EPFL Technical Report IC/2004/67

Survey of Preference Elicitation Methods

Li Chen and Pearl Pu

Human Computer Interaction Group
Ecole Politechnique Federale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
{li.chen, pearl.pu}@epfl.ch

Abstract

As people increasingly rely on interactive decision support systems to choose products and make decisions, building effective interfaces for these systems becomes more and more challenging due to the explosion of on-line information, the initial incomplete user preference and user's cognitive and emotional limitations of information processing. How to accurately elicit user's preference thereby becomes the main concern of current decision support systems. This paper is a survey of the typical preference elicitation methods proposed by related research works, starting from the traditional utility function elicitation and analytic hierarchy process methods, to computer aided elicitation approaches which include example critiquing, needs-oriented interaction, comparison matrix, CP-network, preferences clustering & matching and collaborative filtering.

Keywords: preference elicitation, decision support, multi-attribute utility theory, value function, Analytic Hierarchy Process, tweak, example critiquing, recommender systems, CP-network

1. Introduction

Since the goal of decision support system is to assist users with making decision, it is especially important for them to accurately model the user preferences. If no information is available at the start of interaction, preference elicitation techniques must attempt to collect as much information of users' preferences as possible so that the systems can help users working toward their goals. Because user preferences are always incomplete initially, and tend to change in different context, in addition to user's cognitive and emotional limitations of information processing, preference elicitation methods must also be able to avoid preference reversals, discover hidden preferences, and assist users making tradeoffs when confronting with competing objectives.

The theoretical basis of user preference models can be found in decision and utility theory. Multi-attribute utility theory focuses on evaluation of choices or outcomes for a decision problem. Outcomes are defined by the assignment of values to a set of attribute variables $X = \{X_1, ..., X_n\}$. Attribute variables are either discrete or continuous. The outcome space composed of the space of all possible outcomes is the Cartesian product of $\Omega = \{X_1 \times X_2 \times ... \times X_n\}$. The set of outcomes O considered for a decision problem is contained by Ω. It is common for O to be very large. In order to make decisions based on O, a decision maker often needs a ranking of all outcomes determined by preferences. This is called a

preference relation. Typically, the preference relation under decision problem of certainty is induced by a real-valued function, $v(o): O \to R$. The value function reflects the decision maker's preferences on a particular outcome. In case of uncertain decision scenarios, where the outcomes are characterized by probabilities, a more complex function, utility function, is need to evaluate the "utility" of a decision. The utility function represents the user's attitudes about risk as well as the value of outcomes, so it induces a preference ordering on the probability distributions over the outcome space. When assigning values for an outcome *o*, the utility function *u* must consider the uncertainty of attaining *o* and the user's attitudes toward risk to correctly preserve the user's preference relation for actions.

Given the fact that value (utility) function elicitation over large amount of outcomes is typically time-consuming and tedious, many decision support systems have made various assumptions concerning preferences structures. The normally applied assumption is additive independence [1], where the value (or utility) of any given outcome can be broken down to the sum of individual attributes. The assumption of independence allows for the reduction of the number of outcomes for consideration and the construction of less complicated and more manageable value functions.

However, in many cases, attributes are preferentially dependent and thus assumptions of decomposability are incorrect. In order to elicit full value (or utility) function as well as save user's effort as much as possible, some research works have proposed to elicit the preferences of a new user using the closest existing preference structures as potential default. They don't make any restrictive assumptions on the form of the underlying value functions, but make assumptions about the existence of complete or incomplete preference structures elicited from a population of users.

This paper is a survey of various preference elicitation methods mentioned above. Section 2 describes the traditional elicitation methods which mainly query users about the behavior of value function, or relative importance of every outcome in terms of each decision criterion. Section 3 introduces several decision support systems which made various assumptions concerning preference structures in order to reduce elicitation overhead. Section 4 is a brief summary of preference elicitation methods which refine new user's preference based on other users' preference structures. The collaborative filtering approach, which has been extensively used in e-commerce web site, is also introduced in this section. Finally a conclusion will be given.

2. Traditional Elicitation Methods

2.1 Value Function Elicitation

Keeney and Raiffa provide a procedure of eliciting additive independent value function by creating scales for each component of the value function and querying the user about the behavior of each sub-value function [1]. Let's first see the definition of additive independence:

Preference Independence: A set of attributes $Y \subset X$ is preferentially independent of its complement X-Y when the preference order over outcomes with varying values of attributes in Y does not change when the attributes of X-Y are fixed to any value.

Mutual Preferential Independence: The attributes $X = \{x_1,..., x_n\}$ are mutually preferentially independent if every subset Y of X is preferentially independent of its complementary set.

Theorem of Additive Value Function: Given attributes $X = \{x_1,..., x_n\}$, $n \geq 3$, an additive value function $v(x_1,..., x_n) = \sum_{i=1}^{n} l_i v_i(x_i)$ (where v and v_i are scaled from zero to one, and $\sum_{i=1}^{n} l_i = 1,\ l_i > 0)$ exists if and only if the attributes are mutually preferentially independence.

Additive Independence: If the value function can be wrote as additive model, namely the condition of mutually preferentially independence is met, the attributes are said to be additive independent.

The assessment of the additive value function only needs to determine the component value function of each attribute $v_i(x_i)$ and the component scale constant l_i. Here is given the concrete procedure of assessing the additive value function for two attributes [1].

Let the range of X be $x_0 \leq x \leq x_1$, of Y be $y_0 \leq y \leq y_1$, and assume the value function v can be expressed in the form $v(x, y) = l_1 v_X^*(x) + l_2 v_Y^*(y)$, where $v_X^*(x_0) = 0\ and\ v_X^*(x_1) = 1$;

$v_Y^*(y_0) = 0\ and\ v_Y^*(y_1) = 1;\ l_1 > 0,\ l_2 > 0,\ and\ l_1 + l_2 = 1$.

The assessment procedure is as follows:

1. Obtain v_X^* as follows.
   1) Find the midvalue point of $[x_0, x_1]$; call it $x_{.5}$ and let $v_X^*(x_{.5}) = .5$.
   2) Find the midvalue point $x_{.75}$ of $[x_{.5}, x_1]$ and let $v_X^*(x_{.75}) = .75$.
   3) Find the midvalue point $x_{.25}$ of $[x_0, x_{.5}]$ and let $v_X^*(x_{.25}) = .25$.
   4) As a consistency check, ascertain that $x_{.5}$ is the midvalue point of $[x_{.25}, x_{.75}]$; if not, juggle the entries to get consistency.
   5) Fair in the v_X^* curve passing through points (x_k, k) for $k = 0,1,.5,.75,.25$ and perhaps additional points obtained by a midvalue splitting technique.
2. Repeat the same process for v_Y^*.
3. Find the scale factors l_1 and l_2:

   Choose any two (x, y) pairs that are indifferent, for example, (x', y') and (x'', y''), then $v(x', y') = v(x'', y'')$ or $l_1 v_X^*(x') + l_2 v_Y^*(y') = l_1 v_X^*(x'') + l_2 v_Y^*(y'')$.

   Since $v_X^*(x')$, $v_Y^*(y')$, $v_X^*(x'')$ and $v_Y^*(y'')$ are now known numbers and since $l_1 + l_2 = 1$, we can solve for l_1 and l_2.

Two terms need to be explained in order to understand the procedure.

