Improving Recommendation Effectiveness by Adapting the Dialogue Strategy in Online Travel Planning

Abstract

Conversational recommender systems support a structured human-computer interaction in order to assist online tourists in important online activities such as travel planning and dynamic packaging. In this paper we describe the effects and advantages of a novel recommendation methodology based on Machine Learning techniques. It allows conversational systems to autonomously improve an initial strategy in order to learn a new one that is more effective and efficient. We applied and tested our approach within a prototype of an online travel recommender system in collaboration with the Austrian Tourism portal (Austria.info). In this paper, we present the features of this technology and the results of the online evaluation. We show that the learned strategy adapts its actions to the served users, and deviates from a rigid initial strategy. More importantly, we show that the optimal strategy is able to assist online tourists in acquiring their goals more efficiently than the initial strategy. It can be used by the system designer to understand the limitations of an existing interaction design and guide him in the adoption of a new one that is capable to improve customer relationship, the usage of their web site, and the conversion rate of their online users.

Keywords: Conversational Recommender Systems, Reinforcement Learning, Markov Decision Process, Travel Planning, Dynamic Packaging, Information Presentation and Delivery

Improving Recommendation Effectiveness by Adapting the Dialogue Strategy in Online Travel Planning

Tariq Mahmood[a],
Francesco Ricci[b], and
Adriano Venturini[c]

[a]Department of Information and Communication Technology
University of Trento, Italy

[b]Faculty of Computer Science
Free University of Bozen-Bolzano
via Sernesi 1, Italy
phone: +39-0471-016971 - fax: +39-0471-016009
fricci@unibz.it

[c]ECTRL Solutions
Trento, Italy

Abstract

Conversational recommender systems support a structured human-computer interaction in order to assist online tourists in important online activities such as travel planning and dynamic packaging. In this paper we describe the effects and advantages of a novel recommendation methodology based on Machine Learning techniques. It allows conversational systems to autonomously improve an initial strategy in order to learn a new one that is more effective and efficient. We applied and tested our approach within a prototype of an online travel recommender system in collaboration with the Austrian Tourism portal (Austria.info). In this paper, we present the features of this technology and the results of the online evaluation. We show that the learned strategy adapts its actions to the served users, and deviates from a rigid initial strategy. More importantly, we show that the optimal strategy is able to assist online tourists in acquiring their goals more efficiently than the initial strategy. It can be used by the system designer to understand the limitations of an existing interaction design and guide him in the adoption of a new one that is capable to improve customer relationship, the usage of their web site, and the conversion rate of their online users.

Keywords: Conversational Recommender Systems, Reinforcement Learning, Markov Decision Process, Travel Planning, Dynamic Packaging, Information Presentation and Delivery

1 Introduction

Dynamic packaging (DP) is a cheap and flexible way of composing complex travel products; it involves assembling components into a single priced itinerary. In fact, most consumers prefer to create their own packages rather than purchase pre-packaged tours, and this has created a strong push towards the implementation of DP solutions [Rose, 2004]. But this, apparently simple, functionality requires a number of solutions and technologies: supporting prepackaging at various levels of abstraction; integrating with backend services offered by a networked set of SMEs; supporting (flexible) alternative business models for eDestinations as well as Brokers (e.g. models should be changeable); supporting interoperability at both service and data layer, since packaging is implemented by a set of brokers that endorse the proposed standard interface; supporting business models for such service integration. In addition to these important building blocks, DP requires the information system to enable the end user and the supplier to optimize the package composition based on a selection of criteria (user defined or supplier driven). This can be interpreted as a kind of negotiation between suppliers and customers that should yield a balance between consumer preferences and supplier marketing goals [Staab et al., 2002]. To this goal, the user interface of DP systems is essentially trying to duplicate the offline agency-customer interaction by tracking user preferences and providing flexible itinerary options. This paper aims at illustrating how this agency-customer interaction can be effectively supported by new recommendation technologies extending well established conversational travel planning systems [Fesenmaier et al., 2006].

In general, recommender systems are defined as intelligent applications aimed at assisting users in a decision-making process when they don't have sufficient personal experience to choose one item amongst a potentially overwhelming set of alternative products or services. The first recommender systems have been introduced in the mid 90’s (e.g., the Amazon book recommender system), and since then the application of recommender systems has gradually expanded to several kinds of products (e.g., movies, CDs, electronics, travel services), and various rather sophisticated technologies have been introduced [Adomavicius and Tuzhilin, 2005]. In fact, originally recommender systems were information systems driven by a simple but effective principle: "people provide recommendations as inputs, which the system then aggregates and directs to appropriate recipients." [Resnick and Varian, 1997]. This characterization refers to a particular kind of recommender system technology, namely collaborative or social filtering. This technique has two fundamental merits: firstly it emulates a well know social behaviour (“word of mouth”), hence it is easier for the user to accept the recommendation and for the system to justify it, e.g., Amazon comments its recommendations by saying that “users that bought this also bought that”; secondly, it does not require any particular knowledge of the application domain, e.g., the description of the recommended items, since the algorithm exploits only the ratings/recommendations provided by a community of users to a catalogue of items. So, social filtering has the merit to have started the research in the field and is still inspiring a number of hybrid approaches combining the idea of social filtering with other technologies [Ricci et al., 2006] [Adomavicius and Tuzhilin, 2005] [Burke, 2007]. It is important to stress that social filtering and other earlier technologies focus on the construction of a user specific selection of the available items, or on the personalised ranking of the full catalogue of the items. In other words, their goal is to limit the number of items offered to the user to those that the system conjectures may be more relevant for the user. For this reason they have also been called information filtering systems. Besides, these technologies are not concerned with the process of recommendation, i.e., the way the user-system interaction is structured and unfolds. This is a clear limitation of those system, since it is clear that the effectiveness of a recommendation is also linked to the adopted recommender/user dialogue strategy and recommendations' presentation [Gretzel and Fesenmaier, 2005] [Gretzel and Fesenmaier, 2006].

Conversely, a particular type of recommender systems, called "conversational", is focusing on the human computer interaction and its effects on the recommendation outcome. The main feature of these approaches is primarily their increased interactivity. They assist users in their information-seeking and decision making tasks by offering personalized product recommendations during a structured interaction session [Bridge et al., 2006]. The user and the system are seen as two interacting partners, and, for instance, they may query each other: e.g. the user may ask the system "show me a cheap restaurant", or the system may query the user by asking "please tell me if you prefer a quieter resort". In other situations they may offer each other information and recommendations: e.g., the system may show some list of recommended travel itineraries or the user may point out that she does not like a particular restaurant. It is clear that conversational travel recommender systems may be more helpful than simpler information filtering solutions in assisting online tourists in important activities related to the planning of the travel, e.g., destination and activities selection. In fact, recent conversational travel planning systems have successfully addressed the goal of supporting dynamic packaging [Fesenmaier et al., 2006]. They acquire the user preferences, either explicitly (by querying), or implicitly (by mining the user activity logs), in order to suggest interesting travel products, hence allowing the users to easily construct their travel plans and book their preferred products. In the last years, several conversational travel recommenders have been proposed, and are now operational in major tourism portals [Venturini and Ricci, 2006], travel.yahoo.com [May 15, 2009].

Let us focus now on the interaction control in these systems. At each stage of a recommendation session, a conversational system executes one from amongst a set of available actions and the selected action is specified by the system's *recommendation strategy*. For instance, a conversational recommender for itinerary planning could employ the following strategy: "explicitly acquire all the user preferences before suggesting any points of interest". Hence, according to this strategy the system will first perform actions aimed at asking the user about her travel characteristics and constraints, e.g., will show a form where the user can specify that she is going to travel alone and will not use a car in that planned trip. Then, in another form the system may acquire some more specific preferences, e.g. the user prefers a three stars hotel, and then finally the system will output travel recommendations, e.g., a small set of personalized dynamically built travel packages.

