Adaptive goal recognition

Neal Lesh*
Department of Computer Science and Engineering
University of Washington, Seattle, WA 98195
neal@cs.washington.edu
(206) 616-1849 FAX: (206) 543-2969

Abstract

Because observing the same actions can warrant different conclusions depending on who executed the actions, a goal recognizer that works well on one person might not work well on another. Two problems that arise in providing user-specific recognition are how to consider the vast number of possible adaptations that might be made to the goal recognizer and how to evaluate a particular set of adaptations. For the first problem, we evaluate the use of hillclimbing to search the space of all combinations of an input set of adaptations. For the second problem, we present an algorithm that estimates the accuracy and coverage of a recognizer on a set of action sequences the individual has recently executed. We use these techniques to construct *Adapt,* a recognizer-independent unsupervised-learning algorithm for adapting a recognizer to a person's idiosyncratic behaviors. Our experiments in two domains show that applying Adapt to the BOCE recognizer can improve its performance by a factor of two to three.

1 Introduction

Goal recognition (e.g. [Kautz, 1987; Carberry, 1990]) is the task of inferring a person's intentions given a partial view of their actions. As noted in [Maes and Kozierok, 1993; Bauer, 1994; Ardissono and Cohen, 1995], goal recognition is difficult, in part, because people do not all act alike. The same actions can warrant different conclusions depending on who executed them. For example, when I go to the College Inn Cafe the waitress invariably brings me my usual order, scrambled eggs and rye toast, without asking me what I want. With a few exceptions discussed in section 7, previous approaches can provide user-specific recognition only by having a human expert hand-tune the recognizer for each person.

*Many thanks to Sandra Carberry, Oren Etzioni, Keith Golden, Steve Hanks, Anna Karlin, Nick Kushmerick, Diane Litman, Dan Weld, and the reviewers of the submission draft of this paper for comments and discussion. This research was funded in part by Office of Naval Research grant 92-J-1946, by ARPA / Rome Labs grant F30602-95-1-0024, by a gift from Rockwell International Palo Alto Research, and by National Science Foundation grant IRI-9357772.

We present the *Adapt* algorithm for automatically adapting a recognizer to perform well for an individual, given a goal recognizer, a set of adaptations that can be made to the recognizer, and a sample of the person's recent behavior.

We treat the person's recent behavior as training data, and attempt to find the combination of adaptations with which the recognizer performs best on the sample behavior. For tract ability, we use hillclimbing to search the space of all possible combinations of adaptations. The primary challenge we address is how to evaluate the performance of the input goal recognizer with a candidate set of adaptations on the input sample data. The input data is not annotated with the person's actual goals.

We perform unsupervised learning by treating goals as verifiable predictions of future behavior, as opposed to mental state [Pollack, 1990] or explanations [Hobbs *et al,* 1988]. For example, if the recognizer indicates that a computer user's goal is to delete all her backup files, then we view the recognizer's output as predicting that the user will delete all her backup files. This goal-prediction is correct if she does go on to delete her backup files.

We empirically evaluate Adapt on the BOCE goal recognizer [Lesh and Etzioni, 1995; 1996]. Our experiments in two domains show that applying Adapt to the BOCE recognizer can improve its performance by a factor of two to three.

We use two metrics to gauge the quality of recognition: *accuracy,* the probability a recognizer's inferences are correct, and *coverage,* the probability the recognizer will draw an inference. We formally prove a tradeoff exists between accuracy and coverage in goal recognition.

Our work makes the following contributions:

1. We present a framework for formally characterizing and comparing the performance of various goal recognizers.
2. We present an unsupervised algorithm for training a goal recognizer on samples of action sequences that:
   - Learns to filter out spurious actions.
   - Adapts to the particular distribution of goals that an individual pursues.
   - Learns in the presence of noise, such as if the observed person occasionally abandons tasks.

In section 2, we define the adaptive goal recognition problem. In section 3, we prove there is a tradeoff between accuracy and coverage. In section 4, we present a function which estimates accuracy and coverage. In section 5, we present the Adapt function. In section 6, we describe our experiments. In sections 7 and 8, we discuss related work, future work, and conclusions.

2 Formulation

We now define the adaptive goal recognition problem.

2.1 Basic definitions

Let Q be the set of all persons, $\mathcal{G}$ be the set of all possible goals, and $\mathcal{A}^*$ be the set of all action sequences any person might execute. The *exec* and *goal* functions describe the two relevant relationships among persons, goals, and action sequences. Let *exec(q, A)* hold iff person *q* is observed to execute action sequence *A*. Let *goal(q,g)* hold iff person *q* has goal *g*. For simplicity, in this paper, we assume people have one goal at a time.

