Case-Based Recommender Systems: a Unifying View

Fabiana Lorenzi[1,2] and Francesco Ricci[2]

[1] Universidade Luterana do Brasil
Rua Miguel Tostes, 101
RS, Brasil
lorenzi@ulbra.tche.br
[2] eCommerce and Tourism Research Laboratory
ITC-irst
via Solteri 38
38100 Trento, Italy
ricci@itc.it

Abstract. This paper presents a unifying framework to model case-based reasoning recommender systems (CBR-RSs). CBR-RSs have complex architectures and specialize the CBR problem solving methodology in a number of ways. The goal of the proposed framework is to illustrate both the common features of the various CBR-RSs as well as the points were these systems take different solutions. The proposed framework was derived by the analysis of some systems and techniques comprising nine different recommendation functionalities. The ultimate goal of the this framework is to ease the evaluation and the comparison of case-based reasoning recommender systems and to provide a tool to identify open areas for further research.

1 Introduction

Recommender systems are being used in e-commerce web sites to help customers in selecting products more suitable to their needs. The growth of Internet and the business to consumer e-Commerce has brought the need for such a new technology [32]. In the past years, a number of research projects have focused on recommender systems [25, 7, 30, 8, 27]. These systems learn about user preferences over time and automatically suggest products that fit the learned user model.

Case-Based Reasoning (CBR) is one of the most successful machine learning methodologies that exploit a knowledge-rich representation of the application domain [1, 36, 2]. Basically, CBR is a problem solving methodology that addresses a new problem by first retrieving a past, already solved similar case, and then reusing that case for solving the current problem. In the most straightforward application of CBR to recommendation generation, the case base models the products to be recommended and the set of suggested/recommended products is retrieved from the case base by searching for products similar to that partially described by the user [7]. In these approaches a case and a product are

essentially considered as identical objects. The problem component of the case is typically represented by a set of product features, those specified by the user, and the solution component of the case is the product itself.

In the basic usage scenario, the customer is looking for some product to purchase. He/she makes explicit some requirements about the product being searched for and the system searches the case base for products that match the user requirements. The retrieval process is driven by a similarity metric that computes the similarity of the problem description, i.e., the current user requirements and the products in the case base. A set of cases products is then retrieved from the case base and these products are recommended to the user. If the user is not satisfied with these suggestions he/she can modify the requirements in the query and a new recommendation cycle is started.

In a Case-Based Reasoning Recommender System (CBR-RS) the effectiveness of the recommendation is based on: the ability to match user preferences with product description; the tools used to explain the match and to enforce the validity of the suggestion; the range of available functionalities and the graphical interface that support the user in browsing the information content, either the cases or the products to recommend.

In this paper we propose an interpretation framework that models how CBR-RSs behave. We have derived this model by an analysis of the literature, that even if not complete, includes a good number of radically different approaches. In carrying out such an analysis we realized that the literature is fragmented and even contradictory in explaining the effective adoption of the CBR methodology to generate recommendations. The proposed framework: a) specializes the general CBR steps to the recommendation task; b) describes how a recommender system exploits the classical CBR learning loop (retrieve-reuse-revise-review-retain); and c) illustrates how different classes of CBR-RSs further specialize the general CBR model.

To validate the framework we considered several approaches recently developed either as techniques or as full recommender systems. For instance, Entree [7], is a restaurant recommender system that provides recommendations by finding restaurants in a new city similar to restaurants the user knows and likes. The Interest Confidence Values [22] is a recommendation technique where a new product (restaurant) is suggested reasoning on the explicit and implicit user's interests on previously considered products, that are stored in the case base. This approach exploits also a forgetting mechanism that allows the system to distinguish between current and old interests. In the Comparison-Based Retrieval technique [16] is used a preference-based feedback approach that transforms the user's preference into explicit query modifications. In another system, DieToRecs [28, 11], the goal is to help the user to plan a leisure trip. DieToRecs is able to personalize the current recommendation on the base of previously stored recommendation sessions (cases). In the Compromise-Driven Retrieval technique cases are retrieved and grouped according to the compromises done by the system, i.e., the user requirements which cannot be satisfied, and therefore are relaxed by the retrieval algorithm [17]. In the Order-Based Retrieval technique [4] a number of

different order relationships among cases are managed to optimally sort the recommended products.

In summary, the goal of this paper is to present a good sample of CBR-RSs and explain the proposed methods using a unifying notation that will ease the reader in understanding their relationships, respective benefits and shortcomings. We aim at offering this research as a starting point for further analysis, identifying still unexplored solutions and therefore motivating new research in the field.

The paper is organized as follow. A brief overview of recommender system technologies is given in the next section. Section 3 describes the whole Case-Based Reasoning process and how it is used in recommender systems. Section 4 discusses the proposed framework. Section 5 illustrates the chosen examples of recommendation techniques and discusses how they fit in the interpretation framework. Section 6 presents a summary comparison of the techniques presented in the previous section. Finally, Section 7 summarizes the conclusions of the paper.

2 Recommender Systems

In the past years, a number of research projects have focused on recommender systems [25, 31, 32]. These systems try to learn user preferences over time and to automatically suggest products that fit the learned user model. E-commerce sites are using recommender systems to suggest products to their customers and improve the look to buy ratio.

The most popular recommendation technique is collaborative filtering that aggregates data about customer's preferences (products' ratings) to recommend new products. Amazon.com is a very popular example of an e-Commerce site that exploits a collaborative-filtering approach. In its book section for instance, the system encourages direct feedback from customers about books they already read [32]. After this, the customer may request recommendation for books that he/she might like. Another notable example is MovieLens [19], a well-known movie recommender system that bases its recommendations on collaborative filtering as well.

Content-based filtering is another recommendation technique that basically exploits the preferences (past and current) of a specific customer to build new recommendations to the customer. NewsDude [3], for instance, observes what online news stories the user has read and not read and learns to present the user with articles he/she may be interested to read. Content-based systems are usually implemented as classifier systems based on machine learning research [37].

In collaborative filtering the recommendation depends on customers' information, and a large number of previous user/system interactions are required to build reliable recommendations. In content-based systems only the data of the current user are exploited in building a recommendation. It requires a description of user interests that is either matched in the items' catalog or provided as input

for the learned user model to output a recommendation. Both approaches, if not trained with lot of examples (product ratings or pattern of user preferences), deliver poor recommendations. This limitation mostly motivated a third approach, knowledge-based, that tries to better use preexisting knowledge specific of the application domain (e.g. travels vs. computers) to build a more accurate model requiring less training instances.

The knowledge-based approach is considered complementary to the other approaches [7]. In this approach, knowledge about customers and the application domain are used to reason about what products fit the customer's preferences. The most important advantage is that this approach does not depend (exclusively) on customer's rates, hence avoiding the mentioned difficulty in bootstrapping the system. Knowledge can be expressed as a detailed user model, a model of the selection process or a description of the items that will be suggested. However, the usually complex and error prone process required for extracting the required knowledge and building the needed models (knowledge representation), is seen as a limitation of this approach.

Knowledge-based recommender system can exploit similarity metrics. For example, in the e-commerce portal site recommender.com [7] the system uses knowledge about the customer (the movie's name that the user liked) to search in the database (catalog) for similar movies. The retrieved set is sorted by the similarity to the input movie and the top candidates are recommended to the user. We will further discuss this knowledge-based approach in the next Section.

3 Case-Based Reasoning

Case-based reasoning (CBR) is a problem solving methodology that tries to solve new problems by re-using specific past experiences stored in example cases [12]. A case models a past experience, storing both the problem description and the solution applied in that context. All the cases are stored in the case base. When the system is presented with a new problem to solve, it searches for the most similar case(s) in the case base and reuses an adapted version of the retrieved solution to solve the new problem.

