Question Asking to Inform Procedure Learning

Melinda Gervasio Karen Myers

AI Center, SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025
melinda.gervasio@sri.com myers@ai.sri.com

Abstract

Learning by demonstration technology holds great promise for enabling users to automate routine but time-consuming online tasks. However, correct generalization from demonstration traces presents numerous technical challenges related to limited or noisy training data. To address these challenges, we propose a framework that augments learning by demonstration through a facility for asking questions of the demonstrator. We define a catalog of questions designed to inform learning by demonstration technologies that covers function and causality, abstraction, alternatives, limitations on learned knowledge, and the process of learning. We present approaches for managing when to ask questions and selecting which questions to ask that balance the potential utility of the answers against the cost to the demonstrator of answering. Metareasoning about explicit learning state and goals, along with models of question utility and cost, guide the question-asking process.

Introduction

People spend a lot of time on their computers performing repetitive tasks that could easily be automated, thus saving time and eliminating tedium. Non-programmers lack the skills to develop software to automate these tasks. Capable programmers often will not bother, due to the time investment required to understand the APIs for accessing necessary functionality, even if the resulting program is straightforward to write.

One promising avenue for enabling automation of simple, repetitive tasks is to have the user *demonstrate* rather than *program* the desired functionality. Assuming that appropriate instrumentation can be provided, the user could simply perform the task one or more times, and then draw on machine learning technology to create a generalized procedure for the task that could be used in future problem-solving episodes. The natural interaction provided by *programming by demonstration* has made it particularly popular as an approach to end-user programming (Cypher, 1993; Lieberman, 2001). More recently, there has been renewed interest in the field due to research efforts such as DARPA's Integrated Learning and Bootstrap Learning programs that are focused on learning how to perform complex tasks from few examples.

Learning procedures in a fully automated manner presents significant technical challenges. A single demonstration sequence—or even a few of them—will generally lack the breadth to define the general case. A human demonstrator may be willing to provide a small number of examples, but generally will not provide enough for unambiguous learning of the intended generalized problem-solving knowledge. In addition, the demonstrator may make mistakes or may become distracted and include irrelevant or unnecessary steps in the demonstration, thus further complicating the learning process.

For these reasons, a successful learning by demonstration facility will need supplementary information from the demonstrator to simplify the learning problem. For example, the PLOW system relies on a running narrative by the demonstrator to help focus the learning process (Allen et al., 2007). This type of mixed-initiative interaction is common in other subfields of AI that have sought to build technologies that can be deployed for real-world tasks (Bresina et al., 2005; Myers et al., 2003; Tecuci et al., 2007).

We propose to improve learning performance by augmenting learning by demonstration with a metalevel framework for asking questions of the demonstrator (Figure 1). One contribution of the work is the definition of a catalog of questions designed to inform the learning by demonstration process. These questions cover function and causality of elements in the demonstration trace, abstraction, alternatives and justifications, limitations on learned knowledge, and the process of learning. A second contribution is the definition of a metalevel capability for managing question asking within a learning by demonstration system. This capability reasons about gaps in the learned knowledge to determine appropriate questions. The metareasoning takes a limited rationality perspective in selecting questions to pose to the expert, trading off the utility of the missing knowledge with the cost of obtaining it. By deciding which questions to ask, it affects the behavior of the base-level learning components, thereby indirectly controlling system performance.

We begin the paper by presenting a general framework for learning by demonstration, along with a brief description of the specific system that motivates our work. We then describe our proposed metareasoning framework for determining when to ask questions and which questions to ask, and present a catalog of potential questions for informing the learning process. Finally, we describe an initial prototype for question asking, discuss related work, and present closing remarks.

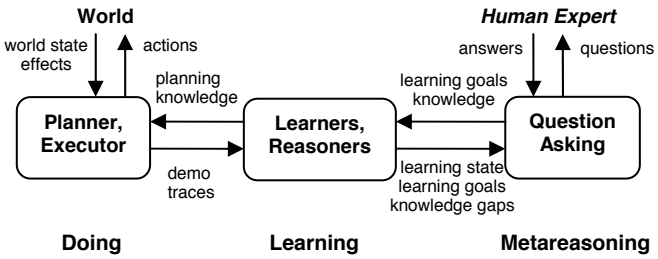

Figure 1. Metareasoning for question asking in a learning by demonstration system.

