# Completeness Criteria for Retrieval in Recommender Systems

David McSherry

School of Computing and Information Engineering
University of Ulster, Coleraine BT52 1SA, Northern Ireland
dmg.mcsherry@ulster.ac.uk

**Abstract.** Often in practice, a recommender system query may include constraints that must be satisfied. Ensuring the retrieval of a product that satisfies any hard constraints in a given query, if such a product exists, is one benefit of a retrieval criterion we refer to as *completeness*. Other benefits include the ease with which the *non-existence* of an acceptable product can often be recognized from the results for a given query, and the ability to justify the exclusion of any product from the retrieval set on the basis that one of the retrieved products satisfies at least the same constraints. We show that in contrast to most retrieval strategies, compromise driven retrieval (CDR) is complete. Another important benefit of CDR is its ability to ensure the retrieval of the *most similar* product, if any, which satisfies all the hard constraints in a given query, a criterion we refer to as *optimal* completeness.

## 1 Introduction

In case-based reasoning (CBR) approaches to product recommendation, descriptions of available products are stored in a product case base and retrieved in response to user queries. The standard CBR approach to retrieval in recommender systems is *k* nearest neighbor (*k*-NN) retrieval. In contrast to traditional database retrieval, *k*-NN does not insist on exact matching, thus having the advantage that the retrieval set (i.e., the *k* most similar cases) is never empty [1].

Regardless of the strategy on which the retrieval of recommended cases is based, a query can also be seen as a set of *constraints*. For example, the preferred attribute values in a *k*-NN query are *equality* constraints that may or may not be satisfied by a given case. One approach to retrieval in CBR that takes account of the constraints satisfied by a given case is compromise driven retrieval (CDR) [2-3]. For example, no case that is less similar than another case which satisfies the same constraints is included in the CDR retrieval set.

While no account is taken of satisfied constraints in *k*-NN, cases which satisfy more constraints may also tend to be more similar. However, there is often a conflict between the goals of retrieval strategies like *k*-NN which reward cases on the basis of overall similarity and those which take account of the constraints satisfied by a given case. As we show in Section 2, for example, it is possible for a case that satisfies a proper subset of the constraints satisfied by another case to be more similar. Also, an

T.R. Roth-Berghofer et al. (Eds.): ECCBR 2006, LNAI 4106, pp. 9 – 29, 2006.

issue that *k*-NN only partly addresses by offering the user a choice of *k* alternatives is that often in practice a given query may include constraints that must be satisfied.

In a holiday recommender, for example, a user seeking a skiing holiday for two persons in December may be unable to compromise on the number of persons and unwilling to compromise on holiday type. If the *k* most similar cases do not include a skiing holiday for two persons then the system has failed to recommend an acceptable case. It might be considered that even with $k = 3$ there is a good chance that an acceptable case, if one exists, will be retrieved. However, *k*-NN is known to be limited in its coverage of cases that may be acceptable to the user, or *compromises* that the user may be prepared to consider (e.g., the timing of the holiday) [2-5]. A related problem is that the most similar cases also tend to be very similar to each other, with the result that the user may be offered a limited choice [6-8].

Ensuring the retrieval of a case that satisfies any hard constraints, if such a case exists, is the weaker of two *completeness* criteria for retrieval in recommender systems that we present in this paper. Of course, it is a simple matter to ensure the retrieval of a case that satisfies any *known* hard constraints in a given query if such a case exists. However, requiring users to identify hard constraints in advance may not be a realistic solution. Often in practice, a user may not have a clear idea of what she is looking for when constructing her query, and may begin to consider hard constraints only when faced with the need to compromise. Instead we assume that any hard constraints in a given query are *unknown* to the recommender system and therefore that no distinction is made between hard and soft constraints in the retrieval process.

In the formal definition of completeness that we now present, we refer to a case that satisfies all the hard constraints in a given query as a *feasible* case for the query. Whether such a case is acceptable to the user may of course depend on the extent to which it satisfies any soft constraints in her query. However, the non-existence of a feasible case in the case base implies the non-existence of an acceptable case. Equally, the non-existence of a feasible case in the retrieval set means that none of the recommended cases are acceptable.

**Completeness.** *We say that a retrieval strategy is complete if the retrieval set for any query is guaranteed to include a feasible case if such a case exists.*

Ensuring the retrieval of a case that may be acceptable to the user whenever possible is just one advantage of completeness. In Section 3, we show that *only* in a complete retrieval strategy can the exclusion of any case from the retrieval set always be justified on the basis that one of the retrieved cases satisfies at least the same constraints as the non-retrieved case. As we show in Theorem 1, another benefit of completeness is the ease with which the *non-existence* of an acceptable case can often be recognized from the results for a given query — an issue often neglected in recommender systems.

**Theorem 1.** *In a complete retrieval strategy, the non-existence of a feasible case in the retrieval set implies the non-existence of an acceptable case in the case base.*

**Proof.** Immediate from the definition of completeness.

As we show in Section 4, *k*-NN is incomplete regardless of the size of the retrieval set. We also show that it is possible for a retrieval strategy that takes no account of a retrieved case's similarity to be complete. An important question, therefore, is whether the benefits of completeness can be combined with those to be gained by taking account of similarity knowledge. For example, enabling an otherwise competitive case to *compensate* for its failure to satisfy one or more of the constraints satisfied by another case is an important advantage of *k*-NN. Moreover, a complete retrieval strategy that fails to retrieve the *most similar* case ignores the possibility that it may be the best option for the user if none of her requirements are hard constraints.

The role of similarity in balancing trade-offs between competing products is implicit in our definition of *optimal* completeness, the second of our completeness criteria for retrieval in recommender systems.

**Optimal Completeness.** *We say that a retrieval strategy is optimally complete if the retrieval set for any query is guaranteed to include a case that is maximally similar among the feasible cases, if any, in the case base.*

Ensuring the retrieval of the most similar feasible case, or one that is maximally similar, is likely to be of most benefit when there are many feasible cases for a given query — which is not unlikely if most of the constraints in a given query are soft constraints. As we show in Theorem 2, optimal completeness also ensures the retrieval of the most similar case in the case base.

**Theorem 2.** *In an optimally complete retrieval strategy, the retrieval set for any query must include the most similar case.*

**Proof.** It suffices to observe that if none of the constraints in a given query are hard constraints, then all the available cases are feasible cases.

