Adaptive Suggestions for Example based Tools

Paolo Viappiani and **Boi Faltings** [1]

Abstract. Preference-based search is the problem of finding an item that matches best with a user's preferences. Recently several researchers have proposed approaches based on examples where preferences are stated in form of critiques on shown options.

User studies show that example-based tools for preference-based search can achieve significantly higher accuracy when they are complemented with suggestions to stimulate preference expression.

This paper proposes a method for producing adaptive suggestions taking in consideration prior distribution of preferences (perhaps learned from a population of similar people) and reactions of the user to the shown examples. At each interaction cycle, a set of best matches (candidate options) and suggestions are retrieved and shown to the user. Using Bayes rule, the distribution is updated at each step. We evaluate the approach with simulations.

1 Introduction

People frequently use the world-wide web to search through a large collection of items. Most commonly, users fill in a form that is directly mapped to a database query and returns a ranked list of the most suitable options. The user has the option to return to the initial page and change his preferences and then carry out a new search. This is the case for example when searching for flights on the most popular travel web sites. Such tools are only as good as the query the user formulates. A study [3] has shown that among the users of such sites only 18% are satisfied with their final choice.

In most cases, users do not know exactly what they are looking for: they might consider different trade-offs or they might even have conflicting desires about the features the item should have. In fact psychological studies have shown that people construct their preferences [12] while learning about the available products. Therefore preference-based search should also help users in formulating accurate preferences.

Decision theory [7] provides a framework for evaluating alternatives based on utility. The classic method for obtaining a precise model of the user's utility function ([8]), known as the value function assessment procedure, asks the user to respond to a long sequence of questions. Suppose the decision outcome involves trading off some preferred values of the size of an apartment against the distance between the apartment and the city center. A typical assessment question is in the form of "All else being equal, which is better: 30 sqm at 60 minutes distance or 20 sqm at 5 minutes distance?".

Even for simple items such as cameras, eliciting such a model would require hundreds of questions, and few users would be ready to undergo such a lengthy process. A major challenge is that in an interaction on the WWW, few users are willing to tolerate more than 5-10 interaction cycles before reaching a result. Therefore, we need to provide users a concise way to express preferences that would be *usable*. At the same time we would benefit from approaches able to mitigate the inaccuracies inevitably created when translating a user's qualitative statement into a quantitative model of preferences suitable for ranking items.

An interesting technique for letting users volunteer their preferences is an interaction where the system shows proposed options and lets users express their preferences as *critiques* stimulated by these examples. This technique is called *example* or *candidate* critiquing, and has been explored by several authors [6, 10, 17, 16].

Optimal elicitation strategies with prior knowledge Many authors have studied how to improve decision aid tools using prior knowledge, usually represented in a probabilistic way. Price and Messinger in [13] optimize the set of examples given an expectation of the user's preferences, without actually asking the users to state their own preferences. This approach does not consider preferences of an individual user but average preferences for a group of users.

Recent works addressed the problem of developing an optimal elicitation strategy, considering the tradeoff between the effort and the decision quality. Chajewska et al. ([2]) considered how to deal with incomplete utility information, assuming a distribution over utility functions. The resulting elicitation procedure is more efficient; the assessment questions are selected to maximize the expected value of information.

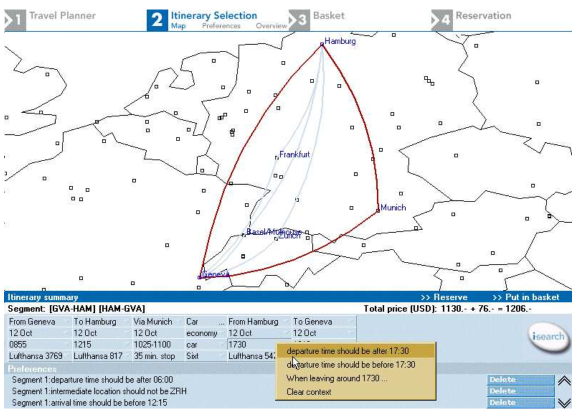
Boutilier ([1]) has extended this work by taking into account values of future questions to further optimize decision quality while minimizing user effort. The elicitation procedure itself is a decision process (represented as a Partially Observable Markov process) where the goal is to minimize the user effort.

These approaches work well when users are familiar with the available choices and thus able to answer the questions correctly. However it is less likely to work in recommendation scenarios where users are not familiar with the options. Even though the results obtained in this way provide a precise model to determine the most preferred outcome for the user, the process is often cognitively arduous. Without training and expertise, even professionals are known to produce incomplete, erroneous, and inconsistent answers [18]. Therefore, such techniques are most useful for well-informed decision makers, but less so for users who need the help of a recommender system.