Framework

We address question asking within the general framework of a community of learners and reasoners having the shared goal of acquiring the ability to perform a complex task from a small number of demonstrations. While most work in machine learning addresses atomic tasks like classification or ranking, here the task being learned involves multiple interacting steps—for example, planning a search and rescue mission, scheduling a medical evacuation, or preparing an information packet for a briefing. Because of the complexity of the task, examples will be hard to come by and learners will face substantial ambiguity, hence the need for a capable question-asking framework to provide supplementary information to aid learning.

In this section, we define the basic performance task to which learning is being applied along with the corresponding learning task. We also describe the POIROT learning by demonstration framework, which is the target system for our question-asking module.

Performance Task

The underlying performance task is essentially a planning problem, defined by a *problem description* consisting of an initial state, a goal state, a collection of actions that can be performed during a demonstration, and a collection of domain constraints. An *action* is defined in terms of a *name*, *parameters*, *preconditions*, and *effects*. Parameters consist of *inputs*, which are used to instantiate an action, and *outputs*, which are functionally determined by the inputs. Preconditions constrain the applicability of individual actions, while *domain constraints* encode additional environmental restrictions.

Given a problem description, the performance task involves determining a sequence of actions that will accomplish the goal from the initial state. Generating solutions involves two basic types of decisions: *action selection* and *parameter instantiation*. In some situations, domain and problem constraints yield a single solution and thus fully determine the decisions that should be made. However, in the more general case, there may be multiple solutions of varying quality, introducing the need to reason about metrics to identify preferred solutions.

Learning Task

The goal of learning in this framework is to acquire from demonstrations by an expert user the problem-solving knowledge necessary to generate high-quality solutions to problem descriptions. Doing so will enable future problems to be solved by the system in automated fashion, rather than by the expert user. Because of this motivation to apply the learned knowledge, the qualities of modularity and understandability for the representation of the knowledge are highly desired. For this reason, we assume the organization of learned knowledge within a hierarchical task network (HTN) representation, which directly supports abstraction and modularity. In particular, procedural knowledge is to be captured in a hierarchy of *methods* that define how tasks can be decomposed into lower-level tasks.

Learning from a demonstration trace involves generalization along a number of dimensions. Collections of actions in the trace may be grouped to form *iterations*, which are defined in terms of a *domain* (i.e., the values over which to iterate), a *body* (the actions to perform for each value), and possibly an *ordering* over the domain. Generalization may identify *conditions* that restrict the applicability of an action or sequence of actions. Generalization can also produce *abstractions* of subsets of actions within the trace into coherent methods for higher-level tasks intended for future use.

POIROT

POIROT is a task-learning system for medical evacuation planning. In POIROT, the performance task involves planning a *workflow* for scheduling patients. The execution of a workflow generates an *execution trace*, and the final result is an *evacuation schedule* for the patients.

POIROT's goal is to learn general planning knowledge for generating evacuation schedules for patients, determining where and how to transport them while taking into account the type and severity of their injuries, their locations, and the availability of transport to move them to an appropriate treatment center. POIROT must learn this from a *single* demonstration trace consisting of an ordered list of actions for information gathering (e.g., to find out about requirements or transport options), booking transportation, and initiating transportation options when current options are oversubscribed or inadequate.

POIROT accomplishes this through the coordinated efforts of a suite of complementary learning and reasoning modules situated within a blackboard architecture (Burstein et al., 2007). Given the demonstration trace from an expert user solving a specific problem instance, POIROT learns a set of HTN methods for solving future problem instances. *Hypothesis formers* posit different generalizations, including looping and branching structures, conditions on branches, temporal ordering between steps, task abstraction, and preferences over parameter selection. *Hypothesis evaluators* are available for testing the current hypothesized methods through experimentation and exploration. Finally, a set of

metareasoners is responsible for combining the disparate hypotheses, detecting conflicts, invoking individual components, and generally coordinating the learning process.

Our question-asking module, QUAIL (Question Asking to Inform Learning), is one of the metareasoners within POIROT. Learning from a single example is a challenging task and even with its powerful suite of learners and reasoners, POIROT may lack sufficient information to be able to learn certain parts of the target planning knowledge. The quality of its learned knowledge could be compromised by missing critical information and the efficiency of its learning may be impaired by the ambiguity arising from insufficient knowledge. QUAIL provides POIROT with the ability to acquire missing information, and thus potentially improve its learning performance, by asking questions of the human expert.

