From *Proceedings of the Third International Workshop on Cooperative Information Agents.* Uppsala, Sweden: Springer (1999).

An Adaptive Conversational Interface for Destination Advice

Pat Langley,[1] Cynthia Thompson,[2]
Renée Elio,[3] and Afsaneh Haddadi[4]

[1] DaimlerChrysler Research & Technology Center
1510 Page Mill Road, Palo Alto, CA 94304 USA
[2] Center for the Study of Language and Information
Stanford University, Stanford CA 94305 USA
[3] Department of Computing Science, Assiniboia Hall
University of Alberta, Edmonton, Alberta T6G 2H1
[4] DaimlerChrysler Research and Technology
Alt-Moabit 96a, 10559 Berlin, Germany

Abstract. In this paper, we describe the Adaptive Place Advisor, a conversational interface designed to help users decide on a destination. We view the selection of destinations as an interactive process of constraint satisfaction, with the advisory system proposing attributes and the human responding. We further characterize this task in terms of heuristic search, which leads us to consider the system's representation of problem states, the operators it uses to generate those states, and the heuristics it invokes to select these operators. In addition, we report a graphical interface that supports this process for the specific task of recommending restaurants, as well as two methods for constructing user models from interaction traces. We contrast our approach to recommendation systems with the more common scheme of showing users a ranked list of items, but we also discuss related work on conversational systems. In closing, we present our plans to evaluate the Adaptive Place Advisor experimentally and to extend its functionality.

1 Introduction

As society becomes more complex, humans are confronted with ever more alternatives in their activities. Today, we have more news to hear, more books to read, more songs to appreciate, and more places to eat than ever before. In addition, access to the World Wide Web has led to substantial growth in the number of information sources. Each new option gives people a greater variety of choices, but the sheer number of alternatives often makes an intelligent choice impossible without some computational assistance.

In response to this need, there have been increased efforts to design and implement intelligent aides for filtering web sites (e.g., Pazzani, Muramatsu, & Billsus, 1996), news stories (e.g., Lang, 1995), and other information sources. A related line of research and development has led to *recommendation systems*, which are not limited to filtering information but can be used for any task that requires choice among a large set of predefined items. Most research in this area has built on the literature in document retrieval, which assumes a specific approach to choice and a particular type of interaction.

However, when developing any intelligent system, especially one that will interact with humans, it seems important to make one's design decisions carefully. In this paper, we describe a *conversational* approach to recommendation systems that diverges from those based on information retrieval. The next section characterizes the traditional framework and its drawbacks, then outlines an alternative that appears to hold some advantages. After this, we present the Adaptive Place Advisor, a prototype system that incorporates this approach to recommendation. The conversational framework also supports user models at a finer grain than the usual item level, and we consider two ways of learning such models from interaction traces. Finally, we discuss related work on conversational interfaces and suggest some directions for future research.

2 Two Approaches to Recommendation Systems

As we have noted, one can view many decision-support tasks in terms of making *recommendations*. A recent workshop on this topic indicates the variety of real-world problems that one can formulate as recommendation tasks, including the selection of videos (Nguyen & Haddawy, 1998), television programs (Das & ter Horst, 1998), and books (Mooney, Bennett, & Roy, 1998). In generic terms, we can characterize the task as:

> Given a large set of items and a description of the user's needs, present to the user a small set of the items that are suited to these needs.

This statement covers all of the examples mentioned above, along with many others. However, note that our task statement is intentionally abstract, in that it does not specify certain details, such as how the user communicates his needs, the manner in which results are presented, or even the number of recommended items. Any specific approach to developing recommendation systems must take a position on these design issues.

We can illustrate this point by describing the most widely used framework for recommendation systems, which we will call the *ranked list* approach. In this scheme, the user specifies his needs with one or more keywords and the system presents an (often) long list of results, ordered by their predicted relevance to the user. This technique has a long history in information retrieval, where it was originally designed to help find documents or reports during library research. However, the basic method is quite general, and it underlies most search engines for the World Wide Web, which millions now use regularly for many quite different selection tasks.

Yet despite its current popularity, the ranked list scheme is not the only approach to making recommendations. We need only look to conversations between humans for a very different model of this process. Consider a situation in which one person, who we will call the *enquirer*, asks another person, who we will call the *advisor*, for assistance in deciding on a restaurant at which to have dinner:

Enquirer: Where do you think I should eat tonight?
Advisor: Well, what type of cuisine would you like?
Enquirer: What types are there?
Advisor: Some examples are Chinese, Indian, and Mediterranean.
Enquirer: Oh, maybe Indian.
Advisor: What quality rating would you like?
Enquirer: I don't care, as long as it's cheap.
Advisor: How do you want to pay?
Enquirer: No, I think I'd like Chinese instead.
Advisor: Okay, we'll switch to Chinese food.
What city do you prefer?
Enquirer: How about Berkeley?
Advisor: I know three cheap Chinese restaurants in Berkeley.
One is the Long Life Vegi House on 2129 University Avenue.
Does that sound alright?
Enquirer: Sure, that sounds fine.

Clearly, one could develop recommendation systems that attempt to mimic this type of human interaction, rather than relying on keyword searches and ranked lists. We will refer to such systems as *conversational* interfaces.

