Learning to Ask the Right Questions

Melinda Gervasio
SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025
1-650-859-4411
melinda.gervasio@sri.com

Eric Yeh
SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025
1-650-859-6134
yeh@ai.sri.com

Karen Myers
SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025
1-650-859-4833
myers@ai.sri.com

ABSTRACT
Asking questions is an integral part of learning. Asking questions can clarify concepts, test hypotheses, add missing information, or provide additional knowledge to facilitate learning. The last item motivates the work described in this paper. By asking questions, our system obtains information that serves as background knowledge for a base learner, enabling the base learner to make useful generalizations even with few training examples. In previous work, we developed static strategies for question asking. Here, we extend that work with a learning approach for acquiring question-asking strategies that better accommodates the interdependent nature of questions. We present experiments validating the approach and showing its usefulness for acquiring efficient, context-dependent question-asking strategies.

Categories and Subject Descriptors
I.2.6 [**Learning**]: Concept learning, Induction, Knowledge acquisition

General Terms
Algorithms, Performance, Experimentation

Keywords
Active learning, decision tree induction, metalearning, preference learning

1. INTRODUCTION
Imagine a car salesperson trying to determine a customer's vehicle preferences to best decide which models to offer. Most likely, before suggesting vehicles, the salesperson asks a few questions to try to quickly determine the kind of cars that interest the customer. Using the customer's reactions, the salesperson refines the model of the customer's preferences and adjusts subsequent offers accordingly, over time figuring out which cars most interest the customer. The salesperson might just offer the customer one car after another, gradually learning the customer's preferences that way. But most customers will not willingly provide feedback on car after car after car. By first asking questions to obtain some background information, the salesman can more quickly learn the customer's preferences. And if the salesperson asks just a few, focused questions, the preference elicitation process is simplified for the customer as well.

Supervised machine learning algorithms primarily rely on labeled training data to acquire models for tasks such as classification and ranking. When training data is scarce, background knowledge can help constrain the hypothesis space and avoid overfitting. One approach to obtaining such background knowledge is by *asking questions* of a human teacher. For example, *active learning* asks for labels on selected examples [[2][3][6]] while *active feature selection* asks questions about feature relevance [12]. In most cases, question asking is tightly integrated into a specific learner.

Our work on QUAIL (*Question Asking to Inform Learning*) addresses an abstracted question-asking problem not tied to any specific learning algorithm [[7][8]]. Although tight integration with a particular learner clearly has its benefits—including the ability to more deeply integrate question asking into the learning algorithm—situations exist in which an independent question-asking system is desirable. QUAIL is part of a large, task learning system, POIROT [1], that includes several different learning and reasoning systems working in concert to address a single, overarching problem. QUAIL enables POIROT to manage question asking in a centralized manner, balancing the demands of the different learners against the benefits to the system as a whole.

Given questions with different costs and utilities, QUAIL's objective is to ask the highest utility questions within a specified budget. Initially, we formulated QUAIL's question-selection problem as a 0/1 knapsack problem [9]. Given a budget for questions and a set of questions with associated costs and utilities, QUAIL applied the knapsack algorithm to find a set of questions with maximal utility within the given budget. To evaluate QUAIL, we integrated it with CHARM [15], a system for learning lexicographic preference models from training examples consisting of pairwise preferences. Although preliminary results looked promising [8], further investigation revealed two complicating factors regarding utility that rendered this formulation problematic. First, question utility turned out to be *non-additive*—that is, the utility of a set of questions was not necessarily equal to the sum of the utilities of the individual questions. Second, utility turned out to be *context-dependent*—that is, the utility of a question was often dependent on the answer to previous questions.

