Characteristics of the Learning Problem in Situated Interactive Task Learning

John E. Laird, Shiwali Mohan, James Kirk, and Aaron Mininger

Abstract

In most learning problems, a single type of task knowledge is learned using a single specialized learning algorithm designed and optimized for that specific type of knowledge and the environment in which it is learned. In contrast, interactive task learning (ITL) involves learning *all* types of task knowledge where such specialization is impossible. This chapter describes these characteristics of the ITL learning problem, which distinguish it from other learning problems, and examines how those characteristics influence the underlying learning algorithms. Throughout our discussion, the Rosie agent is used as an example of an ITL agent that can learn many tasks in a variety of domains. The distinguishing characteristics explored include learning across different domains, learning diverse task knowledge, interactivity in learning, the situated aspects of learning, and how an ITL agent can exploit multiple data sources. Learning approaches are then discussed that can be used in ITL from the perspective of how they address the unique challenges of ITL.

Introduction

Learning research in artificial intelligence (AI) and cognitive science usually focuses on a singular learning problem for a single task where there is a single target type of knowledge to be learned. Much of the current work in statistical machine learning, for example, can be formulated as a supervised learning problem, where the goal is to learn a classification of a set of prespecified data from labeled examples. Outside of research on cognitive architecture and reinforcement learning, learning is rarely coupled to task performance: The learning problem is often studied in isolation, with a focus on a single learning algorithm, and in a noninteractive fashion, where the learner has little control over what training it receives.

From "Interactive Task Learning: Humans, Robots, and Agents Acquiring New Tasks through Natural Interactions," edited by K. A. Gluck and J. E. Laird. Strüngmann Forum Reports, vol. 26, J. R. Lupp, series editor. Cambridge, MA: MIT Press. ISBN 978-0-262-03882-9.

Our interest is not in learning within a single task, or even a class of tasks, but actually learning new tasks from scratch through interactive task learning (ITL) (Laird et al. 2017a). Mitchell et al. (this volume) define ITL as "any process by which an agent (A) improves its performance (P) on some task (T) through experience (E), when E consists of a series of sensing, effecting, and communicating interactions between A, its world, and crucially other agents in the world." ITL stands in stark contrast to approaches that are specialized to a single task. Not only does it utilize a collection of learning algorithms for a variety of learning problems, ITL can occur in many *different domains,* whose very characteristics directly influence the appropriateness of different learning algorithms. ITL also involves learning the complete formulation of a task; this entails learning *multiple types of knowledge*, each of which can depend on the specific task formulation. As defined above, ITL involves an agent learning in real-time through an *interaction* with an instructor. Although there are cases where an instructor can interactively teach an agent about hypothetical situations, our focus in this chapter is on cases where the instructor and agent are *situated* in a shared environment. Because of the flexibility of how an instructor can interact with a learning agent (e.g., a person can demonstrate or give direct instructions) as well as an agent's prior knowledge, *multiple sources of knowledge* can be available during the learning of a task. Some of these characteristics can simplify learning while others may complicate it, making the development of learning algorithms for situated ITL a challenging, but exciting research problem.

Here, we explore the characteristics of situated ITL that distinguish it from other learning problems and discuss how those characteristics affect the agent's learning. We consider each characteristic, independently identifying the impact it has on the ITL problem as defined by Mitchell et al. (this volume). We then consider these characteristics together and discuss potential approaches to learning. When appropriate, we provide examples from our own research in creating an ITL agent named Rosie (Mohan et al. 2012), which was built using the Soar cognitive architecture (Laird 2012). Before diving into the defining characteristics of situated ITL, we briefly introduce Rosie.

Rosie: An Interactive Task Learning Agent

Rosie (RObotic Soar Instructable Entity) is a physically embodied agent that learns new tasks through situated interactive instruction using restricted natural language and limited demonstration. It can learn over forty simple games and puzzles (such as Tic-Tac-Toe or Tower of Hanoi), process-oriented tasks (such as cooking or storing), and navigation tasks (such as delivering or fetching an object). It not only learns various aspects of the tasks (e.g., goals, valid actions, failure states, and policies) but also learns new concepts used in the tasks (e.g., relationships, labels, and functions). Additionally, it learns how to refer to these components of tasks by associating them with words used to describe them. Whenever Rosie encounters a new word, it initiates a new interaction to learn the meaning of the word. It can learn new nouns (names and shapes), adjectives (colors and sizes), prepositions (spatial relations), comparators (quantities), functions (task-specific value calculations), and verbs (actions and tasks). After Rosie learns a new word, the human can use it freely in future instructions ("move the *green* block to the pantry") and the agent can use it when responding to questions ("that is a large *green* block").