Learning Goals

Learning in POIROT is driven by *learning goals*, which are posted to the blackboard by the various components to be addressed by other components, possibly with mediation by a metareasoner. Previously, POIROT's learning goals were all *process oriented,* designed to coordinate the various steps involved in acquiring planning knowledge through learning. Examples of these learning goals include hypothesis creation, hypothesis merging, explanation of observations, and hypothesis validation through experimentation.

Our work introduces the complementary notion of *knowledge-oriented learning goals*. Knowledge-oriented goals provide a means for components to convey their need for additional information—basically, *questions* for which they seek answers in order to let them proceed with their learning or reasoning, or to be more effective in doing so. QUAIL is the metareasoner ultimately responsible for evaluating these questions and managing the question-asking process between POIROT and the expert user.

Question Asking

We formulate the question-asking process in terms of three separate activities: *question nomination, question selection,* and *question posing*.

Question nomination refers to the generation of questions that can potentially provide value to the learning process. Within the POIROT framework, question generation is performed by the individual hypothesis formers and evaluators, as they are best positioned to determine their information needs.

Question selection is the process of choosing from among nominated questions those that should be posed to the user. The creation of appropriate selection strategies lies at the heart of our work and is discussed further below.

Question posing refers to the interactions with the user to obtain answers. Relevant issues include modality (not addressed in this paper) and timing (discussed below).

Question Selection Strategies

Question selection strategies must balance the need for information to further the learning process with the following considerations.

A. Individual questions contribute different levels of value to the learning process.
B. Answering questions imposes a burden on the user.
C. The role of question asking is to inform learning, rather than to replace it by a knowledge acquisition process.

These considerations lead naturally to the formulation of the question selection problem in terms of a cost-benefit analysis, drawing on models of *utility* and *cost* for individual questions.

Utility Model for Questions

The *base utility* of a question is determined by the component that nominates the question. The factors that affect base utility are highly dependent on the specifics of the nominating component. For example, learning components might measure utility in terms of the expected change in the theoretical lower bound on the number of examples needed to learn the target concept. Components faced with ambiguity might factor in the expected reduction in the number of valid alternative hypotheses.

Since our framework involves a community of learners, factors beyond base utility must be considered to determine overall question utility. First, base utility estimates must be normalized to make them comparable across components. Second, the relative importance of contributions from different components must be considered. Finally, multiple learners may benefit from the answer to a given question.

Given these considerations, we propose to model overall question utility relative to a set L of learners as follows:

$$Utility\,(q) = \sum_{l \in L} w_l \times Utility(q,l)$$

$$Utility\,(q,l) = w_B \times BaseUtility(q,l) + w_{LG} \times LearningGoalUtility(q,l)$$

The utility of a question for an individual learner l, denoted by $Utility\,(q,l)$, is defined to be a weighted sum of the base utility $BaseUtility(q,l)$ assigned by the learner and the utility $LearningGoalUtility(q,l)$ of the associated learning goal that motivated the question by the learner. Here, $BaseUtility(q,l) \in [0,1]$ while $w_B + w_{LG} = 1$. The overall utility for question q, denoted by $Utility\,(q)$, is a weighted sum of the utilities assigned by the individual learners, with the w_l encoding the relative weightings assigned to the individual learners subject to the constraint that $\sum_{l \in L} w_l = 1$.

Cost Model for Questions

Prior work on mixed-initiative systems has identified five cost factors for consideration by an agent when deciding whether to interact with a user (Cohen et al. 2005):

1. Inherent difficulty of the question
2. Level of disruption given the user's current focus of attention

3. User's willingness to interact with the system
4. Timing of the interaction
5. Appropriateness of the question

Within a learning by demonstration setting, the user would be expected to focus on interacting with the system and be tolerant of questions that may seem ill motivated or have obvious answers. As such, we can disregard factors (2)-(5) above, focusing instead on the inherent difficulty of the question as the basis for the cost model.