As we show in Section 4, CDR [2-3] is optimally complete, thus ensuring the retrieval of the most similar case as in *k*-NN. Enabling an otherwise competitive case to compensate for its failure to satisfy one or more of the constraints satisfied by another case is another important feature that CDR shares with *k*-NN.

In Section 2, we use an example case base to illustrate some of the limitations of *k*-NN that are well known and others that have received less attention. We also show that some of the problems highlighted can be attributed to the incompleteness of *k*-NN. In Section 3, we present necessary and sufficient conditions for completeness and optimal completeness that can be used to determine whether these criteria are satisfied by a given retrieval strategy. In Section 4, we show that while most retrieval strategies used in CBR recommender systems are incomplete, CDR is optimally complete. Related work is discussed in Section 5, and our conclusions are presented in Section 6.

## 2 Limitations of *k*-NN

Increasing awareness of the limitations of *k*-NN in CBR recommender systems has prompted significant research interest in alternative retrieval strategies (e.g., [2-16]). A detailed account of the issues addressed by this important body of research is beyond the scope of the present discussion. Instead we use an example case base in

the property domain to illustrate some of the limitations of *k*-NN that are well known and others that have received less attention. We also show that some of problems highlighted can be attributed to the incompleteness of *k*-NN.

For example, an issue often neglected in recommender systems is that none of the available cases (i.e., products) may be acceptable to the user. If none of the available cases satisfies all the hard constraints in a given query, it is reasonable to expect that the *non-existence* of an acceptable case should be clear to the user from the system's response to her query. As shown in Section 1, an important benefit of completeness is that the non-existence of an acceptable case can always be inferred from the non-existence of a feasible case in the retrieval set for a given query.

Although *k*-NN is incomplete, it may be possible for a user with a good understanding of similarity to infer the non-existence of an acceptable case from the non-existence of a feasible case in the *k*-NN retrieval set. For example, it may be clear to such a user that a feasible case, if one existed, would be more similar than any of the recommended cases and would thus be included in the *k*-NN retrieval set. In general, however, the non-existence of an acceptable case cannot be inferred from the non-existence of a feasible case in the *k*-NN retrieval set. Before presenting the example that we use to clarify this important point, we outline a typical approach to similarity assessment in CBR recommender systems.

**Global Similarity Measure.** The similarity of a case $C$ to a given query $Q$ is typically defined as:

$$Sim(C, Q) = \frac{\sum_{a \in A} w_a \times sim_a(C, Q)}{\sum_{a \in A} w_a} \quad (1)$$

where $A$ is the set of attributes for which preferred values are specified in $Q$. For each $a \in A$, $w_a$ is an importance weight assigned to $a$ and $sim_a(C, Q)$ is a *local* measure of the similarity between the attribute's values in $C$ and $Q$.

**Local Similarity Measures.** Local similarity measures are often defined in terms of an attribute's values without reference to a specific query or case. For example, the similarity of two values $x$ and $y$ of a numeric attribute is typically defined as:

$$sim(x, y) = 1 - \frac{|x - y|}{\max - \min} \quad (2)$$

where max and min are the attribute's maximum and minimum values in the case base.

### 2.1 Example Case Base

Fig. 1 shows an example case base and query in the property domain that we use to illustrate some of the limitations of *k*-NN. The equally weighted attributes in the case base are location (A, B, or C), bedrooms (2, 3, or 4), type (detached, semi-detached, or terraced), and reception rooms (1, 2, or 3). The user is looking for a 4 bedroom

detached property in location A with 3 reception rooms (RRs). The similarity of each case to the target query is shown in the rightmost column. Similarity assessment with respect to location and type is based on the similarity scores: *sim*(A, A) = 1, *sim*(B, A) = 0.5, *sim*(C, A) = 0, *sim*(det, det) = 1, *sim*(sem, det) = 0.5, *sim*(ter, det) = 0. The standard similarity measure for numeric attributes (2) is used for bedrooms and reception rooms.

| | **Loc** | **Beds** | **Type** | **RRs** | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Query** | A | 4 | det | 3 | **Constraints** | | | | **Similarity** |
| Case 1 | B | 4 | det | 3 | N | Y | Y | Y | 0.88 |
| Case 2 | C | 4 | det | 3 | N | Y | Y | Y | 0.75 |
| Case 3 | A | 3 | sem | 2 | Y | N | N | N | 0.63 |
| Case 4 | A | 3 | ter | 2 | Y | N | N | N | 0.50 |
| Case 5 | A | 2 | det | 1 | Y | N | Y | N | 0.50 |

**Fig. 1.** Example case base and query in the property domain

Constraints in the example query that each case satisfies, or fails to satisfy, are indicated by the entries (Y or N) in the four columns to the right of each case in Fig. 1. No case satisfies all the constraints, a situation that would result in a query failure in traditional database retrieval. In the following sections, we briefly examine some of the issues highlighted by the 3-NN retrieval set for the example query (i.e., the three cases that are not shaded in Fig. 1).

### 2.2 Recognizing the Non-existence of an Acceptable Case

If *loc* = A and *beds* = 4 are hard constraints in the example query, then *none* of the available cases are acceptable to the user. But the user is unable to tell from the system's response to her query that there is no acceptable case. For example, she might be prepared to consider a 4 bedroom terraced property in location A with one reception room. But if such a case existed, it would not appear in the 3-NN retrieval set as its similarity to the user's query (0.50) would be less than the similarities of Cases 1, 2, and 3.

### 2.3 Coverage of Available Cases

As mentioned in the introduction, *k*-NN is known to be limited in its coverage of cases that may be acceptable to the user [2-5]. For example, if *loc* = A and *type* = det are hard constraints in the example query, then 3-NN has failed to retrieve the only case that might be acceptable to the user (i.e., Case 5). However, it can be seen from our definition of completeness (Section 1) that no complete retrieval strategy can fail to retrieve Case 5, the only feasible case in this situation. Thus *k*-NN's limited

coverage of cases that may be acceptable to the user can be attributed to its incompleteness.