Rosie is preprogrammed with only limited initial domain knowledge: action knowledge for performing its primitive actions (e.g., picking up a block, putting down a block, moving down a hallway), feature-space knowledge of object attributes (e.g., the fact that objects have a color, a size, and a shape), and primitive spatial relations about the alignment of objects. As Rosie learns, it creates new structures in its long-term semantic memory that encode the task definition. It interprets this knowledge to perform that task, leading it to learn efficient procedural knowledge (encoded in rules in Soar).

Following the acquisition of the task, the agent uses its innate problem-solving and planning capabilities to play the games or perform the tasks by interacting with its world. Depending on the task, it can also learn policy knowledge to perform the task well. Policy knowledge is used to select actions in each state and can be learned from the agent's own experience, from the specification of a procedure, or from additional advice it receives through instruction (such as heuristics for a game). The knowledge it learns in one task transfers to shared concepts, actions, and constraints in new tasks.

Learning across Different Domains

There is a wide range of domains to which ITL can be applied, including robotics (Mininger and Laird 2018), personal assistants (Azaria et al. 2016), intelligent tutoring authoring (Li et al. 2015), game playing (Hinrichs and Forbus 2013a), constructive agents and virtual humans (Pew and Mavor 1998; Zacharias et al. 2008), and cognitive science research. These domains vary wildly in complexity and characteristics, and these characteristics can significantly impact what needs to be learned as well as the difficulty of the learning problem. This, in turn, informs the appropriateness of different learning algorithms. Some of these characteristics include whether the environment is partially or fully observable, whether the environment is discrete or continuous, whether actions are deterministic or stochastic, whether the environment has complex dynamics, and the extent of uncertainty in sensing and acting in the environment. In general, the more complex and uncertain a domain is, the more difficult it is for an agent to learn new tasks quickly. Current ITL agents usually focus on tasks within a single domain, and designers tailor their learning algorithms to that domain. Whether there are fundamental representations and learning algorithms that will work across all domains remains an open research question, as is the degree to which certain domains require specialized learning approaches. Below we provide examples of two different domains and how their characteristics affect the learning problem.

Virtual Personal Assistants

In the virtual personal assistant domain, an agent performs various tasks that aid in a person's daily activities, such as answering questions, managing a schedule, or handling messages. For these tasks, the agent's actions usually involve interacting with people and software interfaces. There is low uncertainty in its perceptions and actions, and often a discrete, symbolic representation can be used effectively. This can make the learning problem easier and allow more aggressive and data-poor learning mechanisms to be used. The agent can learn in large steps and extract higher-quality information from each example. An example of such an agent is PLOW (Allen et al. 2007), which operates in a web browser and can learn how to answer new types of questions through interaction. PLOW leverages the low uncertainty and direct access to the browser's state to learn new tasks through a single training example, by mapping the linguistic instructions onto the procedural trace of actions performed in the browser. A second example is the LIA system (Azaria et al. 2016), which learns new action sequences for apps on mobile devices using restricted natural language.

Real-World Robotic Domains

A major focus of ITL is in the highly complex domain of real-world robotics. Such environments are often partially observable, involve high-dimensional and continuous data, and contain high uncertainty in both perception and action. It is difficult for an agent to build an accurate and precise model of its environment as well as to predict the results of its own actions. As a result, high levels of uncertainty can force an agent to be conservative in its reasoning and its learning, requiring many examples before any conclusions can be drawn. An incremental approach may be required whereby the agent learns simpler/smaller aspects of the task before learning more complex structures. Unlike the virtual assistant domain, the agent may need to learn not just what an action does when it succeeds, but also contingency plans if an action fails.

In our ITL agent, Rosie, we use the same symbolic state representation formalism across three different robotic embodiments, which allows us to use the same learning algorithms. However, this requires perception and action systems that are sufficiently accurate and precise, so that Rosie does not explicitly need to represent or reason about perceptual or motor uncertainty. In addition, we constrain the environment and the actions so that Rosie has a relatively stable view of the world and can use its memory to pursue tasks effectively. An underlying hypothesis is that these characteristics will be true for other domains in which Rosie will learn new tasks. An interesting question is whether other constraints that arise from the characteristics of ITL, such as the need to learn very quickly, will force ITL systems to make similar assumptions about their perceptual and motor systems as well as the stability of their environments.

For partially observable domains, the agent must use, and possibly learn, strategies for maintaining internal representations of the environment and gathering information. As an example, Rosie can learn some simple strategies for finding objects in the mobile robot domain, such as representing where a type of object is stored and then going there to look for one (Mininger and Laird 2016). Agents in a real-world robotic domain might also need to know, and possibly learn, how to handle complex physical dynamics, in contrast to the above virtual assistants.

Learning new robotic tasks that can generalize to novel situations in a few interactive examples is a difficult problem. Often the environment is constrained to make this feasible. For example, in the task learning mobile robot by Meriçli et al. (2014), constraining actions to driving and speech and using visual fiducial systems to recognize landmarks enabled the agents to learn procedural task specifications in a single example. In further work by Chao et al. (2011), agents learn concepts in task goals (such as object labels and spatial relations) from low-level sensor data. This informs Chao et al.'s choice of using unsupervised learning methods and Bayesian inference and does require more examples (but still not hundreds or thousands).

