Evaluating Example-based Search Tools

Pearl H. Z. Pu
Human Computer Interaction Group
Faculty of Information and Communication Sciences
Swiss Federal Institute of Technology
1015 Lausanne, Switzerland
Tel: +41-21-6936081
pearl.pu@epfl.ch

Pratyush Kumar
Human Computer Interaction Group
Faculty of Information and Communication Sciences
Swiss Federal Institute of Technology
1015 Lausanne, Switzerland
Tel: +41-21-6936656
pratyush.kumar@epfl.ch

ABSTRACT

A crucial element in consumer electronic commerce is a catalog tool that not only finds the product for the user, but also convinces him that he has made the best choice. To do that, it is important to show him ample choices while keeping his interaction effort below an acceptable limit. Among the various interaction models used in operational e-commerce sites, ranked lists are by far the most popular tool for product navigation and selection. However, as the number of product features and the complexity of user's criteria increase, a ranked list's efficiency becomes less satisfactory. As an alternative, research groups from the intelligent user interface community have developed various example-based search tools, including SmartClient from our laboratory. These tools not only perform personalized search, but also support tradeoff analysis. However, despite the academic interest, example-based search paradigms have not been widely adopted in practice.

We have examined the performance of such tools on a variety of tasks involving selection and tradeoff. The studies clearly show that example-based search is comparable to ranked lists on simple tasks, but significantly reduces the error rate and search time when complex tradeoffs are involved. This shows that such tools are likely to be useful particularly for extending the scope of consumer e-commerce to more complex products.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *evaluation/methodology, user-centered design.*

General Terms

Performance, Experimentation, Human Factors.

Keywords

Empirical user study, personalized search with decision support, example-based interface, e-commerce, decision tradeoff, product comparison.

1. INTRODUCTION

A crucial element in consumer electronic commerce is a catalog tool that not only finds the product that best matches the user's needs (personalization), but also convinces him that he has made the best choice (decision support). Personalization is believed to play a key role in converting site visitors to buyers in B2C e-commerce environments [6], while decision support has been considered important for choice problems. However, most search tools used today either provide little support for personalization and decision analysis, or require such a significant interaction effort that users stay away from them. By far, the most common tool is that of a ranked list: products that match the initial request are shown in increasing order of a quantitative attribute, most often price. This paradigm has the advantage that it is easy to implement and gives the user an impression of control over the selection process. Ranked lists provide decision support only for one criterion at a time. However, when user's preferences are a combination of multiple and possibly conflicting criteria, the ranked list's efficiency becomes less satisfactory.

A more advanced search tool should provide decision support for any number of criteria. This requires eliciting a preference model from users, including identifying the criteria to be used for evaluation (criteria enumeration) and how they influence the decision outcome (value function). Traditional elicitation approaches have required users to answer a fixed set of need or preference assessment questions in a fixed order. This practice has been found undesirable because users' decision process is highly adaptive [10] and thus their initial preferences can be uncertain and erroneous, they may lack the motivation to answer cognitively and psychologically demanding questions prior to any perceived benefits [18], they may not have the domain knowledge to answer the questions correctly [15], or any combination of these factors. We believe that the preference elicitation process must be an *integral* part of the search process. Our treatment of personalization and decision support is therefore to provide an intelligent interface to help users construct and reveal their true preferences, and resolve conflicting desires.

The following are further explanations of a set of requirements for such a search tool and its interface:

Decision uncertainty comes from not only the adaptive nature of decision, but also users' lack of domain knowledge. For example, even though a user knows that he has to be in Hamburg by 2pm, he may not be able to articulate a preference for the departure time because he has no idea how long the total travel duration is.

Some preferences may become relevant only in certain contexts. For example, a user will not likely express a preference for intermediate airports unless the catalog shows that that all flights transit somewhere. Only then would he evaluate whether Munich, Frankfurt, or London is a more desirable airport for a stopover.

Search tools must also manage preference conflicts. A user who put in a query for a spacious apartment with a low price range and

EC'04, May 17–20, 2004, New York, New York, USA.

got "nothing found" as a reply learns very little about how to state more suitable preferences. On the other hand, showing apartments which partially satisfy his budget and spatial needs explains the compromises required to meet his preferences.

Finally, because users' decision objectives vary and depend on partial search results, their expression of preferences is often not compatible with a fixed order of questions imposed by a search tool. Pu and Faltings [13] explained that a rigid elicitation order would lead users to state preferences for means objectives, rather than fundamental ones. This is considered as a less optimal decision strategy [7]. For example, suppose that a traveler lives near Geneva, and wants to be in Frankfurt by 3:00 pm (his fundamental objective). However, if he was asked to state departure time first, he would have to formulate a means objective. Unfamiliar with the available flights, he can only estimate the correct departure time, say 9 am. In this case, he would have missed the non-stop flight that leaves Geneva at 12 noon, which will get him there before 3:00 pm.

Table 1. A requirements catalog of preference elicitation for decision search tools.

R1:Incremental effort of elicitation *The interface should allow users to make an incremental rather than a one-shot effort in constructing their preferences due to the highly adaptive nature of decision process and user's lack of initial motivation in stating them.*
R2:Any order The interface should not impose a rigid order for preference elicitation.
R3:Any preference The interface should let users state preferences under relevant contexts.
R4:Preference conflict resolution The decision search tool should solve preference conflicts by showing partially satisfied results with compromises.
R5:Tradeoff analysis In addition to search, the system and the interface should help users perform decision tradeoff analysis, such as "I like this apartment, but can I have something cheaper?" or "I like this apartment, but can I find something bigger?"
***R6:Domain knowledge* The system and the interface should reveal domain knowledge whenever possible.**