A *goal recognizer* maps an action sequence to either a goal *g*, indicating *g* is the person's goal, or *nil*, indicating the recognizer can not determine the person's goal given the current observations. Formally, a *goal recognizer* is a function from $\mathcal{A}^*$ to $\mathcal{G} \cup \{nil\}$. While this definition is fairly broad, it does not allow for recognizers that output a probability distribution over goals (e.g. [Pynadath and Wellman, 1995]).

Samples of people's behavior are stored as episodes. An *episode* is a pair *(A, S)* where *A* is an action sequence the person executed and *S* is the state of the world at the time the person began executing those actions.

2.2 Adaptable recognizers

We refer to adjustments that can be made to a goal recognizer as *adaptations.* Notation ally, for any set of adaptations T, let R_T be recognizer R with adaptations T. Different types of adaptations will be appropriate for different goal recognizers. We now provide a simplified description of the BOCE goal recognizer [Lesh and Etzioni, 1995; 1996] and two adaptations for BOCE.

BOCE recognizes goals based on an input set of action schemas *ACT,* a set of goal predicates $\mathcal{PRED}$, and a set of background goals *BQ.* BOCE composes the actions in *ACT* into candidate plans and the predicates in $\mathcal{PRED}$ into candidate goals based on assumptions about what *type* of plans and goals people have. In particular, BOCE assumes (1) the person's goal is a conjunction of the input predicates, and (2) every action in the person's plan will either enable another action in that plan or enable the person's goal. The background goals in *BQ* arc used to filter out actions that are not causally connected to the person's goal. For example, if *BQ* contains the goal "have cash" then BOCE will essentially disregard observing a person stop at an automatic teller machine rather than conclude that getting cash is in service of her current goal.

Two types of adaptations available for BOCE, which are discussed in greater detail in section 6, are:

- **Add/remove background goal to $\mathcal{BG}$:** Adding a background goal makes BOCE less sensitive to noise.
- **Add/remove goal predicate to $\mathcal{PRED}$:** Adding a predicate increases the number of candidate goals. This can improve accuracy but can also delay recognition because there are more goals to rule out.

Many other types of adaptations are possible. For example, an adaptation might add or remove a precondition or effect from the domain operators. This could potentially improve recognition by allowing the goal recognizer to build plans based on what preconditions and effects the observed person *believes* actions have. In the widely used framework for plan recognition described in [Kautz, 1987], one possible adaptation is to add or remove an *abstraction* link between nodes in the plan hierarchy. Another adaptation in Kautz's framework is adding or removing ordering constraints between steps in the plan hierarchy. Finally, the notion of background goals could be applied to almost any goal recognizer as a way of filtering out actions that typically reduce the performance of that recognizer.

2.3 Analytic model of recognition

We now present terms that allow us to analyze, evaluate, and compare goal recognizers.

The *coverage* of recognizer R is the probability that R will not return nil. Let domain $\mathcal{D}_R$ be all action sequences $\mathcal{A}_i \in \mathcal{A}^*$ such that $R(\mathcal{A}_i) \neq nil$. The *coverage* of recognizer R on person q is:

$$COV(R,q) = \sum_{\mathcal{A}_i \in \mathcal{D}_R} P(\text{exec}(q, \mathcal{A}_i))$$

A recognizer's *accuracy* is the probability that it will correctly identify the person's goal given that it produces a goal. Formally, $ACC(R,q)$, is: 0 if $COV(R,q) = 0$, else

$$\sum_{\mathcal{A}_i \in \mathcal{A}^*} \frac{P(\text{goal}(q, R(\mathcal{A}_i)) \mid \text{exec}(q, \mathcal{A}_i)) \times P(\text{exec}(q, \mathcal{A}_i))}{COV(R,q)}$$

A *score function* maps the accuracy and coverage of a recognizer to a single value. For example, a simple score function is $(5 \times \textit{accuracy}) + \textit{coverage}$.

Different score functions will be appropriate for different applications. An application that uses goal recognition to offer to complete people's tasks might require very high accuracy to avoid bothering people with inappropriate assistance. On the other hand, a security system using a recognizer to detect potentially dangerous behavior might sacrifice accuracy for coverage if it could tolerate false positives but did not want to fail to alert the authorities to any potentially dangerous behavior.