A richer, more faithful utility model was clearly needed; however, it is infeasible to expect such a model to be engineered by hand. Indeed, our experience with the earlier experiments revealed that our intuitions were not always right: our initial utility assignments performed poorly, developing utility assignments that led to better performance required multiple attempts. During our experimentation, we also realized that questions were interdependent: asking particular questions was only useful given particular answers to previous questions, something our selection strategy did not account for. For our car salesperson, our static strategy would have amounted to deciding, ahead of time, to ask a particular set of questions to all customers, regardless of their answers. Thus, the salesperson might ask whether a customer liked luxury cars even if the customer had already indicated a desire for hybrid cars and there were no hybrid luxury cars.

To address these issues, we developed a learning approach to question asking, initially focused on addressing the context-dependent nature of question utilities. Because we address a *metalearning*[1] task—the objective being to find a question-asking strategy that provides the greatest improvement for a base learner—our metalearner does not itself interact directly with a human teacher. However, it uses performance data for a base learner that does rely on a *question-asking agent* to interact with a human teacher. This agent may itself be considered a learner as its goal is to obtain information to improve performance on some task. This agent's learning is completely under its own initiative: communication with the human teacher is restricted to questions from the agent and answers from the teacher. However, the questions may involve any content and be of any form provided the answers can be incorporated by the base learner. The metalearning process does not require a staged curriculum that builds over previously learned concepts. But, for the question-asking agent, the information acquired through question asking *is* dependent on the information gleaned from previous questions—the primary motivation for our work on learning question-asking strategies. The goal of our research is not so much to provide insight on human learning but, rather, to draw inspiration from the way humans ask focused questions to acquire information.

We begin by describing our formulation of a question-asking strategy as a decision tree. We then present our algorithm for inducing such a decision tree from examples consisting of a base learner's performance with the corresponding answers. Next we describe our evaluation domain, starting with a discussion of how question answers are incorporated into CHARM and moving on to a presentation of the synthetic datasets we designed to investigate the effects on learning of different target model populations and different object distributions. We then present our evaluation of L-QUAIL, the learning version of QUAIL, comparing its performance against a set of handcrafted strategies. We conclude with a discussion of related and future work.

2. LEARNING FORMULATION

To enable question selection to depend on previous answers, we need a data structure that supports conditional branching. One such structure is a *decision tree*, where the nodes correspond to questions and the branches to answers. Our basic learning task is thus one of *decision tree induction*. However, some distinctive features about question-asking decision trees distinguish them from the typical decision trees learned for standard supervised learning tasks.

In decision tree induction for classification (Quinlan, 1986), a decision tree is constructed by selecting successive features on which to split the training instances, represented as feature vectors, until all the instances have the same class label or some other stopping criterion is reached. Each node corresponds to a feature and each branch to a feature value, and the class label of the instances at a leaf node is the predicted class. The goal is to classify examples as quickly and as accurately as possible, so key to the construction of good decision trees is the splitting criterion for determining the next feature to test. For classification, the splitting criterion typically relies on some measure of entropy or impurity regarding the instances at a node and their labels.

[1] We use *metalearning* in the sense of learning to learn, rather than learning knowledge that can be transferred across domains.

For our metalearning problem, the training instances are the models to be learned by the base learner—for example, preference models for a preference learner or classifiers for a classification learner. The features are questions and the feature values are answers, so a model is effectively represented by its answers to the questions. Our goal is to provide the base learner with the most informative answers possible, so our splitting criterion is based on the expected performance improvement from the answers to a question across *all* the models at that node. More specifically, the expected performance of the base learner at a particular node is the average of its performance over all the models at that node, given the answers leading to that node, and we choose the question leading to highest expected performance over all the resulting splits. Our decision tree induction algorithm is summarized in Table 1.

Table 1. Decision tree induction algorithm for learning question-asking strategies.