We have implemented SmartClient [12,20,21], initially known as ATP in [19]. It is a personalized decision search tool for finding travel products. Later on, we applied the same technique to a number of online catalogs such as vacation packages, insurance policies, and rental properties. Central to SmartClient is a user system interaction model called *example critiquing*. In the past years, we have performed usability tests as well as a comparative study for evaluating the client-server architecture of SmartClient [12]. Based on the accumulative experience and user evaluation, we have identified a requirements catalog for a decision search tool and its interface (Table 1). R1 through R3 and R6 were derived from behavior decision theories [3,10,11] and value-focused decision thinking [7], and have been tested in our prototypes. Some were recently confirmed by researchers studying interface design issues from marketing behavior theories [18]. Recently, we were further interested in empirical analysis from the point of view of decision support tasks, that is, evaluating R4 and R5. In particular, do example-based search tools require more time to learn than ranked lists for decision tasks? If so, what is this learning period? After the initial training cost, are example-based tools comparable to ranked lists for search and tradeoff tasks? Do more complex tasks make a difference in the evaluation? What are these tasks? We report in this paper a comparative user study to qualitatively and quantitatively evaluate example critiquing against a ranked list. We expected that the results would also shed light on how to improve example critiquing.

This paper will proceed as follows. We first describe the main interaction component of SmartClient, the example critiquing interface. Then we describe our user study in detail: data sets, characteristics of our subjects, the experimental procedure, user tasks, the main hypothesis, and measured data. We then describe the main finding of this experiment. To make some conjectures on how related example-based search tools would perform, we made some extrapolation of our results along the dimensions of ease of use and tradeoff complexities. We then compare our work with other studies of user interaction issues in e-commerce, example based recommender systems using natural language interfaces, and decision evaluation tools that employ visualization techniques. The section on future work describes our plan to further compare SmartClient with ranked list for larger collections of data, and steps to be taken to improve SmartClient's interface.

2. SmartClient

SmartClient is an example-based search tool for finding flights, and subsequently vacation packages, insurance policies and apartments. Each SmartClient implementation consists of a user interface and a search engine.

The SmartClient interface is based on the example critiquing model (see Figure 4). A user starts the search by specifying one or any number of preferences in the query area. Based on this initial preference model, the search engine will find and display a set of matching results (see Faltings *et al* [5] for the optimal number of displayed solutions based on catalog sizes). The user either accepts a result, or takes a near solution and starts posting critiques to that result. Critiques are small revisions of the current preference values. A user who desires to find a less expensive apartment may compose a critique by clicking on the pulldown menu next to the price and selecting the menu item "less expensive". A user can post one or several critiques simultaneously, for example less expensive, closer, etc. In addition to critiques, the user can also modify the relative importance of a preference by setting the bars underlying each of the attributes in four stages: zero, somewhat important, quite important and very important. A full colored bar indicates his strong desire to respect this preference. An almost-empty bar indicates the contrary. Clicking the "compromise" button will set the bar to the empty-color status, meaning that he is willing to accept a compromised value of this attribute. With the possibility to set the weight of a preference, a user can perform tradeoffs while searching for products. For example, a user who wants a less expensive apartment and is willing to commute can select the pulldown menu option for a "less expensive" apartment, while clicking the checkbox button, "compromise", for distance.

Once a set of critiques has been composed, the system will show another set of matching examples. This query/tweaking completes one cycle of interaction, and it continues as long as users want to further refine the results.

The search engine can be a simple ranking function for multi-attribute products, such as the case for fining apartments. For configurable products, SmartClient employs more sophisticated constraint satisfaction algorithms and models user preferences as soft constraints [1]. More detail on the modeling and search engine issues is provided in [22].

3. USER STUDY

3.1 The Implemented Ranked List

Search tools that use ranked lists to show results are still the most common solution for product search and selection. The earliest ranked lists display products in a list in the increasing order of the price attribute. More advanced versions can rank products on any quantitative attribute, but one at a time. The one used in our study implements the advanced version (see Figure 3 for a screen shot of the interface).

Ranked lists are commonly used to display a search engine's results. The scenario we assumed in this study was that a person has already used a search engine to prune the product space based on his strong preferences. A ranked list was then used to compare and select the final product among a set of uncertain decision parameters.

3.2 Data Set and Subjects Used

The data set originally used in SmartClient dealt with multi-attribute and configurable products in the travel industry. However, we chose to evaluate SmartClient for apartment searches in this study. Firstly, it is easier for our subjects to relate to task scenarios used in apartment searches rather than finding flights because they are not likely to be frequent travelers. Secondly, travel data (price, intermediate airports, routes) undergo frequent changes and therefore cannot remain relevant throughout the duration of a research project (in this case two years). On the other hand, apartment data are relevant for up to three years, especially in countries where rent control is tight. Furthermore, the evaluation of SmartClient for apartment searches would demonstrate that the example critiquing interface is useful not only for travel planning, but also for other industries.

We used two sets of 50 rental properties (student apartments) located in the vicinity of our university. Each of the sets is used for evaluating the ranked list (called RankedList henceforth) or the example critiquing interface (called tweakingUI). The entries in these two sets are not identical to avoid any learning effects when users compare and evaluate the two tools. However, they are equivalent with respect to user tasks. That is, each data set contained at least a correct and similar answer for each of the user questions. The rental properties used in the experiment were based on real data with slight modifications. For instance, each property, regardless of its type, was normalized for the purpose of accommodating one person only.

All of the 22 subjects were recruited from our university (EPFL). Since EPFL does not provide sufficient dormitory rooms for our graduate students, our subjects are likely to be familiar with the search tasks. To make the group as diverse as possible, the subjects were selected from a variety of nationalities and educational backgrounds. They were Swiss, Algerian, American, Indian, Vietnamese, Chinese, and Mexican, and have different educational background (undergraduate students, graduate students, research assistants, and secretaries).

Users were given adequate time to familiarize themselves with the interfaces. The data set used for this warm-up exercise was different from the datasets used for the real experiments. To help them learn how to use the interfaces, users were told to perform a test search, for example finding an apartment for the price of 550 Swiss Francs and an area of 20 square meters.

3.3 Experimental Procedure

Before each user experiment, we explained to each subject the experiment's objectives, the meaning of labels on each of the interfaces, and we told them that we will be recording their task performances. We then gave them 5-10 minutes for trying out the interfaces with test scenarios.