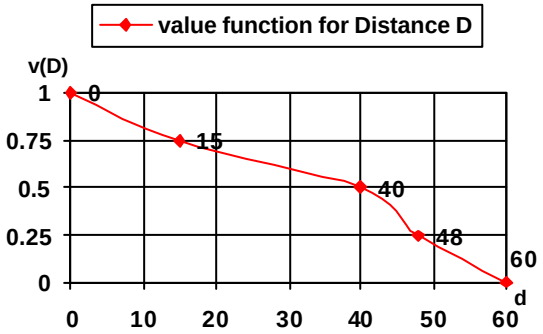
Differentially value -equivalent: The pair (x_a, x_b) is said to be differentially value-equivalent to the pair (x_c, x_d), where $x_a < x_b$ and $x_c < x_d$, if whenever we are just willing to go from x_b to x_a for a given increase of Y, we would be just willing to go from x_d to x_c for the same increase in Y.

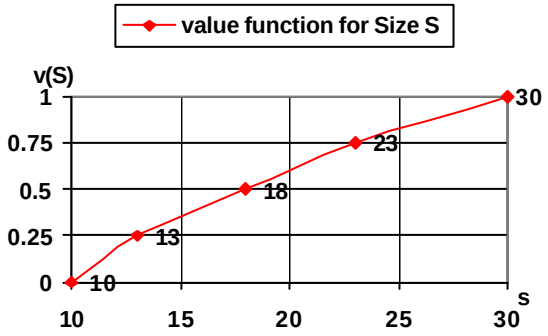
Midvalue Point: For any interval $[x_a, x_b]$ of X, its midvalue point x_c is such that the pairs (x_a, x_c) and (x_c, x_b) are differentially value-equivalent.

According to the above assessment procedure, a hypothetical interaction process between an analyst and decision maker was also given by Keeney and Raiffa. Let's illustrate it in the apartment finder scenario. Assume the value function of apartments contains two attributes: D*istance-D*, and *Size-S*. The *distance* is calculated in minutes of walking from the apartment to working place, and it ranges over the interval 0 minutes to 60 minutes. *Size* is the square meters of the apartment, and it ranges over the interval 10 to 30. The questions and hypothesized answers are as follows:

Question	Hypothesized answer
1. Suppose you are at Size=20. Would you pay more of Size to change *Distance* from 60 to 30 or 30 to 0?	I would pay more to go from 60 to 30.
2. More to go from 60 to 50 or 50 to 0?	More to go from 50 to 0.
3. Give me a value, d' say, such that you would give up the same in *Size* to go from 60 to d' as from d' to 0.	About x' =40
4. In our language, 40 is the midvalue point between 0 and 60. We label 40 by $d_{.5}$. What is your midvalue point between 0 and 40?	Let's say 15, I'd pay the same to go from 40 to 15 as 15 to 0.
5. In that case $d_{.75}$ =15. What is your midvalue point between 40 and 60?	Oh, about 48
6. This means that $d_{.25} = 48$. Does 40 seem like a good midvalue between 15 and 48?	Sure
7. Now let's turn to the *Size* value. What is the midvalue point between 10 and 30?	Say, 18.
8. The midvalue between 18 and 30?	Say, 23.
9. The midvalue between 10 and 18?	13.

Then we can plot these few points and fairs in the curves of v_D (distance) and v_S (size).





Next step is assessing the scale constants l_1 and l_2, which are also called the value tradeoffs between *Distance* and *Size*:

Question	Hypothesized answer
Which (d, s) pair would you prefer, (60, 30) or (0,10)? In other words, if you were at (60, 10) would you rather push *Distance* up to its limit of 0 or *Size* up to its limit of 30?	The *Distance* variable is more critical. I would rather have (0, 10) than (60, 30). (The answer implies that $l_1 > l_2$)
Give me a value *d* such that you are indifferent between (*d*, 10) and (60, 30)? In other words, imagine that you're at (60, 10). How much would you have to push *Distance* up to be equivalent to *Size* going from 10 to 30?	I don't know. I would say about 20, but I feel awfully woozy about that.

If (20, 10) is indifferent to (60, 30), we could conclude that l_1=0.61 and $l_2 = 0.39$, then the value function can be finally described as $v(d, s) = 0.61v_D^*(d) + 0.39v_S^*(s)$.

This procedure can also be extended to additive value function for more than two attributes:

1. Assessment of component value functions
   1) Identify three midvalue points for each attribute $A_i, i = 1,...,n$ $(n \geq 3)$
   2) Consistency check for each component value function
2. Assessment of tradeoff constants $l_i (i = 1,..., n)$, which needs to identify at least (n-1) indifferent pairs.

Therefore, the number of questions asked to a decision maker is at least $4 \times n + (n-1) = 5n-1$ for assessing an additive value function of *n* attributes.

2.2 Analytic Hierarchy Process

The Analytic Hierarchy Process (AHP) [2][3][4] is a decision support tool to solve multi-criteria decision problems. It uses a multi-level hierarchical structure of objectives, criteria, subcriteria, and alternatives. By using pairwise comparisons, it can obtain the weights of importance of the decision criteria, and the relative performance measures of the alternatives in terms of each individual decision criterion. If the comparisons are not perfectly consistent, it provides a mechanism for improving consistency.

The structure of the typical decision problem considered in AHP contains M alternatives and N decision criteria. Each alternative can be evaluated in terms of the decision criteria and the relative importance (or weight) of each criterion can be estimated as well. The core of the typical multi-criteria decision making (MCDM) problem is represented by the following **decision matrix**. The $a_{ij} (i = 1,2,...,M, j = 1,2,..N)$ denotes the **performance value** of the i-th alternative (Ai) in terms of the j-th criterion (Cj), and the Wj is the **weight** of the criterion Cj. Given the decision matrix, the final performance (priority) denoted by A_{AHP}^i of the i-th alternative in terms of all the criteria combined can be determined according to the formula: $A_{AHP}^i = \sum_{j=1}^{N} a_{ij} w_j$, *for* $i = 1,2,...,M$.

	Criterion				
	C_1	C_2	C_3	…	C_N
Alt.	W_1	W_2	W_3	…	W_N
A_1	a_{11}	a_{12}	a_{13}	…	a_{1N}
A_2	a_{21}	a_{22}	a_{23}	…	a_{2N}
A_3	a_{31}	a_{32}	a_{33}	…	a_{3N}
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
A_M	a_{M1}	a_{M2}	a_{M3}	…	a_{MN}

The a_{ij} and w_j are estimated by the use of pairwise comparisons. The decision maker has to express his opinion about the value of one single pairwise comparison at a time. Usually, he has to choose the answer among 10-17 discrete choices, each of which is a linguistic phrase such as "*A is more important than B*" or "*A is of the same importance as B*". The linguistic phrase selected by the decision maker is then quantified by using a scale. Such a scale is a one-to-one mapping between the set of discrete linguistic choices and a discrete set of numbers representing the importance or weight. According to the scale introduced by Saaty [3], the available values for the pairwise comparisons are members of the set: {9, 8, 7, 6, 5, 4, 3, 2, 1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8, 1/9}.

As an illustrative example also consider the apartment finder scenario. Suppose there are three alternative apartments: A(price:400, size:20 square meters, distance:10 minutes, kitchen: private), B(price:500, size:15 square meters, distance:25 minutes, kitchen: not available) and C(price:600, size:25 square meters, distance:20 minutes, kitchen: share), then the decision criteria are price, size, distance and kitchen. Suppose the following is the judgment matrix when the three alternatives are examined by the decision maker in terms of criterion "price".

C1: price	A	B	C
A	1	6	8
B	1/6	1	4
C	1/8	1/4	1

For instance, when apartment A is compared to B then the decision-maker determined that A is between to be classified as *"essentially more important"* and *"demonstrated more important"* than B. Thus, the corresponding comparison is quantified as value of 6. A similar interpretation is true for the rest of entries.