With this perspective, the cost model should measure the cognitive burden incurred by the expert in answering the question. This can be an arbitrarily complex quantity to measure, involving not only readily observable factors such as time to answer and brevity of answer but also potentially more complex metrics such as difficulty in understanding a question. However, there are certain heuristics we can use to approximate cognitive burden. Typically, certain question formats will be easier to answer than others. For example, yes/no questions usually will be less costly to answer than multiple-choice questions, which in turn would be less costly than open-ended questions.[1] Similarly, a question grounded in the demonstration trace most likely will be easier to answer than the same question asked about a hypothetical situation.

With these observations in mind, we define the cost of a question by

$$Cost(q) = w_F \times FormatCost(q) + w_G \times GroundednessCost(q)$$

where $FormatCost(q)$ denotes the cost associated with the format of q, and $GroundednessCost(q)$ is either $C_{concrete}$ or C_{hypo}, depending on whether the question relates to concrete elements in the demonstration trace or a hypothetical situation. Both $FormatCost(q)$ and $GroundednessCost(q)$ are constrained to lie in the interval [0,1]. The weights w_F and w_G allow relative weighting of the two cost factors, subject to the constraint that $w_F+w_G=1$.

Control Strategies for Question Posing

We consider two types of control for managing when to pose questions: *asynchronous* and *synchronous*.

Asynchronous Control

An *asynchronous control strategy* lets questions be asked continuously during the learning process. Asynchronous strategies could possibly lead to faster learning as they would enable early elimination of incorrect or irrelevant hypotheses, leading to a more focused search. For example, questions to resolve substantial ambiguity during learning may better be asked as they arise, rather than waiting until a complete initial hypothesis has been formed.

However, asynchronous control complicates the management of question asking, since the decision of whether or not a question is worth asking has to be made without knowledge of any other questions and possibly even without a hypothesis space against which the value of a component's contribution can be measured.

Management of continuous questioning has been considered previously. Techniques to address this problem generally can be categorized as *backward* or *forward looking*.

Backward-looking approaches incorporate historical information about previous interactions into their cost models, as a way to prevent question overload. For example, (Cohen et al., 2005) uses cost-benefit analysis to determine when the expected utility increase for asking a question outweighs the associated costs. The cost model includes both the associated *base bother cost* for a question and an *accumulation bother cost* derived by summing over costs associated with prior question initiations, scaled by a decay factor that takes into account the temporal distance from prior questions to the present.

In contrast, forward-looking approaches use MDPs to reason about future expected costs and reward in order to determine when to initiate interactions. Forward-looking approaches have been used to support both questions about action selection strategy (e.g., Scerri et al., 2002) and learning user preference models (e.g., Boutilier, 2002).

Forward-looking approaches require detailed models of the likelihood and utility of expected question outcomes, which will be difficult to obtain in practice. For this reason, a backward-looking approach has greater appeal for our question-asking framework.

Synchronous Control

A *synchronous control strategy* lets questions be asked only at a fixed point during the learning process. Synchronous control may sacrifice some learning efficiency as components may not be able to get critical clarifications as early as desired. But it could potentially lead to better use of the resources for question asking as the questions could be considered in groups, all pertaining to some stable state of the hypothesis space.

The synchronous question selection problem can be formulated as follows. We assume the following functions defined for a collection of questions Q.

$$Cost(Q) = \sum_{q \in Q} Cost(q)$$
$$Utility(Q) = \sum_{q \in Q} Utility(q)$$

As noted above, question selection imposes a burden on the demonstrator. Furthermore, our goal is to provide question asking in *support of learning,* rather than devolving into a knowledge acquisition process that obtains extensive procedure knowledge through system-initiated interactions. For these reasons, we impose a budget on question asking to restrict access to the user.

[1] Question format and cognitive load are not perfectly correlated (e.g., the halting problem constitutes a particularly difficult yes/no question). Our question catalog (described below) has been designed to enforce this correlation, with simple question formats used only for questions with low expected cognitive load.

Definition (Synchronous Question Selection). Given a collection of questions $Q=\{q_1 \ldots q_n\}$ and a budget B, determine a subset $Q' \subseteq Q$ with $Cost(Q') \leq B$ such that there is no $Q'' \subset Q$ for which $Cost(Q'') \leq B$ and $Utility(Q'') > Utility(Q')$.[2]

This framing of synchronous question selection maps directly to the *knapsack* problem (Kellerer et al., 2005). Although the knapsack problem is NP-complete, dynamic programming algorithms can generate solutions that run in time $O(nB)$. Given a reasonable number of nominated questions and budget, we anticipate acceptable performance.