The interaction supported by conversational systems seems quite different from that found in the ranked list approach. The most important distinction is that the enquirer never hears about a complete item until only one, or at most a few, choices remain. Rather than being overwhelmed with items that compete for his attention, he interacts with the advisor to narrow down the choices in an iterative, manageable fashion. This interaction takes the form of a sequence of questions, most designed to eliminate some items from consideration. Answering these questions plays a similar role to giving keywords with the ranked list scheme, but the aim is remove alternatives rather than to simply order them. The conversational process can also help the enquirer better understand his own desires, since thinking about possible questions and answers may clarify goals in ways a ranked list does not.

Clearly, such dialogues seem better for recommendations that must be delivered by sound rather than visually. This makes the conversational approach well suited not only for advice between humans, but also for computer interfaces that must rely on speech, such as ones used while the enquirer is driving. However, they also seem ideal, independent of modality, for tasks like restaurant and movie selection, in which the user needs to converge on at most a few items. On the other hand, ranked list methods seem more appropriate for tasks like the selection of web pages or news stories, in which the user may well want to examine many options. In the next section, we describe the Adaptive Place Advisor, a computational assistant that takes this conversational approach to recommendation.

3 The Adaptive Place Advisor

We are interested in developing a conversational interface that has broad applicability to recommendation tasks, but our initial work has focused on a particular class of problems – *destination selection* – that seems especially relevant to drivers. Our prototype system, the Adaptive Place Advisor, aims to help the user select a physical location that is relevant to his goals. Again, the basic approach involves carrying out a conversation with the user to identify a place that matches his needs at the time.

We view this conversational process in terms of heuristic search, similar to constraint satisfaction in that it requires the successive addition of constraints on solutions, but also analogous to game playing in that the user and system take turns. Formulating the problem within a search framework means we must represent not only items, as in ranked lists, but also states of the conversation, as well as operators for advancing the conversation and heuristics for deciding which operator to apply on each step. This treatment also has some important implications for the modeling of user preferences. In this section, we discuss each of these issues in turn.

Our approach to destination advice draws heavily on an earlier analysis of the task by Elio and Haddadi (1998, 1999), which itself borrows ideas from linguistic research on speech acts (e.g., Searle, 1969). One difference between our formulations is that the previous work distinguishes between search through a task space and a dialogue space, whereas we aggregate these into search through a combined space. Another distinction is that user intentions plays a central role in their framework, whereas our less sophisticated approach sidesteps intentions by specifying system actions that are appropriate to different types of user responses. Nevertheless, both analyses view decision making as a process of successively refining constraints, which separates them from the ranked-list scheme.

3.1 Representation of Items and States

Many recommendation systems, since they focus on retrieving documents or Web pages, represent each item as a 'bag of words', that is, in terms of the words in the text that describes each item. In contrast, the Adaptive Place Advisor assumes a more constrained representation, similar to that found in a relational database. The system stores each item as a conjunction of attribute-value pairs, each specifying information likely to interest the user. For restaurants, these characteristics include attributes like their location, the cuisine served, the hours of operation, the method of payment, the price range, the availability of parking, requirements for reservations, and the name.

However, representing an item like a restaurant does not equate to describing the state of a conversation about such items. To encode conversational states, we must allow *partial* descriptions of items, which the Place Advisor specifies as a subset of attribute-value pairs. For example, after receiving answers to a few questions, the system may have determined that the user would prefer to eat someplace in Palo Alto that serves Thai cuisine and that does not require reservations. Earlier in the dialogue, this specification will tend to be less constrained, whereas later states will tend to include more attribute values. We can view each such partial description as a database query that has an associated, but implicit, representation consisting of all restaurants that match the query.

But this tells only half the story, since the Place Advisor must also represent its knowledge about a conversation's history. This includes information about attributes that remain unasked, that the system has asked but the user has indicated are undesired, and that the user has indicated are important enough to be fixed. The system must also keep track of attributes and values it has mentioned in response to a user query, as well as complete items the user has rejected as unacceptable. Most important, it must represent the dialogue operator(s) indicated by the user's most recent utterance, since these determine the system's appropriate response.

3.2 Dialogue Operators

Our view of conversational recommendation as search also requires us to specify the operators that take steps through the search space. Following Elio and Haddadi (1998, 1999), we group conversational actions if they achieve the same effect, so that two superficially different utterances constitute examples of the same operator if they take the dialogue in the same direction. Table 1 summarizes the operators assumed by the Adaptive Place Advisor, which differ somewhat from those in the earlier analysis.

Let us first consider the operators available to the system for advancing the conversation. The most obvious, Ask-Constrain, involves asking a question to constrain items, by proposing some attribute that does not yet have a value. In our example, we saw four examples of this operator, with the advisor asking questions about the cuisine, quality of the food, payment options, and the location (city). Asking such questions is the most central activity of conversational interfaces, at least for recommendation tasks, since it determines the ways in which the system constrains items presented to the user.

Another operator, Suggest-Values, answers a user's query about possible values for an attribute. In our example, this occurred in response to the enquirer's query about cuisine. Note that, in this case, the advisor lists only a few options rather than all possible choices, and if we want our interface to seem natural, its answers should have a similar character. In some cases, the advisor may decide to suggest values for an attribute without an explicit request, especially if the user's response is predictable. A similar operator, Suggest-Attributes, responds to a user query about the possible characteristics of destinations.