The above definition of accuracy measures the probability of the recognizer being correct on the portion of the behavior that the recognizer covers. The score function $\textit{accuracy} \times \textit{coverage}$ ranks recognizers according to their absolute-accuracy, or the probability of the recognizer correctly identifying the person's goal. For many applications, however, *accuracy* × *coverage* is not a useful evaluation metric because it does not differentiate between returning the *wrong* goal and returning *no* goal.

For the remainder of the paper, when clear from context we will omit q. For example, we will use $COV(R)$ instead of $COV(R, q)$.

2.4 Adaptive goal recognition problem

As shown in figure 1, the input to Adapt is a set of episodes $\mathcal{E}$, a score function *score*, a goal recognizer R, and a set of adaptations $\mathcal{T}$. The output should be the $T_i \subseteq \mathcal{T}$ that maximizes the function $score(ACC(R_{T_i}, q), COV(R_{T_i}, q))$.

- **Given** a set of episodes $\mathcal{E}$, a score function *score*, a recognizer R, and a set of adaptations $\mathcal{T}$
- **Find** $T_i \subseteq \mathcal{T}$ that maximizes the function:

$$score(ACC(R_{T_i}, q), COV(R_{T_i}, q))$$

- **Where** q is the person who executed the action sequences in episodes $\mathcal{E}$, and R_{T_i} is recognizer R with adaptations T_i.

Figure 1: The adaptive goal recognition problem

In section 4 we present an algorithm that estimates accuracy and coverage given episodes $\mathcal{E}$. If $\mathcal{E}$ sufficiently reflects q's behavior, then the recognizer that maximizes the above equation is the best recognizer for that person under the given score function. In section 6, we examine how large $\mathcal{E}$ has to be to reflect the person's behavior.

3 Accuracy vs. coverage

A tradeoff between accuracy and coverage has been observed empirically in many fields including machine learning (e.g. [Mitchell *et al.*, 1994]) and natural language processing (e.g. [Soderland, 1997]). In this section, we prove a tradeoff exists between accuracy and coverage in goal recognition.

For a given person q, we define the *most accurate recognizer*, MAR_1, as follows. Recall that for any action sequence $\mathcal{A}$ and goal g there is some probability that g is q's goal given that q has executed $\mathcal{A}$. For example, there might be a 25% chance that q's goal is to print her resume if she has executed the Unix command `lpq -Palpha`. Let P_1 be the highest probability of *any* goal given *any* action sequence. Recognizer MAR_1 outputs goal g if the probability of g given the observed actions is P_1, otherwise MAR_1 outputs *nil*. The second most accurate recognizer, MAR_2, returns g given action sequence $\mathcal{A}$ if the probability of g given $\mathcal{A}$ is the highest *or the second highest* probability, P_2, over all goals and actions.

Formally, let P_i be the ith highest value of $P(goal(g_k) \mid exec(\mathcal{A}_j))$ for all $\mathcal{A}_j \in \mathcal{A}^*$ and $g_k \in \mathcal{G}$. The most accurate recognizer is:

$$MAR_1(\mathcal{A}_i) = \begin{cases} \text{g} & P(goal(g) \mid exec(\mathcal{A}_i)) = P_1 \\ \text{nil} & \text{otherwise} \end{cases}$$

We define MAR_n recursively given the recognizer MAR_{n-1}. Formally, for $n > 1$, $MAR_n(\mathcal{A}_i) =$

$$\begin{cases} MAR_{n-1}(\mathcal{A}_i) & \text{if } MAR_{n-1}(\mathcal{A}_i) \neq nil \\ \text{g} & \text{if } P(goal(g) \mid exec(\mathcal{A}_i)) = P_n \\ \text{nil} & \text{otherwise} \end{cases}$$

The following theorems establish the tradeoff between accuracy and coverage. Theorem 1 says MAR_1 is the most accurate recognizer. Theorems 2 and 3 state that $\mathrm{MAR}_1, \mathrm{MAR}_2, \ldots$ are ordered in terms of decreasing accuracy and increasing coverage. Theorem 4 states that these recognizers represent the maximal points along the tradeoff between accuracy and coverage.

Theorem 1 (Most accurate recognizer) *For all recognizers* R_j, $ACC(R_j) \leq ACC(MAR_1)$.

Theorem 2 (Decreasing accuracy) *For all* $i \geq 1$, $ACC(MAR_i) \geq ACC(MAR_{i+1})$.

Theorem 3 (Increasing coverage) *For all* $i \geq 1$, $COV(MAR_i) \leq COV(MAR_{i+1})$.