Users were asked to test Interface 1 (RankedList) and Interface 2 (tweakingUI, or example critiquing) by performing a list of tasks. The order of the tools given in the evaluation sequence alternated each time we tested a subject. This was to counter balance any biases that users may develop while evaluating one interface and carrying these biases to the evaluation of the other one.

3.4 User Tasks

3.4.1 Task Analysis: Decision Navigation

A decision maker is rarely content with what he initially finds [14]. Instead, he explores the product space by navigating from one product to others, looking for better deals. With example critiquing interfaces, he can conveniently start the navigation from a shown example, post a critique (e.g., a cheaper apartment), and see a new set of products. We call this process the decision navigation process. More precisely, decision navigation involves finding products having more optimal values on one or several attributes, while accepting compromised values for other attributes. This type of tradeoff is known as attribute value tradeoff. [14] discusses in more detail the types of decision tradeoffs in product search.

As the number of attributes becomes larger, the complexity of the tradeoff task increases. Let us define each tradeoff task as having two variables: (`optimize, compromise`), where `optimize` represents the set of attributes to be optimized, and `compromise` the set of attributes to be compromised. So (`{price}, {size of room}`) denotes that a user wants to get a better price by sacrificing the size of his room. (`{price}, {size of room, distance to work}`) denotes that the user wants to get a better price by sacrificing the size of his room, the distance to work, or both. Furthermore, we use pairs (`x, y`) to specify the complexity of tradeoff tasks. (`1,1`) denotes that one attribute is being optimized, while at the same time another attributed is being compromised. (`1, 2`) denotes the participation of two attributes for the compromising process, and one attribute for the optimization process. It is clear that (`1,1`) entails one single tradeoff scenario, while for the (`1,2`)case, there are three scenarios because there are three ways to compromise two attributes. As the number of variables participating in a tradeoff scenarios increases, the optimize/compromise scenario pairs increase exponentially. For the case of (`1, 3`), there are 7

optimize/compromise pairs. That is, there are 7 different ways to compromise in order to gain on one attribute.

3.4.2 User Task Design

The objective of this experiment was to measure users' task performance and error rates while using two interfaces for decision navigation. We thus started with a rather specific decision goal by asking them to identify the most preferred apartment. Then we asked everyone to navigate from that item, and evaluated how quickly he found answers to a set of tradeoff questions:

1. Find your most preferred apartment.
2. Can you find something closer? You can compromise on one and only one attribute.
3. Can you find something bigger than what you found for question #1? You can compromise on one and only one attribute.
4. Find something which is roughly 100 francs less than the answer to question #1. You can compromise on up to two attributes, but not more.
5. Find an apartment which is 5 square meters bigger than the answer to question #1. You can compromise on up to two attributes but not more.

The questions can be broadly divided into three categories. The first question is a simple search task of finding a multi-attribute product from a list of products. This question on one hand ensures that we get an idea of the user's comfort level with the interfaces; it also gives us a starting point for answering subsequent tradeoff questions. The second category of questions (Question 2, 3) deals with multi attribute tradeoff tasks with one attribute in each direction of gain and compromise, i.e., the `(1,1)` tradeoff case. The third category of questions (Question 4, 5) deals with making tradeoffs when we gain on one attribute, and compromise on two attributes, i.e., the `(1, 2)` case.

The entire user study was carried out in experiments scheduled in three phases, with 11, 5, and 6 subjects involved in each of the phases respectively.

3.5 Post Study Questionnaire

After each user evaluation, we asked subjects whether they were more satisfied with RankedList than tweakingUI or vice versa, and why. We also asked them to express any opinions they may have regarding the interfaces, such as which interface gave them higher confidence when answers were found.

3.6 Main Hypothesis and Measured data

Our main hypothesis was that users will not take more time, nor make more mistakes while performing the given tasks using RankedList than tweakingUI. The task completion time was defined to be the amount of time a subject took to answer each of the questions. We also measured the error rate, which was defined to be the total number of wrong answers a subject gave over the total amount of questions.

4. ANALYSIS OF DATA

4.1 Multi Attribute Searching Task

The first question required users to find an apartment of their choice. Our data showed that there were no significant improvement of the response time for answering that question using tweakingUI than RankedList (chi square p=0.617). Furthermore, there were no errors recorded in either interfaces. A number of individuals took longer time to fine the answer while using tweakingUI than RankedList. We believe that this was largely due to the fact that subjects took longer time to learn to use tweakingUI, especially under testing conditions.

4.2 Trade-off with 2 attributes

These questions (#2 and #3) required the subjects to achieve a more optimal value on one identified attribute while compromising the values of one of the four remaining attributes. The improvement for response time in using tweakingUI was not significant (p = 0.617), although the error rate for RankedList was much greater (strong significance p <0.001). We concur that the relatively high error rate was due to the fact that subjects had to do a significant amount of visual search using RankedList, hence they were more susceptible to make mistakes.

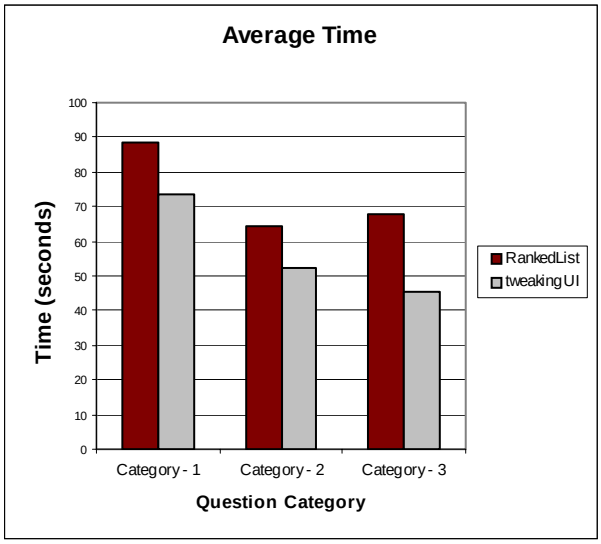


Figure 1: average task completion times in seconds for the three categories of tasks when evaluating RankedList and tweakingUI respectively.

