Systems*, 2017.

John Seely Brown, Allan Collins, and Paul Duguid. Situated cognition and the culture of learning. *Educational researcher*, 18(1):32–42, 1989.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020a.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020b.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1547.

Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91. PMLR, 2018.

Josep Call and Michael Tomasello. Does the chimpanzee have a theory of mind? 30 years later. *Trends in Cognitive Sciences*, 12(5):187–192, 2008. ISSN 1364-6613. doi: https://doi.org/10.1016/j.tics.2008.02.010.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. e-snli: Natural language inference with natural language explanations. In *Proceedings of Advances in Neural Information Processing Systems*, 2018.

David Cameron, Emily C Collins, Adriel Chua, Samuel Fernando, Owen McAree, Uriel Martinez-Hernandez, Jonathan M Aitken, Luke Boorman, and James Law. Help! i can't reach the buttons: Facilitating helping behaviors towards robots. In *Conference on Biomimetic and Biohybrid Systems*, pages 354–358. Springer, 2015.

David Cameron, Ee Jing Loh, Adriel Chua, Emily Collins, Jonathan M Aitken, and James Law. Robot-stated limitations but not intentions promote user assistance. *arXiv preprint arXiv:1606.02603*, 2016.

Angelo Cangelosi. Evolution of communication and language using signals, symbols, and words. *IEEE Transactions on Evolutionary Computation*, 5(2):93–101, 2001.

Malinda Carpenter, Katherine Nagell, Michael Tomasello, George Butterworth, and Chris Moore. Social cognition, joint attention, and communicative competence from 9 to 15 months of age. *Monographs of the society for research in child development*, pages i–174, 1998.

Mauro Cettolo, Christian Girardi, and Marcello Federico. Wit^3: Web inventory of transcribed and translated talks. In *Conference of the European Association for Machine Translation (EAMT)*, Trento, Italy, 2012.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. Report on the 11th IWSLT evaluation campaign, IWSLT 2014. In *International Workshop on Spoken Language Translation (IWSLT)*, 2014.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, Roldano Cattoni, and Marcello Federico. The IWSLT 2015 evaluation campaign. In *International Workshop on Spoken Language Translation (IWSLT)*, 2015.

Elizabeth Cha and Maja Matarić. Using nonverbal signals to request help during human-robot collaboration. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5070–5076. IEEE, 2016.

Harris Chan, Yuhuai Wu, Jamie Kiros, Sanja Fidler, and Jimmy Ba. Actrce: Augmenting experience via teacher's advice for multi-goal reinforcement learning. *arXiv preprint arXiv:1902.04546*, 2019.

Balasubramaniyan Chandrasekaran and James M Conrad. Human-robot collaboration: A survey. In *SoutheastCon 2015*, pages 1–8. IEEE, 2015.

Elliot Chane-Sane, Cordelia Schmid, and Ivan Laptev. Goal-conditioned reinforcement learning with imagined subgoals. In *International Conference on Machine Learning*, pages 1430–1440. PMLR, 2021.

Angel Chang, Manolis Savva, and Christopher D Manning. Learning spatial knowledge for text to 3d scene generation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2028–2038, 2014.

Angel Chang, Will Monroe, Manolis Savva, Christopher Potts, and Christopher D Manning. Text to 3d scene generation with rich lexical grounding. *arXiv preprint arXiv:1505.06289*, 2015a.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*, 2017a.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017b.

K. Chang, A. Krishnamurthy, A. Agarwal, H. Daume, and J. Langford. Learning to search better than your teacher. *arXiv*, 2015b.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. Learning to search better than your teacher. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015c.

Pi-Chuan Chang, Michel Galley, and Chris Manning. Optimizing chinese word segmentation for machine translation performance. In *Workshop on Machine Translation*, 2008.

Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. Gated-attention architectures for task-oriented language grounding. In *Association for the Advancement of Artificial Intelligence*, 2018.

Boxing Chen and Colin Cherry. A systematic comparison of smoothing techniques for sentence-level bleu. In *Association for Computational Linguistics (ACL)*, 2014.

Chun-Yen Chen, Dian Yu, Weiming Wen, Yi Mang Yang, Jiaping Zhang, Mingyang Zhou, Kevin Jesse, Austin Chau, Antara Bhowmick, Shreenath Iyer, et al. Gunrock: Building a human-like social bot by leveraging large scale real user data. *Alexa Prize Proceedings*, 2018.

David L Chen and Raymond J Mooney. Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th international conference on Machine learning*, pages 128–135, 2008.

David L Chen and Raymond J Mooney. Learning to interpret natural language navigation instructions from observations. In *Association for the Advancement of Artificial Intelligence*, volume 2, pages 1–2, 2011.

Howard Chen, Alane Shur, Dipendra Misra, Noah Snavely, Ian Artzi, Yoav, Stephen Gould, and Anton van den Hengel. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019a.

Howard Chen, Alane Suhr, Dipendra Misra, and Yoav Artzi. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019b.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34, 2021.

Qian Chen, Zhu Zhuo, and Wen Wang. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*, 2019c.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. In *Proceedings of the International Conference on Learning Representations*, 2019.

Ta-Chung Chi, Minmin Shen, Mihail Eric, Seokhwan Kim, and Dilek Hakkani-tur. Just ask: An interactive learning framework for vision and language navigation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2459–2466, 2020.

François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

Leshem Choshen, Lior Fox, Zohar Aizenbud, and Omri Abend. On the weaknesses of reinforcement learning for neural machine translation. In *Proceedings of the International Conference on Learning Representations*, 2020.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Baindoor Rao, Parker Barnes, Yi Tay, Noam M. Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Benton C. Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier García, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Oliveira Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S. Meier-Hellstern, Douglas

Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *ArXiv*, abs/2204.02311, 2022a.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022b.

P. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Geoffrey Cideron, Mathieu Seurin, Florian Strub, and Olivier Pietquin. Higher: Improving instruction following with hindsight generation for experience replay. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 225–232. IEEE, 2020.

H. H. Clark and S. E. Brennan. *Grounding in Communication*. Perspectives on Socially Shared Cognition, 1991.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. Driving semantic parsing from the world's response. In *Proceedings of the fourteenth conference on computational natural language learning*, pages 18–27, 2010.

David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine learning*, 15(2):201–221, 1994.

Cédric Colas, Tristan Karch, Nicolas Lair, Jean-Michel Dussoux, Clément Moulin-Frier, Peter Ford Dominey, and Pierre-Yves Oudeyer. Language as a cognitive tool to imagine goals in curiosity-driven exploration. In *Proceedings of Advances in Neural Information Processing Systems*, 2020.

J Edward Colgate, J Edward, Michael A Peshkin, and Witaya Wannasuphoprasit. Cobots: Robots for collaboration with human operators. 1996.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. Textworld: A learning environment for text-based games. In *Computer Games Workshop at ICML/IJCAI*, 2018.

Bob Coyne and Richard Sproat. Wordseye: an automatic text-to-scene conversion system. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 487–496, 2001.

Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.

