

# Evaluating a personalized conversational recommendation system

Cynthia A. Thompson

School of Computing  
University of Utah  
Salt Lake City, UT 84112  
cindi@cs.utah.edu

## Abstract

We describe and evaluate a personalized conversational item recommendation system. In this setting, the system directs conversations that help users find items, and adapts its actions according to information from past conversations with a given user. We present results demonstrating the effectiveness of our approach.

## 1 Introduction

The application of statistical and machine learning techniques to natural language problems has become ubiquitous in the research community. However, there has not yet been much effort in applying these techniques to conversational interfaces. We believe that the benefits of doing so are numerous, and describe here one such application. Our focus is on dialogue systems that support task-oriented conversations, specifically those involving item recommendation and selection. We apply machine learning techniques to improve a dialogue system, after it has been built, for each individual who uses it.

Our system, the Adaptive Place Advisor, aims to help users select a destination, currently instantiated as restaurant selection, from a large set of options that meets their preferences. The system directs the conversation in a manner similar to slot-filling (Seneff et al. 1998; Jurafsky, 1994; Dowding et al. 1993), retrieves and ranks items using a case-based reasoning (Aamodt, 1994) paradigm, and adapts its similarity calculation based on traces of conversations with a user, thereby personalizing the retrieval and the conversation. Our main contributions are twofold. First, we demonstrate the incorporation of user adaptation into a simple dialogue model for destination recommendation. Second, we discuss experimental results

showing our ability to learn a correct model, and the issues involved in evaluating this type of system.

Most of the evolving dialogue system evaluation metrics (e.g., PARADISE (Walker et al. 1997)), while of course useful for the purpose for which they were intended, do not measure all types of improvement that we would like to see for a personalized system. While one measure, dialogue efficiency, is relevant, the accuracy of the learned user models is not reflected by any of these measures. We propose a new measure in Section 3.

## 2 The Adaptive Place Advisor

Consider a situation in which one person, an inquirer, asks a second person, an advisor, for assistance in deciding on a restaurant at which to have dinner:

1. Inquirer: Where do you think I should eat tonight?
2. Advisor: Well, what type of cuisine would you like?
3. Inquirer: What types are there?
4. Advisor: Some examples are Chinese, Indian, and Mediterranean.
5. Inquirer: Oh, maybe an Indian place with free parking.
6. Advisor: What quality rating would you like?
7. Inquirer: I don't care, as long as it's cheap.
8. Advisor: How do you want to pay?
9. Inquirer: No, I think I'd like Chinese instead.
10. Advisor: Okay, we'll switch to Chinese food. What city do you prefer?
11. Inquirer: How about Berkeley?
12. Advisor: I know three cheap Chinese restaurants in Berkeley. One is the Long Life Vegi House on 2129 University Avenue. Does that sound all right?

13. Inquirer: Sure, that sounds fine.

We refer to systems that mimic this approach to recommendation as conversational recommendation systems (Aha and Breslow, 1997). This interaction takes the form of a sequence of questions, most designed to eliminate some items from consideration. While this scheme is common to other slot-filling approaches, in our instantiation of it the order of the slots to fill is not fixed, and not all slots have to be filled in before items are presented. As is typical in conversational systems but not other types of recommendation systems (Resnick and Varian, 1997), the inquirer never hears about a complete item until only one, or at most a few, choices remain.

This type of conversation will ultimately direct the user to a suitable solution. However, such a conversation can become 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 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 the subjective quality of both the results (found items) and the conversation that leads to these results. For example, after several conversations with the Inquirer above, a new interaction may proceed as follows:

1. Inquirer: Where do you think I should eat tonight?
2. Advisor: Well, would you like a cheap Chinese place again?
3. Inquirer: Yes, but not in Berkeley this time.
4. Advisor: What city do you prefer?
7. Inquirer: How about Palo Alto?
8. Advisor: I know two cheap Chinese restaurants in Palo Alto. One is the Jing-Jing Szechuan Gourmet on 443 Emerson. Does that sound all right?
9. Inquirer: Sure, that sounds fine.