Learning Diverse Task Knowledge

An important feature of ITL is that it does not involve learning just one type of knowledge for which a single learning algorithm can be deployed and optimized. Tasks can be formulated in different ways, and the choice of task formulation impacts which types of knowledge need to be learned. Thus, an ITL agent must be capable of learning diverse types of knowledge. Below we distinguish among three different formulations of tasks: problem space, procedure, and optimization (although a given task can include aspects of any of these) and the types of knowledge each requires. This analysis is independent of the underlying knowledge representations, such as production rules, neural networks, semantic networks, or partially observable Markov decision processes. Instead, our analysis focuses on the types of knowledge required to learn a task, regardless of how it is represented.

Knowledge Diversity in Problem Space Formulations

To determine what type of knowledge needs to be learned for a task, we start with Newell's Problem Space Hypothesis, which states: "The fundamental organizational unit of all human goal-oriented symbolic activity is the problem space" (Newell 1991:696). Newell informally defines problem space and problem as follows:

> Problem Space: A problem space consists of a set of symbolic structures (the states of the space) and a set of operators over the space. Each operator takes a state as input and produces a state as output, although there may be other inputs and outputs as well. The operators may be partial, i.e., not defined for all states. Sequences of operators define paths that thread their way through sequences of states.
>
> Problem: A problem in a problem space consists of a set of initial states, a set of goal states, and a set of path constraints. The problem is to find a path through the space that starts at any initial state, passes only along paths that satisfy the path constraints, and ends at any goal state.

The strength of problem spaces is in formulating tasks where an agent learns a goal and must search through the space of states to achieve it. Classic examples from AI are many types of reasoning tasks as well as assembly tasks, puzzles, and games. For these tasks, the problem space formulation provides us with a starting point to discuss the types of knowledge that should be learned by an ITL agent: descriptions of initial states, goal states to be achieved, illegal states, constraints on sequences of operators, and operators that transform states. The assumption is that once the agent acquires this "first principle" knowledge, it can attempt a problem by searching for a solution (either internally or externally) through selecting and applying operators until it achieves the goal. Rather than describing a specific procedure or generalized policy, the teacher communicates the component task concepts. Once an ITL agent has learned the problem space, it knows enough to attempt and possibly solve any problem (as defined by an initial state and a set of goal states) in that problem space. This formulation is especially useful when the instructor does not know how to solve the task, or when it is difficult or time consuming to communicate the solution.

Although there are many different concepts that define a task, most of them involve state descriptions that are associated with some aspect of the formulation of a task, such as goal completion or operator preconditions. These state descriptions usually involve conjunctions of conditions over objects and values in the world, such as unary features of objects or relations between objects. Depending on the domains and tasks, these features can be combinations of continuous and discrete values. Within AI and cognitive science, there are many different ways to represent this type of information, and there are many different learning methods for learning these representations, although tasks often require rich representations beyond simple feature matching. For example, for Rosie, the breadth of games and puzzles it currently learns (over 30, including games such as *Othello* as well as *Missionaries and Cannibals*) requires concepts that include conjunctions of unary features (*red, large, square*), numeric functions (*number of, sum*), comparators (*less than, more than*), words, numbers, and spatial relations (*left of*, *below*, *linear*) (Kirk and Laird 2016). In terms of learning algorithms, constraints (and assumptions) that arise from different characteristics of ITL usually limit them to be online, incremental approaches to concept learning. Statistical techniques that require large numbers of examples can be useful for learning perceptual classifiers, such as for colors and shapes, which the agent would learn before task learning is attempted.

In addition to learning associations between state descriptions and goals, failure states, and operators, task learning can also involve learning hierarchical goal structures, where the operator at one level becomes a new task to solve at a lower level. Lower-level tasks can be any of the types of tasks described in this section: performing a new action or even simpler types of tasks, such as classification or categorization. For example, in learning an assembly task, Rosie may be instructed using a new verb, such as in "*store* the box in the pantry." This will lead to an interaction in which Rosie learns the "store" task (Mohan and Laird 2014). In terms of learning new classifications, Rosie might not know the word "red" when instructed to "pick up the *red* block." In this case, Rosie will engage the instructor to show it examples of red objects, ultimately learning a classifier in the color spectrum that Rosie can use to identify red objects.

Another type of knowledge that can be valuable for a problem space task is heuristic or policy knowledge. This type of knowledge does not define the task, but can be critical for helping the agent solve the task.

Knowledge Diversity in Procedure Formulations

Although the problem space formulation works well for many tasks, in many others it is more efficient for the agent to formulate the task as learning a procedure. For this, however, the agent might not have an explicit description of the goal or be capable of achieving the goal on its own without the specified procedure.