Aron Culotta and Andrew McCallum. Reducing labeling effort for structured prediction tasks. In *AAAI*, volume 5, pages 746–751, 2005.

Felipe Leno Da Silva and Anna Helena Reali Costa. A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research*, 64:645–703, 2019.

Felipe Leno Da Silva, Ruben Glatt, and Anna Helena Reali Costa. Simultaneously learning and advising in multiagent reinforcement learning. In *Proceedings of the 16th conference on autonomous agents and multiagent systems*, pages 1100–1108, 2017.

Felipe Leno Da Silva, Pablo Hernandez-Leal, Bilal Kartal, and Matthew E Taylor. Uncertainty-aware action advising for deep reinforcement learning agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5792–5799, 2020.

A. Das, S. Kottur, J. M. Moura, S. Lee, and D. Batra. Learning cooperative visual dialog agents with deep reinforcement learning. In *International Conference on Computer Vision*, 2017a.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh, and Dhruv Batra. Visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 326–335, 2017b.

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Arun Das and Paul Rad. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*, 2020.

Hal Daumé, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

Leentje De Schuymer, Isabel De Groote, Tricia Striano, Daniel Stahl, and Herbert Roeyers. Dyadic and triadic skills in preterm and full term infants: A longitudinal study in the first year. *Infant Behavior and Development*, 34(1):179–188, 2011.

Harm De Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron Courville. Guesswhat?! visual object discovery through multi-modal dialogue. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5503–5512, 2017.

Harm de Vries, Kurt Shuster, Dhruv Batra, Devi Parikh, Jason Weston, and Douwe Kiela. Talk the walk: Navigating new york city through grounded dialogue. *arXiv preprint arXiv:1807.03367*, 2018.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423.

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. In *Proceedings of the International Conference of Machine Learning*, 2019.

Ahmed Elgohary, Saghar Hosseini, and Ahmed Hassan Awadallah. Speak to your parser: Interactive text-to-sql with natural language feedback. *arXiv preprint arXiv:2005.02539*, 2020.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. Multi30K: Multilingual English-German image descriptions. In *Proceedings of the 5th Workshop on Vision and Language*, pages 70–74, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-3210.

Frank Emmert-Streib, Olli Yli-Harja, and Matthias Dehmer. Explainable artificial intelligence and machine learning: A reality rooted perspective. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(6):e1368, 2020.

Katrina Evtimova, Andrew Drozdov, Douwe Kiela, and Kyunghyun Cho. Emergent communication in a multi-modal, multi-step referential game. In *Proceedings of the International Conference on Learning Representations*, 2018.

Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1625–1634, 2018.

Ben Eysenbach, Xinyang Geng, Sergey Levine, and Russ R Salakhutdinov. Rewriting history with inverse rl: Hindsight inference for policy improvement. *Advances in neural information processing systems*, 33:14783–14795, 2020.

Jerry Alan Fails and Dan R Olsen Jr. Interactive machine learning. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 39–45, 2003.

Angela Fan, Jack Urbanek, Pratik Ringshia, Emily Dinan, Emma Qian, Siddharth Karamcheti, Shrimai Prabhumoye, Douwe Kiela, Tim Rocktäschel, Arthur Szlam, et al. Generating interactive worlds with text. In *AAAI*, pages 1693–1700, 2020.

M. Fang, Y. Li, and T. Cohn. Learning how to active learn: A deep reinforcement learning approach. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

Ethan Fast, Binbin Chen, Julia Mendelsohn, Jonathan Bassen, and Michael S Bernstein. Iris: A conversational agent for complex tasks. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2018.

Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. Pathologies of neural models make interpretations difficult. *arXiv preprint arXiv:1804.07781*, 2018.

Sanja Fidler et al. Teaching machines to describe images with natural language feedback. In *Advances in Neural Information Processing Systems*, pages 5068–5078, 2017.

Jakob N. Foerster, Yannis M. Assael, N. D. Freitas, and S. Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *NIPS*, 2016.

Terrence Fong, Illah Nourbakhsh, and Kerstin Dautenhahn. A survey of socially interactive robots. *Robotics and autonomous systems*, 42(3-4):143–166, 2003.

Daniel Fried, Jacob Andreas, and Dan Klein. Unified pragmatic models for generating and following instructions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1951–1963, New Orleans, Louisiana, June 2018a. Association for Computational Linguistics. doi: 10.18653/v1/N18-1177.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. Speaker-follower models for vision-and-language navigation. In *Proceedings of Advances in Neural Information Processing Systems*, 2018b.

Justin Fu, Anoop Korattikara, Sergey Levine, and Sergio Guadarrama. From language to goals: Inverse reinforcement learning for vision-based instruction following. *arXiv preprint arXiv:1902.07742*, 2019.

Zhe Gan, Yen-Chun Chen, Linjie Li, Chen Zhu, Yu Cheng, and Jingjing Liu. Large-scale adversarial training for vision-and-language representation learning. In *Proceedings of Advances in Neural Information Processing Systems*, 2020.

Jianfeng Gao, Michel Galley, and Lihong Li. Neural approaches to conversational ai. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1371–1374, 2018.

Atticus Geiger, Zhengxuan Wu, Hanson Lu, Josh Rozner, Elisa Kreiss, Thomas Icard, Noah D Goodman, and Christopher Potts. Inducing causal structure for interpretable neural networks. *arXiv preprint arXiv:2112.00826*, 2021.

Sayan Ghosh and Shashank Srivastava. Mapping language to programs using multiple reward components with inverse reinforcement learning. *arXiv preprint arXiv:2110.00842*, 2021.

Dan Goldwasser and Dan Roth. Learning from natural instructions. *Machine learning*, 94(2): 205–232, 2014.

Dave Golland, Percy Liang, and Dan Klein. A game-theoretic approach to generating spatial descriptions. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 410–419, Cambridge, MA, October 2010. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D10-1040.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27:2672–2680, 2014.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proceedings of the International Conference on Learning Representations*, 2015.

N. D. Goodman and M. C. Frank. Pragmatic language interpretation as probabilistic inference. *Trends in Cognitive Sciences*, 20(11):818–829, 2016.

Alison Gopnik and Janet W Astington. Children's understanding of representational change and its relation to the understanding of false belief and the appearance-reality distinction. *Child development*, pages 26–37, 1988.

Daniel Gordon, Kiana Ehsani, Dieter Fox, and Ali Farhadi. Watching the world go by: Representation learning from unlabeled videos. *arXiv preprint arXiv:2003.07990*, 2020.

Prasoon Goyal, Scott Niekum, and Raymond J Mooney. Using natural language for reward shaping in reinforcement learning. *arXiv preprint arXiv:1903.02020*, 2019.

Arthur C Graesser, Natalie Person, and John Huber. Mechanisms that generate questions. *Questions and information systems*, 2:167–187, 1992.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. Can machine translation systems be evaluated by the crowd alone. *Natural Language Engineering*, 23(1):3–30, 2017.

Mary L Gray and Siddharth Suri. *Ghost work: How to stop Silicon Valley from building a new global underclass*. Eamon Dolan Books, 2019.