LEARN(*models, questions, answers, cost, budget, evalset*)
  node = new decision tree node
  node.models = models
  node.answers = answers

  For each $q \in Q$ where $q.cost \leq budget - cost$
    $splits_q = \{<models_a, a> \mid models_a \subseteq models$ that answer a to $q\}$
    $q.perf = PERF(splits_q, answers, evalset)$
  Select q_{best} as the $q \in Q$ with maximum *q.perf*
  node.question = q_{best}

  If q_{best}.perf < *PERFORMANCE_THRESHOLD*
    For each $<models_a, a>$ in $splits_{qbest}$
      Add *LEARN*($models_a$, $Q-q_{best}$, *answers+a*, *cost+q.cost*, *budget*) as a branch to *node*
  Return *node*

PERF(*splits,answers,evalset*)
  perf = 0
  nmodels = 0
  For each $<models_a, a> \in splits$
    For each *model* in $models_a$
      perf = perf + EVAL_BASE_LEARNER(*model*, *answers+a*, *evalset*)
      nmodels++
  RETURN *perf/nmodels*

3. QUESTION ANSWERS AS BACKGROUND KNOWLEDGE

Building on our previous work, we decided to evaluate our learning approach to question asking using the CHARM preference learner as our base learner. CHARM learns *lexicographic preference models* from pairwise preferences—essentially, a partial ordering on object attributes, where the attributes have ordinal values and preferences can be for either direction (high or low) for each attribute [15]. We developed a set of questions for CHARM and modified CHARM to incorporate as background knowledge the corresponding answers for any target model [8]. For the experiments described in this paper, we used the binary attribute questions listed in Table 2.

Table 2. Binary attribute questions used in the experiments.[2] CHARM incorporates answers to these questions as background knowledge.

Question Class	Question
Attribute Relevance	*Is ATT relevant?*
Attribute Ordering	*Is ATT1 more important than ATT2?*
Value Ordering	*For ATT, are LOW values preferred to HIGH?*

Before learning, CHARM considers all attributes to be equally important and, for each attribute, high and low values to be equally preferred. Thus, without any learning or background knowledge, CHARM predicts a tie for any two objects. CHARM incorporates the answers to the attribute-related questions as constraints on the learned model—more specifically, on the relative attribute-ordering and value preferences reflected by the lexicographic preference model. Thus, even without any training examples, answers result in a modified model, typically one that predicts preferences over some objects.

The question types in Table 2 are complete: there are several combinations of question instantiations whose answers provide complete information to CHARM, enabling it to learn a preference model even without any training examples. Asking questions comes at its own cost and our goal is *not* to replace training but, rather, to augment it with the answers to carefully selected questions, particularly when the base learner has very few training examples. However, to focus our investigation purely on the effects of question answers, we do not provide CHARM with any training examples in the experiments discussed here.

4. DATA

Because we address a metalearning problem, we need data for evaluating the base learner (CHARM) as well as data for training and testing the metalearner (L-QUAIL). Specifically, we need the *objects* over which CHARM is to learn preferences, the *target models* CHARM is to learn, the *questions* QUAIL can ask, the *answers* of each target model to the questions, and CHARM's *performance numbers* given different question answers.

To support controlled experimentation, we generated synthetic datasets that varied assorted properties of the objects and the target models. Specifically, given a fixed set of object *attributes*, we created object sets that varied in how the attributes were correlated. We also varied the target models regarding which attributes were relevant, which attributes were equally preferred, and which attribute value direction was preferred.

We decided to exhaustively generate performance data for all possible question subsets for all the target models under the different conditions. Aside from greatly facilitating experimentation, these numbers provide us with a way to compute an upper bound on CHARM's performance with question answers. But because the number of objects, models, questions, and question subsets grows very quickly with the number of attributes, this exhaustive generation approach limits us to a small number of attributes—an issue we address in the discussion on future work. In the experiments reported here, we use four binary attributes, which correspond to the question instantiations in Table 3 and to the datasets in Table 4.

Table 3. Specific questions used in experiments.