Conversational systems typically employ a *rigid* recommendation strategy, i.e., one which is determined at design time and hard-coded into the system. In fact, it is infeasible for system designers to evaluate *all* available strategies for a given task, and they select the strategy based on their experience and knowledge, which doesn't guarantee that this strategy will be optimal for all users. In fact, a strategy may be good for certain users and less applicable for others, or the same user may prefer a certain approach depending on certain contextual conditions. For instance, imagine a user that is familiar with a destination, in this case she may want to have fewer recommendations than those needed by another user that has never visited that area and is accessing the system the first time. In fact, these online user behaviors and preferences are largely unknown and there may be several factors (here called state variables) that can influence the user's decision making process and an intelligent system may try to acquire and exploit to optimize the recommendation dialogue and increase the conversion rate (selection of a travel service).

In a previous paper [Mahmood et al., 2007a], we have tackled these requirements by proposing a new type of recommender system, which exploits a class of Machine Learning techniques called Reinforcement Learning (RL) in order to *autonomously* improve a rigid strategy and learn a newer one that is *optimal*, given the particular model of the interaction represented by the system (the precise definition of "optimal" strategy is given later) [Sutton and Barto, 1998]. In the context of RL, we use the term "policy" rather than "strategy", where a policy specifies how the strategy is implemented in terms of the system's actions. In [Mahmood and Ricci, 2007a], we validated our approach through off-line simulations within the NutKing travel recommender system, improving NutKing's rigid policy with an optimal one. Moreover, in [Mahmood and Ricci, 2008b], we showed that the system learns different optimal policies for different types of user behaviours, and for different representations of the interaction process (state variables).

All these experiments were carried out offline, with simulated interactions, where the system had to learn a simple behavior. In fact, in just *one* point of the interaction the system had one action to execute among a set of alternative actions. Hence, in [Mahmood et al., 2008a] [Mahmood et al., 2009a], we proposed the application of our approach online, specifically, within the Travel Planner tool (TP) prototype that was built for the Austrian tourism web portal (Austria.info). We showed how the rigid (default) user navigation flow of the TP could be made adaptive, by allowing the system to autonomously decide which action to perform in four particular situations of the interaction (system decision points or SDPs). In those papers we presented our proposed system architecture and identified an evaluation strategy.

In this paper, we focus on the practical advantages of this approach, presenting in a unified and self contained way the motivations of our approach and its benefits as they result from the quoted online evaluation. The main conclusion is that the adaptation of the recommendation process, as it is supported by the proposed learning techniques, results in a more efficient recommendation dialogue. In fact, the strategy learned by the system can better serve the users; they are more likely to complete the travel planning task and they can complete it faster, i.e., in a smaller number of steps and in less time.

The structure of this paper is as follows. Section 2 describes the user-system interaction, and illustrates the specific application used to test our technologies. Section 3 elaborates and informally describes the proposed computational model that the system exploits to frame its decision making problems and to learn the optimal policy. Section 4 presents the specific details of our evaluation methodology, e.g., the evaluation phases, user tasks, the evaluation design etc. Section 5 presents an analysis of the learnt optimal policy. Section 6 illustrates our results, wherein we compare the values of several performance variables across the evaluation phases, in order to validate our recommendation approach. Finally, Section 7 discusses our conclusions and some future work.

2 System Interaction

In this section we illustrate the various user interface components, i.e., the system functions and GUI, implemented by the recommender system (TP) that was used to test our approach. The GUI is composed by some dynamically built Web pages. The navigation, i.e., the flow from one page to another is controlled by the system policy, reacting to user requests. In this section we will refer to a generic policy, since the goal is not to explain the behaviour of a single policy but to present the possible user requests and the corresponding system actions and page views. Later on we will illustrate some precise policies, i.e., a default one, defined by the system designer, and the one computed by the system, that is the optimal policy.

The user starts the interaction in the welcome page depicted in Figure 1. Here the user, by clicking on the "Travel Suggestions" hyperlink, can request the system to compute some complete travel plan proposals (dynamically packaged). Before showing the "Travel Suggestions" (as shown in Figure 3), which are personalized for the user, the system may ask the user to provide some travel characteristics as shown in Figure 2.

Otherwise, if the user in the starting page selects a search function, i.e., searches either for destinations, or events or experiences, the system typically asks again for the travel characteristics, similarly as in the previous user request for travel suggestions, in case these characteristics have not been acquired yet (Figure 2), but then it dispatches the user to a query form where she can enter specific preferences for the required service (Figure 4, left hand side). The query search execution can either results in a set of options (Figure 5), which are ranked using a hybrid approach based on social filtering and case based reasoning [Ricci et al., 2006], or it can fail to retrieve any service because none is satisfying all the required preferences, or because the result list is so long that the system may decide not to display it, to not confuse the user with an excessive number of recommendations. In the last two situations (too few or too many results) the system may decide to ask the user to modify the query. In the first case, i.e., when the query returns no results, the user is required to relax (i.e., discard) some preferences, in particular those that are not satisfiable (which are clearly indicated by the system, see Figure 6). In the second case the system may ask the user to enter in the query form some more preferences, in order to retrieve a smaller set of items. Here we said "the system may", to indicate that the exact behaviour of the system is determined by the current policy and can vary according to the current situation, i.e., depending on the values of certain state variables, as for instance the number of requests already made to the system or the user knowledge of the destination. This adaptive system behaviour is described in the following section.

3 Learning the Optimal Policy

The computational model that was used to learn the optimal policy exploits Markov Decision Processes (MDPs) [Sutton and Barto, 1998] for modelling the human-computer interaction and the action selection decisions of the system in response to user requests. Markov decision processes (MDPs) provide a mathematical framework for modelling decision-making in situations where the outcomes are partly random and partly under the control of the decision maker. In our application the system uses a MDP to model its decision-making problems, i.e., what computations to perform and what user interface to show, after the user makes a request (e.g., a search for hotels). The outcome of a system action, for instance the action to provide to a user a selection of hotels, is partly random because the system does not know what the user will do next, e.g., if the user will select one of these hotels or make another search. More precisely, a Markov Decision Process is a discrete time stochastic control process characterized by a set of states; in each state there are several actions from which the decision maker must choose. For a state *s* and an action *a*, a state transition function *T(s,a,s')* determines the transition probability to the next state *s'*. The decision maker earns a reward for each state transition. An important assumption of this model is that the state transitions of an MDP possess the Markov property, i.e., given the state of the MDP at time t is known, transition probabilities to the state at time t + 1 are independent of all previous states or actions. More precisely, our MDP consists of:

- **States:** the state of the interaction is observed by the system and is modelled with a set of variables (state representation). The values of these variables will change as users keep interacting with the system. In Table 1 the selected state representation consisting of 12 variables is shown. These variables have been carefully selected after some trials and off line experiments. They provide a detailed description of the interaction state, the user goals and experience, the reaction of the user to system actions (e.g., if the user accepted or not previous relaxation suggestions) and the type of products selected by the user.

- **System actions**: is the set from which the system selects one action for execution each time it is in a system decision point (SDP), i.e., in a state of the interaction where there is more than one available action. For instance, in the welcome page (Figure 1) if the user selects to query for destinations the system may either ask for travel characteristics (Figure 2) and then show the query form (Figure 4) or it can immediately show this form and avoid asking the travel preferences. The TP system has several decision points that are classified and grouped in Table 2 (we will describe later this table). Every system action is causing a state change, i.e., the system will show a new page and the user will make a new request and the values of some of the 12 variables in Table 1 will change accordingly.