Herbert P Grice. Logic and conversation. In *Speech acts*, pages 41–58. Brill, 1975.

Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, and Andrea L Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. Georgia Institute of Technology, 2013.

Alvin Grissom II, He He, Jordan Boyd-Graber, John Morgan, and Hal Daumé III. Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

Jing Gu, Eliana Stefani, Qi Wu, Jesse Thomason, and Xin Eric Wang. Vision-and-language navigation: A survey of tasks, methods, and future directions. *arXiv preprint arXiv:2203.12667*, 2022.

Pierre-Louis Guhur, Makarand Tapaswi, Shizhe Chen, Ivan Laptev, and Cordelia Schmid. Airbert: In-domain pretraining for vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1634–1643, 2021.

Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Thomas Paine, Sergio Gómez, Konrad Zolna, Rishabh Agarwal, Josh S Merel, Daniel J Mankowitz, Cosmin Paduraru, et al. Rl unplugged: A suite of benchmarks for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:7248–7259, 2020.

Danna Gurari, Yinan Zhao, Meng Zhang, and Nilavra Bhattacharya. Captioning images taken by people who are blind. *arXiv preprint arXiv:2002.08565*, 2020.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018.

Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Stein. The argument reasoning comprehension task: Identification and reconstruction of implicit warrants. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1930–1940, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1175.

Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. Cooperative inverse reinforcement learning. *Advances in neural information processing systems*, 29, 2016.

Braden Hancock, Martin Bringmann, Paroma Varma, Percy Liang, Stephanie Wang, and Christopher Ré. Training classifiers with natural language explanations. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2018, page 1884. NIH Public Access, 2018.

Weituo Hao, Chunyuan Li, Xiujun Li, Lawrence Carin, and Jianfeng Gao. Towards learning a generic agent for vision-and-language navigation via pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13137–13146, 2020.

Stevan Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3): 335–346, 1990.

Serhii Havrylov and Ivan Titov. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. In *Proceedings of Advances in Neural Information Processing Systems*, 2017.

H. He, H. Daume, and J. Eisner. Dynamic feature selection for dependency parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1455–1464, 2013.

He He, Jason Eisner, and Hal Daume. Imitation learning by coaching. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

He He, Alvin Grissom II, Jordan Boyd-Graber, and Hal Daumé III. Syntax-based rewriting for simultaneous machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

He He, Jordan Boyd-Graber, and Hal Daumé III. Interpretese vs. translationese: The uniqueness of human strategies in simultaneous interpretation. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016b.

Charles T Hemphill, John J Godfrey, and George R Doddington. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*, 1990.

Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating visual explanations. In *European conference on computer vision*, pages 3–19. Springer, 2016.

Dan Hendrycks, Nicholas Carlini, John Schulman, and Jacob Steinhardt. Unsolved problems in ml safety. *arXiv preprint arXiv:2109.13916*, 2021.

Jonathan L Herlocker, Joseph A Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *ACM Conference on Computer Supported Cooperative Work*, 2000.

Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Czarnecki, Max Jaderberg, Denis Teplyashin, Marcus Wainwright, Chris Apps, Demis Hassabis, and Phil Blunsom. Grounded language learning in a simulated 3D world. *CoRR*, abs/1706.06551, 2017a.

Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, et al. Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551*, 2017b.

Daniela Hernandez. Ibm's retreat from watson highlights broader ai struggles in health. https://www.wsj.com/articles/ibms-retreat-from-watson-highlights-broader-ai-struggles-in-health-11613839579, 2021.

Felix Hill, Andrew Lampinen, Rosalia Schneider, Stephen Clark, Matthew Botvinick, James L McClelland, and Adam Santoro. Environmental drivers of systematicity and generalization in a situated agent. In *International Conference on Learning Representations*, 2019.

Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. A recurrent vision-and-language bert for navigation. *arXiv preprint arXiv:2011.13922*, 2020.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1031.

Chang Hu, Philip Resnik, and Benjamin B Bederson. Crowdsourced monolingual translation. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 21(4):22, 2014.

Helge Hüttenrauch and Kerstin Severinson-Eklundh. To help or not to help a service robot: Bystander intervention as a resource in human–robot collaboration. *Interaction Studies*, 7(3): 455–477, 2006.

Charles Isbell, Christian R Shelton, Michael Kearns, Satinder Singh, and Peter Stone. A social reinforcement learning agent. In *International Conference on Autonomous Agents (AA)*, 2001.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34, 2021.

Prashant Jayannavar, Anjali Narayan-Chen, and Julia Hockenmaier. Learning to execute instructions in a minecraft dialogue. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 2589–2602, 2020.

Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1215.

Yiding Jiang, Shixiang Gu, Kevin Murphy, and Chelsea Finn. Language as an abstraction for hierarchical deep reinforcement learning. In *Proceedings of Advances in Neural Information Processing Systems*, 2019.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910, 2017.

Kshitij Judah, Saikat Roy, Alan Fern, and Thomas Dietterich. Reinforcement learning via practice and critique advice. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 2010.

Kshitij Judah, Alan P Fern, Thomas G Dietterich, and Prasad Tadepalli. Active imitation learning: Formal and practical reductions to iid learning. *Journal of Machine Learning Research*, 15 (120):4105–4143, 2014.

Sham M Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. Efficient bandit algorithms for online multiclass prediction. In *International Conference on Machine learning (ICML)*, 2008.

Yipeng Kang, Tonghan Wang, and Gerard de Melo. Incorporating pragmatic reasoning communication into emergent language. *Advances in Neural Information Processing Systems*, 33: 10348–10359, 2020.

Thommen Karimpanal. Intentionality in reinforcement learning. 2021.

Rohit J Kate, Yuk Wah Wong, and Raymond J Mooney. Learning to transform natural to formal languages. In *AAAI*, pages 1062–1068, 2005.

Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, pages 1–8. IEEE, 2016.

Kristian Kersting, Christian Plagemann, Patrick Pfaff, and Wolfram Burgard. Most likely heteroscedastic gaussian process regression. In *International Conference on Machine Learning (ICML)*, 2007.

Dong-Ki Kim, Miao Liu, Shayegan Omidshafiei, Sebastian Lopez-Cot, Matthew Riemer, Golnaz Habibi, Gerald Tesauro, Sami Mourad, Murray Campbell, and Jonathan P How. Learning hierarchical teaching policies for cooperative agents. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015a.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2015b.

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

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. Moses: Open source toolkit for statistical machine translation. In *Association for Computational Linguistics (ACL)*, 2007.

P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning (ICML)*, 2017.

J Kolter, Pieter Abbeel, and Andrew Ng. Hierarchical apprenticeship learning with application to quadruped locomotion. *Advances in Neural Information Processing Systems*, 20, 2007.

Eric Kolve, Roozbeh Mottaghi, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. In *SIAM Journal on Control and Optimization*, pages 1008–1014. MIT Press, 2000.

Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, 1999.

Satwik Kottur, José Moura, Stefan Lee, and Dhruv Batra. Natural language does not emerge 'naturally' in multi-agent dialog. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2962–2967, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1321. URL https://www.aclweb.org/anthology/D17-1321.