Once the conversation has reduced the alternatives to a manageable number, the advisor must invoke Recommend-Item, an operator that proposes a complete item to the user. In the restaurant domain, this involves giving the restaurant name, address, and similar information, as we saw at the end of our sample dialogue. In some cases, the process of introducing a constraint can produce a situation in which no candidates are satisfactory. When this occurs, the advisor applies the operator Ask-Relax, which proposes dropping an attribute

Table 1. Dialogue operators supported in the Adaptive Place Advisor, with system operators in boldface and user operators in italics.

Ask-Constrain. Asks the user a question designed to constrain available candidates.

Ask-Relax. Proposes to the user that a constraint be removed to expand candidates.

Suggest-Attributes. Suggests to the user a small set of unused attributes.

Suggest-Values. Suggests to the user a small set of values for a given attribute.

Recommend-Item. Recommends to the user an item that satisfies the constraints.

Answer-Constrain. Answers a system question by giving a value for the attribute.

Reject-Constrain. Refuses to answer a system question by rejecting the attribute.

Accept-Relax. Accepts a system suggestion to relax a constrained attribute.

Reject-Relax. Rejects a system suggestion to relax a constrained attribute.

Replace-Attribute. Replaces a system question with a different attribute.

Accept-Item. Accepts proposed destination and ends the conversation.

Reject-Item. Rejects proposed destination and requests another recommended item.

Query-Attributes. Asks system for information about possible attributes.

Query-Values. Asks system for information about possible values of an attribute.

to expand the candidate set. A less drastic response, which we have not implemented, would replace an attribute's value with another having a similar effect.

Now let us turn to the operators that the system assumes are available to the human user. The most central action the user can take, Answer-Constrain, involves answering a question so as to specify the value of some attribute. Our example included two instances of this operator, in response to questions about cuisine and city. Each such answer further constrains the items the system considers for presentation to the user, and thus advances the dialogue toward its goal of identifying a few recommended restaurants.

But the Place Advisor does not assume the user will always answer its questions. If the person decides that the proposed attribute is inappropriate or less relevant than some other factor, he can reject the attribute or even replace it with another, using the operators Reject-Constrain or Replace-Attribute. We saw the second type of response in our example when the enquirer did not specify a restaurant quality, but instead replied 'I don't care, as long as it's cheap'. Note that, effectively, this utterance not only replaces one question with another, but also answers the second question. Replacements can also apply to attributes and values agreed upon earlier in the conversation, as happened when the user changed his mind about cuisine and decided on Indian food.

In addition, the user can explicitly accept or reject other proposals that the system makes, say for relaxing a certain attribute (Accept-Relax or Reject-Relax) or for a complete item once the system recommends it (Accept-Item or Reject-Item). We saw no examples of such rejections in our earlier scenario, but they take the same form as rejecting questions for contraction. Finally, the user can query about the available attributes (Query-Attributes) or, if the system has asked a question, about possible values of that attribute (Query-Values), as we saw for cuisine in our example.

3.3 Operator Selection and Search Control

Any search process requires some control structure to constrain and direct it. In the Adaptive Place Advisor, selection of the dialogue operators themselves is largely bounded by the communication protocols that govern human dialogue. Of course, one side of the conversation is determined by the user, so here we are concerned with how the system selects its own actions.

Table 2 presents English paraphrases of the rules that specify the Adaptive Place Advisor's selection of questions and recommendations. These control rules are sensitive to the database query that the system constructs during its interaction with the user. Thus, the initial rule (Q1) initializes the query and welcomes the user. The second (Q2) selects an attribute to constrain the query when more than four candidates remain, then asks the user about this attribute. If the query has become so constrained that no items remain, the third rule (Q3) proposes to the user an attribute to relax. Finally, if the query returns only a few candidates, the last rule (Q4) selects one of these items and recommends it to the user.

Table 3 presents another set of control rules that are responsible for dealing more directly with user responses. Many of the conditions here refer to user-applied operators, which we assume another part of the Place Advisor infers from the user's behavior. For example, rule R1 handles situations in which the user answers a question with some value, thus constraining the database query, whereas rule R2 detects when the user has rejected an attribute and notes this fact for future reference.

Rules R3 and R4, respectively, handle the analogous cases in which the user accepts and rejects system proposals to relax a particular constraint. Similarly, rules R5 and R6 deal with acceptance and rejection of system recommendations about particular items. The seventh rule, R7, covers situations in which the user rejects a question the system has asked or decides to replace a question previously answered. The following two rules, R8 and R9, respond to user queries about available attributes and values, keeping track of what they tell the user to avoid repetition. The final condition-action rule, RW, is a default that, if the user has not yet replied, waits for a response.

As we noted above, the current implementation does not make use of an explicit representation of user intentions, as Elio and Haddadi assumed in their analysis. Nor does the Adaptive Place Advisor directly support subdialogues about side topics, such as for clarification of an attribute's values, as does their design. Rather, for both purposes, the system relies on information about the

Table 2. Control rules in the Adaptive Place Advisor that present questions to user.

```
Q1. If there is no database Query,
    Then let Query be an unconstrained query,
         Let Unasked be all known attributes.
         Let Asked be the empty set.
         Let Undesired be the empty set.
         Let Fixed be the empty set.
         Let Rejected be the empty set.
         Welcome the user.
Q2. If Size(Query) > 4,
    Then let Attribute be Select-Constrain(Query, Unasked, Undesired).
         Remove Attribute from Unasked.
         Add Attribute to Asked.
         Ask-Constrain(Attribute).
Q3. If Size(Query) = 0,
    Then let Attribute be Select-Relax(Query, Asked, Fixed).
         Ask-Relax(Attribute).
Q4. If 0 < Size(Query) < 4,
    Then let Recommended be Select-Candidate(Query, Rejected).
         Recommend-Item(Recommended).
```

history of the conversation and the dialogue operators that the user has invoked. We believe these will be sufficient to support natural conversations about destinations, but ultimately this remains an empirical question.