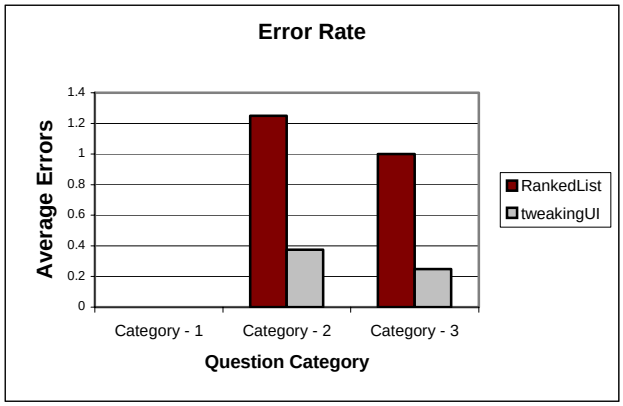


Figure 2: Average error rates for the three categories of tasks when evaluating RankedList and tweakingUI respectively.

4.3 Trade-off with more than 2 attributes

This set of questions (#4 and #5) increased the complexity of human decision making process by engaging them to perform tradeoffs on more than two attributes. An analysis of the statistics gave interesting observations. Not only there was a decrease in average performance times in using tweakingUI as compared to RankedList ($p < 0.001$), but the overall error rate also dropped by a significant margin (approx 75%).

The overall performance of these two interfaces indicates that users took increasingly less time to perform tradeoff tasks in tweakingUI even though task complexities have increased, indicating that learning is a worthwhile investment. On the contrary, the task completion time for RankedList increased as tradeoff tasks became more complex. The obtained data thus suggests that tweakingUI provides a useful tool for making multi-attribute tradeoff functions, especially as the complexity of tradeoff tasks increases. In addition, by observing subjects interacting with the interfaces, we noticed that the tradeoff tasks were made significantly easier in tweakingUI because users can just set an attribute to the "compromise" value when they were willing to sacrifice it, and concentrate on the preferences whose values are to be optimized. Making decisions in such a scenario becomes easier as compared to using RankedList, where decision navigation was done via visual scanning of the data in the list.

4.4 User satisfaction and confidence level

16 subjects participated in the first and second stages of testing. 10 of them were more satisfied with RankedList, while the remaining 6 were more satisfied with tweakingUI. When they were told to provide the reasons for preferring RankedList over tweakingUI, they said that it was easier to use RankedList, and they felt more in control. For the second batch of subjects (6), we gave them more time to get familiar with tweakingUI, and put more effort in explaining the interface. The result was that 4 out of 6 subjects strongly preferred tweakingUI, while 2 only slightly preferred tweakingUI. None of the 6 subjects preferred RankedList over tweakingUI. These results indicate that ease of use influences a user's satisfaction level of the interface. It also points out that if a new interface is not easy to learn, there is a decreased chance that users will prefer it over a traditional one.

A somewhat surprising result came from subjects commenting on their confidence level when a solution was found. Even though fewer of them committed errors using tweakingUI, more of them expressed a higher confidence level with answers found using RankedList. That is, they felt more certain that they found the correct answer in RankedList. A recurring comment was that "the search engine hides something from me," whereas "I can see everything in the ranked list." However, when asked whether they would still feel that way if the data set were very large, they all responded negatively.

5. RESULTS EXTRAPOLATION

5.1 Usability Criteria

In light of the comparative user study described, we were able to make some preliminary conjectures about usability and task performances of various example-based critiquing interfaces such as FindMe, Apt Decision, and ATA (Table 2). Since our findings indicate that a user's satisfaction is reduced significantly by an interface which is unfamiliar and hard to operate, our first criteria for the comparison was ease of use. Secondly, we evaluated whether it is possible to perform value tradeoffs by directly manipulating the value functions (weights) of an attribute. This feature allows the user to make decision tradeoffs by setting certain attributes to the "compromise" value (i.e., zero weight) and other attributes (whose values are to be optimized) to higher weights. Lastly, we assessed whether the interface enables simple tradeoffs, i.e., the `(1,1)` case, and more complex tradeoffs, i.e., the `(1,2)` and `(1,3)` cases.

5.2 Performance Comparison of Example-based Search tools

FindMe [2] aims at providing knowledge support to end-users as they find their way through a large information space. This approach has been implemented in various online product search tools for renting apartments, choosing restaurants, finding cars, selecting videos, etc. An important element in FindMe is tweaking, an interaction model that enables users to navigate to alternatives based on examples. Tweaking starts with a product found by the system based on the initial set of preferences. Once a user selects an almost ideal solution, he can post small changes (tweaking), and view more results. That is, he can find apartments that are cheaper, bigger, closer, or based on any combinations of those criteria. Furthermore, FindMe explains tradeoffs conflicts. For example, if a user wants both a fuel-efficient and high-powered car, FindMe attempts to illustrate the tradeoff between horsepower and fuel efficiency.

We were able to evaluate the most well known FindMe system, Entrée[1] (a restaurant recommender), by performing a cognitive walkthrough on an online version. Example restaurants were shown after an initial query. Choices of critiquing, such as less expensive, quieter restaurants, were clearly presented to the user. After each critique, another set of examples was shown. We found the FindMe system easy to use. However, we are not sure how novice users will judge FindMe. It provided decision tradeoff in the (1,1) case via simple critiques such as less expensive restaurants. It did not, however, allow the direct manipulation of the value function of an attribute, nor provided functionalities for complex tradeoff tasks.