Julia Kreutzer, Artem Sokolov, and Stefan Riezler. Bandit structured prediction for neural sequence-to-sequence learning. In *Association of Computational Linguistics (ACL)*, 2017.

Julia Kreutzer, Shahram Khadivi, Evgeny Matusov, and Stefan Riezler. Can neural machine translation be improved with user feedback? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 92–105, New Orleans - Louisiana, June 2018a. Association for Computational Linguistics. doi: 10.18653/v1/N18-3012.

Julia Kreutzer, Joshua Uyheng, and Stefan Riezler. Reliability and learnability of human bandit feedback for sequence-to-sequence reinforcement learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788, Melbourne, Australia, July 2018b. Association for Computational Linguistics. doi: 10.18653/v1/P18-1165.

Julia Kreutzer, Stefan Riezler, and Carolin Lawrence. Learning from human feedback: Challenges for real-world reinforcement learning in nlp. 2020.

Jayant Krishnamurthy and Tom Mitchell. Weakly supervised training of semantic parsers. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, pages 754–765, 2012.

T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, pages 3675–3683, 2016.

Sawan Kumar and Partha Talukdar. NILE : Natural language inference with faithful natural language explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8730–8742, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.771.

Andrey Kurenkov. Reinforcement learning's foundational flaw. *The Gradient*, 2018.

Minae Kwon, Sandy H Huang, and Anca D Dragan. Expressing robot incapability. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pages 87–95, 2018.

Igor Labutov, Sumit Basu, and Lucy Vanderwende. Deep questions without deep understanding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*

and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 889–898, 2015.

Igor Labutov, Bishan Yang, and Tom Mitchell. Learning to learn semantic parsers from natural language supervision. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1676–1690, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1195.

John E Laird, Kevin Gluck, John Anderson, Kenneth D Forbus, Odest Chadwicke Jenkins, Christian Lebiere, Dario Salvucci, Matthias Scheutz, Andrea Thomaz, Greg Trafton, et al. Interactive task learning. *IEEE Intelligent Systems*, 32(4):6–21, 2017.

John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in neural information processing systems*, pages 817–824, 2008a.

John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20, pages 817–824. Curran Associates, Inc., 2008b. URL https://proceedings.neurips.cc/paper/2007/file/4b04a686b0ad13dce35fa99fa4161c65-Paper.pdf.

John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in Neural Information Processing Systems (NIPS)*, 2008c.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. In *Proceedings of the International Conference on Learning Representations*, 2017.

Hoang Le, Nan Jiang, Alekh Agarwal, Miroslav Dudík, Yisong Yue, and Hal Daumé. Hierarchical imitation and reinforcement learning. In *International conference on machine learning*, pages 2917–2926. PMLR, 2018.

David A Leavens, William D Hopkins, and Roger K Thomas. Referential communication by chimpanzees (pan troglodytes). *Journal of Comparative Psychology*, 118(1):48, 2004.

Kimin Lee, Laura Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In *Proceedings of the International Conference of Machine Learning*, 2021.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Hector Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*. Citeseer, 2012.

D. Lewis. *Convention: A philosophical study*. John Wiley & Sons, 2008.

Gen Li, Nan Duan, Yuejian Fang, Ming Gong, Daxin Jiang, and Ming Zhou. Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training. In *AAAI*, pages 11336–11344, 2020a.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, 2019.

Qing Li, Siyuan Huang, Yining Hong, Yixin Chen, Ying Nian Wu, and Song-Chun Zhu. Closed loop neural-symbolic learning via integrating neural perception, grammar parsing, and symbolic reasoning. In *Proceedings of the International Conference of Machine Learning*, 2020b.

Toby Jia-Jun Li, Yuanchun Li, Fanglin Chen, and Brad A Myers. Programming iot devices by demonstration using mobile apps. In *International Symposium on End User Development*, pages 3–17. Springer, 2017.

Toby Jia-Jun Li, Jingya Chen, Tom M Mitchell, and Brad A Myers. Towards effective human-ai collaboration in gui-based interactive task learning agents. *arXiv preprint arXiv:2003.02622*, 2020c.

Toby Jia-Jun Li, Marissa Radensky, Justin Jia, Kirielle Singarajah, Tom M Mitchell, and Brad A Myers. Interactive task and concept learning from natural language instructions and gui demonstrations. In *The AAAI-20 Workshop on Intelligent Process Automation (IPA-20)*, 2020d.

Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. Mapping natural language instructions to mobile UI action sequences. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8198–8210, Online, July 2020e. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.729.

Henry Lieberman and Hugo Liu. Feasibility studies for programming in natural language. In *End User Development*, pages 459–473. Springer, 2006.

Rensis Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 22(140): 1–55, 1932.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

H. Ling and S. Fidler. Teaching machines to describe images via natural language feedback. In *Proceedings of Advances in Neural Information Processing Systems*, 2017.

Bang Liu, Haojie Wei, Di Niu, Haolan Chen, and Yancheng He. Asking questions the human way: Scalable question-answer generation from text corpus. In *Proceedings of The Web Conference 2020*, pages 2032–2043, 2020.

Changsong Liu, Shaohua Yang, Sari Saba-Sadiya, Nishant Shukla, Yunzhong He, Song-Chun Zhu, and Joyce Chai. Jointly learning grounded task structures from language instruction and visual demonstration. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1482–1492, 2016.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration. *arXiv preprint arXiv:1802.08802*, 2018.

Iou-Jen Liu, Xingdi Yuan, Marc-Alexandre Côté, Pierre-Yves Oudeyer, and Alexander G Schwing. Asking for knowledge: Training rl agents to query external knowledge using language. In *Proceedings of the International Conference of Machine Learning*, 2022.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Robert Loftin, James MacGlashan, Michael L Littman, Matthew E Taylor, and David L Roberts. A strategy-aware technique for learning behaviors from discrete human feedback. Technical report, North Carolina State University. Dept. of Computer Science, 2014.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems*, pages 13–23, 2019.

Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. A survey of reinforcement learning informed by natural language. In *International Joint Conference on Artificial Intelligence*, 2019.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

James MacGlashan, Monica Babes-Vroman, Marie desJardins, Michael L Littman, Smaranda Muresan, Shawn Squire, Stefanie Tellex, Dilip Arumugam, and Lei Yang. Grounding english commands to reward functions. In *Robotics: Science and Systems*, 2015.

Richard Maclin and Jude W Shavlik. Creating advice-taking reinforcement learners. *Machine Learning*, 22(1):251–281, 1996.

Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. Walk the talk: Connecting language, knowledge, and action in route instructions. In *National Conference on Artificial Intelligence*, 2006.

Andreas Madsen, Nicholas Meade, Vaibhav Adlakha, and Siva Reddy. Evaluating the faithfulness of importance measures in nlp by recursively masking allegedly important tokens and retraining. *arXiv preprint arXiv:2110.08412*, 2021.

Gabriel Ilharco Magalhaes, Vihan Jain, Alexander Ku, Eugene Ie, and Jason Baldridge. General evaluation for instruction conditioned navigation using dynamic time warping. In *Proceedings of Advances in Neural Information Processing Systems*, 2019.