The next step is to determine the relative importance implied by the comparisons. Saaty asserted that calculating the **right principal eigenvector** of the judgment matrix can answer the question. Given a judgment matrix with pairwise comparisons, the corresponding maximum left eigenvector is approximated by using the geometric mean of each row. That is, the elements in each row are multiplied with each other and then the n-th root is taken (where n is the number of elements in the row). Next the numbers are normalized by dividing them with their sum. Hence, for the previous matrix the corresponding priority vector is: (0.754, 0.181, 0.065).

AHP methodology also allows for consistency check. If all the comparisons are perfectly consistent, the following relation should always be true for any combination of comparisons taken from the judgment matrix: $a_{ij} = a_{ik}a_{kj}$. In AHP the pairwise comparisons in a judgment matrix are considered to be adequately consistent if the corresponding consistency ratio (CR) is less than 10% [3]. How to calculate the CR is omitted here for the sake of brevity.

After the alternatives are compared with each other in terms of each criterion and the individual priority vectors are derived, the decision matrix is determined. The priority vectors become the columns of the decision matrix, and the weights of importance of the criteria are also estimated by pairwise comparisons. Consider the illustrative example of apartment finder, whose final priority is as follows:

	Criterion				
	C_1	C_2	C_3	C_4	
Alt.	(0.553	0.131	0.271	0.045)	Final Priority
A	0.754	0.233	0.745	0.674	0.680
B	0.181	0.055	0.065	0.101	0.130
C	0.065	0.713	0.181	0.226	0.190

Therefore, the best apartment for the decision maker is A followed by apartment C which is followed by apartment B. If a problem has M alternatives and N criteria, the decision maker is required to perform $O(M^2 \times N + N^2)$ times of pairwise comparisons.

3. Preference Elicitation System Examples

From section 2, we can see that the traditional elicitation methods are typically time consuming, tedious and sometimes error-prone. To simplify the elicitation task, some computer-aided decision support systems made use of assumptions (e.g. additive independence) which allows a high-dimensional value (utility) function to be decomposed into a simple combination of lower-dimensional sub-value (utility) functions. In this section, we will introduce several such systems in detail.

3.1 Knowledge-based FindMe Recommender Systems

The knowledge-based recommender system is to use knowledge about users and products to provide advice to users about items they might wish to purchase or examine. The knowledge about users is users' preferences of products. In the FindMe systems proposed by Robin Burke et al. [5] [6] [7], preferences are elicited by example similarity and tweak application. In the similarity case, the user selects a given item (called the *source*) from the catalog and requests other items similar to it. To perform the retrieval, a large set of candidate alternatives is retrieved from the database and sorted by the similarity to the source. The top few alternatives (called the *examples*) are returned to the user. Tweak application is essentially the same except that the candidate set is filtered prior to sorting to leave only those candidates satisfying the tweak. For

example, if a user responds to item X with the tweak "Cheaper", the system determines the "price" value of X and rejects all candidates except those whose value is cheaper.

There are 5 FindMe systems for different domains: Car Navigator for new cars, PickAFlick movie recommender, RentMe apartment-finding, Entree restaurant recommender, and Kenwood for home theater systems configurations. Here we only give introductions of PickAFlick, RentMe and Entree.

PickAFlick was to let users find movies similar to ones they already knew and liked. It made several sets of suggestions introducing the idea of multiple retrieval strategies, and different ways of assessing the similarity of items. For example, if a user entered the name of the movie "*Bringing Up Baby*" , the system would look for movies with similar genre, actor and director respectively.

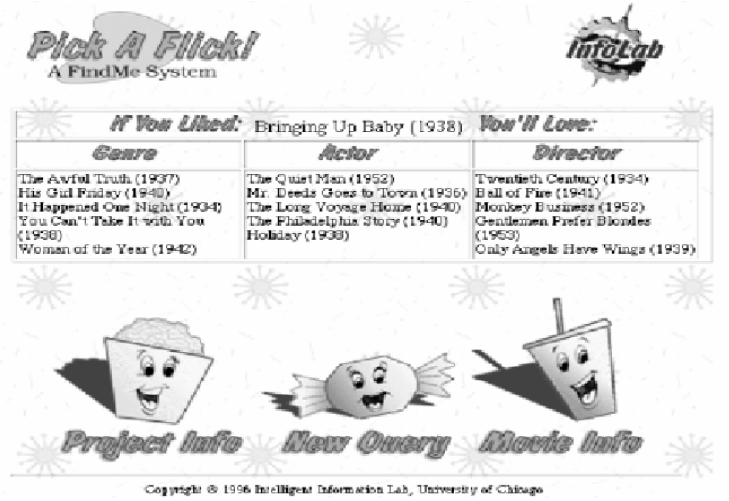


Figure 1 Multi-strategy retrieval in PickAFlick

For apartment finding, the entry point of a known example is not effective in this domain, so users must specify their initialpreferences as a set of constraints. For example, the user's initial constraints are [600<price<650, neighborhood = 'Bucktown', size=2]. System would find an apartment matching user's constraints and let user further tweak based on the apartment returned.

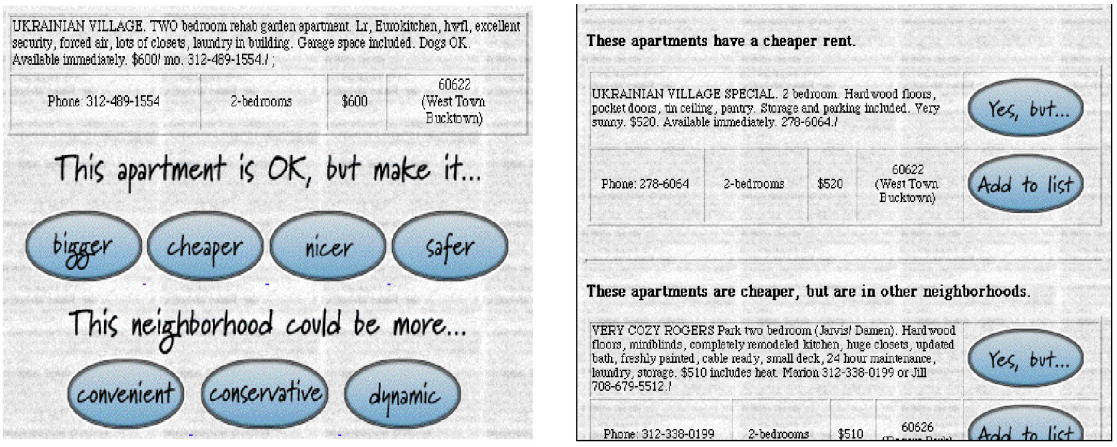


Figure 2 Tweaking an apartment in RentMe

In the Entree restaurant recommender, the user starts with a known restaurant, for example Wolfgang Puck's "Chinois on Main" in Los Angeles. The system finds a similar Chicago restaurant combining Asian and French influences, "Yoshi's Cafe" . If the user is interested in a cheaper meal and selects the "Less $$" button, it would result in a creative Asian restaurant in a cheaper price bracket: "Lulu's".

Figure 3 Similarity-based retrieval in Entree

The Recommender Personal Shopper (RPS) (www.recommender.com) is the culmination of the FindMe techniques. It has aimed to create a generic recommendation capability which could be customized for any domain by the addition of product data and declarative similarity knowledge. In Recommender.com movie recommender, user can apply the tweaks of "Remove Feature", "Add Feature" , " Change Genre" or "With Person", which is different from the interaction of PickAFlick.

Figure 4 Recommender.com movie recommender

The user's preferences model underlying all FindMe systems is a feature vector obtained from entry example or user's initial constraints (like in RendMe). When user performs tweak application, the model will be updated accordingly.