Theorem 4 (Tradeoff) *For all* $i \geq 1$, *there does not exist any recognizer* R *such that both* $ACC(R) > ACC(MAR_i)$ *and* $COV(R) > COV(MAR_i)$.

For brevity we omit the proofs, which will appear in the author's thesis.

4 Estimating accuracy and coverage

We now present an algorithm which estimates the accuracy and coverage of a recognizer on a person given a sample of that person's behavior.

Estimate takes a recognizer R and a set of episodes $\mathcal{E}$ and returns two real numbers in the range [0,1] which are the estimated accuracy and coverage of R on the person whose behavior is described by $\mathcal{E}$

Estimate treats goals as *predictions* of future behavior. Suppose that after observing six actions in a fifteen action sequence the recognizer reports that the person's goal is to compress all large files. This predicts the person will compress all large files. This prediction is correct if the person compresses all large files, i.e. if the remaining nine actions serve to complete the goal.

To implement this idea, we need a means to determine if a given action sequence satisfies a given goal. The procedure *Achieves(A,S,g)* returns true iff action sequence A satisfies goal g assuming that it was executed from initial state S. Our implementation of Achieves returns true iff g does not hold in S but g does hold in state S' where S' is the state reached by executing sequence A in state S.[1] Most planning languages provide a similar def-

[1] A problem with this Achieves function is that it allows up to $|\mathcal{A}| - 1$ irrelevant actions to be executed in service of achieving a goal, and thus might highly reward a goal

inition for *Achieves,* such as the Modal Truth Criterion [Chapman, 1987] for Strips [Fikes and Nilsson, 1971].[2]

We assume the recognizer will be called after each observed action until it infers a goal. For each episode *(A,S),* the recognizer is fed incrementally longer prefixes of *A* until it produces some goal or outputs *nil* when given *A.* If the recognizer does produce a goal, we then determine if *A* achieves that goal from state *S.* In the following pseudo code, the *Inputs* variable counts the number of calls to the recognizer and *Inferences* and *Correct* count the number of inferences and correct inferences, respectively, made by the recognizer:
Estimate(recognizer R, episodes £)

```
1. Let $Inputs = 0$, $Inferences = 0$, $Correct = 0$.
2. For each $\langle\{a_1, ..., a_n\}, S\rangle \in \mathcal{E}$:
  (a) Let $i = 0$
  (b) WHILE $i < n$ AND $R(a_1, ..., a_i) = nil$
      - Let $i = i + 1$
      - Let $Inputs = Inputs + 1$
  (c) IF $R(a_1, ..., a_i) \neq nil$ THEN
      - Let $Inferences = Inferences + 1$
      - IF Achieves($a_1 ... a_n, S, R(a_1, ..., a_i)$)
        THEN let $Correct = Correct + 1$
3. RETURN ( accuracy = $\frac{Correct}{Inferences}$ ,
   coverage = $\frac{Inferences}{Inputs}$ )
```

The Estimate function requires up to $|\mathcal{E}|$ calls to Achieves and up to $l \times |\mathcal{E}|$ calls to recognizer R., where / is the average length of action sequences in $\mathcal{E}$. We further discuss complexity and running time in section 5.

The experiments described in section 6 indicate that good estimates can be produced from reasonably small samples of behavior. We believe that in many domains collecting sample data is relatively cheap and easy. In software domains, for example, the commands executed by a computer user can be recorded. A key problem is *task segmentation:* how to know when a goal-solving episode starts and stops. It may be best that this problem is handled by the goal recognizer, since it will be difficult to perform task segmentation without also performing some form of goal recognition. In this case, the goal recognizer would be fed a continuous stream of goal solving behavior and output a sequence of goals. However, in our experiments and the current formulation of the problem, we assume the person's recent behavior is already segmented prior to adaptation.

recognizer that always returns a goal that is achieved by the first action in *A.* Modifying Achieves to allow *no* irrelevant actions would be problematic in that we are interested in adapting goal recognizers to handle spurious, i.e. irrelevant actions. Our current solution is to restrict the goal recognizer to output only legitimate top-level goals, corresponding to End events in Kautz's plan hierarchies [Kautz, 1987).

[2]We do not actually need the full power of the Modal Truth Criterion since *A* is a totally ordered set of actions.

5 An adaptive goal recognizer

We now present an algorithm that tunes a given recognizer to perform well on a sample of a person's behavior.

