

Learning to Suggest: The Adaptive Place Advisor

Cynthia A. Thompson

Center for the Study of Language and Information
Stanford University
Stanford, CA 94305-4115
cthomp@csl.i.stanford.edu

Mehmet H. Göker

DaimlerChrysler Research & Technology
1510 Page Mill Road
Palo Alto, CA 94304
mehmet.goeker@daimlerchrysler.com

Abstract

In this paper, we describe the Adaptive Place Advisor, a conversational recommendation system designed to help users decide on a destination. We view the selection of destinations as an interactive, conversational process, with the advisory system inquiring about desired item characteristics and the human responding. The user model, which contains preferences regarding items, attributes, values and value combinations, is also acquired during the conversation. The system enhances the user's requirements with the user model and retrieves suitable items from a case-base. If the number of items found by the system is unsuitable (too high, too low) the next attribute to be constrained or relaxed is selected based on the information gain associated with the attributes. We also describe the current status of the system and future work.

1. Motivation

As information becomes abundant, humans are confronted with more difficult decisions for how to access, navigate through and select available options. The sheer number of alternatives often makes a wise choice impossible without some intelligent 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* (e.g. Burke, Hammond, and Young (1996)), which are not limited to filtering information but can be used for any task that requires choice among a large set of predefined items.

Society, on the other hand, is getting more complex and diversified. The differences in personal preferences, social and educational backgrounds and private or professional interests are increasing, and tools to access information are becoming ubiquitous. This causes the need for intelligent systems that process, filter and display available information in a personalized manner. Research on personalization has led to the development of systems that adapt themselves to the characteristics of their user: *user adaptive systems* (e.g. Rich, E. (1979), Langley (1997)).

In this paper we describe the Adaptive Place Advisor, a user adaptive, conversational recommendation system. The system helps the user to select a destination from its database by performing a personalized conversation.

2. Conversational Recommendation Systems

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

The most widely used framework for recommendation systems is the *ranked list* approach. In this scheme, the user specifies his needs with one or more keywords and the system presents a usually long list of results, ordered by their predicted relevance to the user. This technique comes from 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. Consider a situation in which one person, that we will call the *inquirer* asks a second person, that we will call the *advisor*, for assistance in deciding on a restaurant at which to have dinner:

Inquirer: Where do you think I should eat tonight?

Advisor: Well, what type of cuisine would you like?

Inquirer: What types are there?

Advisor: Some examples are Chinese, Indian, and Mediterranean.

Inquirer: Oh, maybe Indian.

Advisor: What quality rating would you like?

Inquirer: I don't care, as long as it's cheap.

Advisor: How do you want to pay?

Inquirer: No, I think I'd like Chinese instead.

Advisor: Okay, we'll switch to Chinese food. What city do you prefer?

Inquirer: 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?

Inquirer: Sure, that sounds fine.

We will refer to systems that mimic this approach to recommendation as *conversational* recommendation systems.

The interaction supported by conversational systems seems quite different from that found in the ranked list approach. The most important distinction is that the inquirer 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 to remove alternatives rather than to simply order them. The conversational process can also help the inquirer better understand his own desires, since thinking about possible questions and answers may clarify goals in ways a ranked list does not.

Such dialogues seem better for recommendations that must be delivered by speech 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 inquirer is driving. 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 in situations where information can be presented visually and for tasks like the selection of web pages or news stories, in which the user may well want to examine many options.

3. User Adaptive Systems

While raw data usually does not change based on the individual initiating its processing, the resulting information and the manner in which it is presented can be influenced by personal differences. Diversification in society has a direct impact on the number of ways in which users may choose their data to be processed and presented. A computer system should ultimately be intelligent enough to take individual variations in preferences, goals and

backgrounds into account and generate personalized information.

Individual differences can have an effect on computer systems at three levels:

- the data processing level,
- the information filtering level, and
- the information presentation level.

On the data processing level, the algorithms used for processing the data may vary depending on the user. The information generated by the processing level can be filtered according to its content (content based-filtering) or according to behavior of similar users (collaborative filtering). Finally, the same piece of information can be presented differently to different users.