Question Class	Question	ID
Attribute Relevance	Is A0 relevant?	q0
	Is A1 relevant?	q1
	Is A2 relevant?	q2
	Is A3 relevant?	q3
Attribute Ordering	Is A0 more important than A1?	q4
	Is A0 more important than A2?	q5
	Is A0 more important than A3?	q6
	Is A1 more important than A2?	q7
	Is A1 more important than A3?	q8
	Is A2 more important than A3?	q9
Attribute Value Ordering	For A0, is LO preferred to HI?	q10
	For A1, is LO preferred to HI?	q11
	For A2, is LO preferred to HI?	q12
	For A3, is LO preferred to HI?	q13

Table 4. Datasets used in experiments.

Dataset	Objects	Target Models		
	Att Correlations	***Relevant Atts***	***Pref Val Dir***	***Equiv Atts***
n4_v2_r4	none	all four	any	none
n4_v2_r2	none	any two	any	none
n4_v2_r2_01	none	only A0,A1	any	none
n4_v2_r2_23	none	only A2,A3	any	none
n4_v2_r2_01_23	none	{A0,A1} or {A2,A3}	any	none
n4_v2_r4_H	none	all four	HI	none
n4_v2_r4_L	none	all four	LO	none
eq_n4_v2_r2_0-1	none	A0,A1	any	{0,1}
eq_n4_v2_r4_0-1_2-3	none	all four	any	{0,1}, {2,3}
pos_n4_v2_r4	positive between A0 and A1, A2 and A3	all four	LO	none
pos_n4_v2_r2	positive between A0 and A1, A2 and A3	any two	LO	none
neg_n4_v2_r4	negative between A0 and A1, A2 and A3	all four	LO	none
neg_n4_v2_r2	negative between A0 and A1, A2 and A3	any two	LO	none

5. EVALUATION

Our primary motivation for developing a learning approach to question asking was to avoid the costly and at times fallible handcrafted question-asking strategies required to address the non-additive and context-dependent nature of question utilities. By learning a question-asking strategy, we can address these issues while also enabling the question asking to adapt to unique characteristics of the object and model populations. Thus, we designed our experiments to explore how well L-QUAIL can learn to ask the right questions.

5.1 Handcrafted Question-Asking Strategies

We compare the performance of the strategies learned by L-QUAIL against handcrafted strategies—both the original static strategies, developed in consultation with the CHARM developers, and a handcrafted decision tree. Based on our previous experiments, we already know that the static strategies

[2] In addition to these attribute-related questions, CHARM can also incorporate answers to questions about objects (e.g., *Is OBJ1 preferred to OBJ2?*) and to more costly, set-based question types (e.g., *Which of these attributes are relevant?*). These questions were selected from a comprehensive question catalog we designed for learning in POIROT [7].

suffer from a failure to account for the interdependent nature of questions. Thus, including them here to verify whether learning indeed benefits question asking is useful. Table 5 shows the static strategies used in the experiments.

Table 5. Original static question-asking strategies, which ask questions according to the utilities of different question types.

Strategy	Utilities
valord	value ordering > attribute ordering, attribute relevance
attord	attribute ordering > value ordering, attribute relevance
attrel	attribute relevance > attribute ordering, value ordering
att	attribute ordering, attribute relevance > value ordering
attval	attribute ordering, value ordering > attribute relevance
valrel	attribute relevance, value ordering > attribute ordering

Based on a better understanding of CHARM and an analysis of its method of incorporating questions answers, we also handcrafted a decision tree that addresses context dependence and that can be used to acquire a perfect model purely by asking questions. The strategy first ascertains an attribute's relevance, then it asks whether high or low values are preferred for the attribute, and finally it asks about the attribute's importance relative to every other attribute that has also been determined to be relevant. By taking previous answers into consideration, the strategy avoids asking useless questions. But the strategy does not consider budget at all and is somewhat biased by the order in which it asks about the attributes.[3] The latter is an issue with the decision tree formulation in general, where the question asked at each node is a deterministic choice.