- **Reward function**: assigns a numerical reward to the system for taking a particular action in a given state. This reward is used by the learning algorithm to compute the optimal policy and it should be centred on the joint goals of the system and the user, i.e., to unfold efficient recommendation sessions. It therefore assigns a large positive reward (+5) when the user adds some product to her travel plan, i.e., the main goal of the user in our system. Moreover, it assigns a small positive reward (+1) when the system shows a result page to the user, i.e., a page showing one or more products. This is the secondary goal of the users, since it dictates an intermediary (though necessary) step in order to achieve the main goal. Finally, in all other situations it assigns no reward (0).

- **Transition function**: defines the probability of a transition to a new state, depending on a particular system action and user response. This function is computed with a maximum likelihood estimate observing the users' reactions to the system actions in a training phase (described later in this paper).

As we mentioned above, SDP states are situations where the system has several available optional actions and therefore it must choose what action to execute. The possible interaction policies are defined by their behaviour at these points. The initial, default policy, i.e., that defined by the system designer, will be illustrated in Section 5. The optimal policy is not defined by any expert, but it is rather computed by a learning algorithm, in our case Policy Iteration [Sutton and Barto, 1998], and its behaviour will be illustrated in Section 5 as well. The optimal policy (by definition) is capable to identify in each state the action that if applied will enable the system to collect the maximum expected discounted cumulative reward. The system is able to compute only an "expected" reward since the outcome of any action is stochastic. Moreover, the expected reward at each future step is "discounted" multiplying it by a factor (λ) smaller than 1 to take into account that actions executed later in the process are lees likely to be executed because the process can terminate before the goal is acquired [Sutton and Barto, 1998]. Since the reward models the utility of state transitions, a policy that can get the maximum expected cumulative reward is basically optimizing the system performance. So for instance a policy that from the starting state (welcome page) can collect an expected discounted cumulative reward of 6 units, is more likely to visit results pages and to make a transition where the user adds an item to the travel cart, than a policy with expected discounted cumulative reward of 2 units. Discounts are used to penalise a reward that is acquired later in the interaction with respect to the same reward (e.g., adding a product to the cart) acquired earlier. Hence optimal policies are also producing faster interactions, i.e., interactions where the goals are acquired first (for more technical details the reader is referred to [Mahmood and Ricci, 2007a]).

The system actions available in the TP recommender system are 31. For the sake of clarity, we have grouped the system decision points (SDPs) in four general decision situations. In Table 2 we enlist these situations and the system actions set that are relevant for each situation. Here, the name of each system action is shown in parentheses, and the word "product" refers to a destination, an event, or an experience.

The decision situation A occurs when the user from the welcome page (Figure 1) requests to start a search by clicking on one of the hyperlinks "destinations", "events" or "experiences". At this point, as we illustrated in Section 2, the system can either execute *ShowTravelCharacterisitcsView,* i.e., requests the user to enter the travel characteristics (Figure 2) or show directly the query form and initial product suggestions, i.e., execute *ShowQueryProductSuggestionsView* (Figure 4). The decision situation B is encountered when in the welcome page the user requests complete travel suggestions. Here again, the system can either request the user to enter the travel characteristics (action *ShowTravelCharacterisitcsView,* Figure 2) or show directly the travel suggestions, i.e., execute *ShowTravelSuggestionsView* (Figure 3). The decision situation C occurs when the user finally requests to execute a query search entering preferences in the query form and there are some products satisfying the query (Figure 4). At this point the system has five possible actions. The first is possibly executed if there are too many results, offering to the user suggestions for tightening the current query, i.e., for adding additional preferences. The second is possibly executed if the user has not previously entered the travel characteristics; the system requests explicitly to enter them, because personal travel characteristics help the system to better rank the results. Ranking is computed by a hybrid approach (using social filtering at the session level) described in [Ricci et al., 2006]. The last three actions are all used to display the results but in different formats. In the first action (in Table 2 this is numbered as action 3) the output is very simple and it is illustrated in Figure 5.The action (numbered 4 in the table) results in a page where the user is more strongly encouraged (pushed) to accept the first recommendation and it is also easily supported with hyperlinks to start related searches (see Figure 7). The rationale is that users may be influenced by this type of presentation and could be more likely to add the recommended item in their cart. The final view for showing query results is basically equal to that produced by action 3 (Figure 5) but it also shows the hyperlinks for starting related searches as for action 4.

The final decision situation, D, is encountered by the system when a query returns no results. In this case the system may either suggest a user preference that the user can remove (action *ShowProductRelaxView,* Figure 6) or offer the user to automatically relax the failing user query (action *ShowProductAutoRelaxView)* presenting a similar view but just mentioning that the system will automatically repair this query and find a good relaxation.

As it is clear from this description, it is not easy to decide what action to execute in all possible situations, i.e., what action will be more likely to conduct the user towards the goal of the interaction (building her travel plan). In fact, the system has to take a sequence of decisions during the interaction, in the decision situations described earlier, where the state of the interaction could be very different, i.e., the twelve state variables described in Table 1 could have different values. Our basic research hypothesis is that the optimal policy, which is learnt using reinforcement learning techniques, is the best one that the system can decide to adopt.

So, in order to validate our model, we shall investigate the following hypotheses:

Hypothesis 1: The optimal policy supports more efficient and effective product search sessions, i.e., session requiring lesser user effort, than the current policy (CP) defined by the system designer.

Hypothesis 2: Users are more inclined to follow the system suggestions generated by the optimal policy than those generated by the CP.

Hypothesis 3: The optimal policy is able to offer the products that will be preferred by the user at higher positions (than the CP) in the displayed recommendation list.

We will test these hypotheses in Section 6.

4 Evaluation Methodology

We implemented two variants of our system, *VariantTrain* and *VariantTest,* which were used in two evaluation phases. In the training phase the *VariantTrain* employed a set of default policies, which were selected randomly (uniform distribution) from a set of meaningful policies. Hence, *VariantTrain* tries out as many actions as possible, in all the SDP states. Then, the ensuing user responses, modelled within the state representation, were used to estimate the transition probabilities *T(s, a, s')*, that were exploited in order to learn the optimal policy using a reinforcement learning algorithm called Policy Iteration [Sutton and Barto, 1998]. This optimal policy was used in the testing phase, by *VariantTest*. During both phases, we logged the following data:

- **The sequential data**, i.e., the state and the system action (amongst other related data), at each stage of each interaction session. The sequential data logged during the training phase was used to learn the optimal policy.

- **The performance data**, i.e., a set of variables that describes the performance of the system, as the total number of interactions during a session. A subset of these variables is shown in Table 4 and they will be discussed later when we will compare the rigid policy with the optimal one.

We note that, as we stated earlier in Section 3, the aim of the evaluation is to prove that the optimal policy, as compared to the best policy that the system designer has selected, supports a more effective interaction for the users. To this end, we compared the performance data across the two phases (with *VariantTrain* as our baseline system) in order to determine any improvement in this data from the training phase to the testing phase. We selected a set of participants for carrying out the testing phases (330 for the first phase, and 214 for the second), according to the guidelines mentioned in [Nielsen, 1993]. These users were selected, and the full evaluation was conducted, in collaboration with Vienna University of Economics and Business Administration, under the supervision of Prof. Josef Mazanec. Each participant was required to evaluate one task among two. *Task 1* was simple and dictated an interaction path for the users in which the system would never encounter a SDP. This task was introduced to perform another test that is not described here for lack of space. However the logged data from

Task 1 were used, along with those for *Task 2*, for learning the optimal policy. *Task 2* was more complex; it dictated an interaction path through the decision situations A, C and D (see Table 2). It also informed the participants of the possible system behavior at these SDPs, i.e., that the system could request for preferences, suggest or request for query changes (i.e., the tightening, relaxation and the auto-relaxation functionality), push to add some destination to the plan, and make related searches on a destination. We employed a *within-subjects* evaluation design, in which each group (one for *Task 1* and one for *Task 2*) participated in both phases and evaluated both *VariantTrain* and *VariantTest*. This caters for the large individual variation in the skill of online users [Nielsen, 1993]. A major drawback of this design is the transfer-of-skill effect, i.e., participants who have evaluated *VariantTrain* might be more experienced when they evaluate *VariantTest*. An ideal solution is to allow two groups to evaluate the two variants in different order. Such a solution was not possible in our case, as the training and the testing phases did not occur concurrently. We ensured, however, that the transfer-of-skill took place across the same task, and we stress here that the two phases were separated by more than two months to limit the transfer-of-skill.

