Critique-based Recommendations for Mobile Users: GUI Design and Evaluation

Quang Nhat Nguyen
e-Commerce and Tourism
Research Laboratory, ITC-irst
via Solteri, n. 38
38100 Trento (TN),Italy
quang@itc.it

Francesco Ricci
e-Commerce and Tourism
Research Laboratory, ITC-irst
via Solteri, n. 38
38100 Trento (TN),Italy
ricci@itc.it

Dario Cavada
e-Commerce and Tourism
Research Laboratory, ITC-irst
via Solteri, n. 38
38100 Trento (TN),Italy
dacavada@itc.it

ABSTRACT

Product suggestions provided by e-commerce recommender systems are very useful when users are overwhelmed by options to consider. Although there exist some successful web-based recommender systems, only a few have been designed for mobile users. In this paper, we focus on the problem of designing a user interface which provides critique-based recommendations to mobile users. The proposed system is a critique-based on-tour recommender system that supports travellers in the selection of their desired travel products, during their move to (or their stay at) the selected destination. The system does not require the user to formulate a complete query at the beginning, but rather collect the user's preferences through his critiques to the products recommended. A graphical user interface of the proposed system has been implemented for Java-enabled mobile phones. An experiment with real users was conducted to validate the system's usability. We present here the lesson learned from the empirical evaluation.

Keywords

decision support, mobile recommender systems, GUI design

1. INTRODUCTION

Both leisure and business travellers need sophisticated system support throughout all travel stages: from pre-travel planning to the on-the-move support during the travel, and even when the travel is finished [12]. The specific system support, in particular the functionality related to service and product selection, strongly depends on the user's travel stage and usage context. In fact, e-commerce travel and tourism portals often contain huge quantities and types of travel products, and hence travellers should be supported in searching for the "right" ones. To accomplish this requirement many web sites have recently incorporated recommendation technologies [11, 1].

The 3rd International Workshop on "HCI in Mobile Guides".
(September 13th 2004, in adjunction to mobileHCI'04, in Glasgow)

One of the major problems still open in designing recommender systems is the collection of user preferences. In the simplest approach, these preferences are collected by asking the user, and letting him specify. The collected preferences are typically a subset of the user's real "needs and wants" because users usually do not like to input data, and GUI designers do not want to bother users with many questions. Given these (few) preferences are encoded in a query to the product catalogue, the query's execution usually yields a large number of options, and therefore identifying the best one(s) could become difficult for user. There are some additional disadvantages of this approach. First, user must have enough of knowledge about the problem domain to be able to formulate a meaningful query (even with a query-by-example interface). Second, the requirement of user's effort in formulating a complete query may be unreasonable; especially for mobile users. The more complex a user's query is the more the user's effort is required. Third, users are not good at comparing amongst candidate products (i.e., those returned by a query). Users usually find it very difficult to take decisions that involve more than one feature. In fact, they must consider simultaneously both 1) trade-offs between different values of the features, and 2) the likelihood that the various outcomes occur.

Another approach, which has recently received much interest, is eliciting user preferences through much structured human-computer dialogues. The goal of a user, when participating in such a dialogue, is to find his desired products; whereas the role of the system is to assist the user to quickly and effectively get those products. In conversational recommender systems, for instance, at each interaction cycle, the system performs one of the two functions, *asking* or *proposing*. The user replies either by *answering* or by *criticizing*. The system raises a selective question when the number of candidate products is unsatisfactory (i.e., no items or too many were found). In the reply, the user may indicate, remove, or modify some constraints so that the system can now retrieve a better result set. Once the number of candidate products is acceptable, the system proposes these products to the user. At this point, the user may either 1) accept one, or 2) criticize one to better specify his preferences. Conversational recommender systems may implement only the asking/answering conversation mode([6]), only the proposing/criticizing conversation mode ([2],[7]), or both ([14],[15]). It is worth noting that a conversational sys-

tem can interact with users either through typing/clicking or with a natural language interface (e.g., [14],[15]). Our discussion in this paper is limited to the first interaction modality.