We now describe the functioning of our system. To be able to recommend a destination based on a conversation, the Adaptive Place

Advisor has to:

- carry out a coherent conversation and generate a partial item specification, called a query,
- improve or complement the query with a model of the user's preferences,
- use this query to retrieve matching items from a database and calculate their similarity to the user's 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. The Dialogue Manager generates, directs, and processes conversations. The Retrieval Engine is a case-based system that uses the query that the Dialogue Manager generates after each system-user interaction to retrieve items from the database. The Personalization 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 Synthesis comprise the natural language processing part of the system. Here, we focus on the Dialogue Manager and Personalization Systems.

## 2.1 Dialogue Management

Since the investigation of personalization in conversational recommendation systems has not been well studied, we start with a simple dialogue model, keeping the number of allowable speech acts (Searle, 1969) to a minimum. Our conversational model aims, at its core, to fill in values for slots, or attributes, of the domain at hand. The slot filling model is somewhat hidden from the user, as he is allowed to ignore the system's questions and specify values in the order that he prefers. Over time, our system adapts to these preferences, our hypothesis being that this will result in fewer interactions needed to find a satisfying item.

The process can also be thought of as an interactive search for items satisfactory to the user. To support this, the system attempts

to obtain constraints on attribute values, while avoiding unsatisfiable constraint combinations, until only a few items are found to satisfy those constraints. In the absence of a user model, the process starts with the system assuming an initially unconstrained search, with all items equally likely to be satisfactory to the user.

During this search, the system tries to keep the initiative by asking leading questions, but the user can ignore its suggestions and strike out in new directions. Before describing this process, we note that there are three classes of response available to the user at any point. First, he can ask for clarification from the system. Second, he can ask to start over. Third, he can ask to quit the interaction.

We next enumerate the types of moves that the system makes and the situations in which they are made. For each, we discuss the possible responses that the user can make. First, and most common, is the situation in which the search is under-constrained, and many items meet the current constraints. In this situation, the system makes an `ATTEMPT-CONSTRAIN` move, in which it asks the user to fill in the value for an attribute. The user can respond by a `PROVIDE-CONSTRAIN`, in which he provides a value for the specified attribute. A second possibility is a `REJECT-CONSTRAIN`, in which the user indicates disinterest or dislike in an attribute. The user can also combine more than one move in a single utterance.

A second search situation is that in which no items satisfy the agreed upon constraints. We call this an over-constrained search. In this case, the system informs the user of the situation, and asks if he would like to relax a given constraint (`SUGGEST-RELAX`). The user can respond by accepting (`ACCEPT-RELAX`) or rejecting (`REJECT-RELAX`) the relaxation. In the latter case, we say that the user has fixed the attribute and the system will not try to relax it again. In both situations, the user can specify other attributes to relax in addition to, or instead, the system suggested attribute (`PROVIDE-RELAX`).

Finally, there is the situation in which only a few items satisfy the constraints, whereupon the system begins to suggest them to the user (`RECOMMEND-ITEM`). The user can either accept or reject an item. If the user accepts an

item (`ACCEPT-ITEM`), we end the conversation, assuming that the user has completed his search for items. If the user rejects an item (`REJECT-ITEM`), the system presents an alternative, if any remain.

There are two situations not yet covered. The first is when the search is over-constrained, but the user has fixed all attributes that could be relaxed. The second is when the user has rejected all items that match the constraints. In either situation, the system informs the user, asks him whether he would like to quit, start over, or modify the search, and reacts accordingly.

The user communicates with the system through spoken natural language. To support this, we use a speech recognition package from Nuance Communications, which also allows the generation of prompts from a pre-recorded set. This allows us to write a different recognition grammar for each of the search situations discussed above (over, under, and desirably constrained). The language understanding is handled by semantic tags in the recognition grammars, to fill in each slot. Besides slots for each attribute, we included slots for rejection or acceptance of the system's suggestions.

## 2.2 Personalization

The system relies on a personalized user model to determine the order of questions to ask and items to present. First, as items matching the current constraints are retrieved from the database, the system ranks them according to the user model. If there are still many items, it chooses a constraining question that is the unasked attribute most preferred by the user. If no items remain, it chooses as a relaxing question the least preferred attribute.

### 2.2.1 Modeled Preferences

With respect to conversational item selection, users may have preferences about:

- specific items,
- the relative importance of attributes, and
- values for an attribute.