Contribution of this paper In this paper we present a mixed initiative approach for preference-based search. Our approach lets the user express only those elements of the preference model that he or she feels confident about; at the same time the system stimulates preference expression by showing suggestions (to make the user aware of possible choices) in addition to the best matches according the current query. Such suggestions are adapted to the user by using probabilistic inference techniques.

[1] LIA EPFL, email: paolo.viappiani@epfl.ch, boi.faltings@epfl.ch

2 Incremental preference model acquisition



Segment: [GVA-HAM] [HAM-GVA] Total price (USD): 1130.- + 76.- = 1206.-

From Geneva	To Hamburg	Via Munich	Car	From Hamburg	To Geneva
12 Oct	12 Oct	12 Oct	economy	12 Oct	12 Oct
0855	1215	1025-1100	car	1730	
Lufthansa 3769	Lufthansa 817	35 min. stop	Sixt	Lufthansa 54	

Preferences

Segment 1:departure time should be after 06:00

Segment 1:intermediate location should not be ZRH

Segment 1:arrival time should be before 12:15

Figure 1. *Isy-travel is an example-critiquing tool for planning business-trips. Here the user is planning a one-day trip from Geneva to Hamburg. The preference model is shown at the bottom, and the user can state additional preferences by clicking on features of the shown example.*

In an example-critiquing interaction, user's preferences are *volunteered*, not elicited. This means that users are never forced to answer questions about preferences they do not consciously have. Figure 1 shows Isy-travel, a commercial tool for business travelers [14]. Here, the user is shown examples of options that fit the current preference model well. The idea is that an example either is the most preferred one, or there is some aspect in which it can be improved. Thus, on any of the examples, any attribute can be selected as a basis for critiquing. For instance, if the arrival time is too late, then this can be critiqued. The critique then becomes an additional preference in the model.

Preferences are stated as reactions to displayed options. We can classify such critiques according to the context in which the statement of a preference takes place:

negative reaction none of the options shown satisfy that preference
positive reaction an option is shown that satisfies (partially or completely) the preference

For instance, if the tool shows the user examples that all arrive at London Stansted airport, and she requests to land in Heathrow, that critique would be a *negative reaction*. If the system indeed showed one flight landing in Heathrow, by stating that preference she would be *positively* reacting to the shown examples.

An option that partially satisfies the preference can also cause preference expression. If the user sees examples of flights landing in Stansted and Gatwick, by seeing the possibility of landing in Gatwick, considered better than Stansted but worse than Heathrow, she is stimulated to state a preference about the landing airports.

It is interesting to note that, according to our user studies, as shown in Table 1, in most cases (79%) a preference is stated as a positive reaction to one of the displayed options. This observation suggests that users are likely to state preferences when they see examples that show options that differ in a possible preference and therefore suggestions are useful.

To use a metaphor, the process of example-critiquing is hill-climbing: the user states preferences as long as she perceives them as improving the solution. However, the process might end in a local optimum; a situation in which the user can no longer see potential improvement. For example, a user looking for a notebook computer might start looking for a low price, and thus find that all models weigh about 3 kg. Since all of the presented models have about the same weight, he or she might never bother to look for lighter models. This influence of current examples prevents the user from refocussing the search in another direction; this is known as the *anchoring effect* [18].

Critiques	Frequency
Positive reactions	0.79
Negative reactions	0.21

Table 1. In the majority of cases a preference is stated when the user sees an example that satisfies it (positive reaction to the displayed options).

For these reasons we display two sets of examples:

- **candidate examples** that are optimal for the preference model, and
- **suggested examples** that are chosen to stimulate the expression of preferences.

2.1 Inaccuracy of the preference model

If all the preferences are stated by the user, we would expect the target (the best option available in the database) to be in the displayed set of candidates. However, this might not be always the case. Example-based tools require the designer to choose from possible preference representations: the user expresses the preferences using an interface by qualitative statements, these are then translated into the internal preference model.

In general the representations of the preferences will be inaccurate. In [5], we have analyzed the impact of the error in the modeling on the tool capacity of providing the target option to the user, for several types of preference models and combination functions: weighted soft constraints, fuzzy-lexicographic soft constraints, and simple dominance relations. A remarkable result is that for both weighted and fuzzy-lexicographic constraint models, assuming a bound on the possible error (deviation between true value and the one used by the application) of the soft constraints modeling the preferences, the probability that the true most preferred solution is within k depends only on the number of the preferences and the error bound of the soft constraints but not on the overall size of the solution set. Thus, this approach is suitable when searching a very large space of items.

2.2 The importance of Suggestions

The importance of the diversity of the example shown was recognized by Linden, Hanks and Lesh [10] who explicitly generated examples that showed the extreme values of certain attributes, called *extreme examples*. However, an extreme example might often be an unreasonable choice: it could be a cheap flight that leaves in the early morning, a student accommodation where the student has to work for the family, an apartment extremely far from the city. Moreover, in problems with many attributes, there will be too many extreme or diverse examples to choose from, while we have to limit the display of examples to few of them.