User adaptive systems are intelligent systems that capture user preferences and use them to customize themselves to individual users on one or more of the above mentioned levels.

4. The Adaptive Place Advisor

Our goal is to develop conversational recommendation systems in which the interaction between the system and user becomes more efficient over time due to the system's adjustments to the preferences of the user. While this approach does extend to item recommendation in general, our initial work has focused on destination selection as the initial application domain.

In the following sections, we describe the *Adaptive Place Advisor*, a conversational recommendation system designed to help users decide on a destination. Our prototype system aims to help drivers select a restaurant that meets their preferences.

To be able to recommend a restaurant based on a conversation, the Adaptive Place Advisor has to

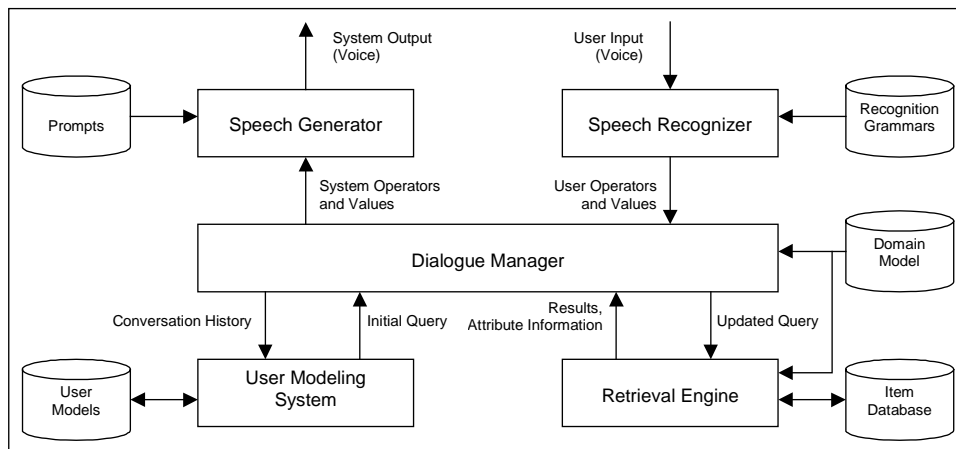


Fig.1: Overall System Architecture of the Adaptive Place Advisor.

- carry out a conversation and generate a partial restaurant specification, i.e. a query,
- improve or complement the query with a model of the user's preferences,
- use this query to retrieve matching restaurants from a database and calculate their similarity to the users' request
- if the number of retrieved items is not acceptable, select the next attribute to be constrained or relaxed during a conversation, and
- learn and update the user model based on these interactions.

The responsibilities for these tasks are distributed among various modules of the system (see Fig. 1). The *Dialogue Manager* generates and recognizes conversations. The *Retrieval Engine* is a case-based system that uses the query that has been generated and updated by the Dialogue Manager to retrieve items from the database. The *User Modeling System* generates the initial (default) query from the user model and updates the user model based on the conversation history. The *Speech Recognizer* and the *Speech Generator* comprise the natural language processing part of the system. We used tools from Nuance¹ to handle recognition and output of the appropriate prompts from a pre-recorded set.

In the following sections we describe the operation of the dialogue manager, the acquisition, modeling and utilization of the user preferences, and the retrieval engine.

5. Talking with the Driver

We view the 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. 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). We extend upon and adapt that work as needed to conform to the requirements of speech recognition technology and the design of the adaptive component.

Our view of conversational recommendation as heuristic search requires us to specify the search states, operators, and operation-selection heuristics. The initial state of the search is that of an unconstrained query, where the system and user have not yet agreed upon any attribute values. Future states, arrived at by the operators discussed below, are (more) constrained queries. A state can consist of an over constrained query with no matching items, and the

final state is reached when only a few items match the query. The search state also includes dialogue history information to help maintain a natural and coherent conversational flow.