CBR is a cyclic and integrated problem solving process (see Figure 1) that supports learning from experience [1] and has four main steps: retrieve, reuse, adaptation and retain [12]. The adaptation phase is split into two sub-steps: revise and review. In the revise step the system adapts the solution to fit the specific constraint of the new problem. Whereas in the review step the constructed solution is evaluated by applying it to the new problem, understanding where it fails and making the necessary corrections.

In a diagnosis task, for instance, the system acquires the patient symptoms (new problem) and tries to give the final diagnosis based on past patient examples (stored in the case base). Sometime the solution retrieved can be straightforwardly reused in the new problem, but in the majority of the situations the retrieved solution is not directly applicable and must be adapted to the specific

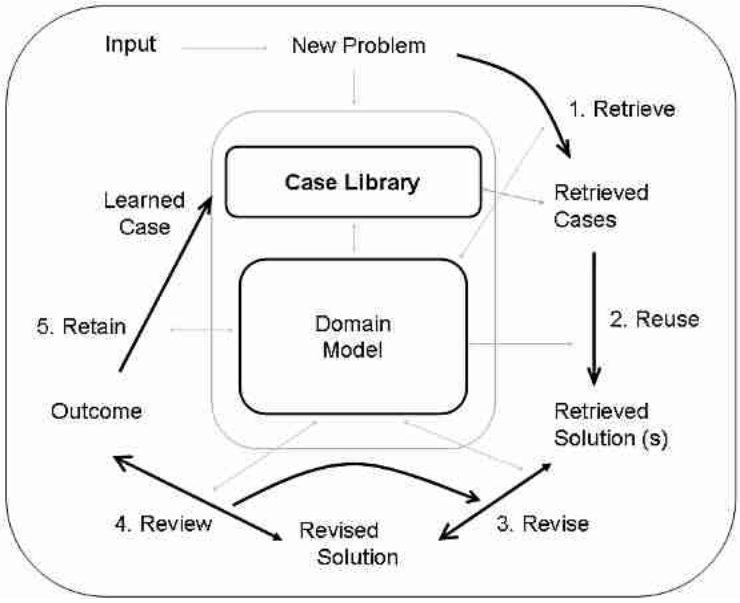

Fig. 1. Case-Based Reasoning problem solving cycle [2]

requirements of the new problem. After this adaptation the system creates a new case and could retain it in the case base (learning).

A fundamental issue in CBR is the case model. This must account for both the problem and solution components. It is necessary to decide which attributes should compose a case and what representation language is better suited to represent the particular knowledge involved in the problem solving process. Hence, the case representation task is concerned with (1) the selection of relevant attributes, (2) the definition of indexes and (3) structuring the knowledge in a specific case implementation. Indexing is related to the creation of additional data structures that can be held in the memory to speed up the search process focussing on the most relevant dimensions. The indexes identify the case attributes that should be used to measure case similarity. Moreover, indexes can speed up the retrieval process by providing fast access to those cases that must be compared with the input case problem. For instance, in a medical diagnosis system, if the system must produce an infection diagnosis then attributes such as profession, gender or age are probably less important than the attributes describing the symptoms.

4 Methodology for the CBR Recommender Systems

In this section we shall illustrate how the generic steps of the CBR problem solving cycle are specialized in a CBR Recommender System, hence providing a unifying description of various systems or techniques. Whereas in the next

Section we shall illustrate some real CBR-RSs and the techniques exploited to generate the recommendations.

In the simplest recommendation process, the user is supposed to be looking for some product to purchase and therefore is asked by the system to provide some product requirements, those that he/she considers as the most important. In reply, the system initiates a search in the case base to identify products that should be recommended, i.e. those that satisfy these requirements. In this process we can identify some basic elements, such as, the input (where the user provides his/her requirements), the products retrieval (where the system searches the products according to user requirements) and the output, where some recommendation is given to the user.

As described in the previous section this flows is very similar to that of a generic CBR system. This starts with a new problem, retrieves similar cases from the case base, shows the retrieved solution to the user or adapts it to better solve the new problem and terminates the process retaining the new case.

We have therefore analyzed four CBR recommender systems and six recommendation techniques and through this analysis we created a condensed model of these recommendation approaches. This model is a general framework for accommodating the description of the specific tasks/functionalities available in the considered systems for product recommendation. Using the framework a number of approaches can be described as specific instantiation of the different steps of the CBR cycle and the evaluation and the comparison of CBR-RSs can be eased.

The CBR recommender systems that we have analyzed are:

- Entree (EN): a recommender system that exploits query tweaking to recommend restaurants to the user.
- DieToRecs (DTR): a travel recommender system that suggests both single travel services (e.g. hotel or an event) and complete travel plans comprising more that one elementary service.
- First Case (CDR): a prototype system that uses the Compromise-Driven Retrieval technique to retrieve and group cases according to the alternative compromises found by the system.
- Expertclerk (EC): a tool for developing dialogue-based recommender systems for e-commerce websites.

The recommendation techniques that we have analyzed, either included in some of the previously mentioned systems or not yet exploited in any prototype, are:

- Interest Confidence Value (ICV): a similarity-based retrieval technique that is used to predict the interest of a user in a product. This technique introduces also a mechanism to progressively forget old not useful cases.
- Single Item Recommendation (SIR): a recommendation technique introduced in the DieToRecs system (DTR) to recommend a single item (product).
- Seeking for Inspiration (SI): this technique, used in DieToRecs, updates travel plans recommendations according to explicit user feedbacks.

– Travel completion (TC): a recommendation technology introduced in DTR to recommend a complete travel a partially defined plan.
– Order-based retrieval (ODR): a retrieval technique based on the application of partial order operators to the case base.
– Comparison-based Retrieval (COB): this technique transforms user's preferences into explicit query modifications.

Figure 2 shows the framework including the classical five steps of the CBR problem solving cycle plus an additional "iterate" step. The iterate step models a peculiar feature of many RSs, i.e., to incrementally update the current set of recommendations acquiring new input from the user, usually in the form of critics or feedbacks. In each stage we list in bold face a general description of the technique or data and then the recommendation techniques or systems that exploit such general technique or data. For instance, in the "Input" box we have "product features" used by the SIR (Single Item Recommendation) technique of the DieToRecs system. All the details and acronyms mentioned in this figure will be explained in the following sections. We provide here a general description of this framework as an introduction to the description of each single technique or system.

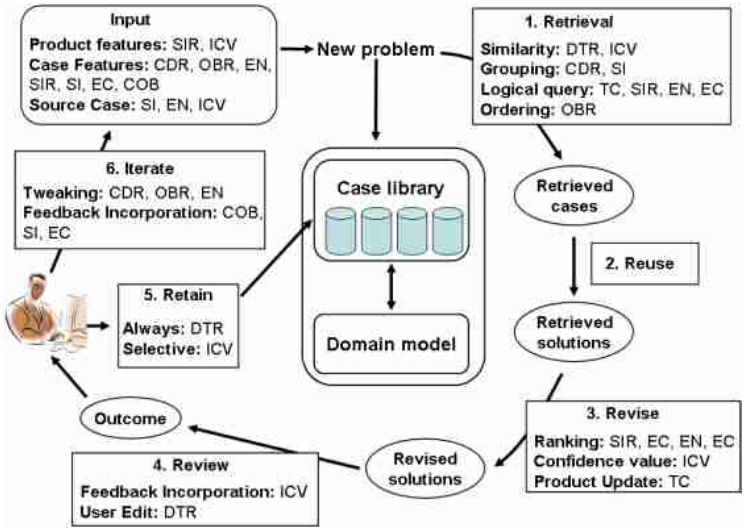

Fig. 2. CBR recommender systems framework

The first stage of many recommendation techniques is the **input** where the system interacts with the user to capture her preferences. According to [24] there are different strategies for interacting with the user. The most popular strategy is dialog-based, where the system offers guidance to the user by asking questions and presenting products alternatives, to help the user to decide. Several CBR

recommender systems ask the user's requirements to have an idea of what the user is looking for. In the Compromise-Driven Retrieval [18], for instance, the user provides the (case) features of a personal computer that he/she is looking for, such as, type, price, processor or speed. Expertclerk [34] asks directly to the user to answer some questions and hence to provide case features as replies to these questions.