Cooking is a good example of a task where procedure formulations are used. Consider trying to bake a key lime four-layer cake. Few of us are able to search through the "baking" problem space and determine all of the steps required to make such a cake, in part because we do not have an operational definition of what the final cake should actually be. Furthermore, the branching factor for baking is immense, and most of us do not have good models for what many of the baking actions actually do at a chemical level. However, most of us can follow a recipe and learn to bake a specific type of cake (or at least learn to follow the recipe on the back of a box). For these types of tasks, the agent might not need to learn detailed descriptions of goals or failure states, because it can assume that the goal is achieved if the correct procedure is performed. Furthermore, the agent might already know the actions that must be performed (e.g., how to break eggs and mix them with a specified amount of sugar). However, the agent does not know when to do each action or which ingredients to use. Instead, the agent is learning a task that is associated with a particular sequence of actions that needs to be performed to accomplish the task. Thus, in a procedure formulation, instead of having to learn descriptions of goals and failure states, the agent needs to learn a procedure. The exact form of that procedure can vary. It can be a linear set of instructions, as in a recipe. It can also be similar to a program, with conditionals, while loops, etc. Alternatively, it can be the value function for a policy, which consists of mappings from states to actions. Those mappings can be probabilistic functions, as is learned in reinforcement learning, or deterministic, as in finite-state machines and rule-based systems.

Tasks formulated as procedures often require learning how to perform complex actions that are specified in the procedure, such as "fold together beaten egg whites with sugar." Learning such complex actions can be formulated as either a problem space or procedure, so that sometimes a goal and appropriate actions may be specified, or sometimes another procedure. Furthermore, a given task may involve knowledge from both formulations, where the instructor provides the goal and actions as well as helps the agent find the solution by giving hints and directions while the agent is attempting to perform the task.

One disadvantage of using a procedure formulation that does not include a problem space formulation is that the agent is bound to the procedure. If there are even small variations to the original task, it can be problematic for the agent to apply the learned procedure, and it can be difficult to recover from interruptions or mistakes because the agent does not know why it is performing each action or whether alternative actions will still achieve the goal.

Knowledge Diversity in Optimization Formulations

The final task formulation type we consider is where the goal is to maximize or minimize some quantity. The traveling salesman problem (TSP) is a classic example, where the goal is to minimize the total distance traveled while visiting every city (node) in a map (graph). In this example, there is both a goal (visit every city) and a value to be optimized (the total distance traveled). Optimization problems can include all knowledge of the problem space formulation, but add the optimization function, which can be specified as a function over the state or, as in TSP, as a function over the path.

Although formulating the TSP and similar tasks as optimization problems might seem like the obvious thing to do, there can be a difficulty. If the agent takes the optimization formulation "seriously," then finding an exact solution to the TSP is NP-hard. That might be necessary in some situations, but in practice, approximations are sufficient, and often the optimization function is used as an evaluation function to guide a greedy search for a solution. This leads to a situation where the task is formulated as a problem space, but with additional knowledge that must be used to guide the search.

Learning Is Interactive

With interactive learning, the human instructor provides information, piece by piece, as the agent needs it. This learning paradigm distributes the onus of learning to both the learner and the instructor. The learner agent can request the information it needs to perform a task as well as clarification and disambiguation of previous instructions. The human instructor can monitor the agent, interrupt it to provide instruction when it seems warranted, and verify the status of agent learning. Thus, with interactive learning, relevant information is not provided as a massive data set or a pile of written documents that the agent must process without help from an instructor. Instead, in interactive learning, the problem of learning a task is deconstructed into subproblems of learning components (e.g., goals, parameters, policies) that can be learned incrementally. Below we analyze important properties of interactive instruction and how it affects the learning of complex tasks.

Interaction Occurs in Real Time

One of the important advantages of using interaction is that the human instructor can watch the learner agent interpret and attempt to perform instructions. This provides real-time feedback to the instructor as to how the agent interprets and executes instructions, and permits the instructor to adapt the training strategy to the needs of the learner. Thus, it is critical for learning mechanisms to produce immediate, useable results that the agent can use to attempt the problem. Furthermore, applying instruction to the problem in real-time facilitates discovery of other related learning problems for which the agent can request training. To exploit these advantages, learning mechanisms must incrementally acquire new knowledge and assimilate it with existing long-term knowledge structures while the agent is behaving. It further implies that during learning, the agent does not need to reevaluate its previous experience or knowledge, two characteristics of batch learning mechanisms. The broader point is that the agent cannot require many examples of a concept before doing any learning. This is not to prohibit corrections or modifications to learned concepts as more data becomes available, just that some learning must occur immediately. It also implies that the learning mechanism is computationally fast in producing results that are quickly incorporated within the agent, so that the agent can use the knowledge immediately. It is hard to quantify exactly how "fast" the agent needs to be in terms of execution time, except that it cannot take so long to respond that it annoys the instructor or breaks the flow of the interaction. Rosie uses online and incremental learning mechanisms that are fast enough so there is no visible delay when it is processing instructions, which includes language understanding, interpretation of internal structures, and compiling of knowledge into high-performance rules.