We assume that the user is minimizing her own effort and will add preferences to the model only when she can expect them to have an impact on the solutions. This is the case when:

- she can see several options that differ in a possible preference, and

- these options are relevant, i.e. they could be acceptable choices, and
- these options are not already optimal, so a new preference is required to make them optimal.

In all other cases, stating an additional preference is likely to be irrelevant. When all options would lead to the same evaluation, or when the preference only has an effect on options that would not be eligible anyway, stating it would only be wasted effort. This leads us to the following *lookahead* principle as a basis for suggestion strategies:

> Suggestions should not be optimal under the current preference model, but should provide a high likelihood of optimality when an additional preference is added.

Since we model preferences by standardized functions that correctly reflect the preference order of individual attribute values but may be numerically inaccurate, when generating suggestions we would like to use a model that is not sensitive to this numerical error. We thus use the qualitative combination concepts of *dominance* and *Pareto optimality* [19].

In the next sections we are now going to formalize criteria for choosing suggestions automatically that adapt to the user.

3 The Framework

3.1 Basics

We assume that O is the set of options $o_1, .., o_n$ defined over A, set of attribute $\{a_1, .., a_n\}$ of domains $D_1, .., D_m$; $a_i(o)$ is the value that o takes on attribute a_i.

Domains can be qualitative or numeric. A **qualitative domain** (such as colors or names) consists of an enumerated set of possibilities; a **numeric domain** has numerical values (as price, distance to center), either discrete or continuous. For numeric domains, we consider a function $range(Att)$ that gives the range on which the attribute domain is defined. For simplicity, we call qualitative (respectively numeric) attributes those with qualitative (numeric) domains.

Preferences are stated on individual attributes. A preference r_i applies to a particular attribute a_i and results in a total or partial order of the values in the domain D_i of a_i. A preference model R consists of a set of preferences.

We assume that a preference r_i is expressed by a cost function c_i. Since a preference always applies to the same attribute we can use $c_i(o)$ instead of $c_i(a_i(o))$.

We assume that the cost functions correctly express the user's *ceteris paribus* preferences, i.e. that for any pair of options o_1 and o_2 that are identical in all preferences except preference r_i, the user prefers o_1 over o_2 if and only if $c_i(s_1) < c_i(s_2)$.

The individual costs obtained by each preference are merged with a particular combination function, for example a weighted sum.

$$C(o) = \sum_i w_i * c_i(o) \tag{1}$$

The *candidates* best options can be found by sorting the database items according to their cost. This is known as the *top-k query* [4]. The set of options retrieved $\{o_1, .., o_k\}$ is such that $C(o_1) \leq C(o_2) \leq .. \leq C(o_k)$ and for any other option $\bar{o}$ in the database $C(\bar{o}) \geq C(o_k)$.

3.2 Pareto-dominance and optimality

We model preferences by standardized functions that correctly reflect the preference order of individual attribute values but may be numerically inaccurate. Pareto-optimality is the strongest concept that would be applicable regardless of the numerical details of the penalty functions.

An option o is **(Pareto) dominated by** another option $\bar{o}$ (equivalently we say that $\bar{o}$ dominates o) if

- $\bar{o}$ is not worse than o according to all preferences in the preference model: $\forall c_i \in R : c_i(\bar{o}) \leq c_i(o)$
- $\bar{o}$ is strictly better than o for at least one preference: $\exists c_j \in R : c_j(\bar{o}) < c_j(o)$

An option o is **Pareto-optimal** if it is not Pareto-dominated by any other option.

The **dominating set** $O^+(o)$ is the set of options that dominates the option o.

A remarkable observation is that a dominated option o' with respect to R becomes Pareto-optimal with respect to $R \cup r_i$ (a new preference r_i is added), if and only if o' is strictly better with respect to r_i than all options that currently dominate it: $o' \succ_{r_i} o, \forall\, o \in O^+_R(o')$.

4 Prior distribution

We suppose that the cost functions have a known form that depends on a parameter θ. It is quite straightforward to deal with the cases of multiple parameters.

$$\mathbf{c}_i = c_i(\theta, a_i(o)) = c_i(\theta, o) \tag{2}$$

We use $r_{i,\bar{\theta}}$ to denote the preference on attribute i with parameter $\bar{\theta}$, corresponding to the cost function $c_i(\bar{\theta}, o)$.

We suppose to have a prior probability distribution over the possible preference models, learnt from previous interactions with the system.

- p_{A_i} the probability that the user has a preference over an attribute,
- $p_i(\theta)$ the distribution of probability over the parametric value of the preference representation for the cost function c_i
- $p(r_{i,\theta})$ the probability that the user has a preference on attribute i and its parameter is θ. It is equivalent to $p_{A_i} * p_i(\theta)$

Such distribution will be updated by the user during the interaction based on the user's action.