The majority of dialogue operators are determined by the task-level goal of finding one or more items that satisfy the user. The remaining, dialogue-level, moves are required for interactions that support progress on that task. While one side of the conversation is determined by the user, the system side of the conversation is governed by a set of control rules, described in detail in Langley, Thompson, Elio and Haddadi (1999). These rules select the next operator based on the search state and the conversational context. The particular instantiation of that operator (for example, which attribute to ask a question about next) is selected by consulting the retrieval engine and conversation history.

We group conversational actions into one operator 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 supported by the Adaptive Place Advisor.

Let us first consider the operators available to the dialogue manager for advancing the conversation. The most obvious, ASK-CONSTRAIN, involves asking the user to provide a value for an attribute that does not yet have one. This question is asked in an effort to constrain the query. In our example conversation, 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.

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 ASK-RELAX, which asks whether the user wants to drop a particular value, which would expand the candidate set.

Another operator, SUGGEST-VALUES, answers a user's query about possible values for an attribute. In our example, this occurred in response to the inquirer's query about cuisine. Note that, in this case, the advisor lists only a few options rather than all possible choices, a desirable characteristic for natural, concise conversations. 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. Finally, the CLARIFY operator is invoked when the system is uncertain about what the user has said, either because of low speech recognition certainty, when an answer diverges significantly from what was expected,

¹ Nuance Communications, Menlo Park, CA.
www.nuance.com

Table 1. Dialogue operators supported in the Adaptive Place Advisor

System Operators	
ASK-CONSTRAIN	Asks a question to obtain a value for an attribute
ASK-RELAX	Asks a question to remove a value of an attribute
SUGGEST-VALUES	Suggests a small set of possible values for an attribute
SUGGEST-ATTRIBUTES	Suggests a small set of unconstrained attributes
RECOMMEND-ITEM	Recommends an item that satisfies the constraints
CLARIFY	Asks a clarifying question if uncertain about the user's most recently performed operator
User Operators	
PROVIDE-CONSTRAIN	Provides a value for an attribute
REJECT-CONSTRAIN	Rejects the proposed attribute
ACCEPT-RELAX	Accepts the removal of a value of an attribute
REJECT-RELAX	Rejects the removal of a value of an attribute
ACCEPT-ITEM	Accepts proposed item
REJECT-ITEM	Rejects proposed item
QUERY-ATTRIBUTES	Asks system for information about possible attributes
QUERY-VALUES	Asks system for information about possible values of an attribute
START-OVER	Asks the system to re-initialize the search
QUIT	Asks the system to abort the search

or when a value could be applicable to more than one attribute.

Now let us turn to the operators that the system assumes are available to the user. The most central action the user can take, PROVIDE-CONSTRAIN, involves specifying the value of some attribute. This can be a value for the attribute just asked for by the systems ASK-CONSTRAIN operator, a value that would answer a different question, or a replacement for a previously specified value. Our example included four instances of this operator, two in response to questions about cuisine and city, one answering a question different from the one posed by the system, and one replacing the previously provided value for cuisine. Each such answer 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.

As we saw above, 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. The REJECT-CONSTRAIN operator captures explicit rejection. We saw this in our example when the inquirer did not specify a restaurant quality, but instead replied 'I don't care, as long as it's cheap'.

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 a form similar to other rejections. The user can also query about

the available attributes (QUERY-ATTRIBUTES) or about possible values of that attribute (QUERY-VALUES), as we saw for cuisine. Finally, the user can reinitialize (START-OVER) or end (QUIT) the search.

We have implemented most of the functionality described in this section. We used Nuance's SpeechObjects to interface to their speech recognition capability and output pre-recorded prompts, with the remainder of the dialogue functionality implemented in Java. Further details on the implementation of the search for satisfying items can be found in Langley et.al. 1999.

6. Acquiring, Modeling and Utilizing User Preferences

The conversation with the user, similar to constraint satisfaction, will ultimately direct the system to a suitable solution. However, such a conversation can become very tiring and the quality of the returned result may not be acceptable for each user.

Just as interactions with a friend who knows your concerns can be more directed and produce better results than those with a stranger, dialogues with the Adaptive Place Advisor should become more efficient and effective over time. 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 want to improve both the subjective quality of the results and the dialogue process itself.