### 2.4 Explaining Why a Case is Not Recommended

If asked to explain why Case 5 is not recommended in 3-NN, the system could point to three recommended cases that are more similar than Case 5, including two that match the given query exactly on bedrooms, type and reception rooms and one that matches it exactly on location. However, such an explanation is unlikely to satisfy a user who is unwilling to compromise on location or type. Of course, one reason why current recommender systems are seldom required to explain their failure to recommend a case which — in the user's opinion — should have been recommended is that the only cases that most users see are those recommended by the system. Nevertheless, explaining why a given case is not recommended is an important test of a system's ability to justify its recommendations [10, 12].

As we show in Section 3, it is *only* in a complete retrieval strategy that the exclusion of a given case from the retrieval set can always be justified on the basis that one of the retrieved cases satisfies at least the same constraints. As the above example shows, *k*-NN's failure to retrieve a given case cannot always be justified in this way — another limitation of *k*-NN that can be attributed to its incompleteness.

### 2.5 Recommendation Diversity

The first two cases in the 3-NN retrieval set are very similar to each other, and both satisfy the same constraints. One might argue, of course, that this makes good sense in a domain in which the recommended cases (i.e., properties) are sought in competition with other users. However, the recommendation engineering technique of providing the user with a link from each retrieved case to non-retrieved cases which satisfy the same constraints provides a simple solution to this problem in any retrieval strategy [14].

Retrieval strategies that aim to increase recommendation diversity by combining measures of similarity and diversity in the retrieval process [7-8] are discussed in Section 4. Instead of relying on a measure of diversity to guide the retrieval process, CDR [2-3] addresses the issue of recommendation diversity by ensuring that no two cases in the retrieval set satisfy the same constraints. Thus for the example query in Fig. 1, the CDR retrieval set would include Case 1 (N Y Y Y) and Case 3 (Y N N N) but not Case 2 (N Y Y Y) or Case 4 (Y N N N).

## 3 Completeness and Optimal Completeness

In this section, we formally define the concepts on which our completeness criteria are based. We also establish necessary and sufficient conditions for completeness and optimal completeness that can be used to determine whether or not these criteria are satisfied by a given retrieval strategy.

**Retrieval Set.** *Given a retrieval strategy S, we denote by r(S, Q) the set of cases that are retrieved in response to a given query Q.*

For example, $r(k\text{-NN}, Q)$ is the set of $k$ cases that are most similar to $Q$. To distinguish $k$-NN from the unrealistic strategy of retrieving all the available cases, we assume that $1 \leq k < n$, where $n$ is the number of cases.

**Query Constraints.** *For any query $Q$, we denote by constraints($Q$) the set of all constraints in $Q$.*

In addition to the *equality* constraints supported by any retrieval strategy, the constraints in a recommender system query might include upper and/or lower limits for numeric attributes and *sets* of preferred values for nominal attributes [2, 10, 13].

**Hard Constraints.** *For any query $Q$, we denote by hard-constraints($Q$) the set of hard constraints in $Q$.*

As mentioned in the introduction, we assume that any hard constraints in a given query are *unknown* to the recommender system.

**Satisfied Constraints.** *For any case $C$ and query $Q$, we denote by satisfied-constraints($C$, $Q$) the set of constraints in $Q$ that are satisfied by $C$.*

**Feasible Case.** *For any case $C$ and query $Q$, we say that $C$ is a feasible case for $Q$ if hard-constraints($Q$) $\subseteq$ satisfied-constraints($C$, $Q$).*

**Exactly Matching Case.** *We say that a case $C$ exactly matches a given query $Q$ if satisfied-constraints($C$, $Q$) = constraints($Q$).*

As well as providing a necessary and sufficient condition for completeness, Theorem 3 confirms our claim that *only* in a complete retrieval strategy can the exclusion of any case from the retrieval set always be justified on the basis that one of the retrieved cases satisfies at least the same constraints as the non-retrieved case.

**Theorem 3.** *A retrieval strategy $S$ is complete if and only if for any query $Q$ and $C_1 \notin r(S, Q)$, there exists $C_2 \in r(S, Q)$ such that satisfied-constraints($C_1$, $Q$) $\subseteq$ satisfied-constraints($C_2$, $Q$).*

**Proof.** If the latter condition holds, and $C_1$ is a feasible case for a given query $Q$, then *hard-constraints*($Q$) $\subseteq$ *satisfied-constraints*($C_1$, $Q$) and there exists $C_2 \in r(S, Q)$ such that *satisfied-constraints*($C_1$, $Q$) $\subseteq$ *satisfied-constraints*($C_2$, $Q$). We have established as required the existence of $C_2 \in r(S, Q)$ such that *hard-constraints*($Q$) $\subseteq$ *satisfied-constraints*($C_2$, $Q$). Conversely, if $S$ is a complete retrieval strategy then for any query $Q$ and $C_1 \notin r(S, Q)$ we can construct another query $Q'$ that differs from $Q$, if at all, only in that *hard-constraints*($Q'$) = *satisfied-constraints*($C_1$, $Q$). As *satisfied-constraints*($C_1$, $Q'$) = *satisfied-constraints*($C_1$, $Q$), $C_1$ is a feasible case for $Q'$, so it follows by the completeness of $S$ that there exists $C_2 \in r(S, Q')$ such that *hard-constraints*($Q'$) $\subseteq$ *satisfied-constraints*($C_2$, $Q'$). As we assume that no distinction is made between hard and soft constraints in the retrieval process, $r(S, Q') = r(S, Q)$, and so we have established the existence of $C_2 \in r(S, Q)$ such that *satisfied-constraints*($C_1$, $Q$) = *hard-constraints*($Q'$) $\subseteq$ *satisfied-constraints*($C_2$, $Q'$) = *satisfied-constraints*($C_2$, $Q$).

As we show in Theorem 4, a necessary and sufficient condition for *optimal* completeness is that the exclusion of any case from the retrieval set can always be justified on the basis that one of the retrieved cases is at least as similar as the non-retrieved case, and satisfies at least the same constraints.

**Theorem 4.** *A retrieval strategy S is optimally complete if and only if for any query Q and* $C_1 \notin r(S, Q)$, *there exists* $C_2 \in r(S, Q)$ *such that similarity*$(C_1, Q) \leq$ *similarity*$(C_2, Q)$ *and satisfied-constraints*$(C_1, Q) \subseteq$ *satisfied-constraints*$(C_2, Q)$.