Although there exist many successful recommender systems for web users, only very few have been designed for mobile users (e.g., [8]). Moreover, to our knowledge, no existing mobile recommender systems are conversational; and most of them run only on PDAs (i.e., Palm or Pocket PC), not on mobile phones (which have smaller screens and limited keypads). In this short paper, we focus on the problem of designing a user interface for mobile phones that supports critique-based product recommendations. In a critique-based recommender system, at every recommendation cycle, the user is allowed to criticize (i.e., to judge) the system's recommendation result. In our system we focussed on mobile users who use a mobile phone to search for some desired travel product (restaurant). This on-tour support system (hereafter called mITR) cooperates with a pre-travel planning system (NutKing). NutKing is a web-based recommender system that supports users in building pre-travel plans [13]. The on-tour support is supposed to be exploited when a mobile traveller, who has possibly built a pre-travel plan, is at the selected destination or on the move to.

We have considered the following requirements, in the design of mITR:

- The products complemented in the on-tour stage should be compatible with those already selected in the pre-travel one.
- Given a user's on-tour request, the travel products recommended to the user should be relevant to his on-tour preferences.
- The user-system interaction should be simple, and the time required to obtain a useful recommendation should be minimized.
- The user should be able to complete the task; i.e., choose and buy a product.

Our working hypothesis was that the user interface (of a mobile recommender system) should be designed so that the user-system interaction is as simple and quick as possible while users still receive good recommendations. We shall describe how difficult it is to manage the trade-off between simplicity, effectiveness and completeness.

The next Section discusses our product representation and user preferences model. Section 2 focuses on how the mITR system produces personalized recommendations to mobile travellers. In Section 3, we firstly present the empirical evaluation of a prototype that implements the proposed methodology, and then discuss on the experiment results that lead to re-design the user interface. Finally, the conclusions are given in Section 4.

2. ON-TOUR RECOMMENDATION

In our model, a travel product is represented as a vector of feature values $x = (x_1, x_2, ..., x_n)$; where a feature value x_i may be numeric, nominal, or symbol-set. For instance, a restaurant may be represented in the 6-dimensional vector space:
$X =< Name, Type, Location, MaxCost, OpeningDays, Characteristics >$. The restaurant $x = (LaBerta, \{pizzeria\}, Trento, 20, \{7, 1\}, \{parking, smoking\_room\})$, for instance, has name $x_1 = LaBerta$, type $x_2 = \{pizzeria\}$, location $x_3 = Trento$, maximum cost $x_4 = 20$, opening days $x_5 = \{7, 1\}$, and characteristics $x_6 = \{parking, smoking\_room\}$.

To produce recommendations personalized for a user, recommender systems need a representation of the user's preferences. Preferences vary from user to user; and even the same user, in different situations (contexts), may have different preferences. User's preferences, in our approach, are represented as a composite query containing three components: 1) a logical query, 2) a favorite pattern, and 3) feature importance weights.

- The logical query models must conditions that need to be independently satisfied by any of the products recommended. The logical query is constructed by the conjunction of logical constraints: $Q_L = (c_1 \wedge c_2 \wedge ... \wedge c_m)$; where c_j is a constraint on a feature. A logical constraint deals with only one feature; and a feature appears in only one constraint. Different feature types have different constraint representations.
- The favorite pattern models wish conditions that are expected to match as much as possible. Differently from must conditions, wish conditions allow trade-offs to be made. The favorite pattern is represented in the same vector space in that travel products are represented: $p = (p_1, p_2, ..., p_n)$; where x_i and p_i belong to the same feature type, $\forall i = 1..n$. A value $p_i =?$ indicates that the value of i-th feature is unknown.
- The feature importance weights model how much each feature is relatively important with respect to the others. $w = (w_1, w_2, ..., w_n)$ is a feature weights vector; where w_i ($\in [0, 1]$) is the importance weight of i-th feature.