Synchronous question selection could be applied at the end of the learning process, thus enabling individual learners to refine their initial hypotheses prior to generation of the final learning output. Another possibility is to support a fixed number of synchronous question selection sessions during the learning process, thus enabling the effects of the answers to be propagated through the system. The available budget for question asking would be distributed across the different sessions, in a manner that provides the most benefit to the learning process. This *multiphase synchronous question selection* provides some of the benefits of asynchronous question selection but with simpler control and better-understood selection methods.

Question Catalog

Here, we present our catalog of questions for aiding in learning generalized planning knowledge in a learning by demonstration setting. In the presentation of the questions, we use *A* to denote an arbitrary action in a plan, *E* for an effect, *P* for an action parameter, *M* for a learned method, and *C* for a world-state condition.

For ease of understanding, our questions are presented in natural language format. We have developed a formal representation for portions of the catalog, which will be expanded during the course of our work to a complete ontology of questions to support metareasoning for question selection.

The questions could be organized along several dimensions. To emphasize how the questions would be used, we have chosen to categorize them according to function. Other possible groupings include by the objects that they reference (e.g., methods, parameters, traces) or question complexity.

For reasons of brevity, the presentation omits variants on questions that obtain the same or related information but through mechanisms with different levels of information content and associated cognitive loads. For example, the question *Which action(s) does A_i enable?* could be decomposed into a family of related yes/no questions of the form *Does A_i enable A_1?, Does A_i enable A_2?*, etc.

Individual yes/no questions may be preferred in certain situations, given that they have lower format costs compared to other question types. In particular, asking the yes/no rather than the more general version of the question may be appropriate when only the more focused information is needed for the learning process, thus preserving resources to expend on other questions.

A. Workflow Function and Causality

These questions seek to identify the specific function of trace elements and relations among them, as a way of understanding the causal structure of the plan. They focus on identifying enabling and dependency relationships (including producer-consumer relationships between parameters), codesignation constraints among variables, and temporal constraints. This information is critical for abstracting the demonstration trace to correct generalizations.

- *Which action(s) does A_i enable?*
- *Which precondition(s) does A_i enable?*
- *Which action(s) does A_i depend on?*
- *Which effect(s) does A_i depend on?*

- *Which action(s) does E_i enable?*
- *Which precondition(s) does E_i enable?*
- *Which action(s) does E_i depend on?*
- *Which effect(s) does E_i depend on?*

- *Do P_j and P_k have to be the same value?*
- *Does P_j determine P_k?*
- *How does P_j determine P_k?*
- *What parameters does P_j determine?*
- *Which of $\{P_1,P_2,\ldots,P_n\}$ determines P_k?*
- *Which of $\{P_1,P_2,\ldots,P_n\}$ does P_j determine?*
- *Is P_j an input to the procedure?*
- *Is P_j an output of the procedure?*

- *Must A_i precede A_j?*
- *Must A_i immediately precede A_{i+1}?*
- *Why must A_i [immediately] precede A_j?*
- *Is it desirable for A_i to precede A_j?*
- *Why is it desirable for A_i to precede A_j?*

- *What is the role of action A_i in the plan?* (Possible responses include (a) to enable one or more actions, (b) to achieve one or more effects, (c) to validate an earlier decision, (d) to provide situational awareness or expose needed information, or (e) unnecessary – the action was included by accident or superseded by some later action.)

[2] This formulation of the problem assumes that the questions in Q are independent of each other, i.e., obtaining the answer to one question does not impact the utility of answering the others.

B. Decision Alternatives and Justifications

These questions seek to clarify why certain decisions were made and what alternatives are possible. This type of knowledge is essential for creating learned procedures that can adapt to execution in new contexts.