The rules described in Tables 2 and 3 establish some general guidelines about the flow of conversation, but some important choices still remain. One decision, denoted by the call to `Select-Constrain` in Rule Q2, involves determining the specific question the system should ask to constrain the query. To this end, the Adaptive Place Advisor selects the attribute that provides the most information and thus minimizes the uncertainty within the items C that satisfy the constraints already established. In mathematical terms, it selects the attribute a_i that minimizes, for the random variable c of possible items, the entropy measure

$$H(c|a_i) = \sum_{v_j \in a_i} \sum_{c_k \in C} - \log(p(c_k|a_i = v_j)) \cdot p(c_k, a_i = v_j) \ ,$$

where c_k is a particular item in the set C that the user may find acceptable and v_j is a possible value for attribute a_i. We can further compute

$$p(c_k|a_i = v_j) = \frac{p(c_k)}{\sum_{c_j \in C'} p(c_j)} \ ,$$

where C' is the set of items which satisfy the revised constraints that include

Table 3. Control rules in the Adaptive Place Advisor that handle user responses.

```
R1. If Response is Answer-Constrain(Attribute, Value),
    Then add Attribute Value as a constraint on Query.
R2. If Response is Reject-Constrain(Attribute),
    Then add Attribute to Undesired.
         Remove Attribute from Asked.
R3. If Response is Accept-Relax(Attribute),
    Then remove Attribute as a constraint on Query.
         Remove Attribute from Asked.
R4. If Response is Reject-Relax(Attribute),
    Then add Attribute to Fixed.
R5. If Response is Accept-Item(Recommended),
    Then halt the conversation.
R6. If Response is Reject-Item(Recommended),
    Then add Recommended to Rejected.
R7. If Response is Replace-Attribute(Attribute, New-Attribute),
    Then add Attribute to Undesired.
         Remove Attribute from Asked.
         Add New-Attribute to Asked.
         Remove New-Attribute from Unasked.
         Let Attribute be New-Attribute.
R8. If Response is Query-Attribute( ),
    Then let Attributes be
           Select-Attributes(Unasked, Asked, Suggested-Attributes).
         Add Attributes to Suggested-Attributes.
         Suggest-Attributes(Attributes).
R9. If Response is Query-Value(Attribute),
    Then let Values be Select-Values(Attribute, Suggested-Values).
         Add Values to Suggested-Values.
         Suggest-Values(Values).
RW. If there is no Response,
    Then wait for some response.
```

$a_i = v_j$. In addition, $p(c_k, a_i = v_j)$ is simply $p(c_k)$ if item c_k meets the revised constraints and zero otherwise.

As a first approximation, we assume that all items have equal probability, which means that $p(c_k) = |C|^{-1}$. The resulting measure behaves sensibly in simple situations. For instance, given a Boolean attribute that divides the remaining items into two groups of equal size and another that divides them into two unequal groups, the metric prefers the attribute that gives the even split.

Using this measure, the Adaptive Place Advisor computes the information to be gained by asking each possible question, then selects the one with the lowest entropy, breaking ties at random. This evaluation metric is similar to one often used in constructing decision trees from training data, except that it does not use class labels and it recalculates the expression during each conversation, rather than creating a permanent tree structure. This makes the question-selection strategy more akin to 'lazy' approaches to decision-tree induction, which construct only a single path through an implicit tree. In our domain, this path corresponds to a search trajectory through the space of conversational actions.

The system must make similar choices when suggesting attributes or values in response to a user query, and when recommending complete items that satisfy the specified constraints. On each conversational step, the Adaptive Place Advisor make predictions about which attribute to present and, given an attribute, about which value to suggest. In responding to a user query, the system simply presents the three most probable options not yet included in the query and not yet seen by the user. At the item level, the system combines its probabilities about attribute values to rank the candidate destinations and presents the most likely one the user has not yet seen.

The Adaptive Place Advisor must also select an attribute when it decides to relax its constraints because no candidate items remain. In this situation, the system selects the attribute that has not yet been rejected for relaxation by the user and that, when removed from the current database query, gives the smallest set of candidates. One could imagine more sophisticated evaluation metrics and even strategies for replacing one value with another, rather than removing the attribute entirely, but the current version takes the most straightforward approach to this issue.

3.4 A Graphical Interface for Conversational Advice

Our long-term plans for the Adaptive Place Advisor call for an interface that supports spoken conversations, drawing on techniques from natural-language processing and speech recognition. The system could then support interactions in a broad range of settings, including situations in which the user is driving an automobile. However, such interfaces are difficult to implement and not always robust, so our initial version instead relies on a graphical interface that we have designed to parallel the conversations we envision with the future system. Figure 1 shows the display, which includes boxes on the left for destination attributes, boxes on the right for their values, menu buttons for information about each, and a box at the bottom for the final candidates.