5 Adaptive model-based suggestions

According to our look-ahead principle, we choose suggestions that have the highest likelihood of becoming optimal when a new preference is added. Since we would like to avoid sensitivity to the numerical error of the cost functions, we use the concept of Pareto-optimality.

In this section we show how to compute the probability that a given option become Pareto optimal (breaking all dominance relations with options in its dominating set). These formulas depend on the probability distributions p_{A_i} and $p_i(\theta)$ that we have introduced before. At the beginning of the interaction, they are initialized by prior knowledge, considering the preference models of previous users. During the interaction, they are updated according to the observations of the user's actions.

For instance, if we have shown to the user an option with value `metro` for attribute "transportation" and she has not stated any critique about the kind of transportation, the probability that the user has a preference for metro as transportation will decrease. In the next interaction cycle, the system is likely to choose options with a different value.

5.1 Probability of escaping dominance

Before discussing the probability of breaking many dominance relations simultaneously, we first consider the probability of breaking one dominance relation alone. To escape dominance, the option has to better than its dominator with respect to the new preference.

The probability that a preference on attribute i makes o_1 be preferred to o_2 can be computed integrating over the values of θ for which the cost of o_1 is less than o_2.

$$\delta_i(o_1, o_2) = \int_\theta H(c_i(\theta, o_2) - c_i(\theta, o_1))p(\theta)d\theta \tag{3}$$

where H is the Heavyside step function $H(x) \equiv$ **if** $(x > 0)$ **then** 1 **else** 0.

For a qualitative domain, we iterate over θ and sum up the probability contribution of the cases in which the value of θ makes o_1 preferred over o_2:

$$\delta_i(o_1, o_2) = \sum_{\theta \in D_i} H(c_i(\theta, o_2) - c_i(\theta, o_1))p(\theta) \tag{4}$$

For breaking the dominance relation with all the options in the dominating set through a_i, all dominating options must have a less preferred value for a_i than that of the considered option. For numeric domains we have to integrate over all possible values of θ, check whether the given option o has lower cost than all its dominators in O^+ and weigh the probability of that particular value of θ.

$$\delta_i(o, O^+) = \sum_{\theta \in D_i} \int [\prod_{o^+ \in O^+(o)} H(c_i(\theta, o) - c_i(\theta, o^+))]p(\theta)d\theta \tag{5}$$

For qualitative domains, simply replace the integral with a summation over θ.

5.2 Probability of becoming Pareto-optimal

We consider the overall probability of becoming Pareto optimal when a preference is added as the combination of the event that the new preference is on a particular attribute, and the chance that a preference on this attribute will make the option be preferred over all values of the dominating options:

$$p^o(o) = 1 - \prod_{a_i \in A_u} (1 - p_{A_i}\delta_i(o, O^+(o))) \tag{6}$$

where $\delta_i(o, O^+)$ is the probability of simultaneously escaping many dominance relations and p_{A_i}, as said before, is the probability that the user has a preference over attribute A_i.

To generate suggestions that are adapted to the user, we update the value p_{A_i} according to the user's actions. The computation of δ_i depends on $p_i(\theta)$, that is also updated according to the user's behavior.

Following the lookahead principle, the options for which p^o is greatest are chosen as the best suggestions

5.3 Belief Update

Model-based suggestions can become very effective if they can adapt to the user, responding to his actions. In particular, after the user has made a query and some example options has been shown (*candidates* and *suggestions*), the user might state or might not state an additional preference. Observing the reaction of the user to the examples, the system can refine the uncertainty over the preference model. Suggestions stimulate the expression of those preferences on the values that are shown; therefore, if the user does not state any preference, the likelihood that there is a preference on those values will decrease.

After each interaction cycle, we need to update the p_{a_i} and $p(\theta_i)$ for all attributes i, on the basis of the user's reaction to the shown example.

We suppose to know a model of the user's reaction behavior, composed of the following probabilities (that refer to the situation in which at least one relevant option is displayed).

- $p(st|r_{i,\theta})$ the probability that the user **states** a preference given that the preference **is** in the user's preference model
- $p(st|\neg r_{i,\theta})$ the probability that the user **states** a preference given that the preference **is not** in the user's model

We will expect the second to be very low: from our experience in the user tests, users of example-based tools state preferences only when these are relevant (in contrast to the form-filling approach, where users are more likely to state preferences they do not have).

We will discuss more about the issue of obtaining a behavior model in the next section.

We use the Bayesian rule to update the probability of a particular preference being present in the user model, in the condition that a relevant option is presented to the user in the set of displayed examples. The probability that a preference on attribute i with parameter θ is present when the user has not stated any critique is the following:

$$p(r_{i,\theta}|\neg st) = \frac{p(\neg st|r_{i,\theta})}{p(\neg st|r_{i,\theta})p(r_{i,\theta}) + p(\neg st|\neg r_{i,\theta})p(\neg r_{i,\theta})} \tag{7}$$

where $p(\neg r) = 1 - p(r)$.