Usually when the user searches for a product, three situations can occur [6, 24]:

- the user knows exactly what he/she wants;
- the user has a desire but does not know the name of the product;
- the user does not know precisely what he/she is looking for.

In each of these situations, to recommend suitable alternative products the system requires some kind of knowledge. In CBR-RSs the knowledge is mainly stored in the case base and analyzing the existing CBR-RSs we noticed that the knowledge contained in a case can refer to many characteristics of the problem domain. In fact, in the systems that were considered for this study, a case stores information about: the products recommended (or to be recommended), the user to whom the recommendation was supplied, and contextual information about the recommendation session when the recommendation was provided. Actually, the systems exploits these basic ingredients to define their own specific case model as a mix of these. Hence, for instance, in one case model we may find a description of the recommended products and user who received the recommendation or the user together with his recommended products' evaluation.

To compare different CBR-RSs we propose here an artificial case model that includes case components found in these RSs. In this perspective a case base CB, can be decomposed in four sub-components:

$$CB \subseteq X \times U \times S \times E$$

where X is the product/content model, U is the user model, S is the session model, and E is the evaluation model (more details on these models are provided below). This means that a general case $c = (x, u, s, e) \in CB$ in a generic CBR-RS consists of four (optional) sub-elements x, u, s, e which are instances of the spaces X, U, S, E respectively. Each CBR-RS adopts a particular model for the spaces X, U, S, E. These spaces could be empty, vector, set of document (textual), labelled graphs, etc. Let us now describe each model separately.

- *Content model*(X): the content model describes the product recommended or to be recommended, and usually adopts a feature-based representation of the product (feature vector). In Compromise-Driven Retrieval [18], for instance, a case is modelled (only) by the content component, which is an n-dimensional vector space $X = \prod_{i=1}^{n} X_i$. Each X_i represents the set of possible values for a product attribute. For instance, when the products are computers, an attribute (symbolic) could be the computer type, or the price of the computer (numeric).

- *User model*(U): the user model usually contains personal user information, such as, name, address, age or information about the user past system usage, such as his/her preferred products. Very few CBR-RSs have exploited this component.
- *Session model*(S): the session model is introduced to collect information about the special recommendation session (problem solving loop). In DieToRecs, for instance, a case describe a recommendation session and stores all the user queries and product selected in that session.
- *Evaluation model*(E): the evaluation model describes the outcome of the recommendation, i.e., if the suggestion was appropriate or not. This could be a user a-posteriori evaluation, or, as in [22], the outcome of an evaluation algorithm that guesses the goodness of the recommendation (exploiting the case base of previous recommendations).

Actually, in CBR-RSs, as we noticed above there is a large variability in what a case really models and therefore what components are really implemented. There are systems that use only the content model, i.e, they consider a case as a product, and other systems that focus on the perspective of cases are recommendation sessions. The example systems described in the following sections will illustrate this variability in case structure.

Going back to the problem solving cycle, let us now consider more in detail how cases are managed by different approaches. The first step of the recommendation cycle is the **retrieval** phase. This is typically the main phase and the majority of CBR recommender systems can be described as sophisticated retrieval engines. For example, in Order-Based Retrieval [4] the system uses special operators to retrieve a lattice of cases, or in the Compromise-Driven Retrieval [18] the system retrieves similar cases from the case base but also groups the cases, putting together those cases that offer the same compromise to the user and presents to the user just a representative case for each group.

After the retrieval, in the **reuse** stage the case solution is considered and the system evaluates if it can be reused in the current problem or what part of the case can be reused. In the simplest CBR-RSs, the system reuse the retrieved cases/products showing them to the user. In more advanced solutions, such as in ICV [22] or DTR [28], the retrieved cases are not recommended but used to rank candidate products identified with other approaches, for instance in DTR, with an interactive query management component.

In the next phase **revise** the reused case is adapted to better fit the new problem. The **review** phase in CBR-RSs is implemented by allowing the user to customize the retrieved set of products. For instance in DieToRecs the user can add to the current case other products either using the CBR functionality or by using other system functions (e.g. browsing the product catalogue).

The iterate step is implemented very often in conversational systems. For example, In Entree [7] the system allows the user to tweak the initial query and search for products having marginal differences with those already shown, with respect to some of the product features (e.g. cheaper products). In Comparison-based Retrieval [16] the system asks the user to provide feedback, either positive

or negative, about the retrieved product and automatically updates the user query using this information.

The last step of the CBR recommendation cycle is the **retain** phase (or learning), where the new case is retained in the case base. In DieToRecs, for instance, all the user/system recommendation sessions are stored as a new cases in the case base.

The next subsections describe some representative CBR-RSs, focusing on their peculiar characteristics.

5 CBR Recommendation Techniques and Systems

5.1 Entree - EN

Entree is a restaurant recommender system that provides recommendations by finding restaurants in a new city similar to restaurants the user knows and likes or those matching some user goals (case features)[7].

The user starts the interaction with Entree, as showed in Figure 3, either by: mentioning a known restaurant in some place (source case) and asking for a similar one in a give city; or selecting a set of high-level features (case features) and searching for a restaurant that matches those features. With this input information, the system first selects from the database, which physically stores the cases, the set of all restaurants that satisfy the largest number of logical constraints generated by considering the input features type and value. The system, if necessary, implicitly relaxes the lowest important constraints until some restaurants could be retrieved.

Then Entree sorts the retrieved cases using a similarity metric. This similarity metric assumes that the user goals, corresponding to the input features (or the features of the source case), could be sorted to reflect the importance of such goals from the user point of view. Hence the global similarity metric sorts the products first with respect the most important goal and then iteratively with respect to the remaining goals (multi-level sort).

If the recommended restaurants satisfies the user then the interaction finishes. But if the user is not satisfied, because of the values of some features of the proposed restaurant, then he can criticize them. This failure situation is determined by the fact that in the similarity retrieval it is possible that the recommended restaurant does not match 100% the good example provided by the user as input. If for instance, the price is too high and the user is looking for something cheaper, then he/she can "tweak" the original request and provide a new input explicitly mentioning that the result must have a cheaper price. This starts a new recommendation cycle and the criticized features is considered the most important user goal.

In Entree the reuse step is trivially implemented, i.e. the products retrieved are passed to the revise step for ranking. The review and retain steps are not implemented.