Before carrying out the experimental evaluation, we performed a Heuristic evaluation of the Graphical User Interface (GUI) of our system, in order to judge its compliance with the standard heuristics of web usability [Nielsen, 1993]. This evaluation was carried out by usability experts as well as by students, who detected several shortcomings in our GUI. We catered for all of these limitations, and modified our GUI before the experimental evaluation.

5 Analysing the Optimal Policy

In this section, we will present an analysis of the learnt optimal policy. Initially, we present the current default policy of the TP tool (See Table 3) by specifying the action it selects for each decision situation. This policy asks for travel characteristics initially at the beginning of the interaction and doesn't ask again them if the user did not enter this information the first time. This is a normal behaviour of conversational non-adaptive systems, as they do not have the capability to decide autonomously when an action is appropriate, and we could not force the system to ask again and again the same information until the user enters it; this behaviour would not be effective. Moreover, the default policy does not offer her tightening suggestions, since in previous user studies we observed that a very small minority (6%) of the users did enter this information when they were queried. It offers relaxation suggestions but not auto-relaxation, since the user, differently from tightening suggestions, where inclined to enter this type of information. It presents search results as in Figure 5 but it does not push the user to add a products to the plan (as in Figure 7).

We will now analyse the optimal policy computed by the system after having observed the users' interactions in the training phase. The optimal policy specifies the system action for 739 SDP states (resulting from the combinations of the 12 state variables), i.e., for all the SDP states that were actually logged during the training phase. In order to facilitate the comparison of the optimal policy with the rigid policy, we grouped the SDP states under the four decision situations illustrated in Table 2. Our objective is to determine whether the optimal policy dictates different (better) system actions, than those dictated by the rigid policy (Table 3). In fact, our analysis shows that, for each situation, the optimal policy specifies different actions for different groups of SDP states, whereas the manually designed rigid policy doesn't.

In order to illustrate that these differences bring an improvement over the rigid policy, we shall present in the next section a comparison of the performance variables. In addition, here, for each situation, we considered the frequency of all possible actions learnt by the optimal policy in that situation. This is important, as actions learnt for a larger number of states might imply that they are more beneficial for the users (in acquiring their goals), as compared to the others. In addition, we were also interested in understanding the exact situations under which the optimal policy specifies some action. To this end, we selected and analyzed the values of a subset of state variables, for each type of action. We now present our analysis for the different decision situations.

Decision Situation A and **Decision Situation B** (the system must cope with the initial request of the user, either for a query search (Situation A), or for a request to show some travel suggestions (Situation B)**).** In Situation A, the optimal policy specifies *ShowQueryProductSuggestionsView,* i.e., to show the query form and initial simple product suggestions, for 66% of the states, and to *ShowTravelCharacterisitcsView,* i.e., to request the user for her travel characteristics at the beginning of her query search session for the remaining 34% of the states. In other words, it is optimal to give immediately to the user some simple, non personalized initial product suggestions and letting the user to formulate a query, almost twice as much as requesting the user for her travel characteristics (that can enable later the system to provide better ranked recommendations). In Situation B, the policy specifies both *ShowTravelSuggestionsView*, i.e., to show the travel suggestions to the user, and

ShowTravelCharacterisitcsView, i.e., to ask for travel characteristics, for 50% of the states, i.e., both actions are specified with the same frequency. For both of these situations, our analysis reveals that, it is optimal to execute *ShowTravelCharacterisitcsView* when she has seen none, or only a small number of result pages, and preferably, has not, as yet, started query-searching for products. This further implies that, it is not convenient to request users for travel characteristics, once they have started searching for the preferred products.

Decision Situation C (the system must cope with a non-failing query). The optimal action with the maximum frequency is *ShowAddRelSearchView*, i.e., to push the user to add the top-ranked destination to her plan, and suggest her to make related searches on this destination, which occurs for 33% of the states. Consider a user who is querying our system for destinations, and has accepted any of the system's suggestions/offers (tightening, relaxation, auto-relaxation) for modifying her query, or has executed a manually-constrained query, such that a product subset has been retrieved. Our analysis reveals that, in this situation, it is optimal to execute *ShowAddRelSearchView*, i.e., to explicitly push the user to add the top ranked destination, and to make searches related to this destination. This behavior makes sense because, when the system pushes the user to add a product, there is greater chance that she will actually add this product to her plan, and acquire her main goal. In doing so, the system also fulfils our secondary goal, that is to show a result page. Furthermore, the optimal policy dictates *ShowProductTightenView*, i.e., suggest tightening features, for 30% of the SDP states (second largest occurrence frequency). Our analysis reveals that the optimal policy dictates tightening, largely for users who are willing to accept it. In case the users are unwilling, the optimal policy suggests tightening only for users whose goals have already been acquired, so the policy can afford the risk of suggesting tightening, and who might require the system's assistance in continuing their interaction. Moreover, the optimal policy dictates *ShowProductRecView*, i.e., to show the simplest type of result page, for 24% of the states (third largest occurrence frequency). Overall, we found that the system largely suggests tightening, and shows the simplest result page, later on in the interaction, when their main and secondary goals have already been acquired and they have viewed a large number of result pages. Moving on to other actions, only 7% of the states, dictate *ShowProductAskCharsView*, i.e., to request the user for travel characteristics during her query search session, which indicates that users were largely unwilling to specify their characteristics during their query search sessions, or that this is not beneficial at that stage of the interaction.

Decision Situation D (the system must cope with a failing query**).** The optimal policy specifies *ShowProductAutoRelaxView* (offer auto-relaxation) for 53% of these states, and *ShowProductRelaxView* (suggest relaxation features) for the remaining 47% states, i.e., these actions are specified with an almost similar frequency. Our analysis reveals that it's optimal to request auto-relaxation and suggest relaxation, earlier on in the interaction, when the main goal of the users has not been acquired.

6 Analysing the Performance Variables

In this section, we will present other results of the experimental evaluation. Specifically, we selected a set of 25 performance variables, and determined whether there was a significant improvement in their values, from the training phase to the testing phase.

The significance is determined through a two-tailed, unequal variance t-test, where a p-value of less than 0.05 is considered statistically significant, while a p-value of less than 0.1 is indicative of a statistical trend. We found significant differences in the values of some variables as it is shown in Table 4. We note that we don't show the other variables due to space constraints. Here, the column "Performance Variable" lists the variables, "Train" and "Test" represent their mean values in the Training and Testing phase respectively.

On the average, users added at least two-three items to their carts in both the phases, i.e., their main and secondary goals were acquired. We note that in the test phase the users added less items and we stress that the task prescribed to add at least one item (destination). Moreover, our results show that the user goals were acquired in the testing phase with a smaller number of result page views (2.7 for Test vs. 3.3 for Train), in a shorter time (15 minutes for Test vs. 29 for Train) query executions (1.5 vs. 1.8) and search requests (1.3 vs. 1.8), i.e., our optimal policy assisted the users in acquiring their goals more efficiently (quickly) in the testing phase, as compared to the training phase. We must note that the users could also have acquired their goals quickly because they were more experienced with our system in the testing phase. However, the search behavior of E-commerce users is strongly influenced by the actions of the system [Katz, 2001], and the quite large distance in time between the training and test phases (two months), implies that the impact of the previously acquired experience should have been marginal. In fact, our optimal policy *did* influence the users in many ways, as we have shown above.