Let T be the set of all possible adaptations. *Adapt* uses steepest ascent hill climbing to search the space of possible combinations of adaptations in set T. The starting point for the search is a recognizer without any adaptations. The *neighbors* of a recognizer are those recognizers with exactly one more, or one less, adaptation. In each iteration, we use *Estimate* to evaluate the accuracy and coverage of the current best recognizer and all its neighbors. We then reset the current search point to the recognizer with the best combination of accuracy and coverage, as determined by the input score function. We repeat the process until we encounter a recognizer with a higher score than all its neighbors, and then return this recognizer's adaptations. The pseudo code is:

```
Adapt(recognizer $R$, episodes $\mathcal{E}$, score function $s$,
      adaptations $\mathcal{T}$ )
1. Let $\mathcal{BEST} = \emptyset$
2. REPEAT
  (a) For all $t_i \in (\mathcal{T} - \mathcal{BEST})$, let $score_i$ =
      $s(Estimate(R_{(\mathcal{BEST}+t_i)}, \mathcal{E}))$
  (b) Let $m$ be the integer for which $score_m$ is the
      highest.
  (c) IF $score_m > s(Estimate(R_{(\mathcal{BEST})}, \mathcal{E}))$
      THEN let $\mathcal{BEST} = \mathcal{BEST} + t_m$.
      ELSE RETURN($\mathcal{BEST}$).
```

The Adapt algorithm requires \T\ *x k* calls to Estimate where *k* is the number of iterations until a local maximum is found. Although Adapt is computationally intensive, our notion is that it would be allowed to run overnight, perhaps once a month, to adapt to changes in a person's behavior. Furthermore, Adapt can be interrupted at any time; set *BEST* will always be a set of adaptations with a higher estimated value than any set of adaptations previously considered. We believe that a small number of adaptations will often significantly improve recognition. Finally, the Adapt algorithm is highly amenable to parallelization. In any iteration, all the calls to Estimate can be processed in parallel, and the Estimate function itself is easily parallelizable.

6 Experimental validation

In this section, we describe experiments that measure how much our adaptive techniques improve recognition.

We apply Adapt to data that contain different idiosyncratic behaviors. For example, one person might spuriously execute the Unix command date while another might spuriously execute the pwd command. Our hypothesis is that adapting the recognizer to each individual behavior will outperform applying any recognizer to all behaviors. We measure:

- Impact: What percentage of someone's work can be done by offering to complete the goal that the recognizer outputs?
- Error: How often does the recognizer produce an incorrect goal?

In all of our experiments, we use separate data for training and testing Adapt.

6.1 Spurious actions

We ran the first set of experiments on action sequences generated by Toast [Agre and Horswill, 1992], a reactive agent that solves goals such as making omelets, cleaning dirty dishes, or setting the table. To generate "noisy" behavior, we randomly insert actions into Toast's behavior, such as causing Toast to randomly add butter to a pan on the stove, or periodically wash its spatula.

The BOCE goal recognizer identifies Toast's goal by ruling out all but one of the goals Toast might have. BOCE rejects a goal if an observed action is not part of any plan to achieve the goal. For example, BOCE would reject the goal "make poached eggs" if Toast adds butter to a pan. Spurious actions can cause BOCE to reject Toast's actual goal which can, in turn, cause BOCE to either fail to infer any goal or infer the wrong goal.

Adding background goals can reduce the confusion caused by spurious actions. For example, adding a background goal "have butter in pans" will essentially cause BOCE to disregard any observations of Toast adding butter to pans. BOCE must, however, add background goals judiciously because ignoring relevant actions can delay or even prevent BOCE from identifying Toast's goal. In the extreme, if all possible background goals are added then BOCE will never output a goal.

We vary two parameters: the frequency of spurious actions and the number of distinct spurious actions. For example, if 10% of Toast's actions are either a spurious washing of the spatula or a spurious adding of butter, then the frequency is 0.1 and there are 2 distinct spurious actions. In each trial, we train BOCE on 100 episodes and test BOCE on another set of 100 episodes. The numbers reflect averages from 500 trials. We use a score function that weights accuracy 20 times as much as coverage.

In any given episode, BOCE outputs at most one goal. In our testing phase, we record the number of times BOCE makes no inference, correctly identifies Toast's goal, and incorrectly identifies Toast's goal. We also measure *plan length,* defined as the number of actions Toast executes per goal. Toast halts execution if BOCE correctly identifies, and presumably completes, its goal.