For instance, the representation of a user's preferences
$< Q_L = (x_3 = Trento) \wedge (x_5 \supseteq \{7, 1\})$,
$p = (?, \{spaghetteria\}, ?, ?, ?, ?)$, $w = (0, 0.6, 0, 0.4, 0, 0) >$ indicates that the user is interested in only those restaurants in Trento that are open on Saturday and Sunday, and he prefers spaghetti restaurants to the others. The user considers the feature 'Type' as the most important one and the cost feature as the secondly important, whereas the remainder as the unimportant.

An on-tour recommendation session starts when a traveller requests the mITR system to find some particular travel products; and ends when the traveller either 1) selects a travel product (i.e., adds this product to his travel notes), or 2) gives up the current session with no selection. A recommendation session evolves in cycles. A recommendation cycle (sometimes also called a recommendation step) finishes when a recommendation of travel products is produced and shown to the traveller. In our model, an on-tour recommendation session is divided into three stages: 1) initialization,

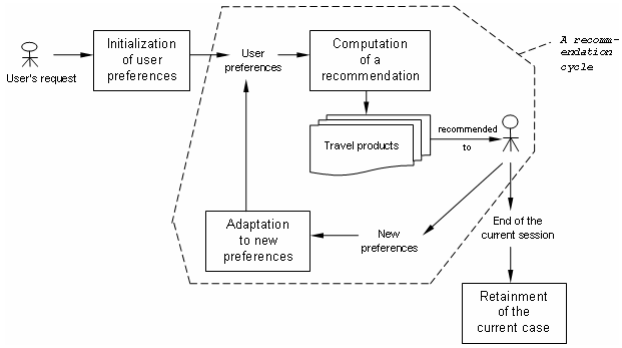

Figure 1: The recommendation process.

2) interaction and adaptation, and 3) retainment; as shown in Figure 1.

Given a mobile traveller's request (e.g., "let me see a good restaurant for this night"), mITR automatically builds an initial representation of the user's preferences. In particular, the logical query (Q_L) is initialized using the contextual constraints extracted from the information about the traveller's position and current time. The favorite pattern (p) is initialized as a blank pattern which has all components unknown. The feature weights vector (w) is initialized so that all the features are equally important.

The mITR system uses this initial representation to compute the first recommendation set. In the computation of a recommendation, the system firstly filters out those products which do not satisfy the logical query (Q_L); and then ranks those Q_L-matched products according to their similarity to the pair $< p, w >$ (i.e., the favorite pattern and the feature weights vector, respectively). The ranking is done so that the more similar to $< p, w >$ a product is the higher that product appears in the ranked list. In case of ties, the cheaper product is ranked higher. Only the k best products in this ranked list (i.e., those which satisfy Q_L and are most similar to $< p, w >$) are shown to the user as recommendation result for the current cycle. The cut-off value (k) depends on the screen size of the travellers' mobile devices. This helps to reduce users' effort, avoiding scrolling.

Having the list of products recommended to the user (see Figure 2a), three situations may occur. In the first situation, the user is satisfied with one of the recommended products. This product is therefore added to his travel notes, and the current session ends successfully. We note that a user's travel notes contain all travel products, services, and information which have been selected by the user for the trip. In the second situation, the user is somewhat interested in one of the recommended products, but one (or some) of its features does not completely satisfy him. Hence, the user is supposed to criticize the product, and to specify his preference on these unsatisfactory features (see Figure 2d). By criticizing, the user at the next recommendation cycle is recommended with other products that are "closer" to his preferences. Such critiques help the system to adapt its previous guess about the user's preferences, and to recompute some new recommendations based on these (adapted) preferences. In the third situation, none of the products recommended