To efficiently provide the users with the solution that matches their needs bests, it is necessary to acquire and

model the preferences of the users. A user may have preferences about:

- items in general,
- an attribute,
- values for an attribute, and
- the combination of certain attribute-value pairs of an item.

A preference about an item manifests itself in the user having a bias for or against a certain item, independent of its characteristics (*item preferences*). The preferences regarding an attribute represent the relative importance a user places on the attribute while selecting an item (i.e. how important is cuisine vs. price: *attribute preferences*). Preferred values show the user's bias towards certain types of items (e.g. Italian restaurants vs. French restaurants: *value preferences*) whereas preferences for certain property combinations represent certain constraints with respect to the combined occurrence of characteristics in an item (accepts Mexican restaurants only if they are cheap: *combination preferences*). While the item preferences are related to single items, the attribute, value and combination preferences are applicable to the retrieval process in general.

Contrary to other adaptive recommendation systems (e.g. Pazzani et.al. 1996, Lang 1995), our basic approach is to create and update the user model without requiring the user to provide direct feedback. The preferences of the users are derived from their interactions with the system.

Item preferences are derived by observing how often a certain item was suggested and afterwards accepted or rejected by the user. *Attribute preferences* are updated according to the item the user selects among the ones the system suggests. If the selected item was not predicted to be the most similar one to the user's query, then the attribute preferences (i.e. weighting factors) have to be adjusted. *Value preferences* are calculated based on the frequencies of the values the user selects for an attribute. *Combination preferences* are derived by looking at the history of selected items and learning association rules.

Since the value preferences can be viewed as a probability distribution over the values for each attribute, the user model can be used as an initial query with default values. In the course of the conversation, this initial query is refined and constrained with the values the user specifies for each attribute. The effects of relevant dialogue operators on the query and the user model can be seen in table 2. Please note that only the user operators have an effect on the query and the user model.

The specification or rejection of values only effects the query and not the user model directly. The user model is updated by using the last version of the query. The update is performed after the user has selected an item or in a situation in which the system is unable to find an item meeting the specifications of the user.

Table 2. The effects of dialogue operators on the query and the user model

<i>Dialogue Operator</i>	<i>Effect on Query / User Model</i>
ACCEPT-ITEM	<ul style="list-style-type: none"> • Update user model with query • Update item preference
REJECT-ITEM	<ul style="list-style-type: none"> • Update item preference
PROVIDE-CONSTRAIN	<ul style="list-style-type: none"> • Set probability of value for the constrained attribute in query to one • Set probability of other values for that attribute in query to zero
REJECT-CONSTRAIN	<ul style="list-style-type: none"> • Drop attribute, i.e. set attribute preference (weighting factor) in query to zero
ACCEPT-RELAX	<ul style="list-style-type: none"> • Update user model with latest query • Reset value preferences for the attribute in query from user model. (Dialogue Manager ensures that the question is not asked again.)
REJECT-RELAX	<ul style="list-style-type: none"> • No effect, Dialogue manager selects next attribute
START-OVER	<ul style="list-style-type: none"> • Initialize query with user model

7. The Retrieval Engine

The retrieval engine of the Adaptive Place Advisor retrieves the items that are most suitable to the users' request and match his preferences. Retrieval engines in Case-Based Reasoning systems are usually indifferent to the users' preferences. They calculate the similarity of the items in the case-base to the query using the similarity metrics and weighting factors in the domain model. The Adaptive Place Advisor has to take the current status of the conversation and preferences of the user into account.

The current status of the conversation determines which of the attributes of the query have values associated with them. Since these values represent explicit choices of the user, we use them generate an SQL-query to retrieve all items that match these values. The set of items that is returned from the database is used as a case-base for similarity based retrieval. This allows the content of the case-base to change from interaction to interaction.

Similarity between a case C and the current query Q is calculated as follows:

$$Sim(Q, C) = R_c \times \frac{\sum_{i=m}^n w_i \times P(V_{A_i})}{n - m}$$

where R_c is the user's preference for the specific case, w_i is

the weighting factor (attribute preference) for attribute A_i , A_m is the first attribute for which the user has not selected a value yet², V_{A_i} is the value of A_i in the case, and $P(V_{A_i})$ is the user's value preference (probability) for this value. The local similarity metric (which calculates the similarity for each attribute of the case and the query) is replaced by the probability of the user requesting the value in the case.