As shown in figure 2, the adapted recognizer performs significantly better than the unadapted recognizer when Toast has 3 distinct spurious actions. As the frequency of spurious actions increases, the unadapted recognizer's performance degrades rapidly. In contrast, the adapted recognizer never mistakenly attributes a goal to Toast.

As shown in figure 3, the performance of the adapted recognizer degrades as the number of distinct spurious actions increases. This happens because Adapt must add more and more background goals, which interfere with BOCE's ability to identify Toast's goals. Note that the adapted case is still far superior to the unadapted case in that it does not make any incorrect inferences.

The table in figure 4 shows that as the frequency of spurious actions increases, the impact of adapting the recognizer grows considerably.

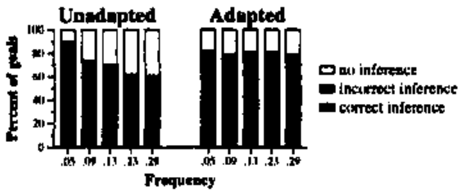

Figure 2: Learning to filter spurious actions. In each trial, there were 3 distinct spurious actions.

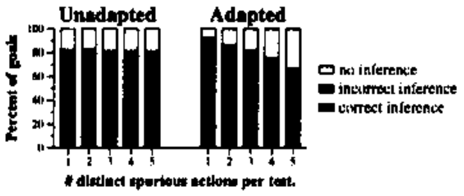

Figure 3: Learning to filter spurious actions. In these experiments, the frequency of spurious actions was 0.1.

Frequency of spurious actions	.05	.09	0.13	0.23	0.29
average plan length					
unadapted	7.5	12.5	14.4	18.3	21.4
adapted	7.2	7.8	7.9	8.4	8.95

Figure 4: Impact of adapting the recognizer: In each trial run, there were 3 different spurious actions.

The results shown in figures 2 and 4 show that Adapt improves recognition by a factor of two to three over unadapted recognition.

6.2 Goal distribution

In the second set of experiments, we simulate people whose goals are to find an object with a conjunction of properties, such as finding a computer with no users working on it that has over 32 MB RAM. We vary the probability with which goals include each predicate.

In our simulations, the person searches until she finds, or the recognizer suggests, an object that satisfies her goal. The recognizer can make one suggestion after each time the person polls an object. We measure *plan length,* the average number of objects polled by the person, and *error,* the average number of incorrect suggestions.

Adapt adjusts the recognizer by adding or removing predicates from $\mathcal{PRED}$, the set of predicates with which BOCE forms goals. If $\mathcal{PRED}$ lacks a predicate a person uses then BOCE can suggest a wrong object to the user. For example, the person might poll five computers with 32 MB RAM but if $\mathcal{PRED}$ does not contain the large.memory predicate, then BOCE would suggest a computer with small memory. On the other hand, if $\mathcal{PRED}$ contains too many predicates then BOCE will be delayed in suggesting a useful object to the person.

Figure 5 compares 18 recognizers, each described by a point where the X-coordinate is the average plan length and the Y-coordinate is the average error. A perfect recognizer would be positioned on the origin of the graph.

Points U1 to U15 represent unadapted recognizers: Ui is a recognizer where $\mathcal{PRED}$ contains *i* random predicates, out of 15 possible predicates. As the size of $\mathcal{PRED}$ increases the number of errors decrease but the plan length increases. Note that no unadapted recognizer has *both* better error and plan length than any other.

T20, T60, and T100 describe recognizers trained on 20, 60, and 100 training examples. In these experiments, a recognizer trained on 60 or 100 examples saves the person twice as much work as any unadapted recognizer with comparable error, and has much less error than any unadapted recognizer that is nearly as effective.

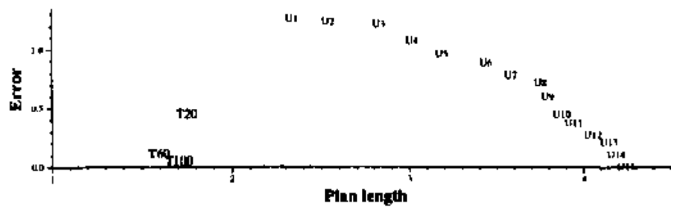

Figure 5: Learning which goals different people pursue: Points T20, T60, and T100 represent a recognizer trained on 20, 60, and 100 training examples respectively. Points Ul through U15 represent 15 different unadapted recognizers.

6.3 Adapting in the presence of noise

We also measured the effect of one type of noise in the training data: we removed the second half of the executed plan from some of the input episodes.