Available Data Are Sparse and High Quality

Human time is costly, and therefore numerous and repetitive interactions with the agent are undesirable. Consequently, the agent must learn from *small* data—a few highly specific training instances. Although the learning data are sparse, it can be assumed that the instructor is well informed on the task and well intentioned in terms of helping the agent learn, and therefore generates high-quality, correct/accurate data for the agent. These characteristics push toward learning approaches where the agent must bring to bear its prior experiences and knowledge, commonsense reasoning, and causal deduction to learn maximally from few instructions. This is in contrast to alternatives such as noninteractive crowd-sourced learning, where the quality of instruction can be an issue, so that more conservative approaches to learning are required.

Rosie employs aggressive generalization mechanisms, such as explanation-based learning (DeJong and Mooney 1986) and version-space induction (Mitchell 1977), to learn procedural and semantic aspects of tasks. These mechanisms allow Rosie to produce actionable task representations even from very few examples. Currently, Rosie assumes that every instruction provided by the instructor is correct. This may not be true in partially observable, noisy environments or when the instructor is a novice. Ongoing research in our lab is looking at how learned knowledge can be corrected upon receiving feedback from the environment.

Structure of Instruction Is Flexible

In interactive learning, an instructor can be freed from using a specific ordering to teach the agent new words and concepts. Often in human-controlled learning (e.g., supervised learning), the instructor must attempt to build and maintain an internal model of what the agent knows and does not know to give it good examples. This is especially challenging when the agent is dynamically learning a variety of concepts (perceptual, spatial, task knowledge in Rosie) from real-world data. In contrast, with interactive learning, the instructor can rely on the agent to initiate an interaction when needed. This approach can speed instruction by eliminating the need for the instructor to carefully structure the interaction or repeatedly check with the agent to ensure it has completely learned a concept. The agent can actively seek examples of concepts that are hard to learn and avoid asking for multiple examples of easily acquired concepts. The instructor can take initiative in presenting interesting examples to the agent that it might have overlooked, refining the agent's learning.

Rosie implements an interaction model that does not impose a strict order on how it is taught new concepts. To support this, Rosie uses the content of the instructions to determine which type of concept is being taught. For example, "if …, then you win" will lead to a goal concept being learned. Additionally, it employs a relational, compositional task representation, which provides flexibility in how the instruction is structured and who takes the onus of learning. The instructor can choose to control how and when Rosie learns. The instructor can teach basic concepts before teaching complex concepts that require knowledge of the basic concepts. However, the instructor may not know or remember the state of Rosie's knowledge. In situations when the agent is learning complex concepts but lacks required knowledge of basic concepts, it will take the initiative and guide the interaction to acquire the relevant basic concepts first. For example, if the concept "empty" is used when teaching Sudoku but the agent does not yet know it, the agent asks the instructor to define the term so it can acquire the concept. Then the instructor can say: "If a square does not contain a value, then it is empty."

Learning Is Situated

Situated ITL occurs in the environment in which the learner agent is attempting to perform a task. Any demonstration will occur in that environment, and instructions will often directly refer to the current situation, although some instruction may refer to past or hypothetical situations. Here we discuss the ways in which a shared, situated environment impacts both the instructor and the learner.

Instructor and Agent Share the Same Environment

By having a shared environment, it is much easier for the participants to build up common ground (Clark 1996), in terms of a shared understanding of the objects, relations, and actions of the environments, and thus decrease the chance of ambiguity. Both participants can use deictic references and concrete descriptions as well as nonlinguistic actions, such as pointing. Within this shared environment, the instructor can focus the agent's attention on aspects of the situation that are relevant to the current problem. When difficulties arise, the agent can use language to describe its knowledge about the situation, its plans, and even reasons for performing actions that are easy for the instructor to understand and evaluate, thus avoiding the necessity of having to "open up" the agent to examine (and try to understand) internal data structures that may be quite distant from the human's representation of the task.

If the agent identifies an instruction as ambiguous, it can immediately ask for clarification. This can make the interaction much easier for both the instructor and agent, and as in the prior section, this allows the agent to be aggressive in learning. To continually learn new tasks (that may be composed of previously taught tasks), the agent must expand its capability to understand references (e.g., to the goals, actions, objects, concepts) that are relevant to the task being learned. This growing common ground ensures that the instructor and the agent can continue to interact using the newly learned knowledge.