In addition, we analyzed the acceptance rates of the users for the various system offers/suggestions. Here for system offer we mean an action of the system that requests an explicit choice of the user. For instance if the system suggests to relax a specific preference that cannot be satisfied in a query search, then the user can either go with this suggestion and relax the query or basically ignore it and make another request that is not strictly related to the request (e.g., to modify autonomously the query or search for another type of products). Our aim was to determine whether the optimal policy was able to positively influence the willingness of the users, in accepting the system's requests, i.e., if the user follows what the system designer thought to be the most rational decision. To check this aspect we measured whether there was some improvement in the acceptance rates from the training to the testing phase. Amongst 12 types of requests, we found that for five of them the acceptance rate improved and decreased for two (precise data are not shown here for lack of space). Hence the optimal behavior did influence the willingness of the users in replying to the system's requests, but not always as the system designer expected, hence in some sense providing a useful critical evaluation of the proposed interaction design.

We will now refer back to our proposed hypotheses, and test them according to the aforementioned results.

- **Hypothesis 1:** *The optimal policy supports more efficient product search sessions (i.e., those requiring lesser user effort) than the current policy (CP) defined by the system designer.* We have validated this hypothesis because, in the testing phase, i.e., with the optimal policy, users added products to their carts with lesser effort (see Table 4). This outcome is clearly indicated by the reduced number of interaction stages, reduced session duration and reduced number of queries. In fact we also observed a 10% increment in the number of successful sessions between the Train and Test phases, i.e., in the Test sessions more users were able to add at least one item to the travel plan, hence accomplishing the required task.

- **Hypothesis 2:** *Users are more inclined to follow the system suggestions generated by the optimal policy than those generated by the CP.* We have only partially validated this hypothesis, because the users' acceptance rate of the system's offers, produced by the optimal policy, increased for certain offers and decreased for others. In order to explain these results, we must observe that the optimal policy is aimed at gathering the largest expected cumulative discounted reward, and the system is rewarded when users acquire their goals. Moreover, the expected responses are those that designers thought to be useful for a positive interaction output, i.e., acquiring the user goals. However, even if some offers were not followed with the expected response it could be still optimal to make these offers, in some situations, in order to acquire the ultimate interaction goal (add a product to the plan).

- **Hypothesis 3:** *The optimal policy is able to offer the products that will be preferred by the user at higher positions (than the CP) in the displayed recommendation list*. This hypothesis is not validated by our experimentation evaluation, as we observed no significant difference between Test and Train. The selected item in the Test was ranked in position 3.0 (average) whereas in the Train was ranked in position 3.1. In fact this result may not be unexpected, as the optimal policy is not directly optimising the position of the preferred products. In some sense this result shows that the item selection behaviour, i.e., the selection decisions of the user are not influenced by differences in the process. In other words the two processes let the user to select the same items but the optimal policy is supporting this selection process in a more efficient way.

7 Conclusions and Future Work

In our research we have proposed a novel methodology for conversational recommender systems, which exploits Reinforcement Learning techniques, in order to autonomously learn an optimal (user-adaptive) strategy for assisting online users in acquiring their goals. In this paper, we have applied our approach within a prototype for an online travel recommender system for the Austrian Tourism portal (Austria.info). We successfully learnt the optimal policy and showed that it dictates intelligent and adapted system actions for the users. We successfully validated its performance against a set of non-adaptive default policies, hence showing that it is able to 1) assist the users in acquiring their goals more efficiently, 2) partially increase the willingness of the users in accepting several of the system's requests/offers, and 3) is doing that without impacting on the user item selection decisions of the user. The proposed method can be used to improve a policy defined by a system designer. Hence, we believe that applying this technology to an existing system can have two major benefits for the system provider. First he can critically reflect on the chosen human-computer interaction and can be suggested to revise this design in the light of the behaviour of the optimal policy. Second, if the designer will enforce in the system the optimal policy he will observe a significant increase in the conversion rate, i.e., more users will be able to complete successfully the travel packaging process and this improved process will let to save a significant amount of resources, since the interactions with the system will be faster and more efficient.

Our work is the first attempt in the domain of Travel and Tourism applications, and particularly in the domain of dynamic packaging systems, to learn a strategy for information presentation/delivery for assisting online tourists in planning their vacations and booking their holidays. This approach can be adopted by any conversational system to improve the currently selected policy. As our future work, we are interested in generalizing our recommendation approach to other tourism portals and to test it with other conversational strategies. For instance we believe that the proposed technique can be beneficial in the implementation of wizard-based interfaces. In this context, we plan to extend some results obtained in [Mahmood and Ricci, 2008] in order to initially determine a generic state representation and a generic reward model which could be used for learning the optimal policy for different portals.

References

Adomavicius, G. Tuzhilin, A. (2005). Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering, vol 17(6): 734-749.

Bridge, D., Goker, M., McGinty, L., and Smyth, B. (2006). Case-based recommender systems. The Knowledge Engineering review, 20(3), 315– 320.

Burke, R. (2007). Hybrid web recommender systems. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), The adaptive web: Methods and strategies of web personalization (pp. 377-408). Heidelberg: Springer.

Fesenmaier, D. R.,Werthner, H., and Woeber, K. (2006). Destination Recommendation Systems: Behavioural Foundations and Applications. CABI Publishing.

Gretzel, U. and Fesenmaier, D. (2005). Persuasiveness of preference elicitation processes in destination recommendation systems. In Information and Communication Technologies in Tourism 2005, pages 194-204. Springer.

Gretzel, U. and Fesenmaier, D. (2006). Persuasion in recommender systems. Int. J. Electron. Commerce, 11(2):81-100.

Katz, M.A. (2001). Searching and Browsing on E-commerce Sites: Frequency, Efficiency and Rationale. Doctoral disseration, Rice University, Houston, TX.

Mahmood, T. and Ricci, F. (2007a). Learning and adaptivity in interactive recommender systems. In Proceedings of the ICEC'07 Conference, Minneapolis, USA, 75-84.

Mahmood, T., Cavada, D., Ricci, F., and Venturini, A. (2007b). Search and recommendation functionality. Technical Report D5.1, eTourism Competence Center Austria, Technikerstr. 21a, ICT-Technologiepark, 6020 Innsbruck.

Mahmood, T., Ricci, F., Venturini, A., and Höpken, W. (2008a). Adaptive recommender systems for travel planning. In O'Connor, P., Höpken, W., and Gretzel, U. (eds), *Information and Communication Technologies in Tourism 2008, proceedings of ENTER 2008 International Conference*, Innsbruck, 2008. Springer, 1-11.

Mahmood, T. and Ricci, F. (2008b). Adapting the Interaction State Model in Conversational Recommender Systems . In Proceedings of the ICEC'08 Conference, Innsbruck, Austria, 1-10.

Mahmood, T., Ricci, F., and Venturini, A. (2009a). Learning Adaptive Recommendation Strategies for Online Travel Planning. In Information and Communication Technologies in Tourism 2009, Pages: 149-160, Springer, Wien New York.

Nielsen, J. (1993). *Usability engineeri*ng. San Francisco: Morgan Kaufmann Publisher.

Resnick P., and Varian, H. R. (1997). Recommender systems. Communications of the ACM, 40, 3, 56-58.