**Proof.** Assuming that the latter condition holds, let $Q$ be any query for which a feasible case exists, and let $C_1$ be a feasible case of *maximal* similarity to $Q$. If $C_1 \in r(S, Q)$ there is nothing more to prove, while if $C_1 \notin r(S, Q)$ then there exists $C_2 \in r(S, Q)$ such that *similarity*$(C_1, Q) \leq$ *similarity*$(C_2, Q)$ and *satisfied-constraints*$(C_1, Q) \subseteq$ *satisfied-constraints*$(C_2, Q)$. Since *hard-constraints*$(Q) \subseteq$ *satisfied-constraints*$(C_1, Q) \subseteq$ *satisfied-constraints*$(C_2, Q)$, $C_2$ is a feasible case and so *similarity*$(C_2, Q) \leq$ *similarity*$(C_1, Q)$. It follows that *similarity*$(C_2, Q) =$ *similarity*$(C_1, Q)$, so we have established as required the existence of a feasible case of maximal similarity $C_2 \in r(S, Q)$.

Conversely, if $S$ is optimally complete then for any query $Q$ and $C_1 \notin r(S, Q)$ we can construct another query $Q'$ that differs from $Q$, if at all, only in that *hard-constraints*$(Q') =$ *satisfied-constraints*$(C_1, Q)$. As $C_1$ is a feasible case for $Q'$, it follows by the optimal completeness of $S$ that there exists $C_2 \in r(S, Q')$ of maximal similarity among all cases $C$ such that *hard-constraints*$(Q') \subseteq$ *satisfied-constraints*$(C, Q')$. In particular, *hard-constraints*$(Q') =$ *satisfied-constraints*$(C_1, Q) =$ *satisfied-constraints*$(C_1, Q')$ and so *similarity*$(C_1, Q) =$ *similarity*$(C_1, Q') \leq$ *similarity*$(C_2, Q') =$ *similarity*$(C_2, Q)$. As we assume that no distinction is made between hard and soft constraints in the retrieval process, $r(S, Q') = r(S, Q)$, and so we have established the existence of $C_2 \in r(S, Q)$ such that *similarity*$(C_1, Q) \leq$ *similarity*$(C_2, Q)$ and *satisfied-constraints*$(C_1, Q) =$ *hard-constraints*$(Q') \subseteq$ *satisfied-constraints*$(C_2, Q') =$ *satisfied-constraints*$(C_2, Q)$.

When the number of constraints in a given query is small, is not unusual for one or more exactly matching cases to be available. In this situation, a single recommended case (i.e., any exactly matching case) is enough to satisfy the condition for completeness in Theorem 3. Equally, a single recommended case (i.e., the most similar of the exactly matching cases) is enough to satisfy the condition for optimal completeness in Theorem 4.

## 4 Comparison of Retrieval Strategies

In this section we compare six possible approaches to retrieval in recommender systems with respect to the following criteria:

1. Is the most similar case always retrieved?
2. Is a feasible case always retrieved if such a case exists?
3. Is the most similar feasible case always retrieved if there is more than one feasible case?

Ensuring the retrieval of the most similar case, the first of the above criteria, is of course a feature of most retrieval strategies in CBR recommender systems. Criteria 2 and 3 are the criteria for completeness and optimal completeness that we use to assess the effectiveness of retrieval when, as often in practice, a given query may include constraints that must be satisfied.

### 4.1 *k*-NN Retrieval

As discussed in Section 1, the standard approach to the retrieval of recommended cases in CBR has some important advantages, such as ensuring the retrieval of the most similar case for a given query. As we show in Theorem 5, however, *k*-NN is incomplete regardless of the size of the retrieval set; that is, the existence of a feasible case does not guarantee that such a case will be retrieved.

**Theorem 5.** *k*-NN *is incomplete regardless of the size of the retrieval set.*

**Proof.** For any $k \geq 3$, we can construct a case base with $k + 1$ cases $C_1$, $C_2$, ..., $C_{k+1}$ and $k + 1$ equally weighted attributes $a_1$, $a_2$, ..., $a_{k+1}$ such that for $1 \leq i \leq k + 1$, $a_i = 1$ in all cases with the following exceptions: (1) for $1 \leq i \leq k$, $a_i = 0$ in $C_{k+1}$, (2) for $1 \leq i \leq k$, $a_i = 0$ in $C_i$, and (3) for $1 \leq i \leq k$, $a_{k+1} = 0$ in $C_i$. For $k = 3$, the cases in the resulting case base are:

$$C_1 = (0, 1, 1, 0),\ C_2 = (1, 0, 1, 0),\ C_3 = (1, 1, 0, 0),\ C_4 = (0, 0, 0, 1)$$

Now consider the query $Q = \{a_i = 1: 1 \leq i \leq k + 1\}$. It can be seen that for $1 \leq i \leq k$, $Sim(C_i, Q) = \frac{k-1}{k+1}$ while $Sim(C_{k+1}, Q) = \frac{1}{k+1}$. The *k*-NN retrieval set therefore includes all cases in the case base except $C_{k+1}$. It can also be seen that while *satisfied-constraints*$(C_{k+1}, Q) = \{a_{k+1} = 1\}$, there is no case in the *k*-NN retrieval set which satisfies the constraint $a_{k+1} = 1$. There is therefore no case $C$ in the *k*-NN retrieval set such that *satisfied-constraints*$(C, Q) \subseteq$ *satisfied-constraints*$(C_{k+1}, Q)$. It follows from Theorem 3 that for $k \geq 3$, *k*-NN is incomplete. It remains only to observe that as *k*-NN is incomplete for $k = 3$ it must also be incomplete for $k = 2$ and $k = 1$.

An important corollary of Theorem 5 is that any retrieval strategy in which the cases retrieved for a given query are selected from the *k*-NN retrieval set is incomplete.

**Theorem 6.** *Any retrieval strategy S in which the retrieved cases are selected from the k*-NN *retrieval set is incomplete.*

**Proof.** As *k*-NN is incomplete by Theorem 5, there exists for any $k \geq 1$ a case base, a query $Q$, and a case $C_1 \notin r(k\text{-NN}, Q)$ for which there is no case $C_2 \in r(k\text{-NN}, Q)$ such that *satisfied-constraints*$(C_1, Q) \subseteq$ *satisfied-constraints*$(C_2, Q)$. As $r(S, Q) \subseteq r(k\text{-NN}, Q)$ it follows that $C_1 \notin r(S, Q)$ and there can be no case $C_2 \in r(S, Q)$ such that *satisfied-constraints*$(C_1, Q) \subseteq$ *satisfied-constraints*$(C_2, Q)$. Thus $S$ is incomplete as required.