Users interact with system through tweaking or altering the characteristics of an example. This process can be called example/tweak interaction. FindMe responds to users by implementing retrieval and sorting algorithms both based on goal similarity measure. There are a handful of standard goals in any given product domain. For example, in the restaurant domain, the goals are cuisine, price, quality, atmosphere, etc. For each goal, the system defines a similarity metric, which measures how closely two products come to meeting the same goal. Because different users might have different goal orderings, a FindMe system may have several different retrieval strategies, each capturing a different notion of similarity. When system retrieves candidate entities, each metric creates a constraint, and all constraints are ordered by the priority of the metric within the current retrieval strategy. If a query is to be used for a tweak, a constraint is created implementing the tweak and is given highest priority. An SQL query is created by conjoining all the constraints and is passed to the database. If no enough entities are returned, the lowest priority constraint is dropped and the query is resubmitted. The returned entities are sorted by their similarity to the *source* in sorting algorithm, which works by first applying the most important metric corresponding to the most important goal in the retrieval strategy, and then applying the second most important metric to the top *n* entities. This process repeats until all metrics are used or no further sorting has effect. The entities most similar to the *source* would be represented to the user, and the user can apply tweak function to get a new set of entities. This interaction would end when the user is satisfied with the returned result(s).

3.2 The Automated Travel Assistant

The Automated Travel Assistant (ATA) [8] is a recommender system that focuses on the problem of flight selection. In ATA, user's preferences are described in terms of soft constraints on the values of attributes. A constraint is a function $C_i(v) : dom(C_i) \rightarrow [0,1]$, where v is the value of the i-th attribute. $C_i(v) = 0$ means the constraint is fully satisfied and $C_i(v) = 1$ means the constraint is fully unsatisfied. Values in the open interval represent partial satisfaction of the constraint. ATA makes the assumptions that the preference structure is additive independence and constructs an *error* function which provides a partial ordering over all solutions, $E((v_1, v_2, ..., v_n)) = \sum_{i=1}^{n} C_i(v_i) \times w_i$ where w_i is the weight indicating the importance of each constraint.

The communication between users and ATA can be called the *candidate/critique* model of interaction, in which communication from the system is in the form of candidate solutions to the problem, and communication from the user is in the form of critiques of those solutions. That is, the system uses the current user model to suggest a set of solutions. Either the user can choose one and end the interaction, or add a new constraint, modify an existing constraint or adjust the

weighting of a constraint. After the user critiques the suggested candidates, the system updates user model and results in a new set of solutions being suggested.

The algorithm of ATA starts with user's initial preferences over itineraries, perhaps only the departure and destination cities and the approximate dates of travel, and incorporates a set of default preferences into the user's expressed preferences: price sensitive, fewer stops preferred to more stops, and few different airlines preferred. The system finds flights that satisfy the given preferences, groups the flights into trips, and ranks the trips using the user model (error function). Of the top-ranked trips, three significantly different, undominated trips will be displayed along with two extrema: the cheapest trip and best non-stop trip. For example, ATA returns the following trips.

Figure 5 Trips displayed by ATA

The criterion for determining when one trip is dominated by another is defined as if $\forall$ constraint C_i, $C_i(v_{i1}) \le C_i(v_{i2})$ and for some constraint C_j, $C_j(v_{j1}) \le C_j(v_{j2})$ (where v_{i1} and v_{i2} are the values of the i-th attribute of solution S_1 and S_2 respectively), the solution S_2 is dominated by solution S_1. The definition of when one trip is significantly different than another is that if $\sum_i w_i \mid v_{i1} - v_{i2} \mid \ge d$ (where v_{i1} and v_{i2} are the values of the i-th attribute of solution S_1 and S_2 respectively, w_i is the weighting of the difference for the attribute and d is the difference threshold), the two solutions S_1 and S_2 are significantly different.

3.3 The Apt Decision Agent

The Apt Decision agent [9] learns user preferences in the domain of rental apartments by observing user's critique of apartment features. User provides a small number of criteria initially, and receives a display of sample apartments. He can then react to any feature of any apartment. The agent uses interactive learning techniques to build a profile of user preferences, which can be

saved and used for further retrievals, for example, taking to a human real estate agent as a starting point for a real-world apartment search.

The user model is represented as a weighted feature vector. Each feature of an apartment has a base weight determined as part of domain analysis. Using an initial profile provided by the user (number of bedrooms, city, price), the system displays a list of sample matching apartments as shown in the figure below.

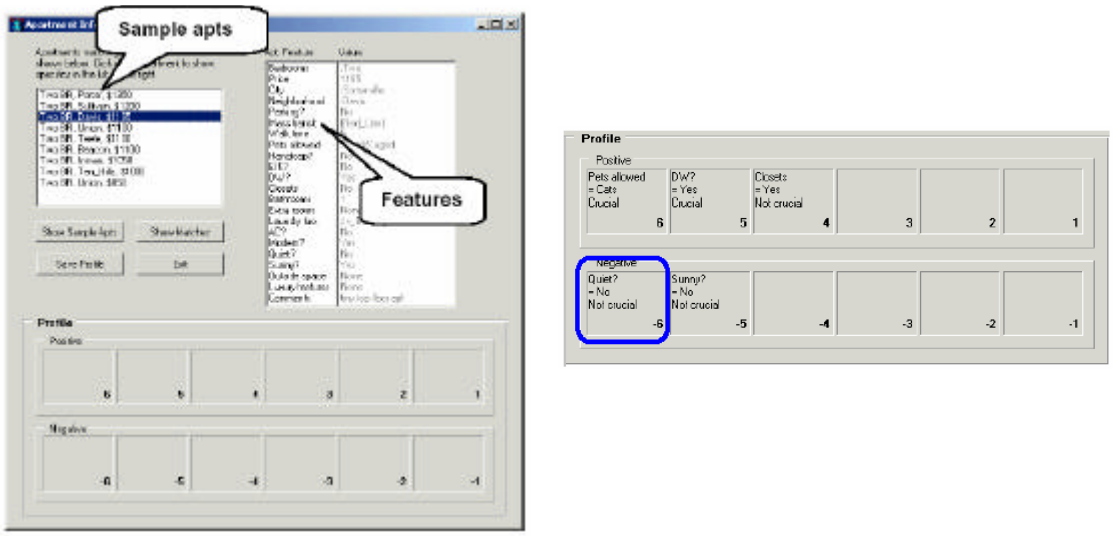


Figure 6 Sample apartments and Profile displayed by Apt Decision agent

The features of the selected apartment are showed on the right side of the window, so user can discover new features of interest and change the weight on individual feature by dragging the feature onto a slot in the profile. The profile contains twelve slots: six positive (1 to 6) and six negative (-1 to -6) with more important slots on the left and less important slots on the right.