Item preferences manifest themselves in the user having a bias for or against a certain item, independent of its characteristics, such as price or location. Attribute preferences represent the

relative importance a user places on each attribute while selecting an item (e.g., how important is cuisine vs. price). Value preferences are manifested by the tendency to respond with one value for an attribute more often than another (e.g., Italian vs. French for cuisine; expensive vs. cheap for price).

Our current implementation uses probabilities to capture attribute and value preferences, and counts to model item preferences. We use simple defaults to initialize these preferences. For item preferences, we use counts to indicate the number of times an item was presented and subsequently accepted; these are initialized by assuming that all items have been presented and then accepted a large percentage of the time, e.g., by setting the presentation count to 10 and the acceptance count to 9. For attribute preferences, we use some knowledge about the domain, for example that price is usually more important than parking, to initialize the weights. For value preferences, we simply set them all equal.

### 2.2.2 Updating the Preferences

While some adaptive recommendation systems (Pazzani et al., 1996; Lang, 1995; Linden, Hanks and Lesh, 1997; Smyth and Cotter, 1999) require the user to provide direct feedback to generate the user model, our approach is to unobtrusively derive the user preferences, by observing the user’s interactions with the system, without introducing unnecessary interactions.

We update preferences in three dialogue circumstances, ACCEPT-ITEM, REJECT-ITEM, and ACCEPT-RELAX. First, we assume that when a user accepts an item, he is indicating 1) a preference for the item itself, 2) preferences for the attributes he constrained to find this item, and 3) preferences for the chosen item’s values of those attributes. Thus, when a user accepts an item presented by the system, we increase the probabilities by a small amount for the appropriate attributes and values, and increase the acceptance count for the item.

On the other hand, when a user rejects an item presented by the system, we only assume that he has a dislike for the particular item, without assuming anything about the particular characteristics of that item. When conversing about destinations, one can consider many options before arriving at an acceptable one, and

the rejection does not necessarily imply a dislike for the destination’s characteristics, just the destination itself. Therefore, we decrease the item preference counts for rejected items.

The final user model update situation is when the search has become over-constrained, the system presents an attribute for relaxation, and the user accepts that relaxation. In this situation, we assume that had there been a matching item, the user would have been satisfied with it, since the characteristics specified in the conversation so far were satisfactory. Therefore, we increase the attribute preferences for attributes constrained by the user thus far, and the value preferences for user-specified values. This enables us to more quickly make inferences about the user’s preferences.

## 3 Evaluation

Our evaluation measures the ability of our system to acquire, through natural recommendation conversations, user models that correctly reflect the user’s preferences. For these tests, we used a version of the system that recommends restaurants in the San Francisco Bay Area. The system uses 7 attributes: cuisine, rating, price, ability to accept reservations, parking, location, and payment options. Most attributes have only a few values, but a few, such as cuisine and location, have dozens. There are approximately 1900 items in the database.

For this paper, we employed “simulated” users to evaluate our system. For recommendation situations, it is difficult to both control the task and get a realistic sampling of users preferences. This is because the user’s preferences determine the task, at least to some extent. Thus, we could ask users to find a particular item, and thereby learn something from the method by which they find it, but that may not reflect their true preferences in the domain. However, simply having users choose destinations freely would lead to a wide variance in the conversations across different experimental situations, *independent* of whether or not user modeling is employed.

Therefore, we generated several artificial user models which were probabilistically used to determine the responses given to the system’s prompts. For each artificial user, we generated a random distribution over attribute and

value preferences. Item preferences, intuitively, should have some relation to attribute and value preferences. In other words, if an item has values that all have high preferences, it should be preferred over one that has values that all have lower preferences, taking into account that attributes with low preference should have less importance in the decision. Therefore, our artificial user models determined item preferences using their attribute-weighted similarity to the values they contained.

Finally, at conversation time, when each question was posed by the system, the simulated user first probabilistically determined, according to attribute weight, whether to answer the question at all or to supply a REJECT-CONSTRAIN. If the question was to be answered, a value was probabilistically chosen according to the value weights. The inverse decision was made in the case of a SUGGEST-RELAX. For item selection, if the simulated item preference exceeded a random number, it was accepted; otherwise, it was rejected.