Matthew Marge, Carol Espy-Wilson, Nigel G Ward, Abeer Alwan, Yoav Artzi, Mohit Bansal, Gil Blankenship, Joyce Chai, Hal Daumé III, Debadeepta Dey, et al. Spoken language interaction with robots: Recommendations for future research. *Computer Speech & Language*, 71:101255, 2022.

Danielle Matthews, Jessica Butcher, Elena Lieven, and Michael Tomasello. Two-and four-year-olds learn to adapt referring expressions to context: effects of distracters and feedback on referential communication. *Topics in cognitive science*, 4(2):184–210, 2012.

Cynthia Matuszek, Dieter Fox, and Karl Koscher. Following directions using statistical machine translation. In *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 251–258. IEEE, 2010.

Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. A joint model of language and perception for grounded attribute learning. In *Proceedings of the International Conference of Machine Learning*, 2012a.

Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. A joint model of language and perception for grounded attribute learning. In *Proceedings of the International Conference of Machine Learning*, 2012b.

Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. Learning to parse natural language commands to a robot control system. In *Experimental Robotics*, pages 403–415. Springer, 2013.

Sahisnu Mazumder and Oriana Riva. Flin: A flexible natural language interface for web navigation. *arXiv preprint arXiv:2010.12844*, 2020.

Sahisnu Mazumder and Oriana Riva. FLIN: A flexible natural language interface for web navigation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2777–2788, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.222.

Tom McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1334.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, 54(6): 1–35, 2021.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2016.

Rakesh R Menon, Sayan Ghosh, and Shashank Srivastava. Clues: A benchmark for learning classifiers using natural language explanations. *arXiv preprint arXiv:2204.07142*, 2022.

Rada Mihalcea, Hugo Liu, and Henry Lieberman. Nlp (natural language processing) for nlp (natural language programming). In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 319–330. Springer, 2006.

George A Miller and Philip N Johnson-Laird. *Language and perception.* Belknap Press, 1976.

Azalia Mirhoseini, Hieu Pham, Quoc V Le, Benoit Steiner, Rasmus Larsen, Yuefeng Zhou, Naveen Kumar, Mohammad Norouzi, Samy Bengio, and Jeff Dean. Device placement optimization with reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2017.

Shachar Mirkin, Scott Nowson, Caroline Brun, and Julien Perez. Motivating personality-aware machine translation. In *The 2015 Conference on Empirical Methods on Natural Language Processing (EMNLP)*, 2015.

Sobhan Miryoosefi, Kianté Brantley, Hal Daume III, Miro Dudik, and Robert E Schapire. Reinforcement learning with convex constraints. In *Proceedings of Advances in Neural Information Processing Systems*, 2019.

Dipendra Misra, John Langford, and Yoav Artzi. Mapping instructions and visual observations to actions with reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2017a.

Dipendra Misra, John Langford, and Yoav Artzi. Mapping instructions and visual observations to actions with reinforcement learning. *Proceedings of Emperical Methods in Natural Language Processing*, 2017b.

Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. Mapping instructions to actions in 3D environments with visual goal prediction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2667–2678, Brussels, Belgium, October-November 2018a. Association for Computational Linguistics. doi: 10.18653/v1/D18-1287. URL https://www.aclweb.org/anthology/D18-1287.

Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. Mapping instructions to actions in 3d environments with visual goal prediction. In *Proceedings of Emperical Methods in Natural Language Processing*, pages 2667–2678. Association for Computational Linguistics, 2018b. URL http://aclweb.org/anthology/D18-1287.

Dipendra Kumar Misra, Jaeyong Sung, Kevin Lee, and Ashutosh Saxena. Tell Me Dave: Context-sensitive grounding of natural language to mobile manipulation instructions. In *Robotics: Science and Systems (RSS)*, 2014.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*, 2013.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2016.

W. Monroe and C. Potts. Learning in the Rational Speech Acts model. In *Proceedings of 20th Amsterdam Colloquium*, 2015.

Cecilia G Morales, Elizabeth J Carter, Xiang Zhi Tan, and Aaron Steinfeld. Interaction needs and opportunities for failing robots. In *Proceedings of the 2019 on Designing Interactive Systems Conference*, pages 659–670, 2019.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. Generating natural questions about an image. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1802–1813, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1170. URL https://www.aclweb.org/anthology/P16-1170.

Peter Mundy, Lisa Sullivan, and Ann M Mastergeorge. A parallel and distributed-processing model of joint attention, social cognition and autism. *Autism research*, 2(1):2–21, 2009.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. Grounding language for transfer in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 63:849–874, 2018.

Anjali Narayan-Chen, Prashant Jayannavar, and Julia Hockenmaier. Collaborative dialogue in Minecraft. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5405–5415, Florence, Italy, July 2019a. Association for Computational Linguistics. doi: 10.18653/v1/P19-1537. URL https://www.aclweb.org/anthology/P19-1537.

Anjali Narayan-Chen, Prashant Jayannavar, and Julia Hockenmaier. Collaborative dialogue in minecraft. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5405–5415, 2019b.

Gergely Neu and Csaba Szepesvári. Apprenticeship learning using inverse reinforcement learning and gradient methods. *arXiv preprint arXiv:1206.5264*, 2012.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287, 1999.

Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

Khanh Nguyen and Hal Daumé III. Help, anna! visual navigation with natural multimodal assistance via retrospective curiosity-encouraging imitation learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, November 2019.

Khanh Nguyen, Hal Daumé III, and Jordan Boyd-Graber. Reinforcement learning for bandit neural machine translation with simulated human feedback. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1464–1474, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1153.

Khanh Nguyen, Debadeepta Dey, Chris Brockett, and Bill Dolan. Vision-based navigation with language-based assistance via imitation learning with indirect intervention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12527–12537, 2019a.

Khanh Nguyen, Debadeepta Dey, Chris Brockett, and Bill Dolan. Vision-based navigation with language-based assistance via imitation learning with indirect intervention. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019b. URL https://arxiv.org/abs/1812.04155.

Khanh Nguyen, Dipendra Misra, Robert Schapire, Miro Dudík, and Patrick Shafto. Interactive learning from activity description. In *Proceedings of the 38th International Conference on Machine Learning*, 2021. URL https://arxiv.org/pdf/2102.07024.pdf.

Khanh Nguyen, Yonatan Bisk, and Hal Daumé III. A framework for learning to request rich and contextually useful information from humans. In *Proceedings of the 39-th International Conference on Machine Learning (ICML)*, July 2022.

Randal S Olson, William La Cava, Patryk Orzechowski, Ryan J Urbanowicz, and Jason H Moore. Pmlb: a large benchmark suite for machine learning evaluation and comparison. *BioData mining*, 10(1):1–13, 2017.

Shayegan Omidshafiei, Dong-Ki Kim, Miao Liu, Gerald Tesauro, Matthew Riemer, Christopher Amato, Murray Campbell, and Jonathan P How. Learning to teach in cooperative multiagent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6128–6136, 2019.

Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J. Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018. ISSN 1935-8253. doi: 10.1561/2300000053.

Charles Packer, Pieter Abbeel, and Joseph E Gonzalez. Hindsight task relabelling: Experience replay for sparse reward meta-rl. *Advances in Neural Information Processing Systems*, 34, 2021.

Aishwarya Padmakumar, Jesse Thomason, Ayush Shrivastava, Patrick Lange, Anjali Narayan-Chen, Spandana Gella, Robinson Piramuthu, Gokhan Tur, and Dilek Hakkani-Tur. Teach: Task-driven embodied agents that chat. *arXiv preprint arXiv:2110.00534*, 2021.

Aishwarya Padmakumar, Jesse Thomason, Ayush Shrivastava, Patrick Lange, Anjali Narayan-Chen, Spandana Gella, Robinson Piramuthu, Gokhan Tur, and Dilek Hakkani-Tur. Teach: Task-driven embodied agents that chat. In *Association for the Advancement of Artificial Intelligence*, 2022.

Vishakh Padmakumar and He He. Machine-in-the-loop rewriting for creative image captioning. *arXiv preprint arXiv:2111.04193*, 2021.

John F Pane, Brad A Myers, et al. Studying the language and structure in non-programmers' solutions to programming problems. *International Journal of Human-Computer Studies*, 54 (2):237–264, 2001.

Rolla E Park. Estimation with heteroscedastic error terms. *Econometrica*, 34(4):888, 1966.

Sang-Min Park and Young-Gab Kim. Visual language navigation: a survey and open challenges. *Artificial Intelligence Review*, pages 1–63, 2022.

Rebecca J Passonneau and Bob Carpenter. The benefits of a model of annotation. *Transactions of the Association for Computational Linguistics (TACL)*, 2:311–326, 2014.

Bei Peng, James MacGlashan, Robert Loftin, Michael L Littman, David L Roberts, and Matthew E Taylor. A need for speed: Adapting agent action speed to improve task learning from non-expert humans. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, 2016.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202.

Patrick M Pilarski, Michael R Dawson, Thomas Degris, Farbod Fahimi, Jason P Carey, and Richard S Sutton. Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning. In *IEEE International Conference on Rehabilitation Robotics (ICORR)*, 2011.

Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

Pragaash Ponnusamy, Alireza Roshan Ghias, Chenlei Guo, and Ruhi Sarikaya. Feedback-based self-learning in large-scale conversational ai agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13180–13187, 2020.

David Premack and Guy Woodruff. Does the chimpanzee have a theory of mind? *Behavioral and brain sciences*, 1(4):515–526, 1978.

Carolyn C Preston and Andrew M Colman. Optimal number of response categories in rating scales: reliability, validity, discriminating power, and respondent preferences. *Acta Psychologica*, 104(1):1–15, 2000. ISSN 0001-6918.

Kathy Pretz. Michael i. jordan explains why today's artificial-intelligence systems aren't actually intelligent. https://spectrum.ieee.org/stop-calling-everything-ai-machinelearning-pioneer-says, 2021.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Di Qi, Lin Su, Jia Song, Edward Cui, Taroon Bharti, and Arun Sacheti. Imagebert: Cross-modal pre-training with large-scale weak-supervised image-text data. *arXiv preprint arXiv:2001.07966*, 2020a.

Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. Reverie: Remote embodied visual referring expression in real indoor environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9982–9991, 2020b.

Hongjin Qian, Xiaohe Li, Hanxun Zhong, Yu Guo, Yueyuan Ma, Yutao Zhu, Zhanliang Liu, Zhicheng Dou, and Ji-Rong Wen. Pchatbot: A large-scale dataset for personalized chatbot. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2470–2477, 2021.

Ella Rabinovich, Shachar Mirkin, Raj Nath Patel, Lucia Specia, and Shuly Wintner. Personalized machine translation: Preserving original author traits. *Association for Computational Linguistics (ACL)*, 2017.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

Wasifur Rahman, Md Kamrul Hasan, Sangwu Lee, AmirAli Bagher Zadeh, Chengfeng Mao, Louis-Philippe Morency, and Ehsan Hoque. Integrating multimodal information in large pretrained transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2359–2369, 2020.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4932–4942, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1487.

Nived Rajaraman, Lin F Yang, Jiantao Jiao, and Kannan Ramachandran. Toward the fundamental limits of imitation learning. In *Proceedings of Advances in Neural Information Processing Systems*, 2020.

Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, et al. Conversational ai: The science behind the alexa prize. *arXiv preprint arXiv:1801.03604*, 2018.

Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *IJCAI*, volume 7, pages 2586–2591, 2007.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831. PMLR, 18–24 Jul 2021.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *International Conference on Learning Representations (ICLR)*, 2016.

Sudha Rao and Hal Daumé III. Learning to ask good questions: Ranking clarification questions using neural expected value of perfect information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2737–2746, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1255. URL https://www.aclweb.org/anthology/P18-1255.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8689–8696, 2020.

Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736, 2006.

Mark O Riedl. Human-centered artificial intelligence and machine learning. *Human Behavior and Emerging Technologies*, 1(1):33–36, 2019.

Homero Roman Roman, Yonatan Bisk, Jesse Thomason, Asli Celikyilmaz, and Jianfeng Gao. RMM: A recursive mental model for dialogue navigation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1732–1745, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.157.

Stephanie Rosenthal and Manuela M Veloso. Mobile robot planning to seek help with spatially-situated tasks. In *AAAI*, volume 4, page 1, 2012.

Stephanie Rosenthal, Joydeep Biswas, and Manuela M Veloso. An effective personal mobile robot agent through symbiotic human-robot interaction. In *AAMAS*, volume 10, pages 915–922, 2010.

S. Ross and D. Bagnell. Efficient reductions for imitation learning. In *Artificial Intelligence and Statistics (AISTATS)*, pages 661–668, 2010.

Stephane Ross. *Interactive Learning for Sequential Decisions and Predictions*. PhD thesis, Carnegie Mellon University, 6 2013.

Stephane Ross and J Andrew Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*, 2014.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Artificial Intelligence and Statistics (AISTATS)*, 2011.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740, 2020.

Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller. *Explainable AI: interpreting, explaining and visualizing deep learning*, volume 11700. Springer Nature, 2019.

Conchi San Martín, Ignacio Montero, M Isabel Navarro, and Barbara Biglia. The development of referential communication: Improving message accuracy by coordinating private speech with peer questioning. *Early Childhood Research Quarterly*, 29(1):76–84, 2014.

Manolis Savva, Angel X Chang, Alexey Dosovitskiy, Thomas Funkhouser, and Vladlen Koltun. Minos: Multimodal indoor simulator for navigation in complex environments. *arXiv preprint arXiv:1712.03931*, 2017.

Juergen Schmidhuber. Reinforcement learning upside down: Don't predict rewards–just map them to actions. *arXiv preprint arXiv:1912.02875*, 2019.

Thom Scott-Phillips. *Speaking our minds: Why human communication is different, and how language evolved to make it special*. Macmillan International Higher Education, 2014.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1009.