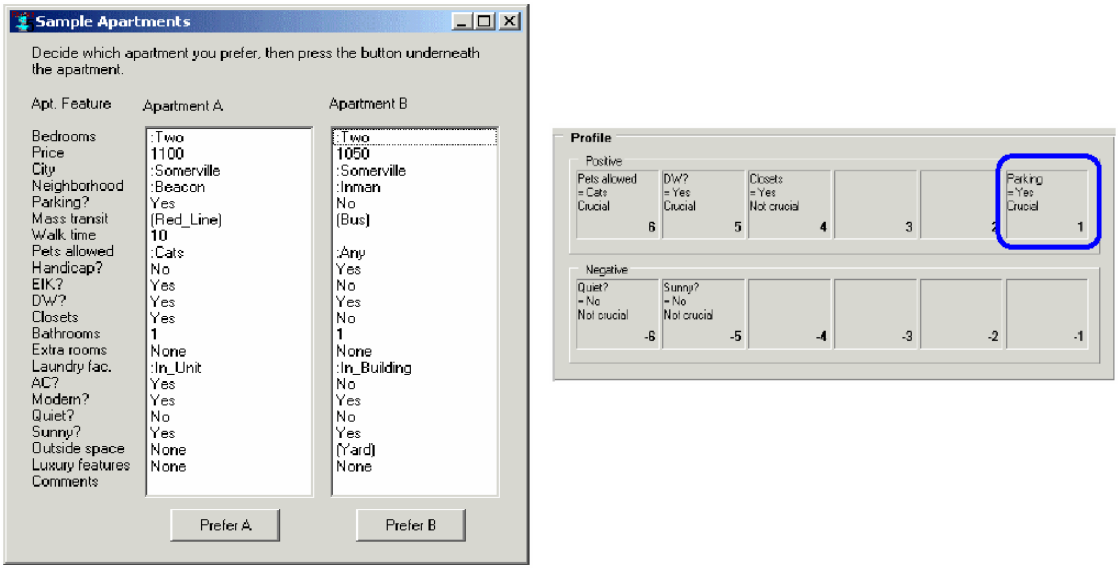


Figure 7 Profile expansion in Apt Decision agent

Profile can also be expanded by expressing pairwise preference among pairs of sample apartments. New features of interest are obtained by examining the current profile and the apartment chosen by the user. The items unique in the chosen apartment and not present in the profile would be added to the right side of the profile. The user can drag the items to different slots in the profile if needed.

The communication between users and Apt Decision agent can be classified as example/critique interaction. The sample apartments are examples and criticized by user while creating the profile. The important advantage of the Apt Decision approach is that the user profile is constantly displayed for user to directly modify it.

3.4 The ExpertClerk Agent

The ExpertClerk [10] is an agent system imitating a human salesclerk. It interacts with shoppers in natural language and narrows down matching goods by asking effective questions (Navigation by Asking). Then it shows three contrasting samples with explanations of their selling points (Navigation by Proposing) and observes shopper's reaction. This process repeats until the shopper finds an appropriate good. Thus we can see the interaction also belongs to sample/critique model.

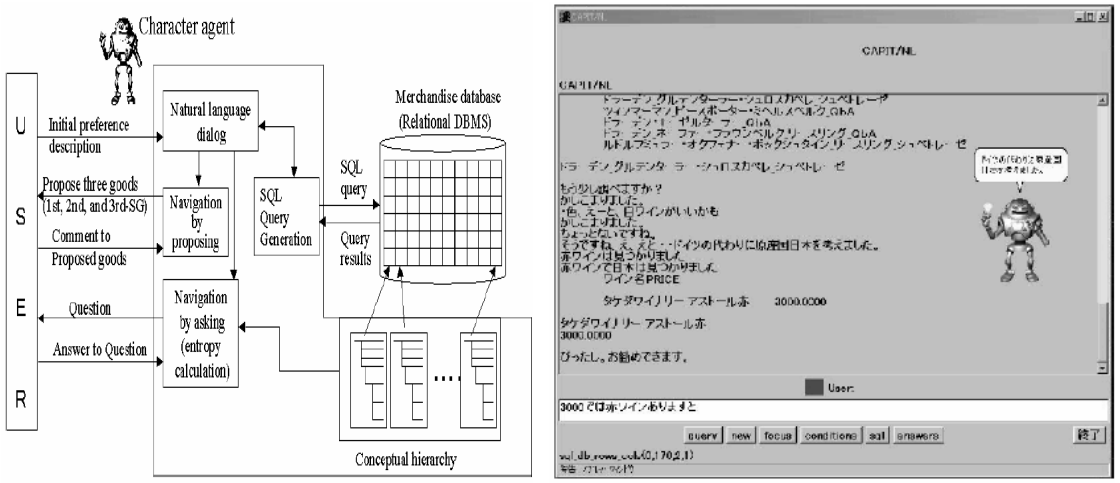


Figure 8 The ExpertClerk architecture and screen image

User's initial preferences (buying points) are identified by asking a few questions in a natural language dialog. The system translates the user's request into a SQL query and passes it to the database. If there exist too many matching goods, the Navigation by asking would calculate the information gain of possible questions and ask appropriate questions to the shopper so as to narrow down the matching goods. After merchandise records are narrowed down to a pre-defined threshold number, Navigation by proposing would show three significantly different samples and explain their selling points. The first sample good is the good record closest to the center point of all matching goods. Its selling points directly reflect the customer's request. The second sample good is the record positioned most distantly from the center point, and the third sample good is the one positioned most distantly from the second sample. The explanation of the sample's selling

point is like “this is twice as expensive as those because it is made of silk and the other two are made of polyester” . While seeing the explanation, the shopper can more easily exclude one of the three proposed goods with a specific reason “this one is too dark for me compared with the other two”. The ExpertClerk would observe the shopper’s reactions and modifie the sample picking strategy accordingly.

3.5 Active Decisions

In online shopping environment, many interactive decision aids obey decision maker’s two-stage process to elicit preferences and assist decision making. In fact, users are often unable to evaluate all available alternatives in great depth prior to making a choice [11], thus, they tend to use two-stage processes to reach their decisions, where the depth of information processing varies by stage [12] [13]. At the first stage, consumers typically screen a large set of available products and identify a subset of the most promising alternatives. Subsequently, they evaluate the latter in more depth, perform relative comparisons across products on important attributes, and make a purchase decision. Given the different tasks to be performed in such a two-stage process, interactive tools that provide support to consumers in the following respects are particularly valuable: the initial screening of available products to determine which ones are worth considering further; and the in-depth comparison of selected products before making the actual purchase decision.

The recommendation agent assists consumers in the initial screening of the alternatives that are available in an online store. Based on information provided by the shopper regarding his/her own preferences, a RA recommends a set of products that are likely to be attractive to that individual. The approaches to preference elicitation can be divided into two groups: feature-oriented and needs-oriented [14]. The systems working in a feature-oriented way require users to specify preferences about product features (e.g. the digital camera should have a resolution of at least 4 Mega Pixel). Needs-oriented approach is to ask users to specify their personal needs (e.g. I want to take baby pictures). It has been argued that the needs-oriented approach should be the preferred method for recommending products to novice users.

The second decision aid, a comparison matrix (CM), is conceptualized as an interactive tool that assists consumers in making in-depth comparisons among those alternatives that appear most promising based on the initial screening. Very basic form of this type of decision aid, usually referred to as a shopping cart or basket, is implemented as an interactive display format in which the product information is presented in an alternatives (rows) × attributes (columns) matrix. It is designed to enable shopper to compare products more efficiently and accurately.

Active Decisions Inc. (www.ActiveDecisions.com) is the world’s leading provider of guided selling solutions. Applications delivered by Active Decisions empower retail staff, branch staff, call center reps, and self-service applications to cost-effectively engage customers, close new and add-on sales, and build long-term customers relationships. Today, over 80 companies use Active Decisions’ proven technology to generate revenues and assist their customers.

The main technology of Active Decisions can be viewed as the combination of recommendation agent and comparison matrix. Initially, it asks the customers what they are looking for and what’s important to them through feature-oriented or needs-oriented preference elicitation

approach; then it makes the right recommendation to the customers. Customers can further choose several preferred alternatives to make in-depth comparisons by comparison matrix. The recommended products will be also displayed with the explanations of why that product or service is right for the customer.

Some typical applications of Active Decisions solutions can be found in A&B Sound DVD Guide (www.absound.ca), Amazon (www.amazon.com), QVC Compare Digital Cameras (www.qvc.com), J&R Music and Computer World (www.jr.com), and Sony Notebook Guide (www.sonystyle.com).