### 4.2 Database Retrieval

Often referred to as filter based retrieval, traditional database retrieval (DBR) insists on exact matching:

$$r(\text{DBR}, Q) = \{C: \textit{satisfied-constraints}(C, Q) = \textit{constraints}(Q)\}$$

If there is no case (or database object) that satisfies all the constraints in a given query, then the DBR retrieval set is empty and the user has no alternative but to start again with a modified query [1, 17].

**Theorem 7.** DBR *is incomplete*.

**Proof.** It suffices to observe that if none of the constraints in a given query are hard constraints then *any* case is a feasible case but the DBR retrieval set may still be empty.

DBR is not guaranteed to retrieve the most similar case for a given query, or indeed any case. Another drawback is that for queries with only a few constraints, the user may be swamped by a large number of exactly matching cases (or database objects) [17].

### 4.3 Retrieval of Non-dominated Cases

In database research, there is increasing interest in the retrieval of database objects that are Pareto optimal with respect to the preferences expressed in a given query (i.e., not *dominated* by another object) as an approach to addressing the limitations of retrieval based on exact matching [17-18]. Elimination of dominated alternatives is also a strategy sometimes used to reduce the number of candidates to be compared by other methods in multiple criteria decision making [19-20]. With respect to each constraint in a recommender system query, it is reasonable to assume that any case that satisfies the constraint is preferred to one that does not satisfy the constraint. A possible approach to retrieval of recommended cases is therefore one in which the retrieval set consists of all cases that are not dominated with respect to the constraints they satisfy. We will refer to this strategy as retrieval of non-dominated cases (RNC).

| | **Constraints** | | | | **Similarity** | RNC | 3-NN | CDR |
|---|---|---|---|---|---|---|---|---|
| Case 1 | N | Y | Y | Y | 0.88 | ● | ● | ● |
| Case 2 | N | Y | Y | Y | 0.75 | ● | ● | |
| Case 3 | Y | N | N | N | 0.63 | | ● | ● |
| Case 4 | Y | N | N | N | 0.50 | | | |
| Case 5 | Y | N | Y | N | 0.50 | ● | | ● |

**Fig. 2.** Cases retrieved (●) by RNC, 3-NN, and CDR for the example case base and query in the property domain

**Dominance Criterion.** *We say that a given case $C_1$ is dominated by another case $C_2$ with respect to a given query if $C_2$ satisfies every constraint that $C_1$ satisfies and at least one constraint that $C_1$ does not satisfy.*

It is worth noting that RNC is equivalent to retrieving the maxima with respect to a partial order induced by a given query $Q$ on the case base:

$C_1 \leq_Q C_2$ if and only if *satisfied-constraints*$(C_1, Q) \subseteq$ *satisfied-constraints*$(C_2, Q)$

It is therefore also related to order based retrieval, a general framework for retrieval in CBR recommender systems in which a partial order constructed from user preferences is applied to the case base to retrieve the maxima [6, 10].

Fig. 2 shows the cases retrieved by RNC for the example case base and query used to highlight limitations of $k$-NN in Section 2. The retrieval sets for 3-NN and CDR [2-3] are also shown. Case 3 (Y N N N) and Case 4 (Y N N N) are dominated by Case 5 (Y N Y N), but none of the other cases are dominated. The RNC retrieval set for the example query is therefore {Case 1, Case 2, Case 5}.

As we show in Theorem 8, RNC is complete. However, it is not optimally complete; that is, it is not guaranteed to retrieve the *most similar* feasible case if there is more than one feasible case. If the only hard constraint in the example query is the first constraint, then Cases 3, 4, and 5 are all feasible cases. But Case 5, the one retrieved by RNC, is not the most similar.

RNC is also not guaranteed to retrieve the most similar case. For example, if Case 1 and Case 2 were removed from the case base, then Case 3 would be the most similar of the remaining cases. However, it would still be dominated by Case 5 and therefore not retrieved by RNC. On the other hand, the RNC retrieval set may include more cases than are needed for completeness. For each $C_1 \in r$(RNC, $Q$), the retrieval set must also include all cases $C_2$ such that *satisfied-constraints*$(C_1, Q)$ = *satisfied-constraints*$(C_2, Q)$. In the RNC retrieval set for the example query, only Case 1 and Case 5 are needed to satisfy the condition for completeness in Theorem 3.

**Theorem 8.** RNC *is complete.*

**Proof.** It suffices by Theorem 3 to show that for any query $Q$ and $C_1 \notin r$(RNC, $Q$), there exists $C_2 \in r$(RNC, $Q$) such that *satisfied-constraints*$(C_1, Q) \subseteq$ *satisfied-constraints*$(C_2, Q)$. If $C_1 \notin r$(RNC, $Q$) there must be at least one case $C$ such that *satisfied-constraints*$(C_1, Q) \subset$ *satisfied-constraints*$(C, Q)$. Among such cases, let $C_2$ be one for which |*satisfied-constraints*$(C_2, Q)$| is maximal. As there can be no case that dominates $C_2$, we have established the existence of $C_2 \in r$(RNC, $Q$) such that *satisfied-constraints*$(C_1, Q) \subset$ *satisfied-constraints*$(C_2, Q)$.

### 4.4 Compromise-Driven Retrieval

In CDR, the constraints satisfied by a given case and its similarity to the target query play complementary roles in the retrieval process [2-3]. The first step in the construction of the retrieval set is to rank all cases in the case base in order of decreasing similarity. Among cases that are *equally* similar to the target query, any case that satisfies a superset of the constraints satisfied by another equally similar case is given priority in the ranking process. The algorithm used to select cases to be included in the CDR retrieval set from the ranked list of candidate cases is shown in Fig. 3.

```
algorithm CDR(Q, Candidates)
begin
  RetrievalSet ← ϕ
  while |Candidates| > 0 do
  begin
    C1 ← first(Candidates)
    RetrievalSet ← RetrievalSet ∪ {C1}
    Candidates ← Candidates - {C1}
    for all C2 ∈ rest(Candidates) do
    begin
      if satisfied-constraints(C2, Q) ⊆ satisfied-constraints(C1, Q)
      then Candidates ← Candidates - {C2}
    end
  end
  return RetrievalSet
end
```

**Fig. 3.** Constructing the retrieval set in compromise driven retrieval

First, the most similar case is placed in the retrieval set and any cases that satisfy a subset of the constraints satisfied by this case are eliminated from the list of candidate cases. The most similar of the remaining cases is now added to the retrieval set and any cases that satisfy a subset of the constraints satisfied by that case are eliminated. This process continues until no further cases remain.