First, we generate three such artificial models, and used each to interact with the system ten times, with user modeling turned on. Our interest is whether the attributes are *ranked* correctly, not whether the learned probability values are close to the desired values. Correct ranking is all that is needed to ensure that questions are asked in a desirable order. To quantify the correctness of attribute ranking, we used a notion of loss as described by Cohen et al. (1998). We define this as:

$$\frac{\sum_{(u,v) \in C} (1 - R(u,v))}{|C|}$$

where the pairs in  $C$  indicate the correct ordering,  $(u,v)$  is one pair in that ordering, and  $R(u,v)$  is the learned order for this pair, defined as 1 if  $u > v$  in the learned order (e.g., it is correct), 0 if  $u < v$ , and  $1/2$  if  $u = v$ . The resulting score is in the range from 0 to 1, where 0 is exact correspondence in ranking and 0.5 is the average loss over all random rankings.

The results for our three users are shown in Figure 1. The loss in ranking decreases dramatically at first, the rate slowing as more conversations are encountered. For comparison, we also show the loss when using one of the learned models to predict for the wrong simulated user

(“WrongModel”). This shows that the model learned for one user is not necessarily appropriate to use for directing conversations with a different user.

## 4 Related Work

We restrict our review here to other types of adaptive dialogue systems. The earliest personalized system was that of Rich (1979), a (type-written) conversational interface for book recommendation. This system put users into one of a set of stereotypical models, and updated that model as the conversation progressed. However, the language understanding capabilities of the system were limited, mostly allowing only yes/no user answers. Related to this was work on user modeling within the dialogue community (Carberry, 1990; Kobsa & Wahlster, 1989). In that paradigm, discourse parameters such as the user’s intentions, and the dialogue context are modeled, typically within a single conversation.

Stolcke, et al. (2000) applied machine learning at the speech recognition and dialogue control level. Also, Litman et al. (2000) and Singh, et al. (2000) applied reinforcement learning to determine the system’s level of initiative and amount of confirmation of user utterances. Their goal is to optimize, over all users, the percentage of dialogues for which a given task is successfully completed. They investigate the control of several system parameters, such as the amount of system initiative and how often to confirm the user’s previous utterance. However, both of these systems apply the learned information across all users, rather than personalizing the information.

## 5 Future Work

Our results to date are promising, but more work is needed. Further experiments will show the applicability across domains and further extend the results to real users (citation withheld). We also plan to measure the long-term effects of the learned models across conversations.

There are further types of preferences that we plan to handle, namely combination and variety preferences. For the first, speakers may have preferences with respect to the combined occurrence of characteristics in an item (e.g., accepts Mexican restaurants only if they are

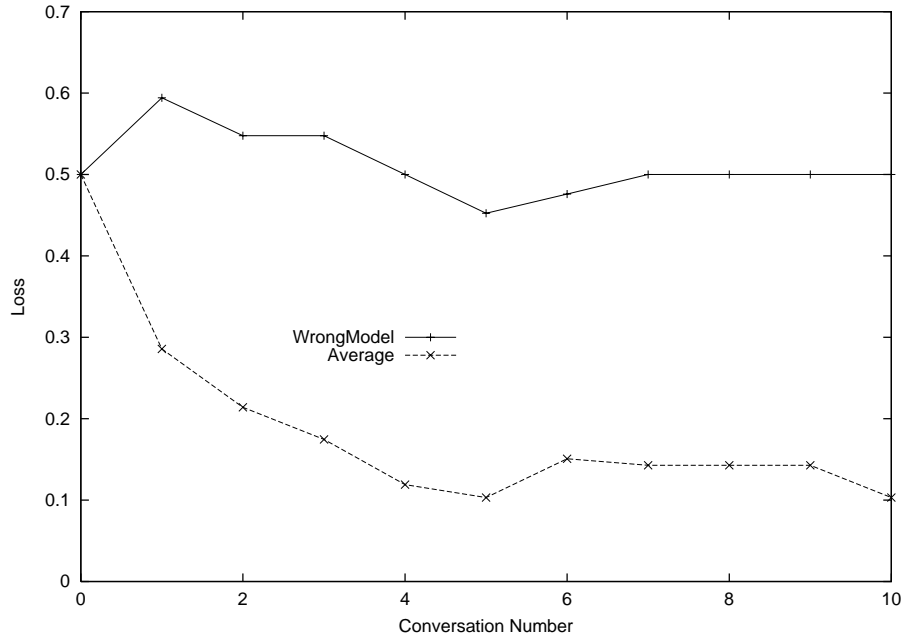


Figure 1: Attribute Ranking Loss

cheap). For the second, if an item or a value has already been suggested in a recent interaction, the user might prefer that it be suggested again only after some time passes. For example, some users want to go to the same restaurant or very similar restaurants every week, while others want more variety. Thus, the variety preferences model the suitability of an item or value at a given time.