Ricci, F., Cavada, D., Mirzadeh, N., and Venturini, A. (2006). Case-based travel recommendations. In Fesenmaier, D. R., Woeber, K. W., and Werthner, H., editors, Destination Recommendation Systems: Behavioural Foundations and Applications, pages 67-93, Oxford, CAB Publishing.

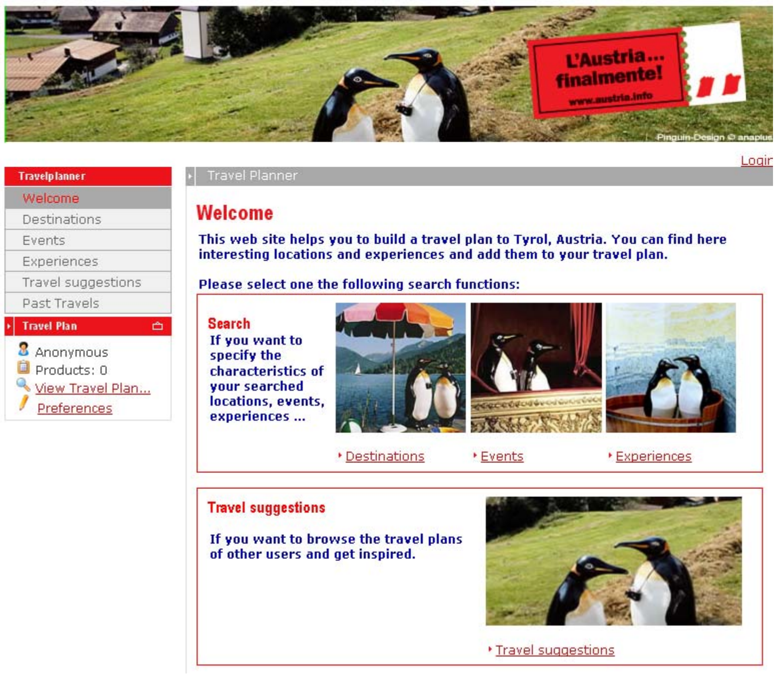
Rose, N. (2004) Selling Complex Leisure Travel Online: Focus on Dynamic Packaging Technology, Sherman: PhoCusWright.

Staab, S., Werthner, H., Ricci, F., Zipf, A., Gretzel, U., Fesenmaier, D. R., Paris, C., Knoblock, C. A. (2002). Intelligent Systems for Tourism. IEEE Intelligent Systems 17(6): 53-64.

Sutton, R. S. and Barto, A. G. (1998). Reinforcement Learning: An Introduction. MIT Press.

Venturini, A. and Ricci, F. (2006). Applying trip@dvice recommendation technology to www.visiteurope.com. In Proceedings of the 17th European Conference on Artificial Intelligence, Riva del Garda, Italy, Aug 28th - Sept 1st, 607-611.

Zanker, M, Fuchs, M., Höpken, W., Tuta, M., and Müller, N. Evaluating Recommender Systems in Tourism — A Case Study from Austria. In Information and Communication Technologies in Tourism 2008, proceedings of ENTER 2008 International Conference, Innsbruck, 2008. Springer, 24-34.



Welcome

This web site helps you to build a travel plan to Tyrol, Austria. You can find here interesting locations and experiences and add them to your travel plan.

Please select one the following search functions:

Search
If you want to specify the characteristics of your searched locations, events, experiences ...

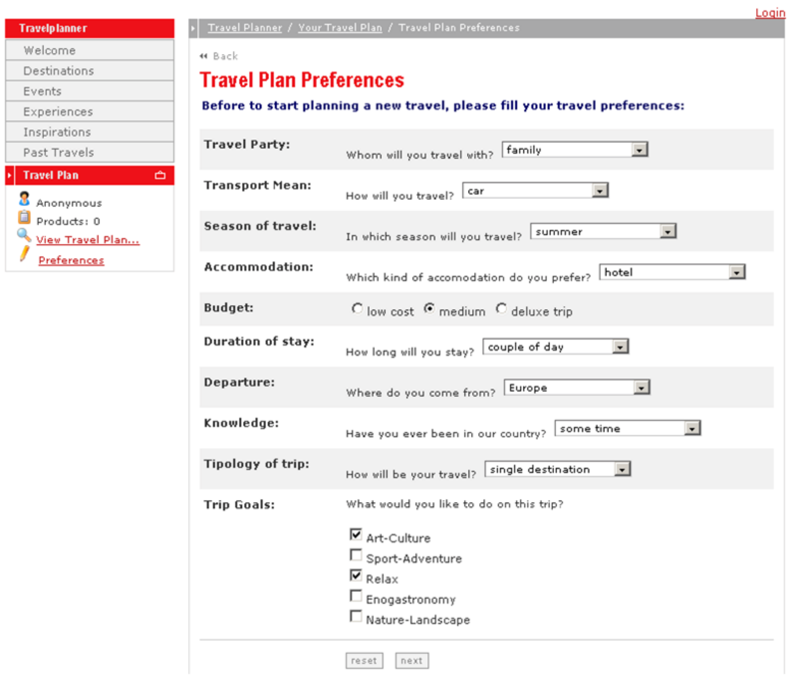
Destinations Events Experiences

Travel suggestions

If you want to browse the travel plans of other users and get inspired.

Travel suggestions

Figure 1. The view State shown at the beginning of the interaction



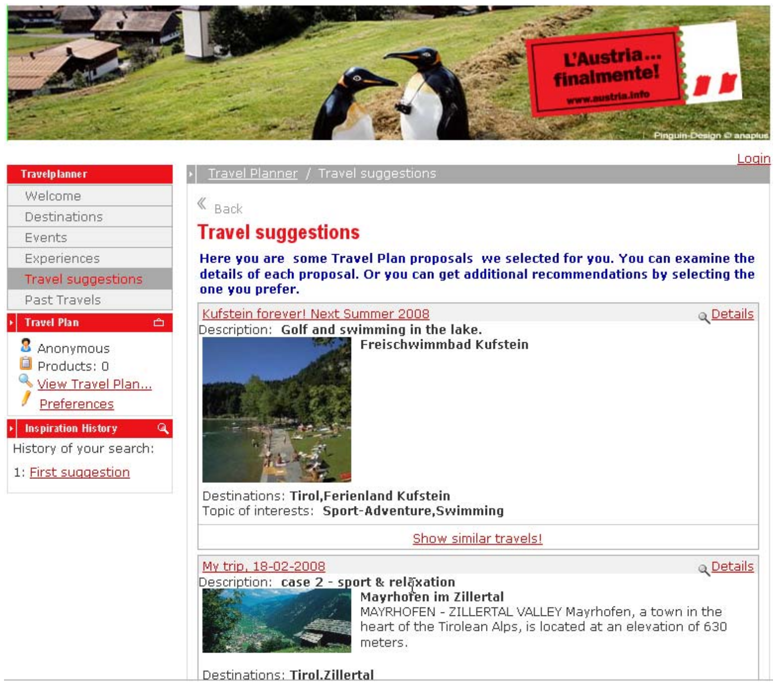
Travel Plan Preferences

Before to start planning a new travel, please fill your travel preferences:

Travel Party:	Whom will you travel with?	family
Transport Mean:	How will you travel?	car
Season of travel:	In which season will you travel?	summer
Accommodation:	Which kind of accomodation do you prefer?	hotel
Budget:	low cost [✓ medium] deluxe trip	
Duration of stay:	How long will you stay?	couple of day
Departure:	Where do you come from?	Europe
Knowledge:	Have you ever been in our country?	some time
Tipology of trip:	How will be your travel?	single destination
Trip Goals:	What would you like to do on this trip?	