A detail not shown in Fig. 3 is that for each case added to the CDR retrieval set, any cases that satisfy the *same* constraints are placed in a separate *reference set*. A link from the retrieved case to the reference set is also created, thus ensuring that cases which satisfy the same constraints are available for immediate inspection at the user's request. In this way, a retrieved case acts as a *representative* for all cases that satisfy the same constraints. This recommendation engineering technique [14] helps to keep the size of the CDR retrieval set within reasonable limits while also addressing the needs of users who are not just seeking a single recommended item, but would like to be informed of all items (e.g., jobs, rental apartments) that are likely to be of interest [2-3].

**Theorem 9.** CDR *is optimally complete.*

**Proof.** For any query $Q$, a given case $C_1$ can fail to be included in the CDR retrieval set only if there exists $C_2 \in r(\text{CDR}, Q)$ such that $similarity(C_1, Q) \leq similarity(C_2, Q)$ and $satisfied\text{-}constraints(C_1, Q) \subseteq satisfied\text{-}constraints(C_2, Q)$. The optimal completeness of CDR immediately follows from Theorem 4.

It follows from Theorem 2 (and is clear from Fig. 3) that in common with *k*-NN, the most similar case is always retrieved in CDR. Another important feature that CDR shares with *k*-NN is that an otherwise competitive case can often compensate for its failure to satisfy one or more of the constraints satisfied by another case. As shown in

Fig. 2, the CDR retrieval set for our example case base and query in the property domain is {Case 1, Case 3, Case 5}. Although Case 3 (Y N N N) fails to satisfy one of the constraints satisfied by Case 5 (Y N Y N), the greater similarity of Case 3 ensures that it is not excluded from the CDR retrieval set.

That the CDR retrieval set may include more cases than are needed for completeness can be seen from the fact that only Case 1 and Case 5 are needed to satisfy the condition for completeness in Theorem 3. However, Case 3 must also be included in the retrieval set for *optimal* completeness. If the first constraint is the only hard constraint in the example query, then Case 3 is the most similar of the feasible cases Case 3, Case 4, and Case 5.

### 4.5 Bounded Greedy

Bounded Greedy (BG) combines measures of similarity and diversity in the retrieval process to achieve a better balance between these often conflicting characteristics of the retrieved cases [8]. It selects $r$ cases from the $b \times r$ cases that are most similar to the target query, where $r$ is the required size of the retrieval set and $b$ is an integer parameter which is usually assigned a small value such as 2 or 3.

BG has been shown to provide significant gains in diversity at the expense of relatively small reductions in similarity [8]. As we show Theorem 10, however, BG is incomplete regardless of the size of the retrieval set or number of candidate cases from which the retrieved cases are selected.

**Theorem 10.** BG *is incomplete regardless of the size of the retrieval set or number of candidate cases from which the retrieved cases are selected.*

**Proof.** For any values of the parameters $b$ and $r$, BG selects $r$ cases from the $k$-NN retrieval set for $k = b \times r$. Its incompleteness immediately follows from Theorem 6.

A feature that BG shares with $k$-NN is that the most similar case for a given query is always retrieved.

### 4.6 Diversity Conscious Retrieval

Usually diversity can be increased only at the expense of some loss of average similarity relative to the $k$-NN retrieval set. Diversity conscious retrieval (DCR) aims to increase recommendation diversity while ensuring that any loss of similarity is strictly controlled [7]. The approach is based on the idea that for any integer $r \geq 2$, a given query partitions the set of cases with non-zero similarities according to the *similarity intervals* in which their similarities lie:

$$(0,\ 1-\frac{r-1}{r}\,],\ (1-\frac{r-1}{r}\,,\ 1-\frac{r-2}{r}\,],\ \ldots,\ (1-\frac{1}{r}\,,\ 1]$$

The retrieval process also depends on $k$, the required size of the retrieval set, and case selection is guided by the measure of relative diversity used in BG [8]. However, the DCR retrieval set is allowed to differ from the $k$-NN retrieval set only in cases whose similarities lie in the *leftmost* similarity interval that contributes to the $k$-NN retrieval set. This ensures that loss of average similarity relative to the $k$-NN retrieval

set can never be more than $\frac{1}{r}$, the width of the similarity intervals on which retrieval is based. For $r = 20$, the loss of average similarity cannot be more than 0.05. With this level of protection against loss of similarity, DCR has been shown to be capable of delivering worthwhile gains in diversity [7]. As we show in Theorem 11, however, DCR is incomplete.

In common with *k*-NN and BG, the most similar case for a given query is always retrieved in DCR.

**Theorem 11.** DCR *is incomplete regardless of the size of the retrieval set or width of the similarity intervals on which retrieval is based.*

**Proof.** For $r \geq 2$ and $k \leq 6$, we can construct a case base with $k + 1$ cases $C_1$, $C_2$, ..., $C_{k+1}$ and equally weighted attributes $a_1$, $a_2$, ..., $a_7$ such that for $1 \leq i \leq 7$, $a_i = 1$ in all cases with the following exceptions: (1) for $1 \leq i \leq 6$, $a_i = 0$ in $C_{k+1}$, (2) for $1 \leq i \leq k$, $a_i = 0$ in $C_i$, and (3) for $1 \leq i \leq k$, $a_7 = 0$ in $C_i$. For $k = 1$, the cases in the resulting case base are:

$$C_1 = (0, 1, 1, 1, 1, 1, 0),\ C_2 = (0, 0, 0, 0, 0, 0, 1)$$

Now consider the query $Q = \{a_i = 1: 1 \leq i \leq 7\}$. As $Sim(C_i, Q) = \frac{5}{7}$ for $1 \leq i \leq k$, and $Sim(C_{k+1}, Q) = \frac{1}{7}$, the *k*-NN retrieval set includes all cases in the case base except $C_{k+1}$. It is also clear that the similarities of all cases in the *k*-NN retrieval set must lie in same similarity interval. For $C_{k+1}$ to be eligible for retrieval in DCR, $Sim(C_{k+1}, Q)$ must therefore be in the *only* similarity interval that contributes to the *k*-NN retrieval set. However, this cannot be the case as the difference in similarity between $C_k$ and $C_{k+1}$ exceeds the width of the similarity intervals on which retrieval is based:

$$Sim(C_k, Q) - Sim(C_{k+1}, Q) = \frac{5}{7} - \frac{1}{7} = \frac{4}{7} > \frac{1}{2} \geq \frac{1}{r}$$

The DCR retrieval set for $Q$ is therefore the same as the *k*-NN retrieval set. As there is no case $C$ in the DCR retrieval set such that *satisfied-constraints*$(C, Q) \subseteq$ *satisfied-constraints*$(C_{k+1}, Q)$ it follows by Theorem 3 that DCR is incomplete for $r \geq 2$ and $k \leq 6$.