3.6 Teaching Salesman

Markus Stolze et al. [15] proposed an approach for interactive Business-To-Consumer (B2C) eCommerce systems that support the required guided transition from a needs-oriented to a feature-oriented interaction, and thereby enable consumer learning and foster confidence building. The user preferences model elicited is a scoring tree with multiple levels of criteria assessing attributes which allows the hierarchical aggregation of utilities to produce a cumulated score for an outcome. In their example scenario, the outcomes are digital cameras and their attributes are camera features such as pixel resolution and weight. The highest level evaluation criteria in the scoring tree are *uses* representing the potential needs for the desired product by the consumer. The score of a use is the weighted score of its associated feature criteria, and the score of a feature criterion is the weighted sum of its attributes' utilities.

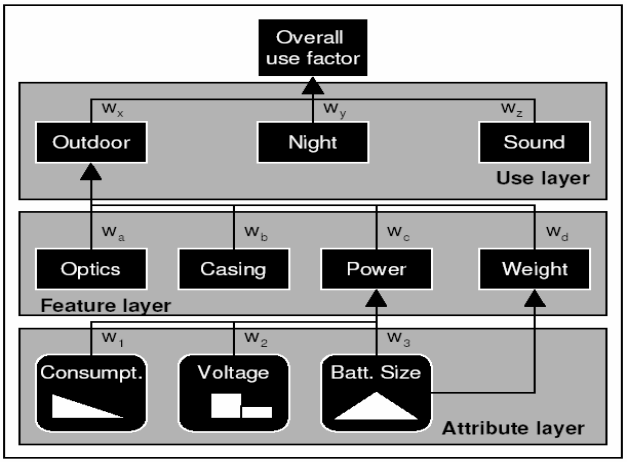


Figure 9 Example of a scoring tree

The hierarchical structure of the user model allows a system to explain to user why a product is recommended for a specific use. If a product achieves a high score for a specific use, the recommendation can be drilled down to the domain features contributing the highest values or having the highest importance, and further down to the attributes, which again might have a high utility or high importance for this use.

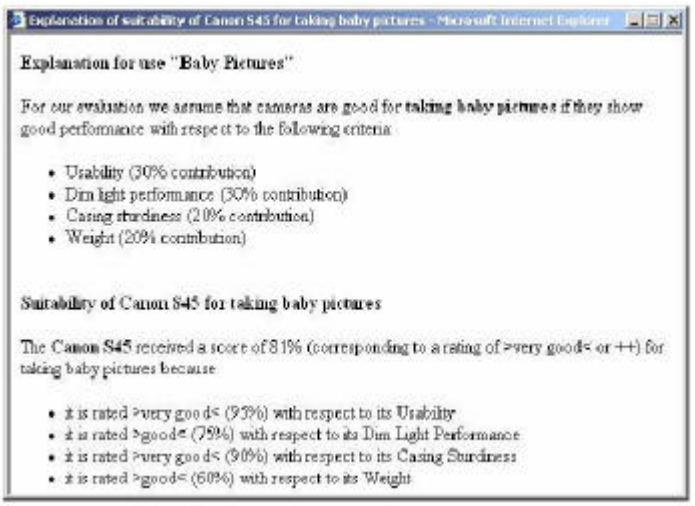


Figure 10 Explanation for camera use "Baby Pictures"

According to the two-stage process of consumer decision making [12], they refined whole interaction into seven phases which emphasize three main aspects: preference discovery, preference optimization and preference debugging. In preference discovery, consumer needs to formalize his potential uses of a product, maybe discover additional uses, and learn how features relate to these uses.

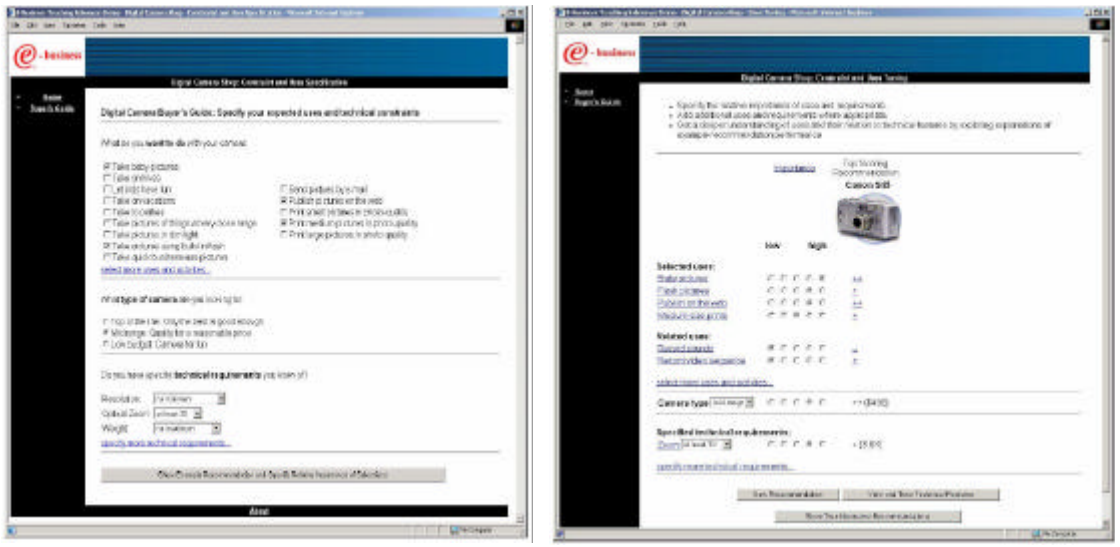


Figure 11 Constraint specification and uses tuning

The preference optimization and debugging are for users to further understand and optimize feature criteria, and verify the completeness and correctness of the evaluation structure (scoring tree) to gain confidence in the final choice. There would be three examples displayed to consumers to increase their confidence in the buying decision.

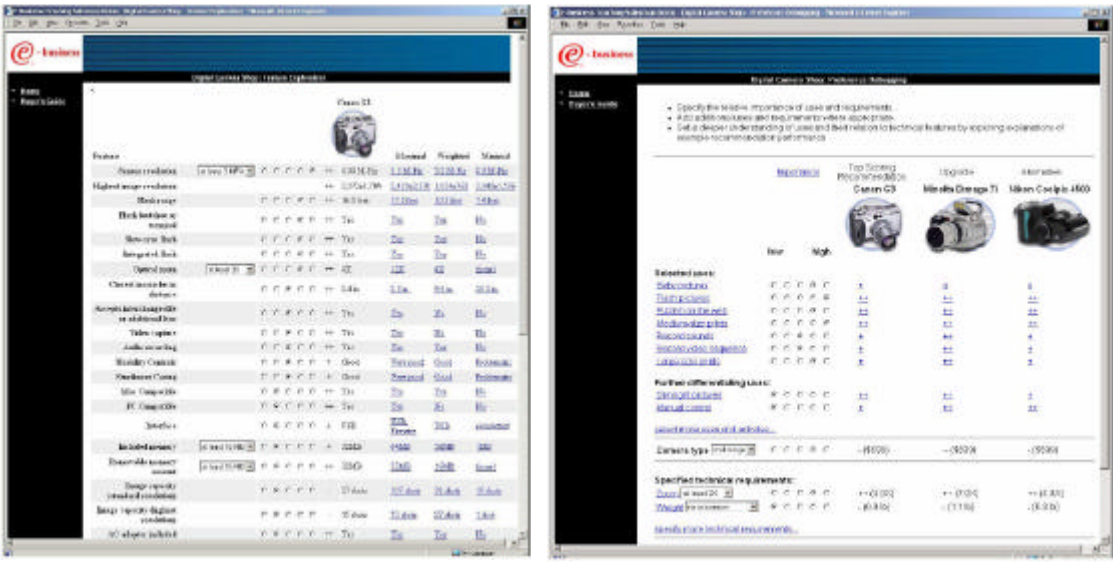


Figure 12 Feature exploration and products comparison

3.7 CP-network

The systems described above are all based on the strong assumption of additive preferential independence. Boutilier et al [16] [17] explored a graphical representation of utilities with the weaker form of additive independence, called the *conditional preferential independence*.