We ran these experiments on actions collected from Toast, as described in section 6.1. We varied the frequency with which Toast would abandon its current goal after executing half its plan. Essentially, corrupting the input data in this fashion causes Adapt to add extraneous background goals in an effort to avoid mistakes in the tainted training cases. As shown in table 6, Adapt performs reasonably well with up to 15% noise levels. We used 400 training examples in these experiments because the results were poor with 100 examples.[3] Note that although the percentage of correct inference goes down, BOCE still never makes an incorrect inference but rather outputs no goal in more cases.

7 Related work

There have been several other approaches to various aspects of the general problem of providing user-specific plan and goal recognition.

[Maes and Kozierok, 1993] apply machine learning techniques to detect recurrent patterns of behavior of computer users. Their *interface agents* learn from observing the user, as we do, but also learn from user feedback and direct training sessions. They focus on predicting the user's next action by matching the current observations to the closest previously encountered situation. In contrast, we do not specifically analyze the past behavior of the observed person but instead evaluate how well a given recognizer, with various adaptations, performs on this behavior. Note that our algorithm, Adapt, might be used to adjust the parameters of interface agents to perform better on a sample of past behavior.

[3]Gathering this many training examples may be infeasible in some domains.

Frequency of abandonment	0.0	.05	0.10	0.15	0.20
plan length	7.2	7.5	7.5	7.9	8.6
percent right	82	80	80	74	63
percent wrong	0	0	0	0	0
percent skipped	18	20	20	26	37

Figure 6: Effect of noise in training data. The frequency of spurious actions was 0.1. and there were 3 distinct spurious actions per trial run.

[Elzer *et* al., 1994; Ardissono, 1996] integrate user modeling and plan recognition to support dialogue understanding. For example, if the user model indicates that John is terrified of flying, then the plan recognizer can reject the plan of flying to Chicago when John says "I want to go to Chicago". In principle, this work could allow a plan recognizer to take advantage of the many techniques developed by the user modeling community for acquiring user preferences and beliefs. While our work has similar motivation, our approach is more direct, or "low-level", in that we find a goal recognizer which works well on a person's recent behavior rather than infer a declarative model of that person's idiosyncrasies or beliefs. Again, our approach could be used in conjunction with this work by adapting a recognizer that uses user-specific information or even adapting the *way* in which the recognizer reasons about the user model.

Our work most closely resembles that described in [Bauer, 1994; 1996]. Bauer runs a plan recognizer on a set of input episodes from a typical user (just as we do), and then gathers statistical data based on the results of running the plan recognizer on the entire observable behavior in each episode. For example, Bauer's techniques can learn that a particular computer user will save email with 80% certainty, unless the email is from her manager, in which case she deletes it with 90% certainty. This analysis enables a plan recognizer to reach the same conclusions with fewer observed actions. Although we both treat past planning episodes as training data, we learn very different information: Bauer learns statistical rules while we learn, for example, a good set of background goals. Bauer's approach is restricted to recognizers that use probabilistic information while our algorithm is restricted to recognizers that produce a single goal, whether or not they use probabilistic information. Additionally, we do not assume our recognizer will work well without adaptation. As shown in figure 2, when the frequency of spurious actions is 0.29, the unadapted recognizer correctly identifies only about 20% of the actor's goals and is wrong on about 40% of the goals. Adapt improves the recognizer so that it correctly identifies 80%

of the goals and never makes a wrong prediction.

[Mooney, 1990] employs explanation-based learning to add new plans to the plan library when a person is observed to execute a plan that is not in her known repertoire. They address the question of how to generalize the new plan so that structurally similar plans can be recognized in the future. One difference between this work and ours is that our adaptations can *remove* as well as add plans and goals from the plan library.

Finally, [Caruana and Freitag, 1994] examine several variations of hillclimbing to select which attributes a concept learner should use. Attribute-selection is similar to predicate-selection, which we explore experimentally in section 6.2. They face different problems than we do in that they receive labeled training data while our algorithm performs unsupervised learning. Furthermore, adding or removing goal predicates is just one of the adaptations that Adapt considers.

8 Future work and conclusions

In future work, we will apply Adapt to more types of adaptations and to other goal recognizers. We are interested in adapting a recognizer to better reflect characteristic mistakes that people make. For example, people sometimes have flawed domain operators. We could adapt for this by introducing a new adaptation that adds or removes a precondition or effect from the domain operators that BOCE forms plans from. We hypothesize that recognition will work best when BOCE builds plans based on what preconditions and effects the observed person *believes* actions have. Another line of future work is to examine search strategies other than hillclimbing.