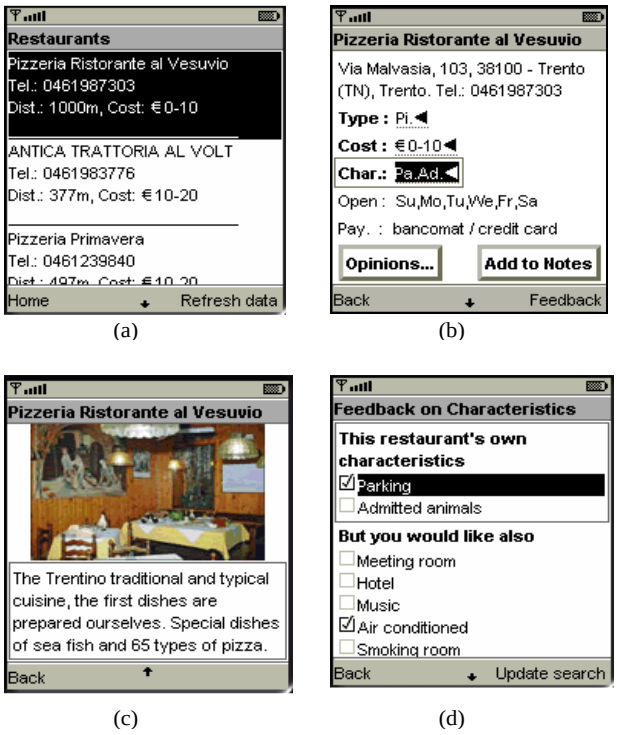

Figure 2: The mobile user interface.

so far satisfy the user, and he does not want to go further. The current session terminates with a failure.

When a recommendation session finishes, regardless the fact that the session terminated with a success or a failure, it is retained as a case for future references. In this way, past recommendation sessions can be exploited by the system in the initialization of user's preferences.

When a user criticizes a recommended product, the user focuses on some particular feature. In our approach, we assume that a critique is related to one of the two kinds of decision criteria: non-compensatory and compensatory ([4]). Non-compensatory criteria represent conditions that should be satisfied completely and independently. Such criteria are encoded (as constraints) in the logical query (Q_L). On the other hand, compensatory criteria represent conditions that would be satisfied to some degree (not necessarily to be completely satisfied), provided that an overall integration function computed on the totality of them is maximized. Compensatory criteria are therefore encoded in the favorite pattern (p) and in the feature importance weights (w) (see [9], for more details about the system's incorporation of user critiques).

The method of eliciting user preferences through critiques has the following advantages: 1) the preferences explicitly collected are much more reliable than those implicitly collected (e.g., by mining a user's navigation), and 2) the user effort is not high (compared to some methods using interviews, early ratings, etc.). Regarding the user-system interaction, a critique to a recommended product is done simply by 2-3 button clicks.

3. EMPIRICAL EVALUATION

3.1 Experiment Procedure

We have performed a usability test to verify the prototype system (mITR) with respect to three dimensions: the functionality, the efficiency, and the convenience [10, 3]. The prototype has been developed using J2ME MIDP 2.0. The experiment involved six users with no experience in WAP- or Java-enabled mobile phones, and was done using a J2ME simulator running on a desktop computer.

The test procedure, which every test user was asked to follow, contains three phases.

- The first phase is for *training*. The tester is introduced to how to use the simulator (e.g., the meaning of the buttons, the meaning of the contextual commands, etc.). Next, he is introduced to the system functionality and usage. In particular, he is provided with a sample of a recommendation session which explains how the system works.

- The second phase is for *testing*. The tester is asked firstly to think about the features of his desired restaurant, and then to try to use the system to find such one. When the tester has found such restaurant, he is asked to add it to his travel notes, and then to open his travel notes to look at that restaurant.

- The third phase is for *evaluating*. The tester is asked to express his (subjective) evaluation of the system's performance, by filling a usability survey. The survey form contains two parts: a pre-defined list of questions, and a free-text space for comment. A seven-point Likert scale (1: strongly agree, 7: strongly disagree) is used by testers for stating their judgement regarding the following statements. (Some of these statements are taken from the Post-Study System Usability Questionnaire [5])
  [s1]. The system was pleasant to use.
  [s2]. The organization of information on the system screen was clear.
  [s3]. This system has all the functions and capabilities that I expected to have.
  [s4]. The information provided by the system was complete.
  [s5]. I've found the restaurant that satisfies my needs.
  [s6]. I've found the possibility to critique a restaurant and get a new sorting of the offers useful and easy to use.
  [s7]. It was simple to use the system.
  [s8]. I was able to efficiently complete the tasks and scenarios using this system.
  [s9]. If the system were available on my phone I would use it.