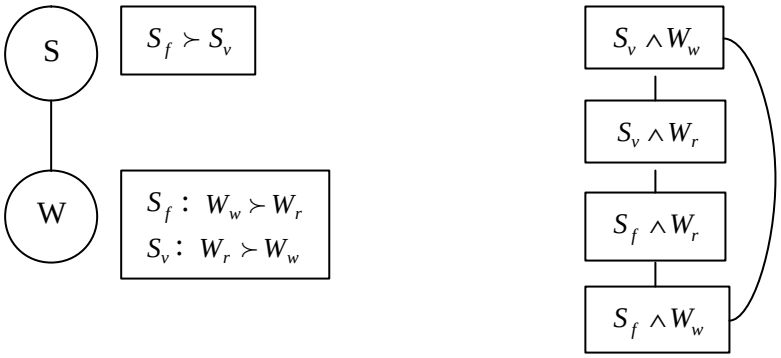
Conditional Preferential Independence: Let X, Y and Z be nonempty sets that partition V, X is conditionally preferentially independent of Y give an assignment z to Z if and only if, for all $x_1, x_2 \in Asst(X)$ (all assignments to X) and $y_1, y_2 \in Asst(Y)$, we have $x_1 z y_1 \geq x_2 z y_1$ iff $x_1 z y_2 \geq x_2 z y_2$. In other words, X is preferentially independent of Y when Z is assigned z. If X is conditionally preferentially independent of Y for all $z \in Asst(Z)$, then X is conditionally preferentially independent of Y given the set of variables Z.

The graphical representation is a *conditional preference network* (CP-network), which creates a node for every attribute. For every attribute A_i, the user must identify a set of parent attributes whose values will influence the user's preference for the value of A_i. Each node has an associated table describing how the parents' values will affect the preference for the value of A_i. Formally, the CP-network is defined as follows.

Conditional Preference Network (CP-network): A CP-network over attributes V= $\{A_1, A_2, ..., A_n\}$ is a directed graph G over V, whose nodes are annotated with conditional preference tables ($CPT(A_i)$ for each $A_i \in V$). Each conditional preference table $CPT(A_i)$ associates a total order $\succ_i (u_i)$ with each instantiation u_i of A_i's parents $Pa(A_i) = U_i$.

With a set of initial conditional preference information, a CP-network can be used to rapidly decide which of two outcomes dominates the other or if there is an insufficient amount of information to determine the dominant outcome. In the case of the latter situation, the CP-network will identify an outcome whose preference information should be elicited from the user.

For example, consider the simple CP-network that expresses user's preferences over dinner configurations. This network consists of two attributes S and W, standing for the soup and wine respectively. If user strictly prefers fish soup (S_f) to vegetable soup (S_v), while his preference between red (W_r) and white (W_w) wines is conditioned on the soup to be served: he prefers red wine if served a vegetable soup, and white wine if served a fish soup, then the CP-network and corresponding induced preference graph over outcomes can be described as below.



An arc in the preference graph directed from outcome o_i to o_j indicates that a preference for o_j over o_i can be determined directly from the CP-network.

The CP-network can be applicable to a much wider situation than the methods based on additive preferential independence, however, it can not represent quantitative utility information. Boutilier et al further extended the CP-network to UCP-network [18] by adding quantitative utility information to the conditional preference table of each attribute. Another problem with CP-network is that there hasn't been concrete interaction design of preferences elicitation.

4. Other Elicitation Methods

Although the elicitation methods under the assumption of additive or conditional preferential independence can reduce the number of questions needed to ask users and make the task easier, the elicitation of a complete value (utility) function may still be too time consuming. Furthermore, in many cases, attributes are preferentially dependent and thus assumptions of decomposability are suspect. One research field on preference elicitation is hence aiming at releasing the assumptions concerning preference structure, but still along with the goal of simplifying elicitation task and saving user's effort. The main idea is to identify the new user's value (utility) function based on previously collected value functions of other users.

Another predominant approach to preference-based selection of products is collaborative filtering, in which the system makes recommendations to a user based on the opinions of other users who have tastes similar to that user, but this approach is not in the framework of decision theory.

4.1 Clustering, Matching and Refining

In this chapter, we describe the elicitation methodology combining attribute-based elicitation of user preferences with matching of a user's preferences against those of other users of the system.

The typical research works are [19] and [20], where the concrete procedure can be summarized as follows [21]:

- Using "complete and reliable" elicitation techniques to elicit a sufficient number of users' preference models;
- Grouping these models into qualitatively different clusters;
- Given the clusters, a new user's preference model is elicited by two sub-processes: find the cluster to which the new user more likely belongs and refine the preference model associated with that cluster for the new user.

The rationale is that finding and refining a matching cluster would require significantly less elicitation steps than building a preference model from scratch.

The Video Advisor [22] is a representative system applying this methodology, which uses the case-based technique described in [20] to elicit the value function representing the user's long-term preferences. At first, it maintains a group of users with their preferences over movies partially or completely specified. When a new user comes in, his preference structure will be determined partially and then matched against the preferences structures of the existing group of users. The retrieved closest matching preference structure is used to supplement the partially elicited new user's preferences.

The new user's initial preferences are determined by asking the user to provide a list of movies he particularly likes and a list of movies he particularly dislikes. If he likes a movie, he is assumed to also like another movie with the same (casting, director, genre) combination. The like/dislike lists are uses as labeled training instance to come up with the value function. The notion of closeness in preferences matching demands a measure of distance among preference structures. Ha and Haddawy [20] have studied various distance measures including Euclidean distance, Spearman's footrule and probabilistic distance.

Figure 13 The screenshot of Video Advisor

In light of the constructive nature of preference elicitation [23] [24], this kind of elicitation methodology can be problematic since the decision-maker's preference construction process is

reduced and consequently any recommendation based on refined preference model will unlikely be understood and accepted by the decision-maker.

4.2 Collaborative Filtering

Collaborative filtering (CF) is also one technique of producing recommendations. Given a domain of choices (like books, movies or CDs), user can express his preferences by rating these choices. These ratings serve as an approximate representation of user's preferences, and the recommender system will match these ratings against ratings submitted by all other users of the system, find the "most similar" users based on some criterion of similarity, and recommend items that similar users rated highly but the user has not rated (presumably not familiar with). The user can further rate the recommended items. Therefore, over time, the system can acquire an increasingly accurate representation of user's preferences.

For example, the MovieLens [25] (http://movielens.umn.edu) is a typical CF system that collects movie preferences from users and groups users with similar tastes. Based on the movie ratings expressed by all the users in a group, it attempts to predict for each individual his opinion on movies he has not yet seen. Other seminal collaborative filtering systems include GroupLens [27], Bellcore Video Recommender [28] and Ringo [29]. The systems varied in how they weighted the ratings of different users (i.e., determined who the similar users were and how close they were) and how they combined the ratings.

There are also many applications of collaborative filtering on the web [26]. Electronic commerce sites like Amazon.com (www.amazon.com) and CDNow (www.cdnow.com) feature recommendation centers, where, in addition to expert reviews, users can rate items and then receive personalized recommendations computed by a collaborative filtering engine.