SmartClient, initially known as ATP [20], was developed around the same time as FindMe and ATA. It went through a number of user studies. The comparative study reported here aimed at measuring user performance for tradeoff tasks between SmartClient's example critiquing interface and a ranked list. In 2000, we conducted a comparative study [12] to measure the number of server contacts and system's response time as 43 users were told to plan four trips using SmartClient and Travelocity.[2] At the same time, a usability test was done which showed that users (mostly undergraduate students) were at ease with preference posting and example critiquing operations in the travel version of SmartClient (Figure 5). However, when secretaries (reserving trips for professors) were studied, a small training period of 10-20 minutes was found necessary. Thus we rated SmartClient easy/medium on usability. In the tradeoff study, we noticed that many users concentrated on the attributes to be optimized. They

[1] http://dent.infolab.nwu.edu/infolab/projects/projectmain.asp

[2] http://www.travelocity.com

initially ignored the use of the compromise feature. After examples were shown, they realized that clicking on the compromise button returned better results. Once they were familiar with manipulating attribute's weight, their performance on tradeoff tasks increased significantly.

Apt Decision [16] is an apartment search and decision support tool. It uses learning techniques to synthesize users' preference models by observing their critiques of apartment features. Users browse through the shown examples to discover new features of interest. The system then revises their preference models accordingly. Its main objective is profiling and predicting users' preferences in apartment search for subsequent interactions. Thus the goal is observing user behavior and making inferences about regularity. The emphasis is on adaptive decision making, although the authors also believe that example critiquing is an effective interaction model to elicit hidden preferences for tradeoffs.

We were unable to evaluate Apt Search's interface because it was not available online. From the interface screenshots published in the article, we conjecture that it is rather easy to use. According to the article, the only way a user can perform tradeoff analysis was to create several profiles. Thus we rated it hard, meaning that it is difficult to perform both types of tradeoffs.

Linden et al [8] described a decision support system for finding flights. Initially only few user preferences need to be expressed. The ATA system (automated travel assistant) uses a constraint solver to obtain several optimal solutions. Five of them are shown to the user, three optimal ones in addition to two extreme solutions (least expensive and shortest flying time). User preferences are modeled as soft constraints in the CSP formalism. To elicit hidden preferences, ATA uses a candidate critiquing agent (essentially an example-based search), which constantly observes user's modification to the expressed preference, and refines the preference model in order to improve solution accuracy. From reading [8], it seemed that posting preferences was easy to perform. However, we did not find much description on how the critiquing agent was used and what the interface looked like. We were thus unable to judge ATA on the three accounts.

The overall comparison of these systems suggests that SmartClient is the only tool that enables complex tradeoffs, a service that can potentially convince online users to switch to decision search tools because of the significant performance gain.

Table 2. Comparison of example-based interfaces and prediction of their performances. [3]

	Ease of Use	Access to Weights	Simple Tradeoff	Complex Tradeoff
FindMe	Easy	✗	✓	✗
SmartClient	Easy/Medium	✓	✓	✓
Apt Search	Easy	✓	Hard	Hard
ATA	Unsure	✓	Unsure	Unsure

[3] Legend: ✗ means that this function is not provided; ✓ means that this function is enabled; hard means that it is rather difficult to perform this function.

6. RELATED WORK

VideoAdvisor [9] uses case-based reasoning techniques to complete the preference structure of a partially established model. When a new user expresses a partial set of preferences, the system will match this preference structure with an existing user in order to give recommendations of movies. Similar to SmartClient, this work uses utility theory to represent user's preferences. A major difference, however, is that preferences are inferred in VideoAdvisor, while SmartClient emphasizes preference construction. Inferred preferences may be valuable to a recommendation system of movies, they are less likely to help buyers select and perform tradeoff analyses of high involvement products. Buyers who do not participate in the preference construction process are not likely to accept products based on inferred preferences. Similar remarks were made in [3,11].

The ExpertClerk system [17] was designed to imitate the interaction between salesclerks and shoppers. Analyses of a conversational corpus and interviewing senior salesclerks indicated that a good salesman typically alternates between asking questions and proposing sample goods to understand a customer's buying points. This model of conversation was them implemented in ExpertClerk as a two-stage interaction. The first stage, navigation by asking, calculates the information gain of possible questions and sorts them according to statistical efficiency. The second stage, navigation by proposal, presents three sample goods with explanations of their selling points to a shopper.

The method used to generate questions does not take into account user's domain knowledge level and may ask a question for which certain users could not answer. In this case, he is likely to quit the interaction or give a wrong answer. Consequently, this wrong answer may lead the subsequent interaction in the wrong direction. Furthermore, the system does not address decision tradeoff, although some proposed products may happen to be contrasting examples of a tradeoff scenario (such as an expensive silk jacket vs. an inexpensive polyester one). ExpertClerk proposes exactly three items to the shopper for selection. According to our experience [5], this number is not sufficient, especially when a large catalog is involved.

Stolze [19] described three subcomponents of a product selection tool: filtering, visualization, and evaluation. It was further remarked that the evaluation process plays a crucial role in convincing users in the selection process. ScoreCat not only ranks products according to user's preferences as was done in SmartClient, it also visualizes the scoring mechanism by displaying how each attribute of products scores in relation to the user's preference model. Such scoring tables provide more detailed sensitivity analysis and augment users' confidence level for what they have selected. Because our users remarked a rather low confidence level for evaluating a decision in SmartClient, we will soon incorporate visualization techniques in the GUI and evaluate the new prototype for further findings.

Previous work has sought to understand the reason for the delayed adoption of advanced search tools by online consumers. Psychological and social factors were some of the initial explanations. In addition, since users' preferences have to be elicited in decision search tools in order to compute the best matches, many users fear that they are revealing private information. As studies indicated that privacy concerns significantly impede an online shopper from making the final

purchase decision [4], it follows that decision search tools may find themselves in a disadvantage compared to simple ranked lists. Spiekermann and Paraschiv [18], for example, proposed to design decision support interface systems with a whole range of risk dimensions in mind: social, psychological, functional, financial, as well as delivery. A detailed set of recommendations mainly concerning the design of user system dialogs were proposed. Several of these recommendations were already implemented in SmartClient [13-15] (see also Table 1). For instance, SmartClient offers a user to make interaction efforts so long as he wants more accurate search results. This any-effort interaction model provides a permanent "opt-out" option for less accuracy-driven as well as expert users. The fact that users can express search criteria in any order, on any preferences, and at any time during the interaction, SmartClient proves to be very adaptive to user's readiness in terms of when they want and can reveal information, their purchase context, and their knowledge level.