- *Could A_i be used as an alternative to A_j in the plan?*
- *What actions other than A_j could be used?*
- *What factors affected the choice of A_j?*
- *Why was A_j used rather than A_k? (e.g., faster, better quality)*
- *Could P_i be used as an alternative to P_j in action A_k?*
- *What values other than P_j could be used in action A_k?*
- *What factors affected the choice of P_j?*
- *Why was P_j used rather than P_k? (e.g., faster, better quality)*
- *Can P_j and P_k be the same value?*
- *Can P_j and P_k be different values?*

C. Limitations and Failure

The questions in this category relate to understanding limitations on the viability and desirability of the given trace, as well as causes and possible solutions to failures that result when workflows are executed. Questions in this category will contribute to the development of learned procedures that are robust to situation dynamics.

Limitations on the current solution:

- *Under what conditions does this plan work?*
- *Under what conditions would A_j cause problems?*
- *Under what conditions would A_j be undesirable?*
- *Under what conditions would the choice of P_i for A_j cause problems?*
- *Under what conditions would the choice of P_i for A_j be undesirable?*
- *Would this plan work if condition C did not hold?*
- *What changes to the world state would cause this plan to fail?*
- *Where should conditional branches be introduced in the workflow?*

Causes and solutions to execution-time failures:

- *Why did the execution of A_j fail?*
- *Which actions contributed to the failure of A_j?*
- *Which effects contributed to the failure of A_j?*
- *How could the failure of A_j be prevented?*

D. Request for Demonstration

Within a learning by demonstration framework, the most natural means to communicate more knowledge is to provide additional demonstrations. Such demonstrations could focus on learning localized pieces of problem-solving knowledge where the system recognizes its current deficiencies, or exploring cases that will broaden system problem-solving knowledge. For example, additional demonstrations may be required to show the appropriate course of action under conditions not observed in the original demonstration.

- *Show how to achieve an effect E_i.*
- *Show an alternative way to achieve effect E_i.*
- *Show how to achieve effect E_i under condition C_j.*
- *Show how to handle a new problem.*
- *Show how to handle the current problem for the initial state $\{ C_{1,...}C_m\}$.*

E. Abstraction

Abstraction questions seek information about the generalization of the trace into higher-level constructs such as iterations and generalized methods. Such questions are valuable for organizing learned knowledge to support reuse, for transfer to related problem-solving domains, and for increased user understandability of learned procedural knowledge.

Iteration:

- *Is there a preferred order for processing elements in the iteration domain?*
- *What is the preferred order for processing elements in the iteration domain?*
- *What defines the domain of iteration for actions A_j through A_{j+k}?*
- *Do actions (or action sequences) A_i and A_j have the same goal?*
- *Which actions (or action sequences) in the trace have the same goal?*

Generalized Methods:

- *Which higher-level actions could A_j contribute to?*
- *Which actions in the plan contribute to higher-level action A_j?*
- *Should $\{A_j , A_{j+1} \ldots, A_{j+m} \}$ be generalized into a method?*
- *What outputs should method M_j have?*

F. Meta-questions

Meta-questions focus on how to learn, rather than the domain-level knowledge being learned. Questions in this area seek information on how to frame the learning problem (e.g., determining appropriate learning biases) as well as priorities for the learning process.

- *Is there a preference function to determine the choice of action A_k?*
- *Is there a preference function to determine the choice of P_j in action A_k?*
- *Which attributes are relevant to the preference function to determine the choice of action A_k?*
- *Which attributes are relevant to the preference function to determine the choice of P_j in A_k?*
- *Given a set of conflicting hypotheses, which should be evaluated first?*

- *There is no identifiable dataflow or precondition/effect relations that necessitate the ordering of actions in the plan: should an experiment be run to determine ordering constraints?*

Discussion

We formulated our question catalog based on an analysis of the general task-learning process. We identified the different types of generalizations required to induce a general set of problem-solving methods from the trace of a successful execution of a plan for a specific problem and then systematically analyzed the demonstration trace and the generated hypotheses to identify the relevant portions about which questions may arise.

As an initial, objective evaluation of the coverage of our questions, we assessed each component in POIROT to determine: 1) whether its primary learning task was covered by these questions; 2) whether its particular learning biases could be determined by answers to these questions; and 3) whether any remaining ambiguities (alternative hypotheses) in its outputs could be resolved through these questions. This led to the identification of some additional question instances but introduced no new categories. Thus, we believe that our categories cover a sufficiently broad subset of the space of possible questions to support effective question asking for informing learning.