Rohin Shah, Noah Gundotra, Pieter Abbeel, and Anca Dragan. On the feasibility of learning, rather than assuming, human biases for reward inference. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5670–5679. PMLR, 09–15 Jun 2019.

Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*, pages 621–635. Springer, 2018.

Kelly K. Shang and Rachel R. Du. *Disciplining Artificial Intelligence Policies: World Trade Organization Law as a Sword and a Shield*, page 274–292. doi: 10.1017/9781108954006.015.

Amr Sharaf and Hal Daumé III. Structured prediction via learning to search under bandit feedback. In *Proceedings of the 2nd Workshop on Structured Prediction for Natural Language Processing*, pages 17–26, 2017.

Jing Shi, Ning Xu, Yihang Xu, Trung Bui, Franck Dernoncourt, and Chenliang Xu. Learning by planning: Language-guided global image editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13590–13599, June 2021.

Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning*, pages 3135–3144, 2017.

Zhengxiang Shi, Yue Feng, and Aldo Lipani. Learning to execute or ask clarification questions. *arXiv preprint arXiv:2204.08373*, 2022.

Nobuyuki Shimizu and Andrew Haas. Learning to follow navigational route instructions. In *Twenty-First International Joint Conference on Artificial Intelligence*. Citeseer, 2009.

Masahiro Shiomi, Daisuke Sakamoto, Takayuki Kanda, Carlos Toshinori Ishi, Hiroshi Ishiguro, and Norihiro Hagita. A semi-autonomous communication robot—a field trial at a train station. In *2008 3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 303–310. IEEE, 2008.

Ben Shneiderman. Human-centered artificial intelligence: Reliable, safe & trustworthy. *International Journal of Human–Computer Interaction*, 36(6):495–504, 2020.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749, 2020.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587): 484–489, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*, 2022.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2008.

Artem Sokolov, Stefan Riezler, and Tanguy Urvoy. Bandit structured prediction for learning from partial feedback in statistical machine translation. In *In Proceedings of MT Summit XV. Miami, FL*, 2015.

Artem Sokolov, Julia Kreutzer, Christopher Lo, and Stefan Riezler. Learning structured predictors from bandit feedback for interactive NLP. In *Association for Computational Linguistics (ACL)*, 2016a.

Artem Sokolov, Julia Kreutzer, and Stefan Riezler. Stochastic structured prediction under bandit feedback. In *Advances In Neural Information Processing Systems (NIPS)*, 2016b.

Dan Sperber and Deirdre Wilson. *Relevance: Communication and cognition*, volume 142. Citeseer, 1986.

Shashank Srivastava, Igor Labutov, and Tom Mitchell. Joint concept learning and semantic parsing from natural language explanations. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 1527–1536, 2017.

Bradly C Stadie, Pieter Abbeel, and Ilya Sutskever. Third-person imitation learning. *arXiv preprint arXiv:1703.01703*, 2017.

Robert Stalnaker. Common ground. *Linguistics and philosophy*, 25(5/6):701–721, 2002.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback. In *Proceedings of Advances in Neural Information Processing Systems*, 2020.

Tricia Striano, Daniel Stahl, and Allison Cleveland. Taking a closer look at social and cognitive skills: A weekly longitudinal assessment between 7 and 10 months of age. *European Journal of Developmental Psychology*, 6(5):567–591, 2009.

Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vl-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, 2019.

Yu Su, Ahmed Hassan Awadallah, Madian Khabsa, Patrick Pantel, Michael Gamon, and Mark Encarnacion. Building natural language interfaces to web apis. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 177–186, 2017.

Yu Su, Ahmed Hassan Awadallah, Miaosen Wang, and Ryen W White. Natural language interfaces with fine-grained user interaction: A case study on web apis. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 855–864, 2018.

Alane Suhr, Claudia Yan, Jack Schluger, Stanley Yu, Hadi Khader, Marwa Mouallem, Iris Zhang, and Yoav Artzi. Executing instructions in situated collaborative interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2119–2130, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1218.

Theodore R Sumers, Mark K Ho, Robert D Hawkins, Karthik Narasimhan, and Thomas L Griffiths. Learning rewards from linguistic feedback. *arXiv preprint arXiv:2009.14715*, 2020.

Chen Sun, Fabien Baradel, Kevin Murphy, and Cordelia Schmid. Learning video representations using contrastive bidirectional transformer. *arXiv preprint arXiv:1906.05743*, 2019a.

W. Sun, A. Venkatraman, G. J. Gordon, B. Boots, and J. A. Bagnell. Deeply aggrevated: Differentiable imitation learning for sequential prediction. In *International Conference on Machine Learning (ICML)*, 2017.

Wen Sun, Anirudh Vemula, Byron Boots, and J. Andrew Bagnell. Provably efficient imitation learning from observation alone. In *Proceedings of (ICML) International Conference on Machine Learning*, June 2019b.

R. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1999a.

R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Articial intelligence*, 112:181–211, 1999b.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Toshitaka N Suzuki. Semantic communication in birds: evidence from field research over the past two decades. *Ecological Research*, 31(3):307–319, 2016.

Umar Syed and Robert E Schapire. A game-theoretic approach to apprenticeship learning. *Advances in neural information processing systems*, 20, 2007.

Catherine S Tamis-LeMonda, Marc H Bornstein, and Lisa Baumwell. Maternal responsiveness and children's achievement of language milestones. *Child development*, 72(3):748–767, 2001.

Hao Tan and Mohit Bansal. Vokenization: Improving language understanding with contextualized, visual-grounded supervision. *arXiv preprint arXiv:2010.06775*, 2020.

Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2610–2621, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1268.

Ars Technica. Tay, the neo-nazi millennial chatbot, gets autopsied. https://arstechnica.com/information-technology/2016/03/tay-the-neo-nazi-millennial-chatbot-gets-autopsied/, 2016.

Stefanie Tellex, Pratiksha Thaker, Josh Joseph, and Nicholas Roy. Toward learning perceptually grounded word meanings from unaligned parallel data. In *Proceedings of the Second Workshop on Semantic Interpretation in an Actionable Context*, pages 7–14, 2012.

Stefanie Tellex, Ross Knepper, Adrian Li, Daniela Rus, and Nicholas Roy. Asking for help using inverse semantics. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014a. doi: 10.15607/RSS.2014.X.024.

Stefanie Tellex, Ross Knepper, Adrian Li, Daniela Rus, and Nicholas Roy. Asking for help using inverse semantics. 2014b.

Ana C Tenorio-Gonzalez, Eduardo F Morales, and Luis Villaseñor-Pineda. Dynamic reward shaping: training a robot by voice. In *Ibero-American Conference on Artificial Intelligence*, pages 483–492. Springer, 2010.

Jesse Thomason, Daniel Gordan, and Yonatan Bisk. Shifting the baseline: Single modality performance on visual navigation & qa. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2019a.

Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. Vision-and-dialog navigation. In *Proceedings of the Conference on Robot Learning*, 2019b.

Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. Vision-and-dialog navigation. In *Conference on Robot Learning*, pages 394–406. PMLR, 2020.

Andrea L Thomaz and Cynthia Breazeal. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence*, 172(6-7):716–737, 2008.

Andrea Lockerd Thomaz, Cynthia Breazeal, et al. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2006.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

Michael Tomasello. *Constructing a language: A usage-based theory of language acquisition*. Harvard university press, 2005.

Michael Tomasello, Malinda Carpenter, Josep Call, Tanya Behne, and Henrike Moll. Understanding and sharing intentions: The origins of cultural cognition. *Behavioral and brain sciences*, 28(5):675–691, 2005.

Lisa Torrey and Matthew Taylor. Teaching on a budget: Agents advising agents in reinforcement learning. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1053–1060, 2013.

Alan Turing. Computing machinery and intelligence. *Mind*, 59(236):433, 1950.

UNESCO. Draft text of the recommendation on the ethics of artificial intelligence. *Intergovernmental Meeting of Experts (Category II) related to a Draft Recommendation on the Ethics of Artificial Intelligence*, 2021.

Jack Urbanek, Angela Fan, Siddharth Karamcheti, Saachi Jain, Samuel Humeau, Emily Dinan, Tim Rocktäschel, Douwe Kiela, Arthur Szlam, and Jason Weston. Learning to speak and act in a fantasy text adventure game. *arXiv preprint arXiv:1903.03094*, 2019.

Mina Valizadeh and Natalie Parde. The ai doctor is in: A survey of task-oriented dialogue systems for healthcare applications. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6638–6660, 2022.

Andrea Vanzo, Francesco Riccio, Mahmoud Sharf, Valeria Mirabella, Tiziana Catarci, and Daniele Nardi. Who is willing to help robots? a user study on collaboration attitude. *International Journal of Social Robotics*, 12(2):589–598, 2020.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

Valeria Villani, Fabio Pini, Francesco Leali, and Cristian Secchi. Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. *Mechatronics*, 55: 248–266, 2018.

Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, Hong Kong, China, November 2019a. Association for Computational Linguistics. doi: 10.18653/v1/D19-1221.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing nlp. *arXiv preprint arXiv:1908.07125*, 2019b.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019a.

Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O Stanley. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753*, 2019b.

Sida I. Wang, Percy Liang, and Christopher D. Manning. Learning language games through interaction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2368–2378, Berlin, Germany, August 2016a. Association for Computational Linguistics. doi: 10.18653/v1/P16-1224. URL https://www.aclweb.org/anthology/P16-1224.

Sida I Wang, Percy Liang, and Christopher D Manning. Learning language games through interaction. *arXiv preprint arXiv:1606.02447*, 2016b.

Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019c.

Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6629–6638, 2019d.

Xin Wang, Jiawei Wu, Junkun Chen, Lei Li, Yuan-Fang Wang, and William Yang Wang. Vatex: A large-scale, high-quality multilingual dataset for video-and-language research. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4581–4591, 2019e.

Zijie J Wang, Dongjin Choi, Shenyu Xu, and Diyi Yang. Putting humans in the natural language processing loop: A survey. *arXiv preprint arXiv:2103.04044*, 2021.

Wei Wei, Quoc Le, Andrew Dai, and Jia Li. Airdialogue: An environment for goal-oriented dialogue research. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3844–3854, 2018.

Astrid Weiss, Judith Igelsböck, Manfred Tscheligi, Andrea Bauer, Kolja Kühnlenz, Dirk Wollherr, and Martin Buss. Robots asking for directions—the willingness of passers-by to support robots. In *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 23–30. IEEE, 2010.

Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.

Jason E Weston. Dialog-based language learning. *Advances in Neural Information Processing Systems*, 29, 2016.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8, 1992a.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992b.

Heinz Wimmer and Josef Perner. Beliefs about beliefs: Representation and constraining function of wrong beliefs in young children's understanding of deception. *Cognition*, 13(1):103–128, 1983. ISSN 0010-0277. doi: https://doi.org/10.1016/0010-0277(83)90004-5.

Terry Winograd. Procedures as a representation for data in a computer program for understanding natural language. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC, 1971.

Terry Winograd. Understanding natural language. *Cognitive Psychology*, 3(1):1–191, 1972.

Ludwig Wittgenstein. Philosophical investigations (orig. 1953), 1995.

Wansen Wu, Tao Chang, and Xinmeng Li. Visual-and-language navigation: A survey and taxonomy. *IEEE transactions on neural networks and learning systems*, 2021.

Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3d environment. In *Workshop Track of the International Conference on Learning Representation*, 2018.

Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: real-world perception for embodied agents. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018.

Wei Xu. Toward human-centered ai: a perspective from human-computer interaction. *interactions*, 26(4):42–46, 2019.

Tsung-Yen Yang, Michael Hu, Yinlam Chow, Peter J Ramadge, and Karthik Narasimhan. Safe reinforcement learning with natural language constraints. *Advances in Neural Information Processing Systems*, 34, 2021.

Ziyu Yao, Yiqi Tang, Wen-tau Yih, Huan Sun, and Yu Su. An imitation game for learning semantic parsers from user interaction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6883–6902, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.559.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Haonan Yu, Xiaochen Lian, Haichao Zhang, and Wei Xu. Guided feature transformation (gft): A neural language grounding module for embodied agents. In *Conference on Robot Learning*, pages 81–98. PMLR, 2018.

Fabio Massimo Zanzotto. Human-in-the-loop artificial intelligence. *Journal of Artificial Intelligence Research*, 64:243–252, 2019.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

Chen Zhao, Yu Su, Adam Pauls, and Emmanouil Antonios Platanios. Bridging the generalization gap in text-to-SQL parsing with schema expansion. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5568–5578, Dublin, Ireland, May 2022. Association for Computational Linguistics.

Li Zhou and Kevin Small. Inverse reinforcement learning with natural language goals. In *AAAI*, 2021a.

Li Zhou and Kevin Small. Inverse reinforcement learning with natural language goals. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11116–11124, 2021b.

Li Zhou, Jianfeng Gao, Di Li, and Heung-Yeung Shum. The design and implementation of xiaoice, an empathetic social chatbot. *Computational Linguistics*, 46(1):53–93, 2020a.

Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason J Corso, and Jianfeng Gao. Unified vision-language pre-training for image captioning and vqa. In *AAAI*, pages 13041–13049, 2020b.

Hao Zhu, Graham Neubig, and Yonatan Bisk. Few-shot language coordination by modeling theory of mind. In *International Conference on Machine Learning*, pages 12901–12911. PMLR, 2021.

Xiaojin Zhu, Andrew B Goldberg, Mohamed Eldawy, Charles R Dyer, and Bradley Strock. A text-to-picture synthesis system for augmenting communication. In *AAAI*, volume 7, pages 1590–1595, 2007.

B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2008.

Matthieu Zimmer, Paolo Viappiani, and Paul Weng. Teacher-student framework: a reinforcement learning approach. In *AAMAS Workshop Autonomous Robots and Multirobot Systems*, 2014.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2017.