Once we know the new value for $p(r_{i,\theta})$, we can compute the new values for p_{A_i} and $p_i(\theta)$ that will be used to generate suggestions at the next interaction cycle.

$$p_{A_i} = \int_\theta p(r_{i,\theta})d\theta \tag{8}$$

$$p_i(\theta) = p(r_{i,\theta}) * (1/p_{A_i}) \tag{9}$$

The new value for p_{A_i}, the probability that there is a preference over a particular attribute, is obtained by the cumulative addition of the joint probability $p(r_{i,\theta}$ over $\theta)$, and the the parametric distribution $p_i(\theta)$ by dividing the joint probability $p(r_{i,\theta})$ by the attribute probability p_{A_i}.

6 Implementation

6.1 Assumptions about the cost functions

We suppose that the parameter θ represents the *reference point* of the preference stated by the user. Some assumptions about the form of the c_i can be made to simplify the calculations.

For qualitative domains, we let θ represent the value preferred by the user; the function $c_i(\theta, x)$ gives a penalty to all but the value θ for attribute i.

This would allow to express statements like "I prefer German cars", meaning that cars manufactured in Germany are preferred to cars manufactured in another country. This kind of preference statement is the most common in practice.

$$c_i(\theta, x) \equiv \textbf{ if } a_i(x) = \theta \textbf{ then } 0 \textbf{ else } 1. \quad (10)$$

For numeric domains we consider that the user can choose between a fixed set of qualitative statement like:

- `LowerThan(`θ`)` : the value should be lower than θ
- `Around(`θ`)` : the value should be around θ
- `GreaterThan(`θ`)` : the value should be greater than θ

We suppose that for a given implementation will have a particular form of representing these statements. The designers of the application will consider the kind of preference statements that the user can state through the user interface and the way to translate the statements into quantitative cost functions. A similar approach is taken by Kiessling in the design of `PREFERENCE SQL` [9], a database system for processing queries with preferences.

A reasonable possibility is to represent their cost functions as graded step functions, with the penalty value increasing from 0 to 1. In case where the degree of violation can worsen indefinitely, a ramp function can be used. For example, in a system for flight reservations where the user states that he prefers to arrive before a certain hour (θ), a possible function is the following:

$$c_i(\theta, x) = \begin{cases} (x - \theta) & \textbf{if } a_i(x) > \theta \\ 0 & \textbf{otherwise} \end{cases} \quad (11)$$

With these assumptions, the calculations for generating suggestions greatly simplify. Considering the cost function $c_i(\theta, x) =$ **if** $a_i(x) = \theta$ **then** 0 **else** 1, the probability of breaking a dominance relation between option o_1 and o_2 simplifies to the probability that the value of option o_1 for attribute i is the preferred value, when it differs from the value of o_2.

$$\delta_i(o_1, o_2) = \begin{cases} p(\theta = a_i(o_1)) & \textbf{if } a_i(o_1) \neq a_i(o_2) \\ 0 & \textbf{otherwise} \end{cases} \quad (12)$$

6.2 Approximation of the user's behavior model

The adaptive suggestions strategy has been implemented in $FlatFinder$, our example-critiquing tool for finding student accommodation.

The main difficulty concerns the belief update. Since we are dealing with a more complicated framework (from the cognitive point of view) than in a utility elicitation procedure, we need a probabilistic behavior model of the user.

In fact, supposing that our look-ahead principle (the user states a preference when she sees options that can be a reasonable choice and a new preference is required to make them optimal) is correct, we should also update the system's beliefs over uncertainty in the same fashion. We should suppose that the user has some fixed probability of stating a preference when she sees an option that would become Pareto-optimal, and update the probability that such preference is present in the model knowing the user's action (he does or does not state the preference) and the fact that the displayed option is Pareto-optimal. In other words, to update the system's beliefs we should know whether an option is optimal or not for the user, but of course if we knew that we would not need to elicit the user model!

To avoid this issue, we use a simplistic probabilistic model of the user. We suppose that $p(st|r_{i,\theta})$ and $p(\neg st|r_{i,\theta})$ are constant and refer to the situation in which an option is shown that satisfy the hypothetical preference $r_{i,\theta}$.

We clarify this with a simple example. As before we suppose there is an attribute "transportation" for an accommodation option. The cost function is of the form $c_i(\theta, x) =$ **if** $a_i(x) = \theta$ **then** 0 **else** 1. By showing an accommodation option whose transportation value is "metro", the probability that the preference $r_{\text{tranportation,metro}}$ is present in the user model is updated by multiplying to its *normalized likelihood* given the user action, that is

- $p(st|r_{\text{transp,metro}})/p(st)$ if the user has stated the preference
- $p(st|r_{\text{transp,metro}})/p(\neg st)$ if the user has not stated the preference

where $p(st)$ and $p(\neg st)$ can be decomposed using conditioned probability over r and $\neg r$, as we have shown in the general case.