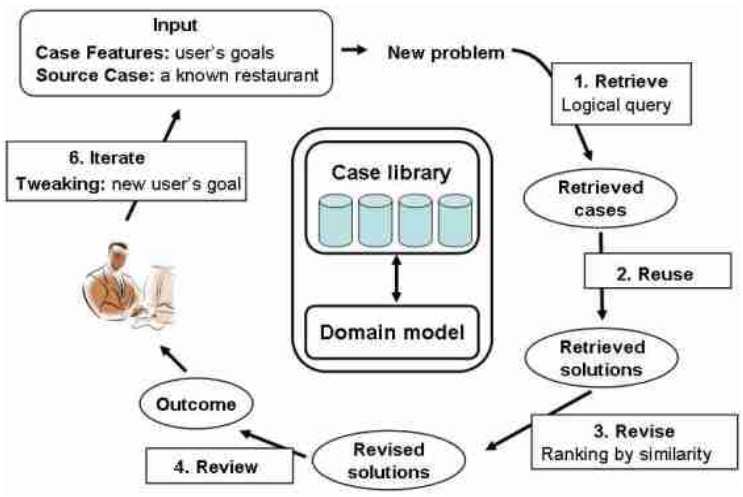

Fig. 3. Entree recommender system

5.2 Interest Confidence Value - ICV

Montaner et al. (in [22]) assume that the user's interest in a new product is similar to the user's interest in similar past products. This means that when a new product comes up, either selected by a user's query or by another method, then the recommender system predicts the user's interest in this product based on the interest attributes/evaluation of similar products.

A case is modelled by objective attributes describing the product (content model) and subjective attributes describing implicit or explicit interests of the user in this product (evaluation model). Formally, the case c is defined as $c \in X \times E$.

The content model (X), in this system, is represented by a vector space X $= \prod_{i=1}^{n} X_i$ where, for instance, x_1 is the restaurant code (*integer*); x_2 is the restaurant name (*string*); x_3 is the restaurant address (*string*); x_4 is the cuisine type (*string*); x_5 is the approximate price (*real*); x_6 is the capacity (*integer*) and x_7 is the air-conditioning (*boolean*).

The evaluation model (E) is also an heterogeneous vector space $E = \prod_{i=1}^{m} E_i$ where some $e_i \in E_i$ describe explicit interests attributes like a general evaluation of the product provided by the user or a quality price ratio. Some other e_i describe implicit evaluation attributes like the rate of time spent by the user to read product information. There is also a special attribute, called drift attribute, that measures how recently the user expressed his interest in the product. When this drift attribute becomes very small the system tends to reduce the importance of the information contained in the case associated to that product, and eventually can discard the case.

As Figure 4 shows, the recommendation process starts with the user providing some preferences about a new restaurant he/she is looking for and with a

new restaurant r (source case). The goal of the system is to evaluate if this new restaurant r could be interesting for the user. With these preferences and input product the system searches similar restaurants in the case base (**retrieval phase**) to find restaurants that could be used to compute the interest prediction of the new restaurant r.

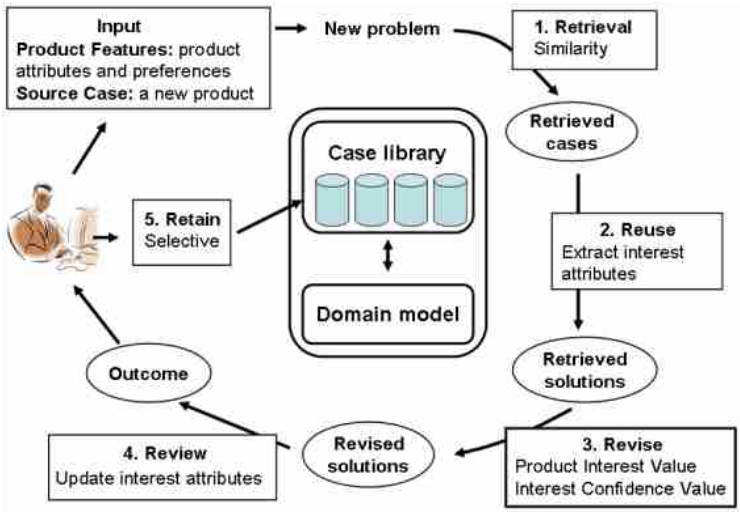

Fig. 4. The Interest Confidence Value technique

In the **reuse** phase the system basically extract from the retrieved cases (c_i, $i = 1, \ldots, k$) the interest attributes, or in our terminology the evaluation model. In the **revise** phase the system assumes that the user's interest in the new restaurant r is similar to his/her interest in the retrieved restaurants, hence, for each retrieved case c_i it extracts the interest attributes (evaluation model) and compute a global interest value $V(i)$ for each retrieved case. $V(i)$ is a weighted average sum of the interest attributes multiplied by the drift attribute [22].

Then a global interest confidence value $I(r)$ for the product r is computed as a weighted average of the interest values of the retrieved cases:

$$I(r) = \frac{\sum_{i=1}^{k} V(i) Sim(r, c_i)}{\sum_{i=1}^{k} Sim(r, c_i)}$$

If the interest confidence value of the new restaurant is greater than a certain value (a confidence threshold), then the new restaurant is recommended to the user. Otherwise, the CBR cycle terminates with no recommendation and the system just provides a negative advice to the user about the queried restaurant.

The **review** phase is implemented by asking the user for the correct evaluation of the restaurant and after that a new case (the product and the evaluation)

is **retained** in the case base. Also implicit evaluation indicators are retained as derived from the analysis of the user/system interaction.

We stress that in this approach the recommended product is not retrieved from the case base, as we saw before in Entree, but the retrieved cases are used to estimate the user interest in a generic new restaurant. The new restaurant could be generated in many ways, including a search in the case base.

5.3 DieToRecs - DTR

DieToRecs is a case-based travel planning recommender system, that helps the user to plan a leisure travel in a selected destination [11]. Three different recommendation techniques were implemented in DieToRecs: the single item recommendation (SIR), the travel completion (TC) and seeking for inspiration (SI).

In DieToRecs, a case represents a user interaction with the system and it is built incrementally during the recommendation session [28]. A case comprises the following main components:

- *Collaborative Features (clf)* are features that describe general user's and travel characteristics, wishes, constraints or goals (e.g. desire to relax or to practice sports). They capture preferences relevant to the user's decision-making process, which cannot be directly mapped into product attributes stored in the electronic catalog. These features are used to measure case (session) similarity. A knowledge of the domain and the decision process is essential to select the right collaborative features [26]. The collaborative features belong to the user and session models.
- *Content Queries (cnq)* are queries posed over the catalogs of products. Content queries are built by constraining (content) features that describe products listed in the catalogs. Products may belong to different types (e.g. an accommodation or an event). The content queries belong to the session model.
- *Cart* contains the set of products chosen by the user during the recommendation session represented by the case. A cart represents a meaningful (from the user's point of view) bundling of different products. For instance, a travel cart may contain some destinations, some accommodations, and some additional attractions. The cart component belongs to the content model.
- *Rate* is a collection of rates given by the user to the products contained in the cart. It represents the user evaluation of the products and therefore belongs to the evaluation model.

In the single item recommendation technique (SIR), the user interacts with the recommender system by querying recommendations about a product type (e.g., a destination). The whole process is shown in Figure 5. In SIR the systems asks the user both some general preferences (the clf) that are used to generate a case (current case or source case) and some specific product preferences that are used to query (cnq) the product catalogue. The system uses the content queries to search in the catalog for products that (logically) match these preferences and computes a result set.