The dialogue strategy can also further incorporate information from the user model. For example, we can stop asking questions and skip to item recommendation as soon as the most highly preferred attributes have been asked about. Or, the values of attributes can be assumed or suggested to the user, as in our second sample conversation in Section 2.

## 6 Conclusions

Work in personalization of dialogue systems is rare. We present an approach for adapting conversations to the destination preferences of users, and present encouraging experimental results. Our measure for success differs from those traditionally used for evaluating dialogue systems, and we feel that this appropriately reflects our difference in focus. Differences among users are reflected in the manner in which they carry out conversations, and this work presents an-

other way to capture such differences. Overall, we have begun to make inroads into unobtrusively acquiring an individual, long term user model, with the purpose of adapting to the needs, goals, and preferences of a user over multiple conversations.

## References

- Aamodt, A., Plaza, E. 1994. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AICom - Artificial Intelligence Communications* IOS Press. 7(1) pp. 39-59
- Aha D., Breslow L. Refining Conversational Case Libraries. in Leake D., Plaza E. (eds.) 'Case-Based Reasoning Research and Development, Second International Conference on Case-Based Reasoning ICCBR 1997', pp. 267-278, Springer Verlag, Berlin 1997.
- Carberry, S. Plan Recognition in Natural Language Dialog. Cambridge, MA, MIT Press, 1990.
- W. Cohen, R. Schapire, and Y. Singer. Learning to Order Things. 1998. Michael I. Jordan and Michael J. Kearns and Sara A. Solla, editors, "Advances in Neural Information Processing Systems" 10, The MIT Press.
- Dowding, J., Gawron, M., Appelt, D., Bear, J., Cherny, L., Moore, R., Moran, D. 1993.

- GEMINI: A natural language system for spoken language understanding. In 31st Meeting of the Association for Computational Linguistics, pp. 54-61. Columbus, OH.
- Jurafsky, D., et.al. 1994. The Berkeley Restaurant Project. In Intl. Conference on Spoken Language Processing.
- Kobsa, A., Wahlster, W., eds. 1998. Computational Linguistics, 14(3): Special Issue on User Modeling.
- 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.
- Linden G., Hanks S., Lesh N. Interactive Assessment of User Preference Models: The Automated Travel Assistant. in Jameson A., Paris C., Tasso C. (eds.), 'User Modelling: Proceedings of the Sixth International Conference, UM97', Springer Verlag, Vienna, 1997. pp 67-78
- D. Litman, S. Singh, M. Kerns, and M. Walker. 2000. NJFun: A reinforcement learning spoken dialogue system. In Workshop on Conversational Systems, ANLP/NAACL Conference, 2000, pages 17-20.
- 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.
- Resnick P., Varian H. (eds). 1997. Recommender Systems. Communications of the ACM, Vol. 40, No. 3, March
- Rich, E. 1979. User modeling via stereotypes. Cognitive Science, 3, pp. 329-354.
- Searle J. 1969. 'Speech Acts', New York, Cambridge University 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.
- Singh, S., Kearns, M., Litman, D., Walker, M. 2000. Empirical Evaluation of a Reinforcement Learning Spoken Dialogue System. In Proceedings of the Seventeenth National Conference on Artificial Intelligence, pp. 645-651, Austin, TX.
- Smyth B., Cotter P. 1999. Surfing the Digital Wave, Generating Personalised TV Listings using Collaborative, Case-Based Recommendation. In Althoff K.D., Bergmann R., Branting K. (eds), 'Case-Based Reasoning Research and Development, Proceedings of the Third International Conference on Case-Based Reasoning ICCBR99', pp. 561-571, Springer Verlag, Berlin
- tolcke, A., et.al. 2000. Dialog Act Modeling for Automatic Tagging and Recognition of Conversational Speech. Computational Linguistics, 26(3):339-373.
- M. Walker, D. Litman, C. Kamm, and A. Abella. 1997. PARADISE: A Framework for Evaluating Spoken Dialogue Agents. In "Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL-97)". pp. 271-280, Madrid, Spain.