6.2.1 *Algorithm for belief update*

Thanks to our simplification in the type of cost functions in use, the algorithm for updating the probability distribution is not particularly complicated. The parameter θ represents the "reference value" of the preference (the most preferred value for qualitative domains; the extreme of acceptable values for numeric domains).

For each of the displayed options o and for each of the attribute i, we update the probability $p(r_{i,a_i(o)})$ (where $\theta = a_i(o)$) conditioned to whether the user has stated an additional preference on i or not, as we have shown in the previous paragraph.

The algorithm is shown in Algorithm 1, where `update` refers to the probability update performed by Equation 7, 8 and 9.

Algorithm 1: **Belief update**
`newPref` contains the indexes of attributes on which a preference has been stated in the last interaction.

```
for all option $o \in$ DISPLAYED-OPTIONS do
    for all attribute $a_i \in$ ATTRIBUTES do
        $\theta = a_i(o)$
        stated = ($i \in$ newPref)
        update($i$, $\theta$, stated)
```

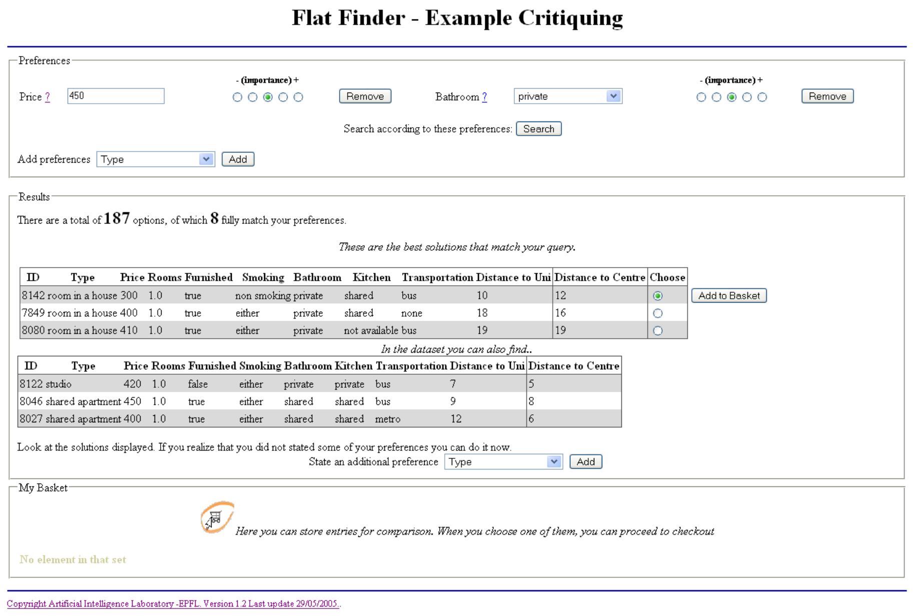
7 Evaluation

We developed FlatFinder (Figure 2), a tool for finding student accommodation using example critiquing, using data available from the faculty housing program (containing around 200 items).

In the following we review some previous results showing that users of the tool with suggestions make better decisions than the users of the tool without suggestion. We then evaluate the performance of adaptive suggestions using prior knowledge and Bayes reasoning.

7.1 Suggestions lead to better decisions

We carried out user studies to validate the effectiveness of example-based tools for preference-based search, with and without suggestions ([15]). In these experiments, model-based suggestions were calculated without prior knowledge (thus, assuming constant probability distribution) and no belief update.



Flat Finder - Example Critiquing

Preferences

Price ? 450 - (importance) + ○ ○ ● ○ ○ Remove

Bathroom ? private - (importance) + ○ ○ ● ○ ○ Remove

Search according to these preferences: Search

Add preferences Type Add

Results

There are a total of **187** options, of which **8** fully match your preferences.

These are the best solutions that match your query.

ID	Type	Price	Rooms	Furnished	Smoking	Bathroom	Kitchen	Transportation	Distance to Uni	Distance to Centre	Choose
8142	room in a house	300	1.0	true	non smoking	private	shared	bus	10	12	●
7849	room in a house	400	1.0	true	either	private	shared	none	18	16	○
8080	room in a house	410	1.0	true	either	private	not available	bus	19	19	○

Add to Basket

In the dataset you can also find..