7. FUTURE WORK

While the results of our user studies point in a positive direction, more work needs to be done for an extensive user study and a more suggestive design for the user interface.

We plan to increase the size of the database from the current count of about 100 apartments to about 300-400. Our goal is to include more cases involving tradeoffs and evaluate example interfaces when the underlying database is larger. We also plan to extend the findings from this experiment towards designing a more general framework of decision tradeoff analysis for configurable products. Configurable products not only provide a more complex domain in terms of the number of available products, but also can make tradeoff analysis more interesting and relevant as the user has potentially many choices of values for each attribute. For the current apartment case, each attribute has two to four values. For general configurable products, this number can be much larger and lead to many more choices that are impossible to show in a RankedList. In such a scenario, we expect the tweakingUI to have a significant edge over RankedList. This is because the user can explore the decision outcomes as they critique examples, whereas in RankedList, the system may have to fetch many new products from a database each time a user posts a set of tradeoff queries.

Some improvement of the SmartClient interface was carried out during the user study. Users had the most difficulty in understanding the manipulation of weights and its effect on tradeoff analysis. After several trial and errors, we settled on using labeled buttons "compromise" together with a bar. We plan to perform an exclusive formative evaluation for the tradeoff interface part.

8. CONCLUSIONS

This paper presented example critiquing, an interaction model used in SmartClient, as well as in other decision search tools such as FindMe and ATA. To analyze the qualitative and quantitative performance of example critiquing, a comparative study was conducted and described in this paper. The important finding was that example critiquing performs only marginally better compared to a ranked list for search and simple tradeoff tasks. As the complexity of tradeoff tasks increases, the performance of a ranked list degrades significantly, not only in terms of average completion time, but also error rate, and example critiquing's performance becomes much stronger, overcoming the initial learning cost. This provides, to our knowledge, the first empirical proof that example-based search is a viable tool for enabling complex consumer e-commerce scenarios that are up to now impossible to implement.

9. ACKNOWLEDGEMENT

We thank the Swiss National Science Foundation for sponsoring the reported research work. We are grateful to the participants of our user studies for their patience and time. We are thankful to the anonymous reviewers of this paper for their constructive comments that helped us reorganize this paper. Li Chen from the human computer interaction group at EPFL helped with conducting the reported user studies. Finally, we thank Anne Standley, Jiyong Zhang and Boi Faltings for contributing to the final editing of this paper.

10. REFERENCES

[1] Bistarelli, S., Montanari, U., and Rossi, F. Semiring-based Constraint Solving and Optimization. *Journal of the ACM, 44*, 2 (Mar. 1997), 201-236.

[2] Burke, R., Hammond, K., and Young, B. The FindMe Approach to Assisted Browsing. In *IEEE Expert: Intelligent Systems and Their Applications, 12*, 4 (Jul. 1997), 32-40.

[3] Carenini G. and Poole D. Constructed Preferences and Value-focused Thinking: Implications for AI research on Preference Elicitation. *AAAI-02 Workshop on Preferences in AI and CP: symbolic approaches*, Edmonton, Canada, 2002.

[4] L.F. Cranor, J. Reagle, and M.S. Ackerman. *Beyond Concern: Understanding Net Users' Attitudes About Online Privacy.* In Ingo Vogelsang and Benjamin M. Compaine (editors). *The Internet Upheaval: Raising Questions, Seeking Answers in Communications Policy*. The MIT Press, Cambridge, Massachusetts, 2000, 47-70.

[5] Faltings, B., Torrens, M., and Pu, P. Solution Generation with Qualitative Models of Preferences, *International Journal of Computational Intelligence and Applications*, 2004. To appear in 2004.

[6] Fink, J., and Kobsa, A. A Review and Analysis of Commercial User Modeling Servers for Personalization on the World Wide Web. *User Modeling and User-Adapted Interaction, 10*, 3-4 (2000), 209-249.

[7] Keeney, R. *Value-Focused Thinking: A Path to Creative Decision Making.* Harvard University Press, 1992.

[8] Linden, G., Hanks, S., and Lesh, N. Interactive assessment of user preference models: The automated travel assistant. *Proceedings of User Modeling '97*, 1997, 67-78.

[9] Nguyen, H., and Haddawy, P. The Decision-Theoretic Video Advisor. In *workshop notes, Recommender Systems, the Fifteenth National Conference on Artificial Intelligence (AAAI'98),* 77-80.

[10] Payne, J.W., Bettman, J.R., and Johnson, E.J., *The Adaptive Decision Maker*. Cambridge University Press, 1993.

[11] Payne, J.W., Bettman, J.R., and Schkade, D.A. Measuring Constructed Preference: Towards a Building Code. *Journal of Risk and Uncertainty, 19*, 1-3 (1999), 243-270.

[12] Pu, P., and Faltings, B. Enriching Buyers' experiences: the SmartClient Approach. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (CHI '00) (The Hague, The Netherlands, April 1-6, 2000). ACM Press, New York, NY, 2000, 289-296.

[13] Pu, P., Faltings, B., and Torrens, M. User-Involved Preference Elicitation. In *workshop notes, workshop on Configuration, the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03)*, 2003.

[14] Pu, P., Kumar, P., and Faltings, B. User-Involved Tradeoff Analysis in Configuration Tasks. In *workshop notes, the Third International Workshop on User-Interaction in Constraint Satisfaction, Ninth International Conference on Principles and Practice of Constraint Programming (CP2003)*.

[15] Pu, P. and Faltings, B. User Preference Elicitation Via Tradeoff. Submitted to the *International Journal of Constraints*, December, 2003.