The primary contribution of this work is the Adapt algorithm for adapting a goal recognizer to perform well on a sample of behavior. Adapt is an unsupervised-learning algorithm in that the sample data is not annotated with the person's actual goals. Adapt is a recognizer-independent algorithm in that it can be applied to any goal recognizer for which there is an available set of adaptations. Our experiments in two domains show that applying Adapt to the BOCE recognizer can improve its performance by a factor of two to three.

References

[Agre and Horswill, 1992] P. Agre and I. Horswill. Cultural support for improvisation. In *Proc. 10th Nat. Conf, on AI,* pp. 363-368, 1992.

[Ardissono and Cohen, 1995] L. Ardissono and R. Cohen. On the value of user modeling for improving plan recognition. In *Proceedings of the workshop on The Next Generation of Plan Recognition Systems: Challenges for and Insight from Related Areas of AI,* pp. 8-12, August 1995.

[Ardissono, 1996] Liliana Ardissono. *Dynamic User Modeling and Plan Recognition in Dialogue.* PhD thesis, Department of Information, University of Torino, Italy, 1996.

[Bauer, 1994] M. Bauer. Quantitative modeling of user preferences for plan recognition. In *Proceedings of the Fourth International Conference on User Modeling,* pp. 73-78, 1994.

[Bauer, 1996] M. Bauer. Acquisition of user preferences for plan recognition. In *Proceedings of the Fifth International Conference on User Modeling,* pp. 105-112, January 1996.

[Carberry, 1990] S. Carberry. Incorporating default inferences into plan recognition. In *Proc. 8th Nat. Conf. on AI,* vol. 1, pp. 471-8, July 1990.

[Caruana and Freitag, 1994] Rich Caruana and Dayne Freitag. Greedy attribute selection. In *Proc. 11th Int. Conf. on Machine Learning,* pp. 28-36, 1994.

[Chapman, 1987] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence,* 32(3):333-377, 1987.

[Elzer *et al,* 1994] Stephanie Elzer, Jennifer Chu-Carroll, and Saundra Carberry. Recognizing and utilizing user preferences in collaborative consultation dialogues. In *Proceedings of the Fourth International Conference on User Modeling,* pp. 19-24, 1994.

[Fikes and Nilsson, 1971] R. Fikes and N. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence,* 2(3/4), 1971.

[Hobbs *et al.,* 1988] J.R. Hobbs, M. Stickel, P. Martin, and D. Edwards. Interpretation as abduction. In *Proc. of Annual Meeting of the Association for Computational Linguistics,* pp. 32 37, 1988.

[Kautz, 1987] H. Kautz. *A Formal Theory Of Plan Recognition.* PhD thesis, University of Rochester, 1987.

[Lesh and Etzioni, 1995] Neal Lesh and Oren Etzioni. A sound and fast goal recognizer. In *Proc. 14th Int. Joint Conf. on AI* pp. 1704-1710, 1995.

[Lesh and Etzioni, 1996] Neal Lesh and Oren Etzioni. Scaling up goal recognition. In *Proc. 5th Int. Conf. on Principles of Knowledge Representation and Reasoning,* pp. 178 189, 1996.

[Maes and Kozierok, 1993] Pattie Maes and Robyn Kozierok. Learning interface agents. In *Proceedings of AAAI-93,* 1993.

[Mitchell *et al.,* 1994] Tom Mitchell, Rich Caruana, Dayne Freitag, John McDermott, and David Zabowski. Experience with a learning personal assistant. *C. ACM,* 37(7):81-91, 1994.

[Mooney, 1990] Raymond J. Mooney. Learning Plan Schemata From Observation: Explanation-Based Learning for Plan Recognition. *Cognitive Science,* pp. 483-509, 1990.

[Pollack, 1990] M. Pollack. Plans as complex mental attitudes. In P. Cohen, J. Morgan, and M. Pollack, eds., *Intentions in Communication,* pp. 77-101. MIT Press, Cambridge, MA, 1990.

[Pynadath and Wellman, 1995] D.V. Pynadath and M.P. Wellman. Accounting for context in plan recognition, with application to traffic monitoring. In *Proc. 11th Conf. on Uncertainty in Artifical Intelligence,* pp. 472-81, August 1995.

[Soderland, 1997] S. G. Soderland. *Learning Text Analysis Rules for Domain-Specific natural language processing.* PhD thesis, University of Massachusetts, 1997.