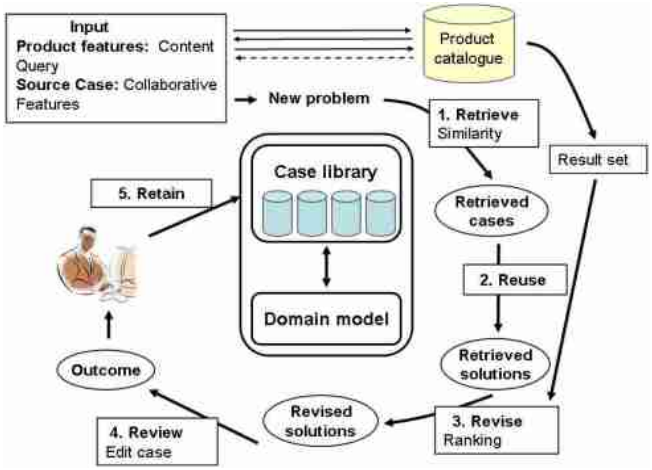

Fig. 5. DieToRecs - Single Item Recommendation technique

The system supports an interaction flow that allows the user to eventually refine the initial content query. That is depicted in Figure 5 as a set of parallel arrows from the product features to the product catalogue. In fact, If too many products matches the input query, then a tightening function suggests to the user some additional features he may use to further constrain the search [21]. Conversely if no result can be found then the system explains to the user the cause of the failure, i.e., it lists those constraints that if relaxed would allow the query to return some results (relax function) [20]. When the number of items retrieved is satisfactory then the system proceeds with the **revise** phase to rank the result set.

In parallel, the collaborative features are used to retrieve the ten most similar cases, and in the reuse phase the products contained in these case are extracted. In the revise phase the results set is ranked with a double similarity process where the product contained in the result set that are more similar to products contained in similar cases obtain a higher rank [28].

Finally the user can edit the current case adding new products, using one of the available recommendation techniques (review). The case is always stored in the case base and it is updated every time the user changes some of its components (for instance adding a new travel product to the cart).

The second recommendation technique introduced in DieToRecs is called Travel Completion (TC). Here the system recommends additional travel products or services to complete the current travel plan of the user. In TC the cycle starts with the current case as source case. This is used by the system to retrieve from the case base similar cases, i.e., travel plans built by other travellers that match the collaborative features of the source case (see Figure 6). Some of the collaborative features are used to generate some logical constraints, hence the retrieval combines a similarity-based one piped after a logical filter.

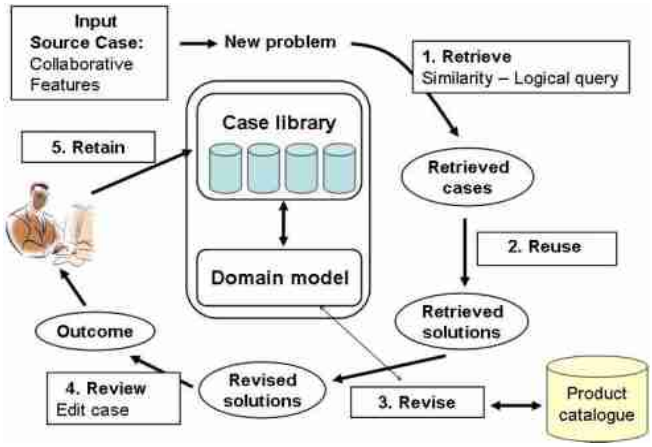

Fig. 6. DieToRecs - Travel Completion technique

Before recommending to the user the products contained in the retrieved cases (solutions), the system in the **revise** stage updates, or replace, the travel products contained in the cases exploiting up-to-date information taken from the product catalogs. This adaptation stage is constrained-based and the constraints are stored as structural properties of the travels. For instance if the destination is x, then the system cannot recommend an accommodation in y where the distance of x and y is larger than a given threshold.

In the **review** phase the system allows the user to reconfigure the recommended travel plan. The system allows the user to replace, add or remove items in the recommended travel plan. If the user accepts the outcome (the final version of the recommendation showed to the user), then the system **retains** this new case in the case base.

In the third recommendation technique introduced in the DieToRecs system, that is Seeking for Inspiration (SI) [29], the user is prompted with complete travel recommendations to choose. SI proceeds as a loop, which is initiated with a source case, and terminated when the user selects one of the recommended case. At each loop six cases are shown. The initial probe is randomly selected by the system if the user has not created yet any case (e.g. with any of the other recommendation techniques). These six displayed cases are computed by, first **retrieving** k most similar cases to the source case, and then selecting with a greedy algorithm the six more diverse among the k retrieved. When these six cases are shown the user can provide some feedback, by checking a "I like this" option. The system then iterates the process using the liked case as the current source case. As in the other techniques when the recommendation process terminates the newly generated case is stored in the case base.

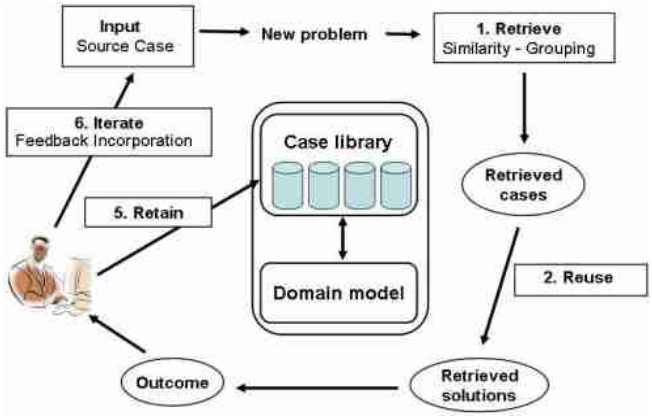

Fig. 7. DieToRecs - Seeking for Inspiration technique

5.4 Order-Based Retrieval - OBR

The Order-Based Retrieval (OBR) technique integrates different type of sorting criteria [4]. Using ORB the authors have developed a prototypes that helps a user to find a place to rent in London. A case is modelled using only the content model, $c = (x)$. The content model X is a vector space $X = \prod_{i=1}^{n} X_i$. Typical attributes are: the price of the apartment (a real number); the number of bedrooms (integer); the number of bathrooms (integer); the location (string); the type of the property (string) and whether it is furnished or not (boolean). The features were classified as ordered or unordered values.

The recommendation process starts when the user provides some preferences, as ideal values, or as maximum or minimum values of the searched case. In the **retrieval** phase, the system converts the user input into order relationships on the attributes. Then all these order relationships are combined in a pre-order to produce a lattice of products. The operators used in this task were defined in [5].

Figure 8 illustrate how ORB can be described in the proposed framework. In the **reuse** phase the system extracts from the lattice the maximal products. The last implemented step is the **iterate**, where the system waits for additional preference constraints from the user to refine the lattice structure. The user can provide modifications to the query, and these are encoded as filters and finally converted into orders using Filter-Ordering (FO) operators. The system will recommend products that satisfy the filter but will not eliminate products that do not satisfy the filter. There are no **revise**, **review** and **retain** tasks implemented in this technique.

The advantage of OBR, compared to pure similarity-based retrieval, is that it allows the user to provide some soft constraints. Suppose the user wants to define the following query: "Rent a property in Clapham with 2 bedrooms but the rent cannot be more than 400". The system takes this upper bound condition ("not

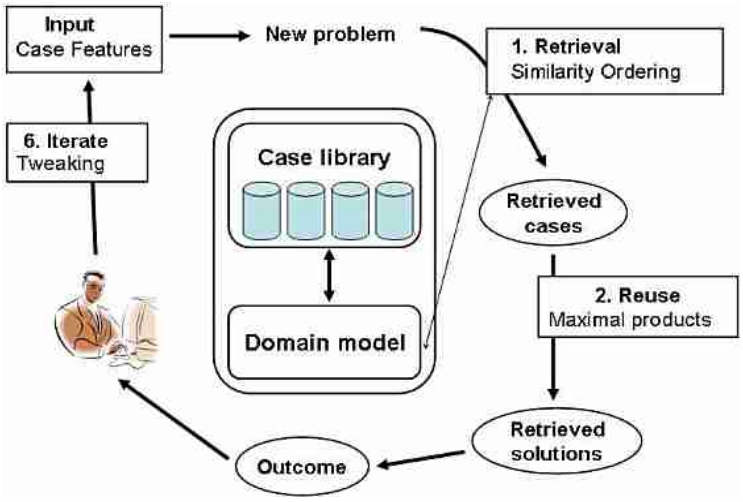

Fig. 8. The Order-Based Retrieval technique

more than 400") and build an unary predicate. But instead of filtering away products it builds an ordering from the products using the Filtering-Ordering (FO) operator.

$$<_{FO(\lambda x[price(x)\leq 400])}$$

The values that satisfy the predicate are higher in the ordering than ones that do not.