Instruction Can Be Performed Immediately in the Environment

One advantage of situated instruction is that as soon as the agent learns some aspect of a task, it can attempt it in the world. If there is failure, it can ask questions and request training to improve its task performance. Conversely, instructions provided by the human instructor are immediately applicable to the relevant situation. This simplifies instruction generation for the human instructor as well as the learning problem for the agent. The instructor can focus on what they are observing and guide the agent to solve the specific instance they both are situated in without having to reason about how the task can be solved in general. The agent, on the other hand, does not have to reason about how and when to leverage the instruction. It can apply the instruction immediately and observe how it affects task performance, which guides learning.

In addition, when the agent attempts to solve a problem, it does not need internal models of the effects of all of its actions; it just needs the ability to execute them. Once it knows the task, it can attempt it by taking action in the world, possibly through exploration. If a hypothetical problem is given to an agent, it must have such models to internally search for a solution. When Rosie is learning new tasks while embodied in a mobile robot, it does not need detailed models of how the robot moves through the world; it can employ abstract models of moving from location to location. This simplifies the learning problem because only the goals need to be learned from instruction.

Learning Is Multimodal

In many learning problems, the source of available knowledge is structured and singular, such as a large database of labeled images. However, with interactive task learning, there are many potential sources of knowledge that arise from the different ways the human can interact with the learning agent and from the agent's access to prior knowledge. Possibly the simplest interaction a teacher can provide is a positive or negative reward as an evaluation of the agent's moment-to-moment task performance. Below we describe additional modalities for learning in ITL.

Language Can Be Used to Instruct Task Knowledge

When language is available, the human can explicitly define and describe different aspects of a task (goals, actions, task-specific concepts, and heuristics) through natural language (Crangle and Suppes 1994). Language can also be used for drawing attention to important aspects of the task as well as to label important collections (and parts) of objects and actions. The human can also lead the agent through the task by providing step-by-step instructions.

Language also makes it possible for the human to answer questions posed by the agent. At the extreme, unrestricted natural language provides a rich, robust vehicle. However, its use is currently beyond existing state-of-the-art methods. Thus, various forms of restricted languages are popular, which usually limit vocabulary, syntactic structures, and semantic content. From a learning perspective, language can provide direct definitions of many of the components of a task, so that the agent is essentially compiling natural language into internal structures that it can later interpret to direct its behavior. This is the approach that Rosie and other agents have taken (Hinrichs and Forbus 2013a). Important challenges are achieving generality and robustness in language understanding and "compiling" the knowledge into an efficient executable representation in real time.

Demonstration and Imitation Can Provide Examples of Task Performance

In combination with language, an instructor can use pointing and gestures to communicate important objects or actions. Without language, the human can *demonstrate* appropriate task behavior (Argall et al. 2009), which can vary from directly controlling a robot via teleoperation to manually moving joints or having the human perform the desired behavior. For tasks that are intimately tied to motor actions, this directly defines both the task and how it is performed. In some cases, the demonstrations can be taken to be instances of appropriate behavior in the task, which can then be analyzed to determine task definitions. Demonstrations can also include the instructor manipulating the environment to create important states, such as the goal of a task, but where the manipulations themselves are not important, as in creating a stack of disks for the final state in Tower of Hanoi (Kirk et al. 2016). Each of these creates different challenges and opportunities for learning. For direct robot control, the challenge is often how to generalize or transfer specific movements with specific objects to new tasks, where exact movements are insufficient to solve the problem.

For some demonstrations, the agent is not trying to learn motor commands but rather the abstract aspects of the task, such as learning the rules of Tic-Tac-Toe. In these cases, the agent needs to use these instances of behavior as positive examples of appropriate behavior. Just seeing positive examples, however, can make it difficult for an agent to learn illegal actions and failure states. More generally, this type of learning is usually not *interactive* because the number of examples is often very large, and it is tedious for the human to show many examples. A second problem with learning from demonstrations without language is that it is difficult for the agent to derive accurate meaning from the visualized examples. Often, demonstrations are tightly scripted and the agent is led through a series of situations.

Combinations of Language and Demonstration Are Possible

By combining language and physical interaction, it may be possible to get the best of both. Some physical actions or situations are difficult to describe with language, whereas with pure physical interaction it is difficult to determine which aspects (objects, relations, and features) of the current situation are relevant. Language can be used to identify which aspects of a task are being taught, define new words and concepts, and identify which objects, properties, and relations are most important during a demonstration.

Prior Knowledge Acquired from Other Tasks Can Transfer

Over time, an agent will accumulate knowledge about many different tasks, with knowledge learned in one task being relevant to future tasks. During learning, the agent needs to be able to take advantage of prior knowledge to simplify and accelerate learning. Similarly, the human instructor must have the means to recognize that the agent already possesses certain knowledge, so that it does not need to be taught. The direct effect of this on learning is that a learning agent must be able to use what it has learned in one context when it is learning in another context. All knowledge learned in one task cannot be sequestered solely to that task; however, some concepts are task specific, so the agent must have some way of learning when a concept does not transfer, such as through additional instruction.