There exist direct analogues between interactions that occur with this interface and the dialogue operators we considered above. For example, the Adaptive Place Advisor asks a question by showing a new attribute in the left column, and the user answers a question by typing his preferred value in the corresponding box on the right. The person can also query the system about the attribute's

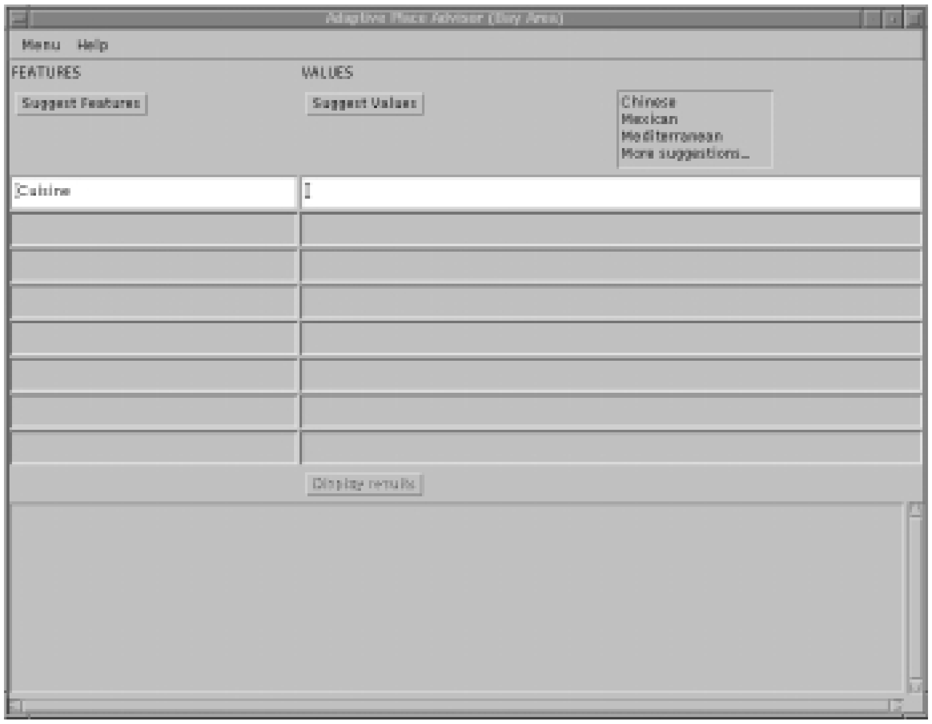
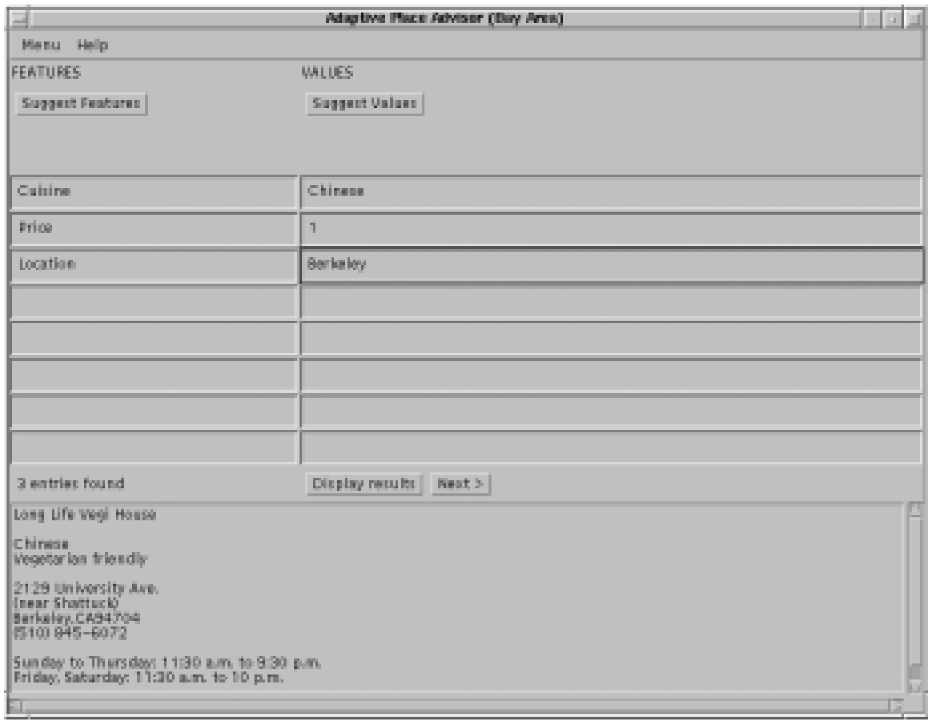


Fig. 1. Graphical display for the Adaptive Place Advisor, showing a state after the user has asked for information about the attribute 'cuisine' and received answers from the system.

values by clicking the menu on the top right, but Figure 1 shows that this menu presents only three values at a time, to reflect the limited bandwidth of spoken conversations. The user can request more values, but this requires an explicit request, which corresponds to a separate dialogue operator.

If the user wants to reject a question rather than answering it, he simply highlights and deletes the one proposed by the system, in which case it suggests another attribute. He can also replace one attribute with another by typing over the highlighted proposal. On a related note, the user can request information about alternative attributes by clicking the menu button on the top left, though again this presents only three options at a time, to reflect the nature of spoken dialogues. Similar actions are possible for values, which the user can reject by deletion or replace by typing over them. When the system decides it should relax an attribute, it highlights the proposed retraction, which the user can accept by typing 'return' or reject by taking any other keyboard action.