For $r \geq 2$ and $k > 6$, we can construct as in the proof of Theorem 5 a case base with $k + 1$ cases $C_1$, $C_2$, ..., $C_{k+1}$ and $k + 1$ equally weighted attributes $a_1$, $a_2$, ..., $a_{k+1}$, and a query $Q$ such that for $1 \leq i \leq k$, $Sim(C_i, Q) = \frac{k-1}{k+1}$ while $Sim(C_{k+1}, Q) = \frac{1}{k+1}$. Once again, the *k*-NN retrieval set includes all cases in the case base except $C_{k+1}$, and the latter case is not eligible for inclusion in the DCR retrieval set because:

$$Sim(C_k, Q) - Sim(C_{k+1}, Q) = \frac{k-2}{k+1} = \frac{7k-14}{7(k+1)} = \frac{3k+4k-14}{7(k+1)} > \frac{18+4k-14}{7(k+1)} = \frac{4(k+1)}{7(k+1)} = \frac{4}{7}$$

There is therefore no case $C$ in the DCR retrieval set such that *satisfied-constraints*$(C, Q) \subseteq$ *satisfied-constraints*$(C_{k+1}, Q)$. Thus DCR is also incomplete for $r \geq 2$ and $k > 6$.

### 4.7 Discussion

The results of our comparison of retrieval strategies are summarized in Table 1. Of the six retrieval strategies included in our analysis, only RNC and CDR are complete (i.e., guaranteed to retrieve a feasible case if one exists), and only CDR is optimally complete (i.e., guaranteed to retrieve the most similar feasible case if there are two or more feasible cases). All except DBR and RNC are guaranteed to retrieve the most similar case for a given query.

As shown by our analysis, the effectiveness of some retrieval strategies may be open to question when, as often in practice, a given query may include constraints that must be satisfied. In terms of assumptions about the nature of the constraints in a given query, DBR and *k*-NN can be seen to represent two extreme positions. In DBR, all the constraints in a given query are treated as hard constraints, whereas in *k*-NN they are essentially treated as soft constraints. However, recognizing that any combination of hard and soft constraints is possible in practice — as in a complete retrieval strategy — seems a more realistic basis for the retrieval of recommended cases. While the completeness of RNC ensures the retrieval of a feasible case if such a case exists, no account is taken of a retrieved case's similarity in the retrieval process. In contrast, the optimal completeness of CDR ensures the retrieval of the most similar feasible case, if any, for a given query.

**Table 1.** Comparison of retrieval strategies with respect to completeness, optimal completeness, and retrieval of the most similar case

| **Retrieval Strategy** | Feasible Case? | Most Similar Feasible Case? | Most Similar Case? |
|---|---|---|---|
| Database Retrieval (DBR) | N | N | N |
| *k*-NN | N | N | Y |
| Bounded Greedy (BG) | N | N | Y |
| Diversity Conscious Retrieval (DCR) | N | N | Y |
| Retrieval of Non-Dominated Cases (RNC) | Y | N | N |
| Compromise Driven Retrieval (CDR) | Y | Y | Y |

## 5 Related Work

### 5.1 Recommendation Dialogues

We have focused in this paper on approaches to the retrieval of recommended cases in response to a query provided by the user in advance. In approaches related to conversational CBR [21], a query is *incrementally* elicited in an interactive dialogue with the user, often with the aim of minimizing the number of questions the user is asked before a recommended product is retrieved (e.g., [22-30]).

Incremental nearest neighbor (iNN) is one such strategy that uniquely combines a goal-driven approach to selecting the most useful question at each stage of the recommendation dialogue with a mechanism for ensuring that the dialogue is terminated only when it is certain that the most similar case (or cases) will be the same regardless of the user's preferences with respect to any remaining attributes [25-28]. One important benefit is that recommendations based on incomplete queries can be justified on the basis that any user preferences that remain unknown cannot affect the recommendation. However, like any retrieval strategy in which only a single case (or set of equally similar cases) is recommended, and no attempt is made to identify constraints that must be satisfied, iNN is incomplete.

In any recommender system, of course, more than a single recommendation cycle may be needed to retrieve a case that is acceptable to the user. As we have seen, a complete retrieval strategy ensures the retrieval of a feasible case whenever possible, but whether such a case is acceptable to the user may depend on the extent to which it satisfies any soft constraints in her query — or additional constraints not mentioned in her query. Approaches to extending recommendation dialogues beyond an initially unsuccessful recommendation include critiquing approaches to elicitation of user feedback (e.g., [2, 10, 20, 31-32, 41]), referral of the user's query to other recommender agents [26], and the recommendation engineering technique of providing the user with a link from each recommended case to other cases that satisfy the same constraints [2-3, 14].

### 5.2 Retrieval Failure and Recovery

By ensuring the retrieval of a case that satisfies any hard constraints in a given query whenever possible, a complete retrieval strategy avoids the need to identify constraints in a given query that must be satisfied. In some retrieval strategies, the non-existence of an exactly matching case (i.e., one that satisfies all the constraints in a given query) is treated as a query *failure* (or retrieval failure) that triggers a recovery process based on query relaxation (e.g., [30, 32-37]). Usually the aim of the relaxation process is to identify constraints in the user's query that need *not* be satisfied and can thus be treated as soft constraints.