5.5 First Case - CDR

First Case is a CBR-RS that uses the Compromise-Driven Retrieval (CDR) technique to recommend computers to the user. First Case models a case exploiting only the content component [18]. The content model X is a vector space $X = \prod_{i=1}^{n} X_i$ where n is the number of attributes. Typical attributes are: the computer type (string); the price (real); the manufacturer (string); the processor (string); the speed (integer); the monitor size (integer); the memory (integer) and the hard disk size (integer). In CDR, if a given case c_1 is more similar to the target query than another case c_2, and differs from the target query in a subset of the attributes in which c_2 differs from the target query, then c_1 is more acceptable than c_2.

As showed in Figure 9, the CBR recommendation cycle starts with the user providing his/her requirements in a query. The user can specify how many requirements he/she wants. For example, let's consider a query q, defined by the following conditions: Intel Pentium; speed = 900 (or higher); 17" monitor size; desktop or tower and price not higher than 1400. Let us further imagine that the hard disk and memory are not important for the user.

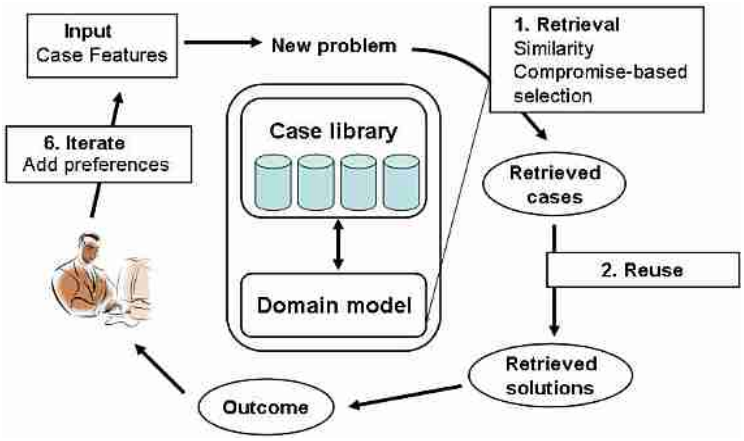

Fig. 9. First Case recommender system

In the CDR **retrieval** algorithm the system sorts all the cases in the case-base according to the similarity to a given query. The combination of attributes in which the case differs from the user query is important and not just the number of attributes that differ.

For any case c and query q, the author defines the set of compromised attributes as:

$$compromises(c, q) = \{a \in A_q : \pi_a(c) \text{ fails to satisfy the user preference}\}$$

where A_q is the set of attributes constrained in the query, a is an attribute, and $\pi_a(c)$ is the value of attribute a in c.

In a second step the algorithm groups together the cases making the same compromise (do not match a user preferred attribute value) and builds a reference set with just one case for each compromise group. In the **reuse phase** the reference set is recommended to the user without modifications and the user has immediate access to it.

The user can also refine (**iterate**) the original query, accepting one compromise, and adding some preference on a different attribute (not that already specified). The system will further decompose the set of cases corresponding to the selected compromise. The **revise**, **review** and **retain** phases are not implemented in CDR.

In this approach similarity and compromise play complementary roles, increasing the probability that one of the retrieved cases will be acceptable to the user. CDR shows alternative compromises and groups cases according to the compromise that is done. In this way it helps the user to immediately grasp the alternatives available and therefore increase the diversity of recommendations. Other researches had proposed different ways to retrieve a diverse set of cases [35]. Mongouie et al [23] have also studied the concept of generalized cases and

presented a method to build a retrieval set of cases that are enough different and are also representative for a set of similar cases.

5.6 Comparison-based retrieval - COB

According to McGinty and Smyth [14], a key feature that differentiates recommender systems from more conventional information retrieval systems, such as search engines, is their conversational character.

As we saw before, the majority of the recommendations techniques support a cycle that initiates with a first query/problem of the user. Then, each feedback provided by the user during the recommendation cycle is used to update the user's query. The goal is to refine the query so that user needs are better captured and a better recommendation can be produced.

Some researches have focused on different strategies for capturing the user feedback [15, 33]. We have:

- *Value-elicitation*: where the user is asked to provide a specific value for a specific feature of the recommended product;
- *Tweaking*: where the user is asked to provide a directional preference for a particular feature;
- *Rating-based*: where the user is asked to rate the recommended cases according to his/her preferences;
- *Preference-based*: where the user is asked to select one of the current recommendations, that is closest to his/her requirements.

These strategies have been classified (in [33]) as: *navigation by asking*, when the system can ask to the user to specify individual feature values as search criteria or *navigation by proposing* when the system invites the user to rate recommendation as relevant or not relevant.

The comparison based retrieval technique (COB) is a navigation by proposing [34] recommender system that exploits preference feedbacks by transforming the user's preference into explicit query adaptations [15]. McGinty and Smyth have used COB to build a prototype aimed at supporting the user in selecting a computer. In this prototype the case model includes only the content model.

The recommendation process starts with the user providing his/her requirements as attribute-value pair of the preferred case. In the **retrieval** phase the system retrieves the cases with a traditional similarity-based process. The retrieved cases are shown to the user in the **review** phase.

Figure 10 shows the recommendation cycle in COB. In the **iterate** phase the user selects a preference case as feedback (positive if the user likes the case recommended or negative if he/she does not like it). This feedback are interpreted as a user evaluation of the difference between the selected product and the products not chosen. This information is used to learn from the user's feedback and update the current query. Some update strategies are used in the review

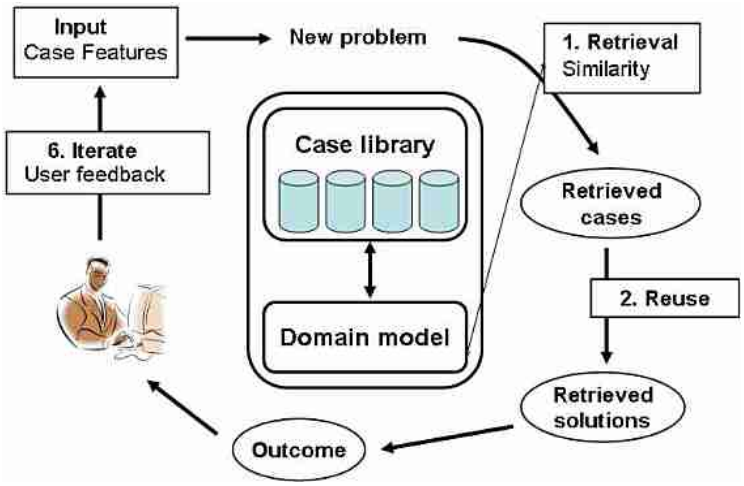

Fig. 10. Comparison-Based Retrieval

phase, such as, *More-Like-This* that takes each feature of the preferred case as a new query feature, or *Partial-More-Like-This* that transfers a feature value from the preference case if none of the rejected cases have the same feature value. The process terminates when the user is presented with an acceptable item or when he/she gives up. In this approach there are no **revise**, **review** and **retain** phases.