- ☑ Art-Culture
- ☐ Sport-Adventure
- ☑ Relax
- ☐ Enogastronomy
- ☐ Nature-Landscape

Figure 2. Page used to insert travel characteristics



Login

Travelplanner
Welcome
Destinations
Events
Experiences
Travel suggestions
Past Travels

Travel Plan
Anonymous
Products: 0
View Travel Plan...
Preferences

Inspiration History
History of your search:
1: First suggestion

Travel Planner / Travel suggestions

Back

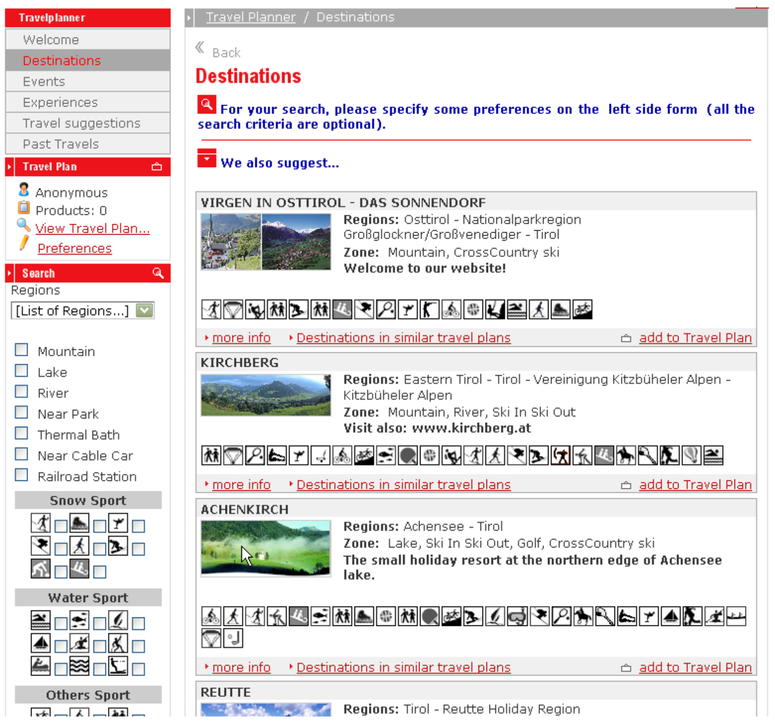
Travel suggestions

Here you are some Travel Plan proposals we selected for you. You can examine the details of each proposal. Or you can get additional recommendations by selecting the one you prefer.

Kufstein forever! Next Summer 2008 — Details
Description: Golf and swimming in the lake.
Freischwimmbad Kufstein
Destinations: Tirol,Ferienland Kufstein
Topic of interests: Sport-Adventure,Swimming
Show similar travels!

My trip, 18-02-2008 — Details
Description: case 2 - sport & relaxation
Mayrhofen im Zillertal
MAYRHOFEN - ZILLERTAL VALLEY Mayrhofen, a town in the heart of the Tirolean Alps, is located at an elevation of 630 meters.
Destinations: Tirol,Zillertal

Figure 3. Travel suggestions view


Travelplanner
Welcome
Destinations
Events
Experiences
Travel suggestions
Past Travels

Travel Plan
Anonymous
Products: 0
View Travel Plan...
Preferences

Search
Regions
[List of Regions...]

- Mountain
- Lake
- River
- Near Park
- Thermal Bath
- Near Cable Car
- Railroad Station

Snow Sport

Water Sport

Others Sport

Travel Planner / Destinations

Back

Destinations

For your search, please specify some preferences on the left side form (all the search criteria are optional).

We also suggest...

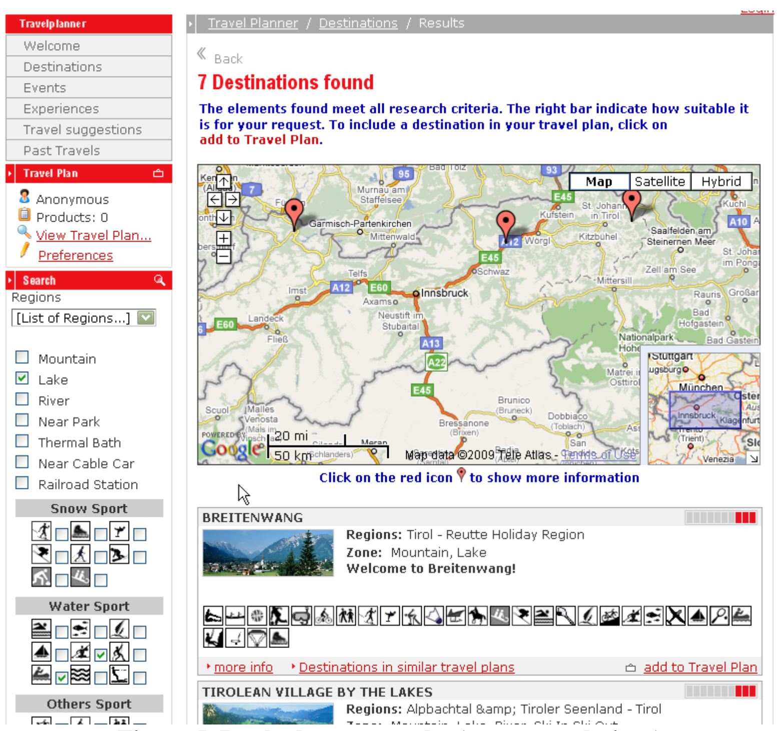
VIRGEN IN OSTTIROL - DAS SONNENDORF
Regions: Osttirol - Nationalparkregion Großglockner/Großvenediger - Tirol
Zone: Mountain, CrossCountry ski
Welcome to our website!
more info · Destinations in similar travel plans · add to Travel Plan

KIRCHBERG
Regions: Eastern Tirol - Tirol - Vereinigung Kitzbüheler Alpen - Kitzbüheler Alpen
Zone: Mountain, River, Ski In Ski Out
Visit also: www.kirchberg.at
more info · Destinations in similar travel plans · add to Travel Plan

ACHENKIRCH
Regions: Achensee - Tirol
Zone: Lake, Ski In Ski Out, Golf, CrossCountry ski
The small holiday resort at the northern edge of Achensee lake.
more info · Destinations in similar travel plans · add to Travel Plan

REUTTE
Regions: Tirol - Reutte Holiday Region

Figure 4. Query form (left) and some suggested destinations



Travel Planner / Destinations / Results

Back

7 Destinations found

The elements found meet all research criteria. The right bar indicate how suitable it is for your request. To include a destination in your travel plan, click on add to Travel Plan.

Anonymous
Products: 0
View Travel Plan...
Preferences

Regions
[List of Regions...]

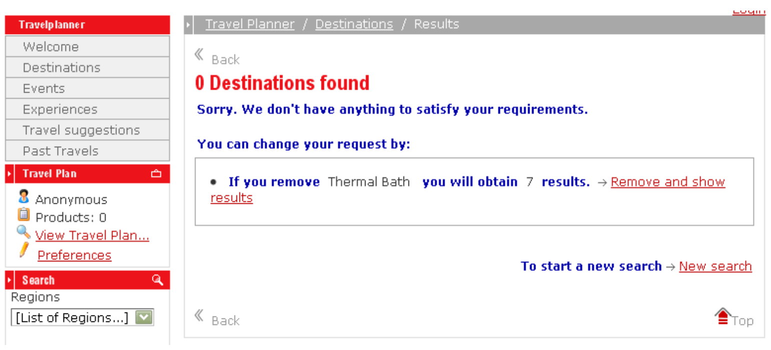
- [ ] Mountain
- [✓] Lake
- [ ] River
- [ ] Near Park
- [ ] Thermal Bath
- [ ] Near Cable Car
- [ ] Railroad Station

Click on the red icon to show more information

BREITENWANG
Regions: Tirol - Reutte Holiday Region
Zone: Mountain, Lake
Welcome to Breitenwang!