5.2 Performance Upper Bound

The performance data generated for the datasets provides us with a means to compute a performance upper bound (or gold standard). For each budget and each model, we can identify the set of questions that results in the highest accuracy that CHARM can achieve for that model with that budget. Note that because the best performing subset of questions may be different for different models and distributions, in general, creating a strategy to ask precisely these questions is not possible.

5.3 Hypotheses

Our primary hypothesis was:

- The learned question-asking strategy will outperform the handcrafted strategies.

In addition, we also hypothesized that the learned decision trees would be adapted to the underlying characteristics of the object and model populations of the individual datasets. Specifically,

- Attribute relevance questions will gain prominence when there were irrelevant attributes.
- Attribute value questions, although generally the most useful, will not be used when the models always preferred a particular attribute value direction.
- Attribute-ordering questions will not be asked between attributes with equal importance.
- Learning preferences will be easier with positively correlated attributes and harder with negatively correlated attributes.

[3] With no prior information on relevance, we simply select an arbitrary ordering over the attributes for question asking.

5.4 Evaluation Metrics

Our primary evaluation metric was CHARM's performance, measured as the percentage of correct predictions (of pairwise preferences) over the objects in a dataset. Because we are interested in question asking under a constrained budget, we plot this performance over increasing budgets. We say one strategy outperforms another for a particular budget if it has higher accuracy for that budget. We also say it outperforms another if it reaches 100% accuracy sooner (i.e., with fewer questions). Because L-QUAIL is dealing with a metalearning problem, we average CHARM's performance over all the target models in a dataset. We also inspect the questions included and relations between them in the learned trees.