In another paper of the same authors the COB approach is improved using the Adaptation Select technique, that adapts the way the new products are selected, making the preference-based feedback more efficient [16].

5.7 ExpertClerk - EC

Expertclerk is a general recommendation methodology aimed at implementing virtual salesclerk systems as front-end of an e-commerce website [34]. The system implements a question selection method (decision tree with information gain). Using navigation-by-asking, the system starts the recommendation session by asking the user some questions. The questions are nodes in a decision tree. A question node subdivides the set of answer nodes and each one of these represents a different answer to the question posed by the question node. The system concatenates all the answer nodes chosen by the user and then builds the SQL retrieval condition expression.

In EC the user can answer the question by choosing an answer node or ignore the question (Figure 11). The system concatenates all the answer nodes chosen by the user and then constitutes the SQL retrieval condition expression. This query is applied to the case base to **retrieve** the set of cases that best match the user query.

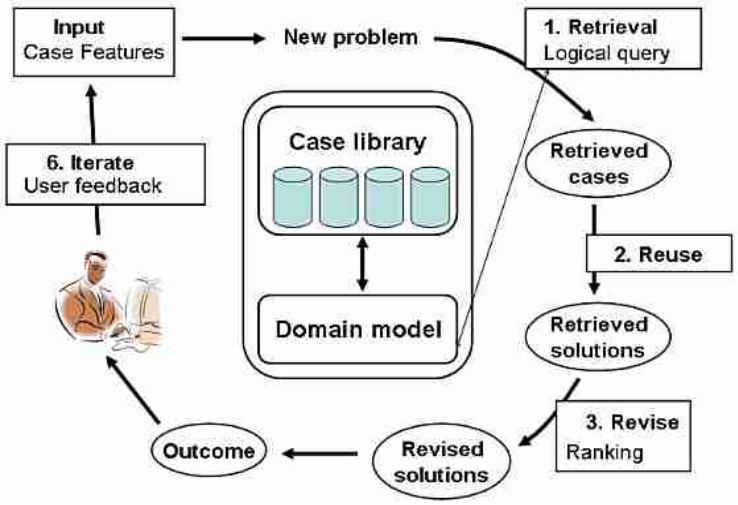

Fig. 11. ExpertClerk

The system ranks the products in the **revise** phase, recommends three sample products to the user, and explains their characteristics (positive and negative).

In the **review** phase, the system switches to the navigation-by-proposing conversation mode and allows the user to refine the query. After refinement, the system applies the new query to the case base and retrieves new cases (iterate). These cases are ranked and shown to the user. This cycle continues until the user finds a good product. In this approach the **retain** phases is not implemented.

6 Comparison

In this Section we present a couple of summary tables to quick compare the CBR-RSs that we have illustrated. From the analysis of these systems or techniques we have detected some common patterns. Tables 1 and 2 show a cross-dimension analysis of the systems, taking into account the features described in the proposed CBR framework .

In Table 1 we summarize the analysis of the case model adopted in the various approaches (techniques or systems). In the majority of them the case includes essentially the content model. In other words a case is considered equal to the product to be recommended. This has the implication that no real learning process is supported by these systems. This is severe limitation which is overcome only by two systems: ICV, where they suggested a case representing the user interest and DieToRecs where the authors proposed a case representing a user interaction with the system. Hence in our opinion CBR-RSs research must devote much more attention on the case model and particularly in finding a more comprehensive way to include in the case information and knowledge about the user, the recommendation process and especially the outcome of such a process.

Table 1. Cross-dimensional analysis of the case model

Approach	**Case model**
Entree EN	Content
Interest Confidence Value-ICV	Content, Evaluation
DieToRecs DTR	All
Order-Based Retrieval OBR	Content
Compromise-Driven Retrieval CDR	Content
Comparison-Based Retrieval COB	Content
ExpertClerk EC	Content

Table 2. Cross-dimensional analysis of the selected approaches (Retrieval, Reuse, Revise, Review and Retain)

Technique	**Retrieval**	**Reuse**	**Revise**	**Review**	**Retain**	**Iterate**
EN	Sim, Logic	All	Ranking	none	none	Tweaking
ICV	Sim	Eval model	ICV computation	feedback	selective	none
SIR	Sim	Content	Ranking	User edit	All	none
TC	Sim, Logic	Content	Constraints	User edit	All	none
SI	Sim, Grouping	All	none	none	All	Feedback
OBR	Sim, Ordering	All	none	none	none	Tweaking
CDR	Sim, Grouping	All	none	none	none	Tweaking
COB	Sim	All	none	none	none	Feedback
EC	Sim	All	none	none	none	Feedback

Referring to the application of the CBR cycle to recommendation (see Table 2), the majority of the CBR-RSs stress the importance of the retrieval phase. Some systems perform retrieval in two steps. First, cases are retrieved by similarity, then the cases are grouped or filtered. The use of pure similarity does not seem to be enough to retrieve a set of cases that satisfy the user. This seems to be true especially in those application domains that require a complex case structure (e.g. travel plans). Hence in these domains similarity is piped after a first retrieval performed with a more efficient logic based filtering (e.g. in SQL on a data base implementation of the case base).

The default reuse phase is used in the majority of the CBR-RSs, i.e, all the retrieved cases are recommended to the user. ICV and SIR have implemented the reuse step in a different way. In SIR, for instance, the system can retrieve just a component of the case (e.g. the destination of a travel and discard the selected accommodation). The same systems that implemented non-trivial reuse approaches, have also implemented both the revise phase, where the cases are adapted, and the retain phase, where the new case (adapted case) is stored.

Not all the CBR-RSs analyzed implement the review phase, allowing the user to modify/configure the proposed solution (recommendation), or implementing an automatic solution to product reconfiguration. Conversely, many systems cycle the recommendation process, either letting the user to tweak the original

query or incorporating (system-driven) explicit or implicit feedbacks collected during the interaction.

7 Conclusions

Looking for products on the Internet is not an easy task. There is a huge quantity of information and on-line there is no human advisor that can help customers to identify what products better fit their preferences. The CBR methodology has been used extensively and successfully to build intelligent applications helping users to cope with these problems.

In this paper we have presented a partial review of the CBR recommender systems literature. We have found that it is often unclear how and why the proposed recommendation methodology can be defined as case-based. In fact, the classical CBR problem solving loop, most of the time, is implemented only partially and sometime is not clear whether a CBR stage (retrieve, reuse, revise, review, retain) is implemented or not. For this reason, we have proposed the unifying framework illustrated in this paper to make possible a coherent description of different CBR-RSs. This framework helps to describe to what extent a recommender system exploits the classical CBR cycle.

We believe, that with such an initial common view it will be easier to understand what the research projects in the area have already delivered, how the existing CBR-RSs behave and which are the topics and the features that could be improved in future systems. We plan to extend this work analyzing more CBR recommendation techniques and to formalize the case model representation using a case representation language such as CBML [10, 9] or CASUEL [13].

References

1. A. Aamodt and E. Plaza. Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994.
2. D. W. Aha. The omnipresence of case-based reasoning in science and application. *Knowledge-Based Systems*, 11(5-6):261–273, 1998.
3. D. Billsus and M. Pazzani. A hybrid user model for news story classification. In *Proceedings of the Seventh International Conference on User Modeling, UM '99*, Banff, Canada, 1999.
4. D. Bridge and A. Ferguson. Diverse product recommendations using an expressive language for case retrieval. In S. Craw and A. Preece, editors, *Advances in Case-Based Reasoning, Proceedings of the 6th European Conference on Case Based Reasoning, ECCBR 2002*, pages 43–57, Aberdeen, Scotland, 4 - 7 September 2002. Springer Verlag.
5. D. Bridge and A. Ferguson. An expressive query language for product recommender systems. *Artificial Intelligence Review*, 18:269–307, 2002.
6. M. L. Brigitte Bartsch-Sporl and A. Hbner. Case-based reasoning - survey and future directions. In *XPS 99*, pages 67–89. Springer Verlag, 1999.
7. R. Burke. Knowledge-based recommender systems. In J. E. Daily, A. Kent, and H. Lancour, editors, *Encyclopedia of Library and Information Science*, volume 69. Marcel Dekker, 2000.