The selection of the attribute to be constrained or relaxed next is based on an information gain measure. The attribute to constrain is selected by determining the attribute with the lowest entropy (highest information gain) among the attributes the user has not yet constrained. If no items were returned from the database query, the attribute with the highest entropy (lowest information gain) is selected among the attributes the user has constrained so far and suggested for relaxation. This insures that the search stays focussed, i.e. information is preserved. The case-base of the previous query is used as a basis for this calculation.

Since we only need to find one item, the entropy of an attribute A_i can be calculated as:

$$H = \sum_{V_j \in A_i} P(V_j) \times |CB_s| \times \frac{1}{|A_i = V_j|_{CB_s}} \times \log \frac{1}{|A_i = V_j|_{CB_s}}$$

Where $P(V_j)$ is the probability of the value of the attribute A_i to be V_j (this is not the probability coming from the user model), $|CB_s|$ is the number of cases above a certain similarity threshold, and $|A_i = V_j|_{CB_s}$ is the number of items in CB_s in which A_i has the value V_j .

8. Summary and Future Work

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. We also described a framework for modeling user preferences and utilizing them for guiding the conversation and during retrieval.

Although we have a detailed design and a partial implementation of the Adaptive Place Advisor, clearly more work lies ahead. Our current similarity calculation does not take the effects of combination preferences into account. We believe however, that this plays an important role in the user's approach of selecting an item and are planning to incorporate it. Obviously we need to perform

² This, simplified, representation assumes that the attributes A_1 to A_{m-1} are the ones the user has explicitly specified during the conversation. Obviously the real system does not require the specified attributes to be in a sequence.

evaluations and measure the effects of the user model on the conversation and the resulting selection. We are also planning to transfer the system to similar domains (e.g. selecting books, music) and translate the system to German.

Acknowledgements

We thank Pat Langley, Renée Elio and Afsaneh Haddadi for initial conception and design of the adaptive place advisor, Cynthia Kuo for help with the speech interface version of the Adaptive Place Advisor, and Stanley Peters for enlightening discussions about the design of conversational interfaces.

References

- Burke, R., Hammond, K., and Young, B. 'Knowledge-based navigation of complex information spaces', In Proceedings of the 13th National Conference on Artificial Intelligence, pp. 462-468. American Association for Artificial Intelligence, 1996.
- Elio R., Haddadi A., 'Dialog management for an adaptive database assistant', Technical Report 98-3, Daimler-Benz research and Technology Center, Palo Alto, CA, 1998.
- Elio R. Haddadi A., 'On abstract task models and conversation policies', in Proceedings of the Agents'99 Workshop on Specifying and Implementing Conversation Policies, Seattle, WA, 1999.
- Lang K, 'NEWSWEEDER: Learning to filter news', in 'Proceedings of the Twelfth Conference on Machine Learning', pp.331-339, Lake Tahoe, CA, Morgan Kaufmann, 1995.
- Langley, P. 'Machine learning for adaptive user interfaces', in: 'Proceedings of the 21st German Annual Conference on Artificial Intelligence', pp. 53-62. Freiburg, Germany: Springer, 1997.
- Langley, P., Thompson, C., Elio, R., Haddadi, A., 'An adaptive conversational interface for destination advice' in: 'Proceedings of the Third International Workshop on Cooperative Information Agents', pp. 347-364, Uppsala, Sweden, 1999.
- Pazzani M., Muramatsu J., Billsus D., 'Syskill and Webert: Identifying interesting web sites', in Proceedings of the 13th National Conference on Artificial Intelligence, pp. 54-61, American Association for Artificial Intelligence, 1996.
- Rich, E., 'User modeling via stereotypes', Cognitive Science, 3, 329-354, 1979.
- Searle J., 'Speech Acts', New York, Cambridge University Press, 1969.