A Prototype for Question Asking

As a first step toward providing a capable question-asking system for POIROT, we are developing an initial prototype that will implement a synchronous strategy for question selection. We will use this prototype to understand better how to model question utility and costs, how to distribute resources across multiple-phase synchronous question asking, and how learning performance (speed, quality) can be improved through appropriate question asking. To support this effort, we are currently formalizing the representation of questions, including the formulation of specific cost and utility models in line with the general approach presented above.

A second phase in our work will explore the use of backward-looking question management strategies, including an investigation of their relative strengths and weaknesses compared to single- and multiple-phase synchronous approaches.

One drawback to using utility-based methods for question selection is the problem of formulating appropriate utility models. Longer term, we are interested in exploring learning strategies for acquiring and evolving question utility models.

Related Work

Most previous learning by demonstration systems involved a single learner (Cypher, 1993; Lieberman, 2001), unlike POIROT, which utilizes several learners and reasoners. In these systems, proactivity on the part of the learner is typically limited to classification questions or suggestions to facilitate automation. In collaborative learning systems such as Collagen (Garland et al., 2001) and PLOW (Allen et al., 2007), the learning process is cast as a dialogue between the user and the learner and thus questions to the user arise naturally from the dialogue models that drive the interaction.

Ram and Hunter (1992) introduce the notion of explicit knowledge goals to capture gaps in the system's knowledge, defining knowledge goals as comprising both the specification of the missing knowledge and the task enabled by it. In addition, they propose augmenting knowledge goals with a utility measure to help drive the inference process. The field of knowledge acquisition has focused primarily on developing effective techniques for eliciting factual knowledge from experts, but some work has also addressed the acquisition of problem-solving knowledge. For example, EXPECT (Melz & Gil, 1996; Kim & Gil, 1999) exploits explicit representations of factual and problem-solving knowledge and their interdependencies to identify potential gaps or problems and to generate follow-up questions. In POIROT, the primary task is to acquire planning knowledge from a single demonstration and in QUAIL, we focus on the management of question selection and question posing .

In supervised machine learning, *active learning* has been used to address the issue of small (labeled) training sets (e.g., Seung et al., 1992; Lewis & Gale, 1994; Freund et al., 1997). Through the use of various techniques, a supervised learner can select for labeling by the human expert those unlabeled examples most likely to provide the greatest value to the learning process. We are performing a form of active learning with QUAIL, but one that is inherently more complex as it is in service of several different components learning a multistep task rather than a single learner learning an atomic task.

Several prior efforts have defined catalogs of questions to support improved understanding of complex activities but have had the objective of furthering *human* rather than *machine* understanding of tasks. Furthermore, they were motivated by activities that differ from the workflows addressed in our work. These differing motivations resulted in a focus on different types of questions from those we have identified as central to support learning by demonstration, although some overlap does occur.

For example, the ICEE framework defines a collection of questions aimed at human understanding of the behavior of an intelligent task executor (McGuinness et al., 2007). Many of these questions provide visibility into execution state and history, which are not relevant to question answering for task learning. Overlap occurs for questions related to exposing information about causal relationships among executed actions (e.g., Why did you execute (or not execute) a certain action?). However, these questions are oriented toward human understanding of the details of a specific execution case, rather than extracting more general problem-solving knowledge for subsequent reuse.

Gruber and Russell (1995) define a catalog of questions for supporting user understandability of an electromechanical design, derived from sessions in which designers asked questions or formulated hypotheses as to design structure and rationale. Many of these questions relate to understanding background knowledge for the design task and validation of requirements. Not surprisingly, given the similarity between understanding a design and understanding a plan, there is some overlap in question scope. Hypothetical reasoning is significant for both cases, as are justifications for choices (relative to alternatives), and causal dependencies. Question details differ, however, as a result of the focus on *designs* rather than *plans*, and *human* rather than *system* explanation.

Summary

This paper presents a metalevel framework for augmenting a learning by demonstration system with the capacity to ask questions of a demonstrator in order to improve the quality and speed of learning. We believe that this kind of question-asking facility will be important in enabling procedure-learning technology to be used broadly and effectively in a wide range of application domains.

The main contributions to date in this work are the creation of a question catalog that captures the types of questions that would be most relevant for the learning system as well as an initial metareasoning approach to support question asking that is grounded in a cost-benefit analysis.