TIROLEAN VILLAGE BY THE LAKES
Regions: Alpbachtal &amp; Tiroler Seenland - Tirol

Figure 5. Ranked query results (recommendations)



Travel Planner / Destinations / Results

Back

0 Destinations found

Sorry. We don't have anything to satisfy your requirements.

You can change your request by:

- If you remove Thermal Bath you will obtain 7 results. → Remove and show results

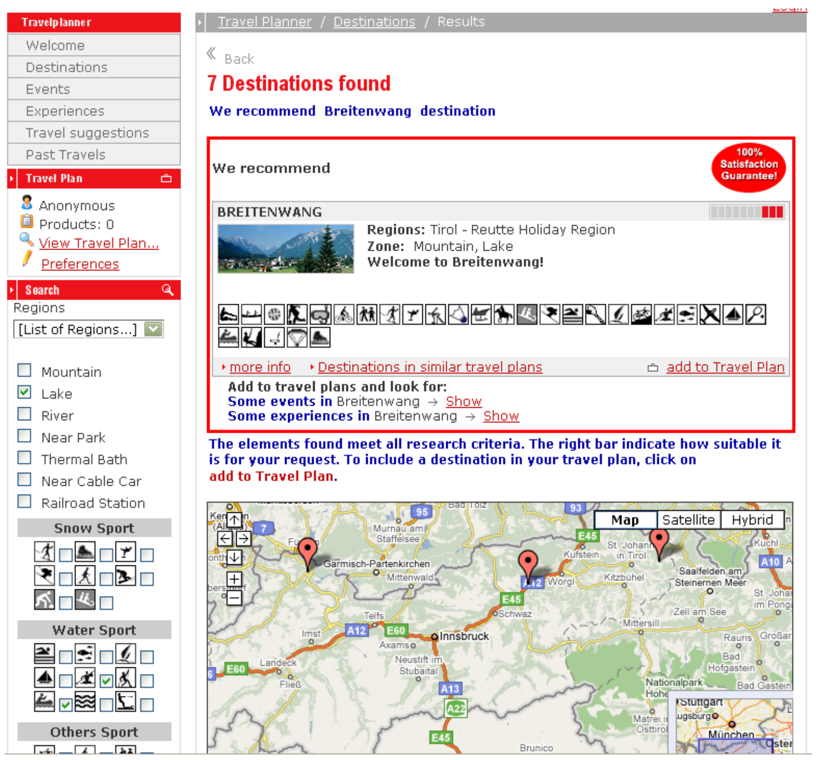
To start a new search → New search

Back

Anonymous
Products: 0
View Travel Plan...
Preferences

Regions
[List of Regions...]

Figure 6. Query relaxation suggestion


Travelplanner

Welcome

Destinations

Events

Experiences

Travel suggestions

Past Travels

Travel Plan

Anonymous

Products: 0

View Travel Plan...

Preferences

Search

Regions

[List of Regions...]

- [ ] Mountain
- [x] Lake
- [ ] River
- [ ] Near Park
- [ ] Thermal Bath
- [ ] Near Cable Car
- [ ] Railroad Station

Snow Sport

Water Sport

Others Sport

Travel Planner / Destinations / Results

Back

7 Destinations found

We recommend Breitenwang destination

We recommend

100% Satisfaction Guarantee!

BREITENWANG

Regions: Tirol - Reutte Holiday Region

Zone: Mountain, Lake

Welcome to Breitenwang!

more info

Destinations in similar travel plans

add to Travel Plan

Add to travel plans and look for:

Some events in Breitenwang → Show

Some experiences in Breitenwang → Show

The elements found meet all research criteria. The right bar indicate how suitable it is for your request. To include a destination in your travel plan, click on add to Travel Plan.

Figure 7. Strongly recommended query results

Table 1. State Variables and their Descriptions

State Variable	Description
UserAction	label ranging on all possible user actions
CurrentResultSize (CRS)	the number of products retrieved by a query
CharacteristicsSpecified	whether the user, up to the current stage, has specified her travel characteristics (or not),
CartStatus	whether the user, up to the current stage, has added some product to her cart (or not),
ResultPagesViewed (RPV)	the number of result pages viewed by the user up to some stage
UserGoal	the goal of the user during her session. In our application this is always "travel planning"
UserExperience	the user experience on tourism in Austria
UserTightResponse	the response of the user to the system requests to provide more preferences (tightening suggestions)
UserRelaxResponse	the response of the user to the system suggestions to remove some failing conditions in query searches (relaxation suggestions)
UserAutoRelaxResponse	the response of the user to the *auto-relax* offer, i.e., the action that enables the system to autonomously remove some failing conditions in a query search
Position of the most Recent product which the user has Added to her travel Plan (PRAP)	"Position" refers to the product's location in the ranked list of displayed products on a given result page
Score of the most Recent product which the user has Added to her travel Plan (SRAP)	The product "Score" is a value between 1 and 100 (for details, see [Venturini and Ricci, 2006]) and it is the recommender system's estimation of the goodness of the recommendation

Table 2. The Decision Situations, their Explanation, and the set of System Actions available under each Situation.

Decision Situation	**Description**	**System Action Set**
Decision Situation A	The user enters the system and submits a request to initiate the query search for products, at the SDP *Start Query Search*	**1)** show the initial query form and product suggestions to the user (*ShowQueryProductSuggestionsView*), **2)** request the user to specify her travel characteristics (*ShowTravelCharacterisitcsView*) and then show the initial product suggestions
Decision Situation B	The user enters the system and requests the travel suggestions (complete travel plans), computed by the system at the SDP *Show Proposals*	**1)** show the product proposals computed by the system **(***ShowTravelSuggestionsView*), **2)** request the user to specify her travel characteristics (*ShowTravelCharacterisitcsView*) and then show the product proposals.
Decision Situation C	After Situation A, the user submits a product query at the SDP *Execute Query*, and one or more products have been retrieved, i.e., *CurrentResultSize* > 0	**1)** suggest features for tightening the current query (*ShowProductTightenView*), **2)** show a result page in which the system requests the user to specify the travel characteristics (*ShowAskTravelCharacterisitcsView*), **3)** show the simplest type of result page to the user (*ShowProductRecView*), **4)** show a result page which pushes the user to add the top ranked destination to her plan and also offers her to make related searches on this destination (*ShowAddRelSearchView*), and **5)** show a result page which suggests the user to make related product searches on the top ranked destination (*ShowRelSearchView*)
Decision Situation D	After Situation A, the user submits a product query at the SDP *Execute Query*, and the query fails, i.e., *CurrentResultSize = 0*	**1)** suggest a set of features for relaxing the current failing product search query (*ShowProductRelaxView*), **2)** acquire the consent of the user for an automatic relaxation of her product query (*ShowProductAutoRelaxView*)

Table 3. Current Rigid Policy of the TP tool

Decision Situation	Rigid Policy Action
Decision Situation A	*ShowQueryProductSuggestionsView*
Decision Situation B	*ShowTravelSuggestionsView*
Decision Situation C	*ShowProductRecView*
Decision Situation D	*ShowProductRelaxView*

Table 4. Performance Variables

Performance Variable	Train	Test	p-value
Number of elapsed interaction stages	10.5	8.1	0.0005
Interaction Session Time (minutes)	29.03	15.8	0.0242
Number of destination queries executed	1.8	1.5	0.014
Number of requests for destination search	1.8	1.3	0.00001
Number of products retrieved while searching for destinations through QS	22.6	30.8	0.467
Number of result pages viewed by the user	3.3	2.7	0.095
Number of products added to the cart	2.8	2.4	0.079