[16] Shearin, S., and Lieberman, H. Intelligent Profiling by Example. *Proceedings of the 6th international Conference on Intelligent User Interfaces* (Santa Fe, New Mexico, USA, 2001) ACM Press, New York, NY, 2001, 145-151.

[17] Shimazu, H. 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)* (Seattle, Washington, USA, August 4-10, 2001).

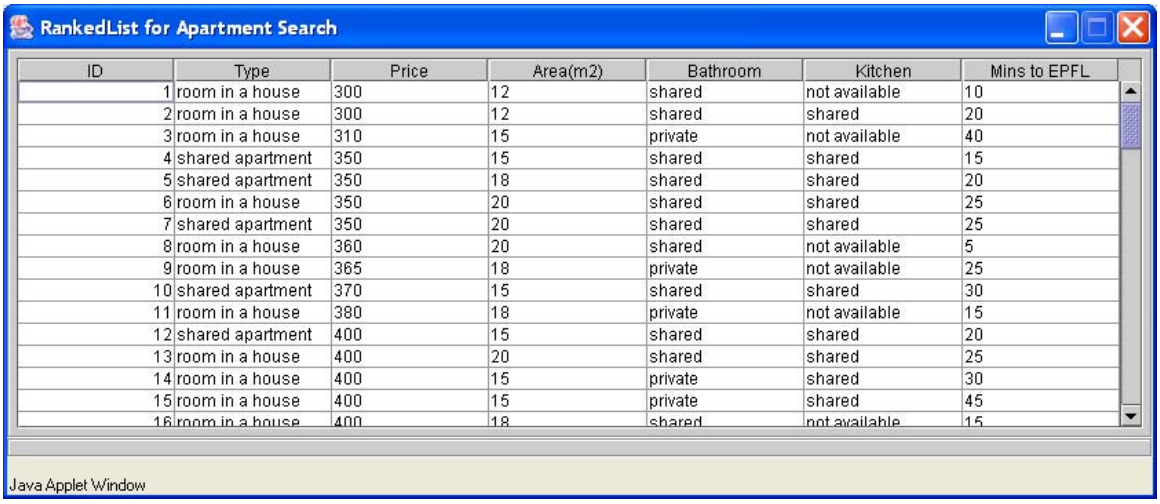
[18] S. Spiekermann and C. Paraschiv. Motivating Human-Agent Interaction: Transferring Insights from Behavioral Marketing to Agent Design. *Electronic Commerce Research, Kluwer Academic Publishers, the Netherland*s, (2002), 255-285.

[19] Stolze, M. Comparative Study of Analytical Product Selection Support Mechanisms, *Proceedings of INTERACT 99*, Edinborough, UK, August 30 – September 3, 1999.

[20] M. Torrens, R. Weigel and B. Faltings. Java Constraint Library: bringing constraints technology on the Internet using the Java language. In *workshop notes, Constraints and Agents, the Fourteenth National Conference on Artificial Intelligence* (AAAI'97), July 1997.

[21] M. Torrens and B. Faltings. SmartClients: Constraint satisfaction as a paradigm for scaleable intelligent information systems. In *workshop notes, Artificial Intelligence for Electronic Commerce, the Sixteenth National Conference on Artificial Intelligence* (AAAI'99), July 1999, 10-15.

[22] Torrens, M., Faltings, B., and Pu, P., SmartClients: Constraint Satisfaction as a Paradigm for Scaleable Intelligent Information Systems. *International Journal of Constraints, 7, 1* (Jan. 2002), 49-69.



ID	Type	Price	Area(m2)	Bathroom	Kitchen	Mins to EPFL
1	room in a house	300	12	shared	not available	10
2	room in a house	300	12	shared	shared	20
3	room in a house	310	15	private	not available	40
4	shared apartment	350	15	shared	shared	15
5	shared apartment	350	18	shared	shared	20
6	room in a house	350	20	shared	shared	25
7	shared apartment	350	20	shared	shared	25
8	room in a house	360	20	shared	not available	5
9	room in a house	365	18	private	not available	25
10	shared apartment	370	15	shared	shared	30
11	room in a house	380	18	private	not available	15
12	shared apartment	400	15	shared	shared	20
13	room in a house	400	20	shared	shared	25
14	room in a house	400	15	private	shared	30
15	room in a house	400	15	private	shared	45
16	room in a house	400	18	shared	not available	15

Figure 3: The RankedList Interface. Apartments are currently ranked in the ascending order of their prices.