Figure 2 shows the graphical display after the conversation has reduced the candidate set to only three restaurants. In this case, the system displays detailed information about one candidate in the lower box, and the user can either accept



Cuisine	Chinese
Price	1
Location	Berkeley

3 entries found

Long Life Vegi House

Chinese
Vegetarian friendly

2129 University Ave.
(near Shattuck)
Berkeley,CA94704
(510) 845-6072

Sunday to Thursday: 11:30 a.m. to 9:30 p.m.
Friday, Saturday: 11:30 a.m. to 10 p.m.

Fig. 2. Graphical display for the Adaptive Place Advisor, showing a state after the system has asked three questions and received answers from the user.

this recommendation or ask to see another alternative by clicking the 'Next' button. We could have designed the system to display all remaining candidates, but again we have constrained presentation to imitate the style of communication that occurs during human interactions.

3.5 Modeling User Preferences

We refer to the Adaptive Place Advisor as 'adaptive' because its design includes a module that constructs user models from interaction traces. However, our goal for user modeling differs from the one commonly assumed in recommendation systems, which emphasizes improving accuracy or related measures like precision and recall. We assume that the conversational nature of the Place Advisor will already give acceptable recommendations at the level of items like restaurants, so the system will not construct a user model at this level.

On the other hand, the dialogue process, if not well guided, can be tedious and time consuming, so we need user modeling to produce more *efficient* conversations. Just as interactions with a friend who knows your concerns can be more directed than one with a stranger, so dialogues with the Adaptive Place Advisor should become more directed over time, giving a form of speedup learning. This suggests modeling user preferences at a finer-grained level than in typical adaptive interfaces, focusing on the questions a user prefers to answer and the responses he tends to give, rather than at the level of entire items.

We are exploring two approaches to user modeling at the conversational level. The first represents the learned user model in direct probabilistic terms. For this, we need some way to predict the distribution of questions (attributes), as well as the distribution of answers (values), given other attribute-value pairs. One straightforward response is to use the naive Bayesian classifier (Langley, Iba, & Thompson, 1992), which we can train to predict the accepted attribute given other attribute values, as well as the value of some attribute given the same conditions. This method estimates and stores the conditional probability of each attribute value given the predicted entity, along with the probability of each entity. Naive Bayes fares well on many domains, despite its assumption of attribute independence, because it must estimate very few parameters, and it also deals cleanly with missing attributes for questions not yet answered. For example, suppose the system wants to predict the distribution of values for 'city' from the values of other attributes. If only 'cuisine' and 'price' are known, then naive Bayes simply ignores other attributes and uses these in predicting the city.

However, we need some way to combine statistics about the user's behavior with the system's generic mode of operation. Recall that the Place Advisor's default technique assumes a uniform distribution over items that, combined with the database, produces a probability distribution for each attribute conditioned on the attribute values already specified during the dialogue. Rather than using these as the final probabilities, we can use them as prior probabilities in the user model. Naive Bayes typically assumes uniform priors over classes and attribute values, but reliance on database queries seems a clean way to generate more informed priors that are relevant to the domain at hand. If the user's preferences disagree with these priors, his responses will alter the distributions in the user model, but to the extent that they do agree, the system will behave according to his wishes all the sooner.

Another approach is suggested by the Adaptive Place Advisor's similarity to case-based reasoning systems for help desks and related applications, which (as we discuss shortly) also engage in conversations of a sort. The basic idea is to store cases in a manner that represents details of a user's past dialogues, in terms of the questions he has accepted and the answers he has given, along with their order. The system would compare its description of the current dialogue state against stored traces to find cases that match closely, then use the selected case to decide on the next question. For instance, suppose the user and system have agreed on Chinese cuisine and medium price; if, during a previous interaction, the user accepted a question about the city under similar conditions, the system would ask about the city this time as well. We can apply the same basic scheme to predict the user's answers, which (if predictable enough) the system could offer as alternatives without an explicit user request.

A well-known characteristic of case-based methods is their sensitivity to the distance metric used for case selection. Here it seems natural to use the entropy measure described earlier to calculate the weights on attributes, since we intended it precisely to determine their relevance. But this raises another design issue, since we must decide when the system should rely on a retrieved case to select attributes or values and when it should use the entropy measure directly. One scheme would invoke a retrieved case only when its distance to the current situation fell below a given threshold and fall back on the entropy measure otherwise. This two-part response is less elegant than the probabilistic method discussed above, but a case-based approach offers many attractions, so we intend to explore this method as well.

4 Related Work on Conversational Interfaces

Although the ranked-list framework remains the most common approach to computer-assisted recommendation, we are not the first researchers to realize the potential of conversational interfaces. Rich (1979) describes one of the earliest systems, which carried out a textual conversation with a user, asking directed questions to infer their reading tastes for the purpose of book recommendation.

More recently, Allen et al. (1995) report on their ambitious Trains system, an intelligent assistant for planning tasks that converses with users in spoken natural language. Like the Place Advisor, the program interacts with the user to progressively construct a solution, though the knowledge structures are partial plans rather than constraints, and search involves operators for plan modification rather than database contraction and expansion. Each conversational step addresses issues about when, where, and how to complete a plan while ensuring no conflicts occur, rather than selecting a single item, as in our work. Trains lacks any mechanism for user modeling, but the overall system is considerably more mature and has been evaluated extensively.