3.2 Experiment Results

The experiment results are summarized in Table 1. Regarding the fifth question (s5), all the test users stated that they were able to find a restaurant that satisfies their needs. Moreover, all of them rated the found restaurant at the highest rating score (i.e., with a rating of 1).

Table 1: The experiment results.

	user							
Sta.	u1	u2	u3	u4	u5	u6	Ave.	Std.dev
s1	4	1	1	3	1	3	2.17	1.33
s2	3	1	3	2	4	3	2.67	1.03
s3	6	1	1	1	2	3	2.33	1.97
s4	5	2	1	1	2	4	2.50	1.64
s5	1	1	1	1	1	1	1.00	0.00
s6	1	2	7	1	3	1	2.50	2.35
s7	2	2	3	3	4	2	2.67	0.82
s8	1	1	1	1	1	1	1.00	0.00
s9	1	3	1	1	1	1	1.33	0.82
Ave.	2.67	1.56	2.11	1.56	2.11	2.11	2.02	0.42

All the test users were able to complete the evaluation scenario (s8). This result was checked again when we exploited the content of the log file, where we saw that every test user had really added a restaurant to his/her travel notes.

In general, all the test users said that they would definitely use this on-tour support service if available on their mobile phone. This is proved in the testers' ratings to the ninth question (s9).

Mobile travellers need not only to find their desired travel products but also to find them quickly (i.e., after a few recommendation cycles). When mining the log file (which records the test users' recommendation sessions) we found that all testers had been able to find their desired restaurant within 3 recommendation cycles. In the mobile context, this limit is quite acceptable.

However, the experiment results also show some critical points. First, some test users felt a bit confusing with the organization of the information on the user interface. For example, in order to click on the command *"Add to Notes"* on the user interface, users had to navigate (scroll) through several other display objects. Second, the tested system lacked some functions which were considered necessary to many test users (e.g. the route information to the restaurant). Third, not all test users found the user-system interaction simple enough. We will go into details on these critical points in Section 3.3.

Using the log, we also computed the time spent by each tester to complete the test procedure. On average, the test users spent 3-5 minutes for the training phase, and 2-4 minutes for the evaluating phase. The testing phase took much time for every test user. In particular, for this second phase, four test users spent 6-8 minutes, and the other two spent 12-15 minutes. A rather longer time spent was foreseen, since the experiment utilized a mobile phone simulator running on a desktop computer, not a true mobile phone. In the simulation, the real phone's command buttons were mapped to some number keys of the PC keyboard. The user-system interaction through the simulator is more difficult, and takes longer time, than that with a real mobile phone. Looking at the log file, we found two additional reasons for these longer interactions. First, some testers tried to learn the system functionality (especially the critique function) during the testing phase. This means that the training phase was not well done. In other words, some testers were not re-

ally familiar with the system functionality as they advanced in the testing phase. Second, the three testers, who spent the longer time (8-15 minutes) for the testing phase, did not follow exactly the pre-defined task scenario. In fact, they searched (and added to their travel notes) two restaurants, rather than only one (as defined in the task scenario).

3.3 Discussion

In this Section, we present and discuss in detail the critical points stated by some test users. These critical points are summarized from the testers' ratings given in Table 1, and also extracted from their comments collected in the usability survey forms.

- Test users found it difficult to use the additional buttons placed on the screen ("Add to Notes" and "Opinions", shown in Figure 2b).
- Some of them preferred to initialize the search by explicitly specifying some preferences.
- Some of them preferred a longer (even the full) list of recommended products at each recommendation cycle.
- All of them asked for a map-based navigation support to reach the selected restaurant from their current position.