ID	Type	Price	Rooms	Furnished	Smoking	Bathroom	Kitchen	Transportation	Distance to Uni	Distance to Centre
8122	studio	420	1.0	false	either	private	private	bus	7	5
8046	shared apartment	450	1.0	true	either	shared	shared	bus	9	8
8027	shared apartment	400	1.0	true	either	shared	shared	metro	12	6

Look at the solutions displayed. If you realize that you did not stated some of your preferences you can do it now.

State an additional preference Type Add

My Basket

Here you can store entries for comparison. When you choose one of them, you can proceed to checkout

No element in that set

Copyright Artificial Intelligence Laboratory -EPFL. Version 1.2 Last update 29/05/2005..

Figure 2. *The FlatFinder example-critiquing interface. The user has posted preferences about having a cheap accommodation (price less than 450) with a private bathroom. The best options are all rooms in a family, with bus or no public transport at all. Suggestions show that paying a bit more, the user can have a studio or a small private apartment, both are closer to the university and the centre of town; the second also is close to a metro station. The last option is still a room but is closer to the center than the options currently showed as best options. Seeing the suggestions, the user could realize that he cares about these features, and then state a preference on the transportation, for the accommodation type or on distance from centre and university.*

We found that with the aid of suggestions, users state more preferences and, more importantly, achieve a much higher decision accuracy (up to 70%). Decision accuracy was measured by asking the users to determine their best choice by carefully examining the entire database, after picking their choices with different versions of the tool; the decision of the tool was deemed accurate whenever this choice agreed with the tool.

We also carried out a user study to validate the hypothesis that obtaining preferences incrementally through example critiquing increases decision accuracy over the form-filling approach of eliciting preferences in the beginning of the process. We performed an experiment with 3 different groups of users, each using a different tool for their search with no previous exposure to the options.

The between-group experiment used 20 users in each of the three groups. They were students looking for new or better housing and thus were very motivated to carry out the experiment. Each group used one of the three versions of our preference search tool:

1. a form-filling interface
2. FlatFinder. The tool returns 6 best options according to the current preferences;
3. FlatFinder, but returning the 3 best options and 3 best suggestions according to our model-based suggestions strategy (with neither prior knowledge, nor belief update).

The results (table 2) confirm the importance of suggestions.

Using the traditional form-filling approach, only 25% of users found their most preferred solution. This fraction only increased to 35% when they could repeat the use of this interface as many times as they liked. On the other hand, example-critiquing reached a 45% accuracy. We obtained the strongest increase in accuracy, to 70%, when using example-critiquing with suggestions.

Version	Accuracy	Average Time
Form-filling	0.25	2:45
Iterated form-filling	0.35	5:30
Example-critiquing	0.45	8:09
Example-critiquing (with suggestions)	0.70	7:39

Table 2. Accuracy and task execution time for different kind of interfaces.

7.2 Effect of prior knowledge and belief update

We evaluated our adaptive strategy of generating suggestions with simulations. Assuming that our look-ahead strategy is correct, we look at the effectiveness of the suggestions in stimulating preference expression. The simulated user behaves according to an opportunistic model by stating a preference whenever the suggestions contain an option that would become optimal if that preference was added to the model with the proper weight.

To obtain realistic prior distributions we considered at logs from previous user experiments. A total of 100 interaction logs were considered.

In the simulations, users have a randomly generated set of preferences generated according to a particular probabilistic distribution. The interaction continues until either the user model is complete or the simulated user states no further preference. We measured the fraction of discovered preferences.

We compare the adaptive method for generating suggestions with the standard model-based suggestions (with no prior knowledge and no belief update, i.e uniform and constant probability distributions), the *extremes* strategy (suggesting options where attributes take extreme values, as proposed in [10]), the *diversity* strategy (computing the 20 best solutions according to the current model and then gen-

	prior distribution		
	uniform	random	real users
adaptive model based suggestions	79%	52%	65%
model based suggestions	76%	53%	60%
diversity strategy	25%	19%	20%
extreme strategy	13%	19%	17%
random suggestions	11%	9%	8%

Table 3. The average number of discovered preferences. For three different scenarios (uniform prior distribution, random prior distribution and real distribution from user studies) we compare the adaptive model-based suggestions, the standard model-based suggestion strategy, the strategy of maximizing diversity, the extreme strategy, the random selection.

erates a maximally diverse set of 5 of them, following the proposal of [11]).

We consider three sample distributions of the preferences:

- one where the probabilities are fixed and the user is equally likely to have preference on any of the attributes (p_{a_i} have same value for each i; the reference point of the preference θ is uniformly distributed)
- one where the probabilities are themselves generated by random distribution,
- and one where the probabilities are obtained by analyzing real user interaction.

In the simulations, we set $p(st|\neg r_{\mathrm{i},\theta}) = 0$, meaning that the user states only preferences that he really has. This is reasonable because the user can only state preferences on her own initiative. We do not have a strong argument for setting $p(st|r_{\mathrm{i},\theta})$, but given experience from user studies we set it to 0.66.