8. R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.
9. L. Coyle, D. Doyle, and P. Cunningham. Representing similarity for cbr in xml. In P.Funk and P. Calero, editors, *Advances in Case-Based Reasoning, Proceedings of the 7th European Conference on Case Based Reasoning, ECCBR 2004*, pages 119–127, Madrid, Spain, 2004. Springer Verlag.
10. L. Coyle, C. Hayes, and P. Cunningham. Representing cases for cbr in xml. In *Proceedings of 7th UKCBR Workshop*, Peterhouse, Cambridge, UK, 2002.
11. D. R. Fesenmaier, F. Ricci, E. Schaumlechner, K. Wöber, and C. Zanella. DIETORECS: Travel advisory for multiple decision styles. In A. J. Frew, M. Hitz, and P. O'Connors, editors, *Information and Communication Technologies in Tourism 2003*, pages 232–241. Springer, 2003.
12. J. Kolodner. *Case-Based Reasoning*. Morgan Kaufmann Publishers, San Mateo, CA, 1993.
13. M. Manago, R. Bergmann, N. Conruyt, R. Traphoner, J. Pasley, J. L. Renard, F. Maurer, S. Wess, K.-D. Althof, and S. Dumont. *CASUEL:A Common Case Representation Language*. Esprit Project 6322, Deliverable D1.
14. L. McGinty and B. Smyth. Comparison-based recommendation. In S. Craw and A. Preece, editors, *Advances in Case-Based Reasoning, Proceedings of the 6th European Conference on Case Based Reasoning, ECCBR 2002*, pages 575–589, Aberdeen, Scotland, September 2002. Springer Verlag.
15. L. McGinty and B. Smyth. Deep dialogue vs casual conversation in recommender systems. In F. Ricci and B. Smyth, editors, *Recommendation and Personalization in eCommerce, Proceedings of the AH'2002 Workshop*, pages 80–89, Malaga, Spain, May, 28th 2002. University of Malaga Press.
16. L. McGinty and B. Smyth. On the role of diversity in conversational recommender systems. In A. Aamodt, D. Bridge, and K. Ashley, editors, *ICCBR 2003, the 5th International Conference on Case-Based Reasoning*, pages 276–290, Trondheim, Norway, June 23-26 2003.
17. D. McSherry. Diversity-conscious retrieval. In S. Craw and A. Preece, editors, *Advances in Case-Based Reasoning, Proceedings of the 6th European Conference on Case Based Reasoning, ECCBR 2002*, pages 219–233, Aberdeen, Scotland, 4 - 7 September 2002. Springer Verlag.
18. D. McSherry. Similarity and compromise. In A. Aamodt, D. Bridge, and K. Ashley, editors, *ICCBR 2003, the 5th International Conference on Case-Based Reasoning*, pages 291–305, Trondheim, Norway, June 23-26 2003.
19. B. Miller, I. Albert, S. Lam, J. Konstan, and J. Riedl. Movielens unplugged: Experiences with an occasionally connected recommender system. In *Proceedings of ACM 2003 International Conference on Intelligent User Interfaces (IUI'03)*. ACM Press, 2003.
20. N. Mirzadeh, F. Ricci, and M. Bansal. Supporting user query relaxation in a recommender system. In *E-Commerce and Web Technologies, Proceedings of the 5th International Conference, EC-Web 2004*, LNCS 3182, pages 31–40, Zaragoza, Spain, August/Semptember 2004. Springer.
21. N. Mirzadeh, F. Ricci, and M. Bansal. Feature selection methods for conversational recommender systems. In *Proceedings of the IEEE International Conference on e-Technology, e-Commerce and e-Services*, Hong Kong, 29 March - 1 April 2005. IEEE Press.
22. M. Montaner, B. López, and J. L. de la Rosa. Improving case representation and case base maintenance in recommender systems. In S. Craw and A. Preece, editors,

Advances in Case-Based Reasoning, Proceedings of the 6th European Conference on Case Based Reasoning, ECCBR 2002, pages 234–248, Aberdeen, Scotland, 4 - 7 September 2002. Springer Verlag.
23. B. Mougouie, M. M. Richter, and R. Bergmann. Diversity-conscious retrieval from generalized cases: a branch and bound algorithm. In A. Aamodt, D. Bridge, and K. Ashley, editors, *ICCBR 2003, the 5th International Conference on Case-Based Reasoning*, pages 319–331, Trondheim, Norway, June 23-26 2003.
24. S. S. Ralph Bergmann and A. Stahl. Intelligent customer support for product selection with case-based reasoning. In P. S. S. Javier Segovia and M. Niedzwiedzinski, editors, *E-Commerce and Intelligent Methods*, pages 322–341. Physica-Verlag, 2002.
25. P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.
26. F. Ricci, B. Arslan, N. Mirzadeh, and A. Venturini. ITR: a case-based travel advisory system. In S. Craw and A. Preece, editors, *6th European Conference on Case Based Reasoning, ECCBR 2002*, pages 613–627, Aberdeen, Scotland, 4 - 7 September 2002. Springer Verlag.
27. F. Ricci and B. Smyth, editors. *Recommendation and Personalization in eCommerce, Proceedings of the AH'2002 Workshop*, Malaga, Spain, May, 28th 2002. Universidad de Malaga.
28. F. Ricci, A. Venturini, D. Cavada, N. Mirzadeh, D. Blaas, and M. Nones. Product recommendation with interactive query management and twofold similarity. In A. Aamodt, D. Bridge, and K. Ashley, editors, *ICCBR 2003, the 5th International Conference on Case-Based Reasoning*, pages 479–493, Trondheim, Norway, June 23-26 2003.
29. F. Ricci, K. Woeber, and A. Zins. Recommendations by collaborative browsing. In *Information and Communication Technologies in Tourism 2005*, pages 172–182. Springer Verlag, Wien - New York, 2005.
30. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of WWW10 Conference*, pages 285–295, Hong Kong, May 1-5 2001. ACM.
31. B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *ACM Conference on Electronic Commerce*, pages 158–167, 2000.
32. J. B. Schafer, J. A. Konstan, and J. Riedl. E-commerce recommendation applications. *Data Mining and Knowledge Discovery*, 5(1/2):115–153, 2001.
33. H. Shimazu. ExpertClerk: Navigating shoppers buying process with the combination of asking and proposing. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001*, pages 1443–1448, Seattle, Washington, USA, August 4-10 2001. Morgan Kaufmann.
34. H. Shimazu. Expertclerk: A conversational case-based reasoning tool for developing salesclerk agents in e-commerce webshops. *Artificial Intelligence Review*, 18:223–244, 2002.
35. B. Smyth and P. McClave. Similarity vs diversity. In *Proceedings of the 4th International Conference on Case-Based Reasoning*, Springer-Verlag, 2001.
36. I. Watson. *Applying Case-Based Reasoning: Techniques for Enterprise Systems*. Morgan Kaufmann, 1997.
37. I. H. Witten and E. Frank. *Data Mining*. Morgan Kaufmann Publisher, 2000.