The first point shows that some testers did not understand how to use the command buttons which, in the first design, were embedded in the screen. In traditional web interfaces, to execute a command users typically move their pointing device (i.e., mouse) to that command and activate it. However, some testers did not find how to execute the screen-embedded commands. As shown in Figure 2b, to execute such a command, users are required 1) to navigate through several display objects (using the device's "push-down" button), and 2) to activate the command (using the device's "select" button).

The second point shows that some testers wished (and even wanted) to specify some initial preferences. In a traditional web-based system, when searching for some items users usually state their conditions before the system's retrieval. In our experiment, the system always tries to automatically recommend some tentative products (i.e., the list of the cheapest restaurants close to the user's current position) without asking the user (see Figure 2a). However, some testers did not like this approach. Instead, they wanted to have more control on the initialization of the search process, similarly to traditional search interfaces.

The third point seems to disprove the hypothesis that mobile users need system support in reducing the time and cognitive effort required to fulfil a task. There could be two explanations for that. The first explanation is the difference in user's goals. Some users simply search for an *"acceptable"* item whereas some others want to find the *"best"* one. For the first group, a small recommendation set is ok; for the second group, this is not enough. Hence, it could be difficult to find a balance between the two conflicting goals: convincing a user that one product is better than others while showing short recommendation lists. This is similar to the trade-off between *precision* and *recall* in Information Retrieval. The second explanation is the familiarity of testers with web-based interfaces. Test users were used to browsing the "complete" list of items, and they therefore were surprised when they found a different interface. We intend to better understand this finding in a future test.

The fourth point simply stresses a missing function which should be supported in the next version of the system.

3.4 New Interface

To address the problems mentioned in the previous Section, we designed a new interface as shown in Figure 3.

At the start-up, the system now offers the user three options for search initialization (see Figure 3a).

- To let the system automatically construct the initial representation of the user's preferences by exploiting the contextual constraints, the pre-travel plan, and the past on-tour recommendations.
- To let the user explicitly specify some initial preferences.
- To let the user specify a known product (i.e., the user has consumed) as the starting point of the system's search.

When showing a list of recommended products to the user, for each product the system shows the product's abstract information together with an icon which provides a hint of how close the product is to the user's preferences. This is supposed to help the user (when looking at the recommendation list) to quickly realize 1) the "appropriateness" of each recommended product, and 2) the improvement of the current cycle's recommendation set compared to the previous one's. In every recommendation cycle, the products which have been criticized so far appear at the bottom of the recommendation list. Moreover, to help the user to understand the size of the recommendation list, the system displays at the top of the screen the total number of products recommended (i.e., including also the criticized products).

All the system commands are now grouped into contextual menus. Depending on a particular interaction context, an appropriate set of system commands are available; and these commands are grouped into a single menu. This makes the user interface consistent through the interaction session of a user.

The new interface also provides some necessary functions which were absent in the previous version (e.g., the command *"How to go there?"* shows a static map which guides the user to the selected restaurant).

4. CONCLUSIONS

In this paper, we have introduced a critique-based mobile recommender system which provides on-tour support. We have illustrated the GUI of such a system, and discussed the results of an empirical evaluation. We have, once more,

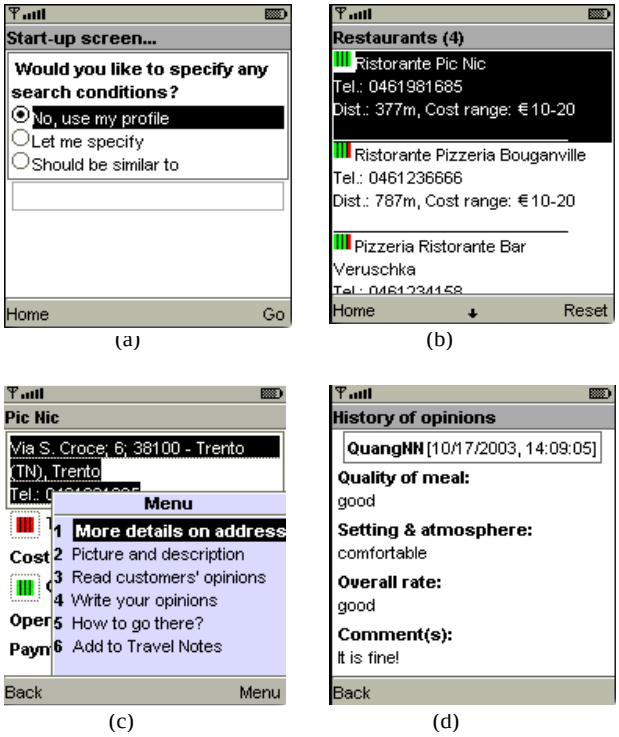

Figure 3: The new interface.