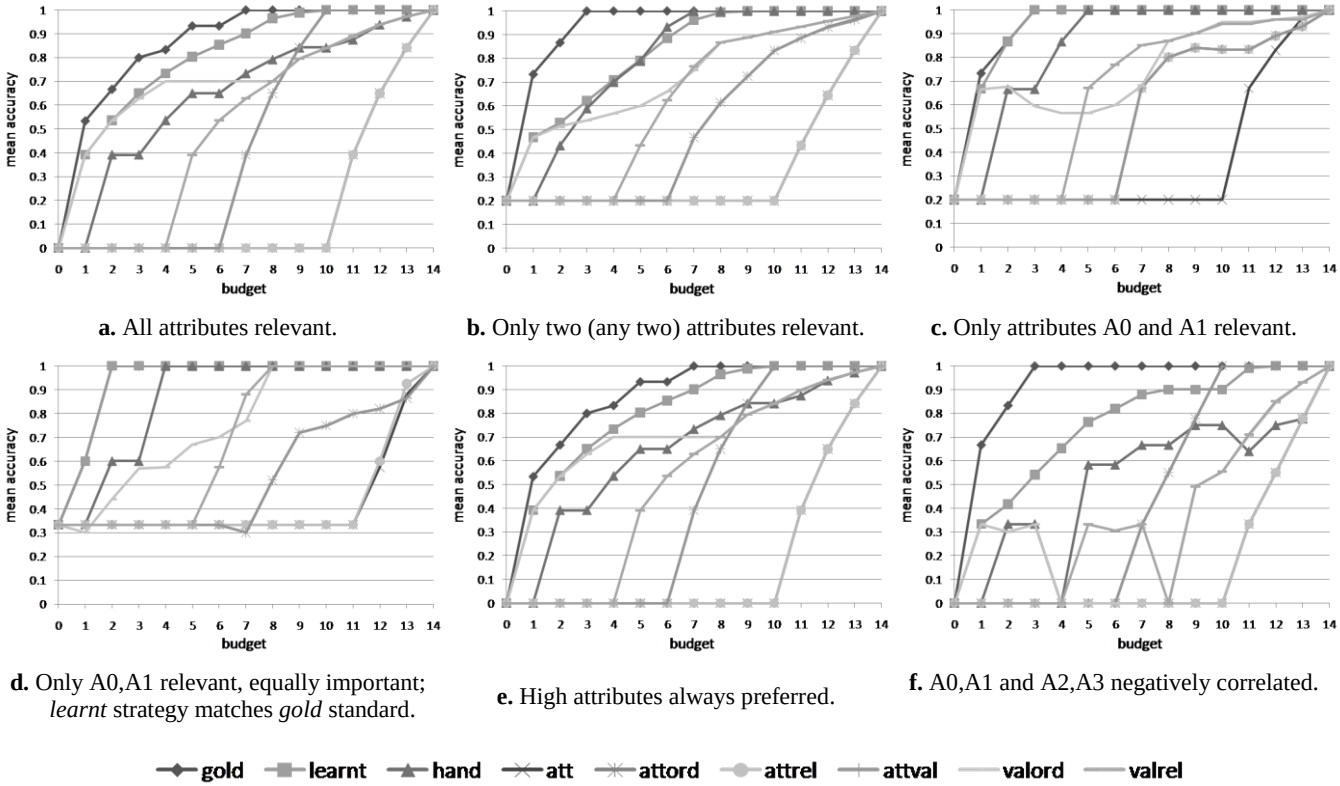
5.5 Experimental Results

Over all the datasets, the results predominantly supported our hypotheses (Figure 1).[4] The learned strategy outperformed all the handcrafted strategies (i.e., achieved higher accuracy for the same budget and achieved 100% accuracy sooner) for all the datasets except under two conditions. First, for the dataset where any two attributes were relevant (Figure 1b), the handcrafted decision tree performed better for budgets ≥ 6. This may be explained by the fact that the handcrafted decision tree explicitly tests for attribute relevance first. This is particularly valuable when there are irrelevant attributes but is inefficient when all attributes are relevant (Figure 1a). Second, when negatively correlated attributes were involved (Figure 1f), the *attord* static strategy preferring attribute-ordering questions reached 100% accuracy sooner. Negatively correlated attributes present a particular challenge in that if two attributes are negatively correlated and a model has the same preferred value direction for both, if a value-ordering question is only asked for the less important attribute, the resulting model will predict incorrect preferences in cases where objects differ in value for the more important attribute. However, the results show that this is an even greater problem for the other strategies than for the learned strategy, which is only outperformed at high budgets. Also interesting to note about these graphs is the nonmonotonic behavior of the performance plots for some of the static strategies (Figure 1c,f), providing evidence of the non-additive nature of question utility.

The results also showed that L-QUAIL learned question-asking strategies adapted to unique characteristics of the objects and models in each dataset. For example, Figure 2 shows the trees learned for datasets where a specific two of the four attributes were relevant (see Table 3 for the corresponding complete questions). Each tree only asked value-ordering questions for the relevant attributes and in the case where the two attributes were equally important (Figure 2c), the tree did not ask any attribute-ordering questions.

Figure 3 shows another learned strategy, where the models fell into two different classes—those where only attributes A0 and A1 were relevant and those where only A2 and A3 were relevant. Interestingly, the attribute-relevance questions are never asked—they are in the tree but their answers are implied by previous answers, illustrating a situation where the learned tree benefits from taking previous answers into consideration. The learned tree is not particularly intuitive but results in much better performance than the handcrafted strategies.

[4] Space constraints preclude the presentation of the complete results; the rest are consistent with the findings reported here.



a. All attributes relevant.

b. Only two (any two) attributes relevant.

c. Only attributes A0 and A1 relevant.

d. Only A0,A1 relevant, equally important; *learnt* strategy matches *gold* standard.

e. High attributes always preferred.

f. A0,A1 and A2,A3 negatively correlated.

Figure 1. Results for different datasets showing learned strategy (*learnt*) generally outperforming handcrafted strategies.

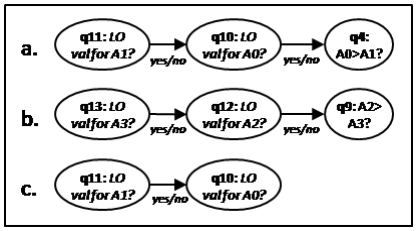


Figure 2. Decision trees learned for datasets with (a) only A0 and A1 relevant, (b) only A2 and A3 relevant, and (c) only A0 and A1 relevant and also equally ranked.