Smith and Hipp (1994) describe another related project, this one concerning a conversational interface for circuit diagnosis. Their system aims to construct not a plan or a set of constraints, but rather a proof tree. The central dialogue operator, which requests knowledge from the user that would aid the proof process, is invoked when the program detects a 'missing axiom' that it needs for its reasoning. This heuristic plays the same role in their system as does the Place Advisor's heuristic for selecting attributes to constrain destination selection. The interface does some limited user modeling, but only infers user knowledge during the course of one conversation, not over the long term, as in our approach.

Perhaps the most closely related effort comes from Rich and Sidner (1998). Their Collagen system supports conversations about complex tasks like scheduling a sequence of airline trips, relying on a graphical interface for interactions between the user and an advisory system. Their approach also builds upon theories of discourse to support an interactive decision-making process. One difference from the Adaptive Place Advisor is that Collagen includes a more sophisticated model of dialogue that makes provision for interruptions, as well incorporating a more general representation of the resulting plan. However, their system's initiative seems to focus on suggesting tasks that the user should next address, making it less directive than the approach we have taken.

Dowding et al. (1993) and Seneff et al. (1998) have also developed conversational interfaces that give advice about air travel. Like the Place Advisor, their systems ask the user questions in order to reduce candidates, treating the choice of selecting airline flights as the interactive construction of database queries. Natural language and speech play a central role in both decision aids, making them more mature than our work on destination advice. However, neither system includes a component for user modeling to make conversations more efficient, despite the clear differences among individuals in this domain.

Linden, Hanks, and Lesh (1997) present another interactive travel assistant that carries out conversations through a graphical interface. Their system also asks questions in an effort to narrow down the available candidates, using similar dialogue operators to those we described earlier, and they share our aim of satisfying the user with as few interactions as possible. Their response to this challenge relies on a 'candidate/critique' approach to problem solving, in which the system presents candidate solutions to the user, who then critiques the solutions. From these responses, the system infers a user model stated as weights on attributes of travel choices, such as price and travel time. Unlike the Adaptive Place Advisor, it does not carry these profiles over to future conversations, but one can envision a version that stores longer-term models.

Aha, Breslow, and Maney's (1998) work on 'conversational case-based reasoning' also has connections to our research. Their effort focuses on troubleshooting printers and similar tasks, and their approach relies on interactions with the user to retrieve cases from memory that will recommend actions to correct some problem. The system supports simple textual interactions, using part-of-speech tags and keyword matching, but it relies primarily on a graphical interface for presenting and answering questions. The dialogue operators and basic flow of control have much in common with the Adaptive Place Advisor, in that answering questions increasingly constrains available answers. However, one significant difference is that they let the user select which of several system-generated questions to answer next, and which of several presented cases (items) is closest to his needs. Their system also supports inference of some fields based on the values of others, which reduces the number of questions it must ask the user.

We have focused here on conversational interfaces, but research on user modeling has been more widespread in traditional recommendation systems. Pazzani, Muramatsu, and Billsus (1996) report a content-based approach that recommends web pages, whereas Shardanand and Maes (1995) describe a collaborative method that suggests movies. Other systems that induce user models focus on filtering news stories (Lang, 1995) and electronic mail (Segal & Kephart, 1999). Nor has this idea been limited to recommendation tasks, with other adaptive user interfaces addressing generative problems like note taking (Schlimmer & Hermens, 1993), route advising (Rogers, Fiechter, & Langley, 1999), and interactive scheduling (Gervasio, Iba, & Langley, in press). Langley (1999) gives a more thorough review of research on the topic of adaptive interfaces.

5 Concluding Remarks

In this paper, we described the initial version of the Adaptive Place Advisor, an intelligent assistant designed to help people select a destination, specifically a restaurant. Unlike most recommendation systems, which accept keywords and produce a ranked list, this one carries out a conversation with the user to progressively narrow his options. And unlike other adaptive interfaces, it constructs user models at the level of dialogue actions rather than the level of complete items. We viewed the dialogue process in terms of problem-space search, describing the various states, operators, and heuristics involved. We also described two approaches to user modeling, one relying on case retrieval and the other on probabilistic summaries, that should lead to more efficient conversations as the system gains experience with the user.

Although we have a detailed design and a partial implementation of the Adaptive Place Advisor, clearly more work lies ahead. First, we should carry out pilot studies with the current graphical version to get experience with users' responses. This may reveal design flaws in the system, such as the need for additional dialogue operators. If these studies are encouraging, we should then run more systematic experiments to evaluate our two approaches to user modeling. Our main hypothesis here is that conversation time will decrease as the system gains experience with a user, and that this reduction will occur more rapidly than when user modeling is not included. But we are also interested in the details of this improvement, including the learning rate and the asymptotic accuracy, so we will collect learning curves that plot performance against the number of conversations.

On another front, we must still replace the graphical interface with one that incorporates natural language and speech. The generation module should be straightforward, since we can rely on canned phrases to ask questions and propose answers. The understanding process will be more challenging, but our design should encourage constrained dialogues, often with one-word answers, that can be handled by existing software packages. Moreover, the presence of a user model should aid speech recognition by providing probability distributions over the user's answers.

In the longer term, we intend to expand the framework to take into account other factors, such as the user's willingness to explore new cuisines or cities and his desire for variety. We might represent the former with a single factor indicating the probability, based on past responses, that the user will accept an attribute value the system has not suggested before. We could represent the user's bias about variety as the average time between selections of a given attribute value, though we might also associate a variety parameter with items themselves. For example, some users may be willing to eat Chinese food once a week and others only once a month, and for some people variety among restaurants may be more important than variety in cuisine or location.