During the instruction of Rosie, much of the knowledge is encoded in the new concepts it learns, such as "three in a row" for Tic-Tac-Toe, Three Men's Morris, and Connect-3. Newly acquired lexical items, together with their meaning, directly transfer to new tasks. This represents both an opportunity and a challenge, as some knowledge can transfer to new situations whereas other learned concepts (and lexical items) acquired for a specific task might not be transferable. Rosie does not have any special "transfer" reasoning except when a term is used that it cannot instantiate in the current situation. In this case, it will attempt to use a different meaning or learn a new definition.

Summary of ITL Characteristics

In Table 17.1, we summarize many of the defining characteristics of interactive task learning and provide examples of those characteristics in the agent Rosie.

Learning Approaches for ITL

In this section, we summarize the major constraints and challenges associated with ITL and describe some of the possible variants of how an agent interactively learns a task with an instructor.

Table 17.1 Summary of ITL characteristics, as exemplified by the agent Rosie.

ITL Characteristics	Rosie's Capabilities
Learning across different domains	Games, puzzles, and simple cooking tasks for a tabletop robot. Navigation and delivery tasks for a mobile robot.
Learning diverse task knowledge	
Perceptual categories	Elaboration rules that extract features from perceptions
Task formulation knowledge	Initial states, goal states, illegal states, constraints on operators, operator preconditions, operator selection knowledge including ordering, heuristics, policy knowledge, and hierarchical task structure
Lexical items	Parts of speech, word definitions, relation to concept definitions
Concept definitions	Conjunctions of unary features, relations, numeric functions, comparators, and numbers
Learning is interactive	
Real-time interactions	Integrates interaction with task performance and learning in real time
Sparse, high-quality data	Learns from sentence-by-sentence interactions and relies on aggressive generalization
Flexible instruction structure	Mixed initiative interaction and concepts defined before or after they are referenced
Learning is situated	
Shared environment	Instructor and Rosie are situated in the same environment. Instructor uses deictic reference, pointing, and demonstrations.
Immediate environmental feedback	Rosie applies instructions as they are provided, which the instructor can view and provide immediate feedback.
Learning is multimodal	
Language	Learns task knowledge from natural language descriptions
Demonstration	Learns goal descriptions from demonstration
Mixed	Language is used to identify important relations in demonstrations.
Prior knowledge	Transfers knowledge from previously taught tasks

When considering various approaches to learning tasks through human interaction in a situated environment, there are fundamental characteristics inherent to the ITL problem that influence and constrain the learning approaches used. Learning multiple aspects of a task and different task formulations requires an approach that can acquire different types of knowledge, possibly using very different techniques. Learning from interaction, where sparse examples are provided in real time, constrains the approaches to avoid reliance on huge data sets. Instead, the agent must be able to start learning something useful from the very first teaching example and incrementally build on that learned knowledge. Fortunately, information communicated through interaction is often high quality, highly relevant, and tied to a shared context. Learning through interaction requires the learning approach of the agent to be flexible to instructional variations as well as capable of taking advantage of the corrections, clarifications, and missing knowledge provided by the human instructor. Characteristics of the domain—such as its dynamics, the level of uncertainty in sensing and acting, and observability—will also affect the approach used. More than anything, we conclude that no one learning algorithm is going to be sufficient for all ITL systems. In fact, learning different types of task knowledge from multiple sources of available knowledge and multiple forms of interaction will almost certainly require using multiple learning algorithms in the same system.

Below we delineate different types of task information that can be provided to the agent from an instructor using the different modalities described earlier. We also consider various learning approaches that can learn from each kind of information, although not all meet each of the constraints we have described (such as being online, real time, with few interactions).

Learning from Reinforcement

The simplest type of information is a reward signal: the teacher provides negative and positive feedback in response to the agent's actions to guide the agent toward the goal. This is usually most appropriate for control tasks, where the agent is attempting to learn a value function using reinforcement learning. An early example was in the commercial computer game *Black and White*, where a player would attempt to train a character by giving it positive or negative reward as it performed actions in its simulated world. This ended up being tricky because the player had to wait for the character to do a relevant behavior and then immediately provide the reward. If the reward was a bit late the character would think that the reward applied to a different action than the instructor intended. Another example is an agent that was interactively trained in a computer game called *Sophie's Kitchen* (Thomaz and Breazeal 2008). Although interactive, online, and incremental, it was restricted to a single task or set of tasks that shared the same reward function. The advantage of instructing through reward is that it is simple for a person to learn what they need to do. However, it is limited in the types of knowledge it can teach, it may require the instructor to carefully monitor the agent's behavior for an extended period, and it usually only works for learning behaviors for a single task.

Learning from Execution Examples

An agent can also learn from the execution of a task in different ways:

1. By observing the instructor performing the task
2. Via a physical demonstration of the task through teleoperation or physical manipulation of the agent, or
3. From natural language instructions that lead the agent through the task

In the first case, the agent observes the instructor performing the task, recognizes the actions taken, and maps them onto its own capabilities.