To handle the probability distribution of continuous attributes, we discretized the domains in few intervals, because otherwise the probabilistic update would be untractable.

The results show that using adaptive suggestions, we can improve the average number of discovered preferences, even with a very simple model of the user's behavior. However, the improvement is much smaller than what we would have expected (and there is no gain at all in the case of randomly generated distributions). We suspect that this can be due to some form of over-fitting. Moreover, the efficiency can depend on the particular distribution shape that is used to fit the preference models learnt from the logs.

More work is needed to study the variation of the efficiency of suggestions considering different values for $p(st|r_{\mathrm{i},\theta})$. It would also be interesting to make test with real users to evaluate adaptive suggestions.

8 Conclusion

This paper proposed a mixed-initiative approach for preference-based search. The system shows a set of options generated on the basis of the user's current preference model, and the user reacts by either picking one as her choice or modifying her preference model.

To increase the quality of the final decisions, the system provides suggestions to stimulate preference expression. These suggestions are adapted to the previous user's response by refining the uncertainty over the user's preferences.

We evaluate the effectiveness of suggestion strategies in eliciting user preferences. One problem in implementing this adaptive strategy is modeling user reaction to displayed examples. The simulations show that even with a very simple model for the user behavior, the efficiency of the suggestions is likely to increase.

ACKNOWLEDGEMENTS

This work has been funded by the Swiss National Science Foundation under contract No. 200020-103421.

REFERENCES

[1] Craig Boutilier, 'A pomdp formulation of preference elicitation problems.', in *AAAI/IAAI*, pp. 239–246, (2002).

[2] Urszula Chajewska, Daphne Koller, and Ronald Parr, 'Making rational decisions using adaptive utility elicitation', in *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pp. 363–369. AAAI Press / The MIT Press, (2000).

[3] Morgan Stanley Dean Witter Equity. Transportation e-commerce and the task of fulfilment, 2000.

[4] Ronald Fagin, 'Fuzzy queries in multimedia database systems', in *PODS '98: Principles of database systems*, pp. 1–10, New York, NY, USA, (1998). ACM Press.

[5] Boi Faltings, Marc Torrens, and Pearl Pu, 'Solution generation with qualitative models of preferences', in *Computational Intelligence*, pp. 246–263(18). ACM, (2004).

[6] Seung-won Hwang Hwanjo Yu and Kevin Chen-Chuan Chang, 'Rankfp: A framework for supporting rank formulation and processing.', in *ICDE 2005*, pp. 514–515, (2005).

[7] Ralph L. Keeney and Howard Raiffa, *Decisions with Multiple Objectives: Preferences and Value Tradeoffs.*, John Wiley and Sons, New York, 1976.

[8] R.L. Keeney and H. Raiffa, *Decisions with Multiple Objectives: Preferences and Value Tradeoffs.*, Cambridge University Press, New Edition, 1993.

[9] Werner Kießling and Gerhard Köstler, 'Preference sql - design, implementation, experiences.', in *VLDB*, pp. 990–1001, (2002).

[10] Greg Linden, Steve Hanks, and Neal Lesh, 'Interactive assessment of user preference models: The automated travel assistant', in *Proceedings, User Modeling '97*, (1997).

[11] David McSherry, 'Diversity-conscious retrieval.', in *ECCBR*, eds., Susan Craw and Alun D. Preece, volume 2416 of *Lecture Notes in Computer Science*, pp. 219–233. Springer, (2002).

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

[13] Robert Price and Paul R. Messinger, 'Optimal recommendation sets: Covering uncertainty over user preferences.', in *AAAI*, eds., Manuela M. Veloso and Subbarao Kambhampati, pp. 541–548. AAAI Press AAAI Press / The MIT Press, (2005).

[14] Pearl Pu and Boi Faltings, 'Enriching buyers' experiences: the smartclient approach', in *SIGCHI conference on Human factors in computing systems*, pp. 289–296. ACM Press New York, NY, USA, (2000).

[15] Pearl Pu, Paolo Viappiani, and Boi Faltings, 'Increasing user decision accuracy using suggestions', in *CHI*, p. to appear, (April 2006).

[16] Hideo Shimazu, 'Expertclerk: Navigating shoppers buying process with the combination of asking and proposing', in *(IJCAI'01)*, volume 2, pp. 1443–1448, (2001).

[17] Barry Smyth and Lorraine McGinty, 'The power of suggestion.', in *IJCAI-03 Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, 2003*, pp. 127–132, (2003).

[18] A. Tversky. Judgement under uncertainity: Heuristics and biases, 1974.

[19] Paolo Viappiani, Boi Faltings, Vincent Schickel-Zuber, and Pearl Pu, 'Stimulating preference expression using suggestions', in *IJCAI-05 Multidisciplinary Workshop on Advances in Preference*, pp. 186–191, (August 2005).