We also hope to extend our conversational approach to other types of destinations, such as hotels and theaters, and to link the system to other driving assistants like the Adaptive Route Advisor (Rogers, Fiechter, & Langley, 1999), which recommends routes to a specified destination. Our goal for such additions is to provide new functionality that will make the Adaptive Place Advisor more attractive to users, but also to test the generality of our framework for adaptive recommendation. In turn, these should bring us closer to truly flexible computational aides that carry out natural dialogues with humans.

Acknowledgements

We thank Lei Wang for implementing the basic version of the Adaptive Place Advisor described in this paper, Sean Stromsten for the information-theoretic analysis of attribute selection, and Stanley Peters for enlightening discussions about the design of conversational interfaces.

References

Aha, D., Breslow, L., & Maney, T. (1998). Supporting conversational case-based reasoning in an integrated reasoning framework. *Proceedings of the AAAI Workshop on Case-based Reasoning Integrations* (pp. 7–11). Madison, WI: AAAI Press.

Allen, J., Schubert, L., Ferguson, G., Heeman, P., Hwang, C., Kato, T., Light, M., Martin, N., Miller, B., Poesio, M., & Traum, D. (1995). The Trains project: A case study in building a conversational planning agent. *Journal of Experimental and Theoretical Artificial Intelligence*, *7*, 7–48.

Das, D., & ter Horst, H. (1998) Recommender systems for TV. *Proceedings of the AAAI Workshop on Recommender Systems* (pp. 35–36). Madison, WI: AAAI Press.

Dowding, J., Gawron, J., Appelt, D., Bear, J., Cherny, L., Moore, R., & Moran,D. (1993). Gemini: A natural language system for spoken-language understanding. *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics* (pp. 54–61). Columbus, OH.

Elio, R., & Haddadi, A. (1998). *Dialog management for an adaptive database assistant* (Technical Report 98-3). Daimler-Benz Research & Technology Center, Palo Alto, CA.

Elio, R., & Haddadi, A. (1999). On abstract task models and conversation policies. *Proceedings of the Agents '99 Workshop on Specifying and Implementing Conversation Policies*. Seattle, WA.

Gervasio, M. T., Iba, W., & Langley, P. (in press). Learning user evaluation functions for adaptive scheduling assistance. *Proceedings of the Sixteenth International Conference on Machine Learning*. Bled, Slovenia: Morgan Kaufmann.

Lang, K. (1995). NewsWeeder: Learning to filter news. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 331–339). Lake Tahoe, CA: Morgan Kaufmann.

Langley, P. (in press). User modeling in adaptive interfaces. *Proceedings of the Seventh International Conference on User Modeling*. Banff, Alberta: Springer.

Langley, P., Iba, W., & Thompson, K. (1992). An analysis of Bayesian classifiers. *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 223–228). San Jose, CA: AAAI Press.

Linden, G., Hanks, S., & Lesh, N. (1997). Interactive assessment of user preference models: The automated travel assistant. *Proceedings of the Sixth International Conference on User Modeling* (pp. 67–78). Chia Laguna, Sardinia: Springer.

Mooney, R. J., Bennett, P. N., & Roy, L. (1998). Book recommending using text categorization with extracted information. *Proceedings of the AAAI Workshop on Recommender Systems* (pp. 70–74). Madison, WI: AAAI Press.

Nguyen, H., & Haddawy, P. (1998). The decision-theoretic video advisor. *Proceedings of the AAAI Workshop on Recommender Systems* (pp. 77–80). Madison, WI: AAAI Press.

Pazzani, M., Muramatsu, J., & Billsus, D. (1996). Syskill & Webert: Identifying interesting web sites. *Proceedings of the Thirteenth National Conference on Artificial Intelligence* (pp. 54–61). Portland, OR: AAAI Press.

Rich, C., & Sidner, C. (1998). Collagen: A collaboration manager for software interface agents. *User Modeling and User-Adapted Interaction*, *8*, 315–350.

Rich, E. (1979). User modeling via stereotypes. *Cognitive Science*, *3*, 329–354.

Rogers, S., Fiechter, C., & Langley, P. (1999). An adaptive interactive agent for route advice. *Proceedings of the Third International Conference on Autonomous Agents* (pp. 198–205). Seattle, WA: ACM Press.

Schlimmer, J. C., & Hermens, L. A. (1993). Software agents: Completing patterns and constructing user interfaces. *Journal of Artificial Intelligence Research*, *1*, 61–89.

Searle, J. (1969). *Speech acts*. New York: Cambridge University Press.

Segal, R. B., & Kephart, J. O. (1999). MailCat: An intelligent assistant for organizing e-mail. *Proceedings of the Third International Conference on Autonomous Agents* (pp. 276–282). Seattle, WA: ACM Press.

Seneff, S., Hurley, E., Lau, R., Pao, C., Schmid, P., & Zue, V. (1998). Galaxy-II: A reference architecture for conversational system development. *Proceedings of the Fifth International Conference on Spoken Language Processing* (pp. 931–934). Sydney: ASSTA.

Shardanand, U., & Maes, P. (1995). Social information filtering: Algorithms for automating 'word of mouth'. *Proceedings of the Conference on Human Factors in Computing Systems* (pp. 210–217). Denver, CO: ACM Press.

Smith, R., & Hipp, D. (1994). *Spoken natural language dialog systems: A practical approach*. New York: Oxford University Press.

This article was processed using the LaTeX macro package with LLNCS style