Our near-term work will focus on completing an initial prototype and evaluating it within the context of the POIROT learning by demonstration application. Longer term, we are interested in exploring learning strategies for acquiring and evolving the question utility models that lie at the heart of our approach.

Acknowledgments. This work was supported by the Defense Advanced Research Projects Agency and the United States Air Force through BBN Technologies Corp. on contract number FA8650-06-C-7606.

References

Allen, J., Chambers, N., Ferguson, G., Galescu, L., Jung, H., Swift, M., and Taysom, W. 2007. PLOW: A collaborative task learning agent. *Proc. of the 22nd Conference on Artificial Intelligence*. Vancouver, Canada.

Boutilier, C. 2002. A POMDP formulation of preference elicitation problems. *Proc. of the 18th National Conference on Artificial Intelligence*, Edmonton, Canada.

Bresina, J., Jonsson, A., Morris, P. and Rajan, K. 2005. Activity planning for the Mars exploration rovers. *Proc. of the 15th International Conference on Automated Planning and Scheduling*. Monterey, CA

Burstein, M., Brinn, M., Cox, M., Hussain, M., Laddaga, R., McDermott, D., McDonald, D., and Tomlinson, R. 2007. An architecture and language for the integrated learning of demonstrations. *Proc. of the AAAI Workshop on Acquiring Planning Knowledge through Demonstration*, AAAI Technical Report WS-07-02.

Cohen, R., Cheng, M., and Fleming, M. W. 2005. Why bother about bother: Is it worth it to ask the user? *Proc. of the AAAI Fall Symposium on Mixed-Initiative Problem-Solving Assistants*.

Cypher, A. (ed.). 1993. *Watch What I Do: Programming by Demonstration*. MIT Press: Cambridge, MA.

Freund, Y., Seung, H., Shamir, E., and Tishby, N. 1997. Selective sampling using the Query by Committee algorithm. *Machine Learning*, 28, 133–168.

Garland, A., Ryall, K., and Rich, C. 2001. Learning hierarchical task models by defining and refining examples. *Proc. of the First International Conference on Knowledge Capture*.

Gruber, T. and Russell, D. 1995. Generative design rationale: Beyond the record and replay paradigm. In *Design Rationale: Concepts, Techniques, and Use*, T. Moran and J. H. Carroll (eds.). Lawrence Erlbaum Assoc.

Kellerer, H., Pferschy, U., and Pisinger, D. 2005. *Knapsack Problems*. Springer Verlag.

Kim, J. and Gil, Y. 1999. Deriving expectations to guide knowledge-base creation. *Proc. of the 16th National Conference on Artificial Intelligence*.

Lewis, D. and Gale, W. 1994. A sequential algorithm for training text classifiers. *Proc. of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*.

Lieberman, H. (ed.). 2001. Your Wish is My Command: Programming by Example. Morgan Kaufmann: San Francisco, CA.

McGuinness, D.L., Glass, A., Wolverton, M., and Pinheiro da Silva, P. 2007. A categorization of explanation questions for task processing systems. *Proc. of the AAAI Workshop on Explanation-aware Computing*. Vancouver, Canada.

Melz, E. and Gil, Y. 1996. Explicit representations of problem-solving strategies to support knowledge acquisition. *Proc. of the 13th National Conference on Artificial Intelligence*.

Myers, K. L., Jarvis, P., Tyson, W. M., and Wolverton, M. J. 2003. A Mixed-initiative framework for robust plan sketching. *Proc. of the 13th International Conference on Automated Planning and Scheduling*. Trento, Italy.

Ram, A. and Hunter, L. 1992. The use of explicit goals for knowledge to guide inference and learning. *Journal of Applied Intelligence*, 2(1):47-73.

Scerri, P., Pynadath, D., and Tambe, M. 2002. Towards adjustable autonomy for the real world. *Journal of Artificial Intelligence Research*, 17: 171-228.

Seung, H. S., Opper, M., and Sompolinsky, H. 1992. Query by committee. *Proc. of 5th Annual ACM Conference on Computational Learning Theory*.

Tecuci, G., Boicu, M., and Cox, M. T (eds.) 2007. *AI Magazine*, Special Issue on Mixed-initiative Assistants, Volume 28(2).