The second case constitutes a difficult learning problem, as the agent is learning from the high-dimensional and noisy space of joint angles and motor movements. Again, the agent can just record the trace and play back the same motor commands, but this is not robust to variations in the initial state. Another approach would be to learn special key frames, which can either be learned or given by the instructor, and offers better generalization.

Finally, in the third case, the instructor can verbally lead the agent through a task, step by step, with individual commands. The agent can employ many different learning mechanisms to learn from this information. The simplest is to memorize the instructions and replay them when given the same task in the same initial state. It is a simple learning approach, but without further generalization capabilities, the agent is restricted to procedure tasks that can be repeated exactly, over and over. To increase generality, the agent can use a few different techniques. One is to use inverse reinforcement learning (Abbeell and Ng 2004) to induce a reward function that is consistent with multiple step-by-step actions. Inverse reinforcement learning usually requires many examples, and is not necessarily appropriate given the interactive aspect of ITL. The agent can also attempt to generalize its experience by learning a description of the goal through comparing the final state to the initial state, and possibly asking the instructor which aspects of the goal are definitional. This allows the agent to formulate future instances of problems that share the same (or similar) goals, but with different initial states.

Once the agent has derived a description of the goal, the agent can use its knowledge about the causal structure of its actions to learn a general policy using explanation-based learning (DeJong and Mooney 1986). This is the approach Rosie uses when it is learning from a sequence of actions: “Open the pantry. Move the plate into the pantry. Close the pantry. You are done.” It first derives and learns a description of the goal (possibly from further interaction with the instructor) and then performs a retrospective analysis of the solution that is used to learn general policy knowledge through Soar’s procedural learning mechanism. In Rosie, this approach is also used when the agent solves a problem space formulation of a task, where only the goal is specified: “The goal is the plate is in the pantry and the pantry is closed” (Mohan and Laird 2014).

Learning from Procedure Specification

An instructor can also provide a more general specification of desired behavior by providing an explicit description of the procedure the agent should follow, such as with the recipe example described earlier. From such a specification, the learner can attempt to learn a procedure which represents the actions needed to perform the task and their ordering. One example of this is the instruction graph (Meriçli et al. (2014), which is a directed graph that represents the flow of execution for the task and can include looping and conditional structures. An alternative is where the instructor provides an explicit procedure, such as "do *y* until condition *x* is met, then…" (Langley et al. 2010; Salvucci 2013), which is essentially a high-level programming language that the agent must dynamically compile into a procedure to execute in the future.

Learning from Task Concept Specifications

The final form of information from which an agent can learn is the explicit specification of high-level task concepts. For example, the instructor could directly communicate a problem space formulation of the task, providing descriptions of goal and failure states, specifications of legal and illegal actions, and definitions of newly introduced concepts. These instructions can be via natural language or demonstration (Huffman and Laird 1995; Hinrichs and Forbus 2013a). This is the method Rosie uses to learn new games and puzzles, by leading the instructor through each aspect of the problem space, asking for definitions, and then interpreting them in the shared context, and operationalizing them for future use (Kirk and Laird 2016).

A variant for teaching policy information is where the instructor provides heuristic advice, instead of a complete procedure or specific steps, to guide the agent's behavior (McCarthy 1968). The advice is usually more general and informative than is possible with a simple reward (e.g., specifying an action as "good" or "bad") and can include descriptions of the situation in which an action applies. In a problem space formulation, advice can be used so that the agent not only learns how to perform the task, but also can both solve a problem quickly and learn to perform it well. For example, in Othello, Rosie can learn specific heuristics such as "prefer moving a block onto a corner location over an edge location."

Another approach is to engage the instructor in an interaction to learn more general concepts that allow it to generalize its task concepts. When Rosie is presented with a demonstration of a goal state in its environment (e.g., a stack of disks for Tower of Hanoi), it engages in an interaction with the instructor to determine which features and relations can be ignored, and which are necessary for defining a goal state (Kirk et al. 2016). Finally, Rosie also allows the instructor to describe general hypothetical situations, so that it is not bound to specific instances.

Conclusion

In this chapter, we have reviewed the learning problem for ITL. What makes learning both challenging and exciting is that it is not a single problem, where there is one type of knowledge to be learned, through one type of interaction. Nor is there a single algorithm that is applicable to all aspects of ITL. Instead, the learning problem for ITL is a very broad constellation of problems, defined by diverse types of interaction and instruction, diverse domains and available knowledge, diverse types of knowledge to be learned, and diverse approaches for learning that knowledge.

Acknowledgments

The work described here was supported by the Office of Naval Research under Grants N00014-08-1-0099 and by Air Force Office of Scientific Research under Grant Number FA9550-15-1-0157. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of the ONR, AFOSR, or the U.S. Government.