discovered that the simplicity can not be achieved sacrificing effectiveness. Some users like the system to automatically start from a default sorting (e.g., by price), but some others like to specify initial conditions. Users interested in finding the "optimal" product prefer to spend more time and effort for customizing the system's search behavior. They want to be able to set some initial conditions; and hence, to have more control on the system's search process.

Another critical issue relates to the choice of the number of products to be shown at each recommendation cycle. On one hand, this number should be as small as possible to fit into the screen of the mobile device to avoid scrolling up and down, or jumping between screens. On the other hand, a small size of the recommendation list restricts the number of candidate products shown to the user, and therefore decreases the "recall" of the system. In the future, we want to conduct a test to find out the best approach for dealing with this problem.

Almost all mobile users have been using web browser interfaces. Hence, some of them are heavily influenced by the familiarity with the classical interfaces/ways of searching. For such users, a mobile recommender system should both support more traditional approaches and offer the innovative ones, providing a seamless learning path.

5. REFERENCES

[1] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[2] R. Burke. Interactive critiquing for catalog navigation in e-commerce. *Artificial Intelligence Review*, 18(3-4):245–267, 2002.

[3] N. Chin. Empirical evaluation of user models and user-adapted systems. *User Modeling and User-Adapted Interaction*, 11(1-2):181–194, 2001.

[4] W. Edwards and B. Fasolo. Decision technology. *Annual Review of Psychology*, 52:581–606, 2001.

[5] J. R. Lewis. IBM computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use. *International Journal of Human Computer Interaction*, 7(1):57–78, 1995.

[6] G. Linden, S. Hanks, and N. Lesh. Interactive assessment of user preference models: The automated travel assistant. In *Proceedings of the 6th International Conference on User Modeling, UM'97*, pages 67–78, 1997.

[7] L. McGinty and B. Smyth. Deep dialogue vs. casual conversation in recommender systems. In *Proceedings of the Workshop on Personalization in eCommerce, at the 2nd International Conference on Adaptive Hypermedia and Web-Based Systems, AH'02*, pages 80–89, 2002.

[8] B. N. Miller, I. Albert, S. K. Lam, J. A. Konstan, and J. Riedl. Movielens unplugged: Experiences with an occasionally connected recommender system. In *Proceedings of the 7th International Conference on Intelligent User Interfaces, IUI'03*, pages 263–266, 2003.

[9] Q. N. Nguyen, D. Cavada, and F. Ricci. On-tour interactive travel recommendations. In *Proceedings of the 11th International Conference on Information and Communication Technologies in Travel and Tourism, ENTER2004*, pages 259–270, 2004.

[10] J. Nielsen. *Usability Engineering*. Morgan Kaufmann Publisher, San Francisco, 1993.

[11] P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[12] F. Ricci. Travel recommender systems. *IEEE Intelligent Systems*, 17(6):55–57, 2002.

[13] F. Ricci, B. Arslan, N. Mirzadeh, and A. Venturini. Itr: A case-based travel advisory system. In *Proceedings of the 6th European Conference on Case Based Reasoning, ECCBR'02*, pages 613–627, 2002.

[14] H. Shimazu. Expertclerk: Navigating shoppers buying process with the combination of asking and proposing. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence, IJCAI'01*, pages 1443–1450, 2001.

[15] C. A. Thompson, M. H. Goker, and P. Langley. A personalized system for conversational recommendations. *Artificial Intelligence Research*, 21:393–428, 2004.