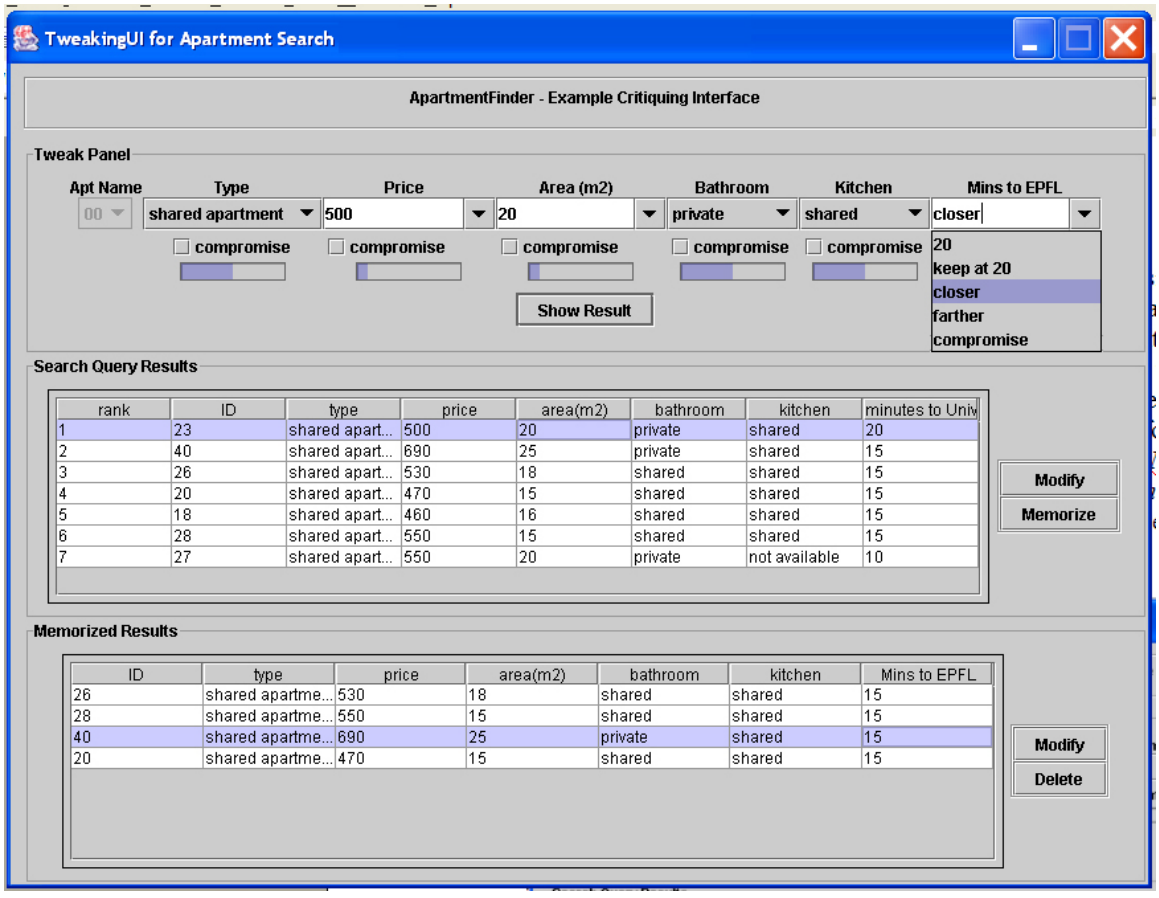


Figure 4: The front end of SmartClient, a personalized decision search tool. Shown in the "tweak panel" is an example just selected from the "search query results" panel. It can serve as a starting point for tradeoff queries. For example, a user can post a critique, "closer," for the distance attribute as shown in the dropdown menu, while compromising on price and area attributes.

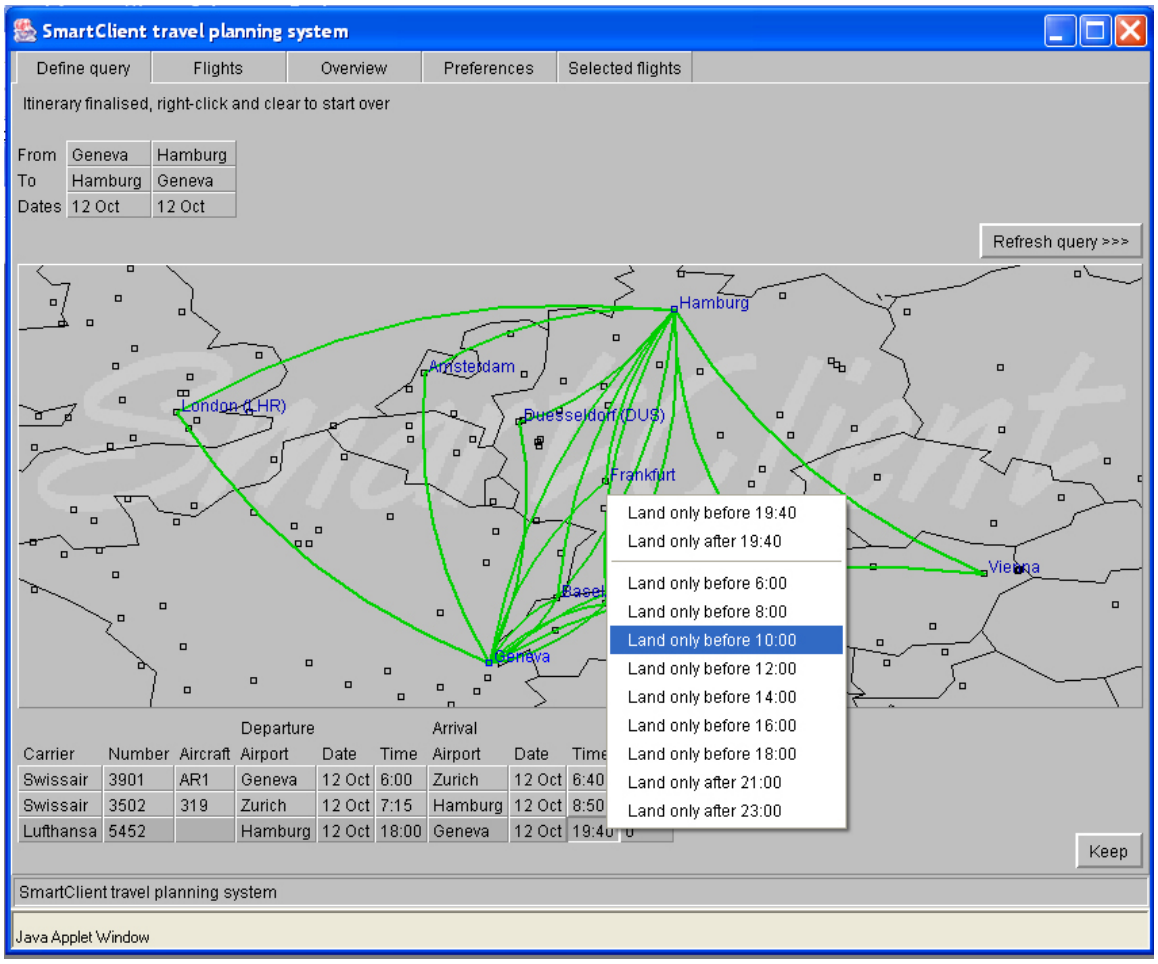


Figure 5. Example critiquing interface used in SmartClient for travel planning.