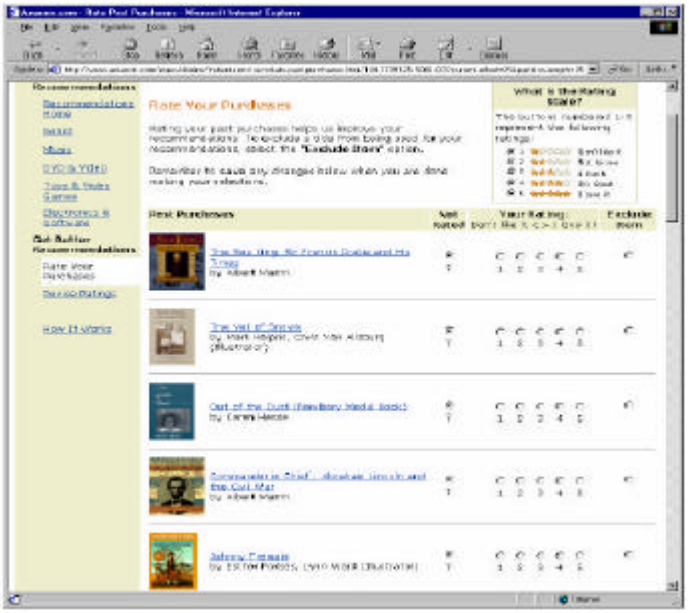


Figure 14 The Amazon.com ratings page

However the CF approach has the drawback that it considers for recommendation only those items that have been rated by at least one user in the collaborative group. Furthermore, this

approach does not make direct use of attributes characterizing the items, for example, the genre, running time and casting attributes of movies. In addition, the work on collaborative filtering is not cast in the framework of decision theory, and there hasn't been theoretical framework or justification provided for the similarity measure used.

5. Conclusion

In this paper, we surveyed the typical preference elicitation methods employed in current decision support systems. Since the traditional methods are too time-consuming and tedious, the computer aided decision support systems have appeared to simplify the task by making the assumption of additive preferential independence. Several representative systems were described, including FindMe, ATA, Apt Decision and ExpertClerk agents.

According to decision maker's two-stage information processing, the Active Decisions has implemented the combination of recommendation agent and comparison matrix. The recommendation agent can recommend products through two approaches, needs-oriented and feature-oriented. The Teaching Salesman supports guided transition from needs-oriented to feature-oriented interaction. The CP-network has explored a graphical representation of utilities with the weaker form of additive independence, which is conditional preferential independence.

In order to improve the accuracy of preferences elicited as well as save decision maker's effort, another research branch on preference elicitation has aimed at releasing all assumptions on preference structure by matching new user's preferences to other users' preference models. The Collaborative Filtering is also based on the similar idea, but the preferences matched are item ratings provided by different users.

This survey gives us a good knowledge about current research progresses on preference elicitation methods. There hasn't been one system which can elicit user preferences neither based on preference structure assumptions nor based on other users' preferences, therefore the work on resolving this problem will be of great challenge and significance.

6. References

[1] Ralph L. Keeney and Howard Raiffa. *Decisions with Multiple Objectives: Preferences and value tradeoffs.* Cambridge University Press, 1976.

[2] Satty, T.L. A Scaling Method for Priorities in Hierarchical Structures. *Journal of Mathematical Psychology*, 1977.

[3] Satty, T.L. *The Analytic Hierarchy Process*. McGraw-Hill International, New York, U.S.A., 1980.

[4] Satty, T.L. *Fundamentals of Decision Making and Priority Theory with the AHP.* RWS Publications, Pittsburgh, PA, U.S.A., 1994.

[5] Burke, R. *Knowledge-based Recommender Systems*. In A. Kent (ed.), Encyclopedia of Library and Information Systems. Vol. 69, Supplement 32. New York, 2000.

[6] Burke, R., Hammond, K. and Young, R. The FindMe Approach to Assisted Browsing. *IEEE Expert*, volume 12(4):32-40, 1997.

[7] Burke, R., Hammond, K. and Cooper, E. *Knowledge*-based navigation of complex information spaces. In *Proceedings of the 13th National Conference on Artificial Intelligence*, pages 462-468, AAAI, 1996.

[8] Greg Linden, Steve Hanks, and Neal Lesh. Interactive Assessment of User Preference Models: The Automated Travel Assistant. In *Proceedings of User Modeling'97*, 1997.

[9] Sybil Shearin and Henry Lieberman. Intelligent Profiling by Example. In *Proceedings of the Conference of Intelligent User Interfaces* (IUI'01). ACM Press, 2001.

[10] 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). USA, August 2001.

[11] Beach, Lee R. Broadening the Definition of Decision Making: the Role of Prechoice Screening of Options. *Psychological Science*, 4 (4) 215-220, 1993.

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

[13] Haubl, G., and Trifts V. Consumer Decision Making in Online Shopping Environments: The Effects of Interactive Decision Aids. *Marketing Science*, Vol.19, No.1: 4-12, 2000.

[14] Markus Stolze, Fabian Nart. Well-integrated needs-oriented recommender components regarded as helpful. *CHI Extended Abstracts*, 2004.

[15] Markus Stolze and Michael Ströbel. Dealing with Learning in eCommerce Product Navigation and Decision Support: the Teaching Salesman Problem. In *Proceedings of World Congress on Mass Customization & Personalization* (MCPC), Munich, 2003.

[16] Boutilier, C., Brafman, R.I., Geib, C. and Poole, D. A Constraint-Based Approach to Preference Elicitation and Decision Making. *AAAI Spring Symposium on Qualitative Decision Theory*, Stanford, CA, pp. 19-28, 1997.

[17] Boutilier, C., Brafman, R.I., Domshlak, C., Hoos, H.H., and Poole, D. CP-nets: A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements. *Journal of Artificial Intelligence Research* (JAIR), 2003.

[18] Boutilier, C., Bacchus, F. and Brafman, R. UCP-Networks: A Directed Graphical Representation of Conditional Utilities. In *Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence* (UAI-01), 56-64, Seattle, 2001.

[19] Chajewska, U., Getoor, L., Norman, J. and Shahar, Y. Utility Elicitation as a Classification Problem. In *Proceedings of the Uncertainty in Artificial Intelligence Conference*, 79-88, 1998.

[20] Ha, V. and Haddawy, P. Toward Case-Based Preference Elicitation: Similarity Measures on Preference Structures. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, 193-201, 1998.

[21] 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.

[22] Nguyen, H. and Haddawy, P. The Decision-Theoretic Video Advisor. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, Stockholm, Sweden, 1999.

[23] Slovic, P. The Construction of Preference. *American Psychologist*, 50 (August), 364-371, 1995.

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

[25] Rashid, A.M., Albert, I., Cosley, D., Lam, S.K., McNee, S.M., Konstan, J.A. and Riedl, J. Getting to know you: Learning new user preferences in recommender systems. In *Proceedings of the International Conference on Intelligent User Interfaces*, pp. 127-134, 2002.

[26] Schafer, J.B., Konstan, J.A. and Riedl, J. E-Commerce Recommender Applications. *Data Mining and Knowledge Discovery*, vol. 5 pp. 115-152, 2001.

[27] Resnick, P., Iacovou, N., Sushak, M., Bergstrom, P., and Riedl, J. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 Computer Supported Collaborative Work Conference*, 1994.

[28] Will Hill, Larry Stead, Mark Rosenstein, George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, p.194-201, 1995.

[29] Shardanand, U., and Maes, P. Social Information Filtering: Algorithms for automating "word of mouth". *In Conference on Human Factors in Computing Systems (CHI'95)*, 1995.

[30] Pearl Pu, Pratyush Kumar. Evaluating Example-based Search Tools. In *ACM Conference on Electronic Commerce* (EC'04), May 2004, New York, USA.