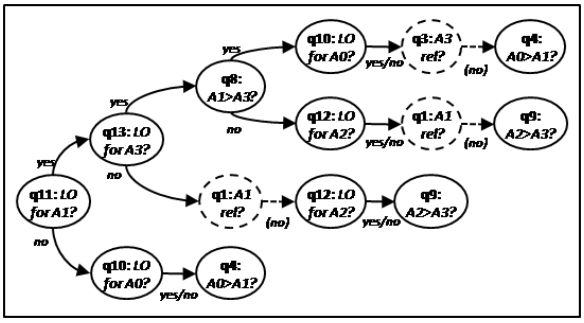


Figure 3. Decision tree learned for dataset where models only cared about attributes A0 and A1 or attributes A2 and A3.

Although our experiments involved a comprehensive set of questions with which CHARM could learn a perfect model, the results should also apply to situations involving questions providing only partial coverage and even to situations involving base learners that may imperfectly incorporate answers. Because L-QUAIL learns over a learner's actual performance with question answers, it acquires question-asking strategies naturally adapted to different situations.

6. RELATED WORK

Asking questions to obtain the most useful information for a learner has much in common with *active learning* [[2][3][6]]. Most active learning work focuses on acquiring class labels for the most informative examples. An extension, *active feature selection* [11] attempts to obtain information about feature relevance. In contrast, QUAIL obtains information of different kinds through a variety of questions, some (like attribute relevance) of which may result in direct modifications to the hypothesis space.

Proactive learning augments active learning with decision-theoretic techniques to accommodate imperfect oracles [4]. Because L-QUAIL learns over performance numbers with the answers rather than from the answers themselves, to some extent it already accommodates imperfect oracles and even imperfect learners. *Active model selection* attempts to find an optimal sequence of tests on learned models with different costs for the purpose of identifying the model with the highest expected accuracy [10]. While active model selection addresses budgetary constraints on testing or evaluation during learning, L-QUAIL addresses budgetary constraints during performance.

Games with a purpose (e.g., [13][14]) involve asking questions in a manner specifically designed to entice humans to provide

answers to fill knowledge gaps. QUAIL differs in its specific focus on asking questions to help a learner learn. This paper contributes a learning approach to finding effective question-asking strategies for supervised learners.

7. FUTURE WORK

Our experimental results, although promising, investigated a synthetic domain and relied on exhaustively generating performance data for all possible question subsets. More realistic settings will require CHARM's performance to be assessed during L-QUAIL's learning. Because CHARM will only have to be run over the subset of models at a node and learning typically terminates before all questions are asked, this approach should be less expensive than exhaustive generation. However, some method to further reduce the set of candidate questions at each node is likely needed.

L-QUAIL's greedy heuristic for question selection does not yet fully address the non-additive nature of question utilities. L-QUAIL remains susceptible to sometimes missing questions that individually lead to small performance improvements but together lead to large gains, and to selecting useless questions when none lead to immediate improvement. We are considering two possible solutions to this problem: 1) limited lookahead for considering small sets of questions rather than single questions, and 2) random decision trees enabling less immediately useful questions to be selected with some probability.

The experiments in this paper only considered questions of equal cost. Questions with differing costs mean learning must factor in question cost when considering the next question to ask—for example, by considering sets of questions of the same total cost rather than individual questions. Within a fixed budget scenario, the resulting tree can be used directly for question asking. But for a flexible budget scenario, something like a conditional decision tree that selects questions based on the remaining budget would be needed.

Our experiments also assume access to some pool of target models together with their corresponding question answers and preference judgments over the objects of interest. Although a large pool of training data clearly benefits learning, obtaining such data comes at its own cost. *Cost-sensitive decision trees* [5], which trade off the cost of obtaining better utility estimates against the value of acquiring a higher-accuracy tree, may help in solving this problem.

8. SUMMARY AND CONCLUSIONS

We presented a learning approach to acquiring effective question-asking strategies under constrained budget scenarios. The approach addresses the issue of context-dependent question utilities and partially addresses the issue of non-additive utilities. We formulated question-asking strategies as decision trees and presented a decision tree learning algorithm for acquiring such strategies. We presented experiments exploring the effects of different target model populations and object distributions on L-QUAIL, testing its ability to learn decision trees adapted to different datasets. The results of our experiments provided strong support for our hypotheses regarding L-QUAIL's ability to adapt to different conditions and to generally outperform handcrafted strategies. By relying on a base learner's actual performance with the answers in the situations it is likely to encounter, our learning approach avoids the costly hand-engineering of effective question-asking strategies for different learners while providing question-asking strategies that are naturally adapted to specific base learners and their environments.

9. ACKNOWLEDGMENTS

This work was supported by the Defense Advanced Research Projects Agency and the United States through BBN Technologies Corp. on contract number FA8650-06-C-7606. We thank Omid Madani for helping us formulate the learning problem, and Fusun Yaman for extending CHARM to integrate QUAIL.

10. REFERENCES

[1] Burstein, M., Laddaga, R., McDonald, D., Cox, M., Benyo, B., Robertson, P., Hussain, T., Brinn, M., & McDermott, D. (2008). POIROT - integrated learning of web service procedures. *Proc. AAAI.* Chicago, IL.

[2] Cohn, D., Atlas, L., & Ladner, R. (1994). Improving generalization with active learning. *Machine Learning, 15* (2), 201-221.

[3] Dasgupta, S., Hsu, D., & Monteleoni, C. (2007). A general agnostic active learning algorithm. *Proc. NIPS.* Vancouver, B.C., Canada.

[4] Donmez, P., & Carbonell, J. (2008). Proactive learning: cost-sensitive active learning with multiple imperfect oracles. *Proc. CIKM.* Napa Valley, CA.

[5] Esmeir, S., & Markovitch, S. (2008). Anytime induction of low-cost, low-error classifiers: a sampling-based approach. *JAIR*.

[6] Freund, Y., Seung, H. S., Shamir, E., & Tishby, N. (1997). Selective sampling using the query by committee algorithm. *Machine Learning, 28*, 133-168.

[7] Gervasio, M., & Myers, K. (2008). Question asking to inform procedure learning. *Proc. AAAI Workshop on Metareasoning: Thinking about Thinking.* Chicago, IL.

[8] Gervasio, M., Myers, K., desJardins, M., & Yaman, F. (2009). Question asking to inform preference learning: a case study. *Proc. AAAI Spring Symposium on Agents that Learn from Human Teachers.* Stanford, CA.

[9] Kellerer, H. P. (2005). *Knapsack Problems.* Springer Verlag.

[10] Madani, O., Lizotte, D., & Greiner, R. (2004). Active model selection. *Proc. UAI.* Banff, Canada.

[11] Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*, 81-106.

[12] Raghavan, H., Madani, O., & Jones, R. (2006). Active learning with feedback on features and instances. *Journal of Machine Learning Research*.

[13] Speer, R., Krishnamurthy, J., Havasi, C., Smith, D., Lieberman, H., & Arnold, K. (2009). An interface for targeted collection of common sense knowledge using a mixture model. *Proc. IUI.* Sanibel Island, FL.

[14] von Ahn, L., Kedia, M., & Blum, M. (2006). Verbosity: a game for collecting common-sense knowledge. *Proc. CHI*. Montreal, Quebec, Canada.

[15] Yaman, F., Walsh, T. J., Littman, M. L., & desJardins, M. (2008). Democratic approximation of lexicographic preference models. *Proc. ICML*. Helsinki, Finland.