For example, the Adaptive Place Advisor [30] is an in-car recommender system for restaurants that converses with the user through a spoken dialogue interface. If there is no restaurant that exactly matches the user's requirements, the system suggests a constraint to be relaxed (e.g., *price range* or *cuisine*) based on its current understanding of the user's preferences. If the suggested constraint is one that must be satisfied, the system may suggest another constraint to be relaxed (i.e., treated as a soft constraint). In the Intelligent Travel Recommender [37], the choice of constraint to be relaxed is left to the user.

However, recovery may not be possible by relaxing a *single* constraint, or at least not one that the user is willing to relax [33-36]. To address this issue, McSherry [34] proposes an *incremental* relaxation process that aims to minimize the number of constraint relaxations required for recovery. An explanation of the query failure is followed by a mixed-initiative dialogue in which the user is guided in the selection of one or more constraints to be relaxed. If the constraint suggested for relaxation at any stage is one that must be satisfied, the user can select another constraint to be relaxed.

Expressed in terms of minimally failing sub-queries, the explanations of query failure provided in the approach are adapted from research on co-operative responses to failing database queries (e.g., [38-40]).

### 5.3 Compromise Driven Retrieval in First Case

First Case is a CDR recommender system that supports queries involving attributes and constraints of different types [2-3]. A *nominal* attribute is one whose values do not have a natural ordering that determines their similarity (e.g., the type or make of a personal computer). A *more-is-better* (MIB) attribute is one that most users would prefer to maximize (e.g., memory). A *less-is-better* (LIB) attribute is one that most users would prefer to minimize (e.g., price). A *nearer-is-better* (NIB) attribute is one for which most users have in mind an ideal value and prefer values that are closer to their ideal value (e.g., screen size). A query in *First Case* may include upper limits for LIB attributes (e.g., *price* ≤ 500), lower limits for MIB attributes (e.g., *memory* ≥ 256), and ideal values for NIB or nominal attributes (e.g., screen = 15, *type* = laptop).

An ideal value for a NIB or nominal attribute, if any, is treated as an equality constraint and also provides the basis for assessment of a given case's similarity with respect to the attribute. Assessment of similarity with respect to LIB/MIB attributes for which upper/lower limits are provided in a given query is based on *assumed* preferences [3, 28]. That is, the preferred value of a LIB attribute is assumed to be the *lowest* value in the case base, while the preferred value of a MIB attribute is assumed to be the *highest* value in the case base.

McSherry [3] investigates the potential benefits of also taking account of assumed preferences with respect to *non-query* attributes in CDR. One such benefit is that competitive cases that might otherwise be overlooked can more easily *compensate* for their failure to satisfy one or more of the constraints in a given query. Assumed preferences also play an important role in a CDR recommender system's ability to explain the benefits of a recommended case relative to another case that is less strongly recommended [2-3].

### 5.4 Balancing User Satisfaction and Cognitive Load

Balancing the trade-off between user satisfaction (or solution quality) and cognitive load is an important issue in recommender systems (e.g., [3, 5, 9]). A simple measure of cognitive load is the number of cases, on average, recommended in response to user queries, while possible measures of solution quality include precision and recall [13, 25]. The aspect of user satisfaction on which we have focused in this paper is the ability to ensure the retrieval of a case that may be acceptable to the user if such a case exists. As we have shown in Section 4, the optimal completeness of CDR ensures the retrieval of the most similar feasible case, if any, for a given query. In contrast, the existence of a feasible case does not guarantee that such a case will be retrieved regardless of the size of the *k*-NN retrieval set.

While the size of the *k*-NN retrieval set is the same for all queries, the size of the CDR retrieval set depends on the query and cannot be predicted in advance. That retrieval set size increases in CDR as query length increases is confirmed by our empirical results on the digital camera case base [41], with average retrieval set sizes

of 1.5 for short queries (3 attributes), 3.0 for longer queries (6 attributes), and 5.6 for queries of maximum length (9 attributes) [3]. Taking account of assumed preferences with respect to non-query (LIB/MIB) attributes had only a minor effect on cognitive load, with average retrieval set sizes increasing to 1.9 for short queries (3 attributes) and 3.3 for longer queries (6 attributes).

For queries of maximum length (8 attributes) on the well-known Travel case base, the average size of the CDR retrieval set was 7.7 [2]. Even for $k = 30$ in this experiment, *k*-NN was unable to match CDR's ability to ensure the retrieval of a case that satisfies any hard constraints in a given query if such a case exists. This provides empirical confirmation of our analysis in Section 4 showing that *k*-NN is incomplete regardless of the size of the retrieval set.

## 6 Conclusions

Completeness is a term used in search and planning, and other areas of artificial intelligence, to describe an algorithm's ability to guarantee that a solution will be found if one exists [42-44]. In this paper we have extended the notion of completeness to retrieval in recommender systems. We say that a retrieval strategy is *complete* if it ensures the retrieval of a product that satisfies any hard constraints in a given query if such a case exists.

We have shown that incompleteness is a limitation that *k*-NN shares with most retrieval strategies, and highlighted other limitations of *k*-NN which can be attributed to its incompleteness. One such limitation that appears not to be widely recognized is that the *non-existence* of an acceptable case cannot always be inferred from the non-existence of a feasible case (i.e., one that satisfies any hard constraints in a given query) in the *k*-NN retrieval set. Also in contrast to a complete retrieval strategy, *k*-NN's failure to retrieve a given case cannot always be justified on the basis that one of the retrieved cases satisfies at least the same constraints.

On the other hand, *k*-NN has important advantages that are not necessarily shared by a complete retrieval strategy, such as enabling an otherwise competitive case to *compensate* for its failure to satisfy one or more of the constraints satisfied by another case. However, the role of similarity in balancing trade-offs between competing cases is implicit in our definition of *optimal* completeness. A retrieval strategy is optimally complete if it ensures the retrieval of the *most similar* case, if any, which satisfies all the hard constraints in a given query. Optimal completeness also has the advantage of ensuring the retrieval of the most similar case as in *k*-NN.

Finally, we have shown that the ability to justify the exclusion of any case from the retrieval set on the basis that one of the retrieved cases is at least as similar, and satisfies at least the same constraints, is one of several benefits of CDR [2-3] that can be attributed to its optimal completeness.

## Acknowledgements

The author would like to thank Chris Stretch for his insightful comments on an earlier version of this paper.

## References


1. Wilke, W., Lenz, M., Wess, S.: Intelligent Sales Support with CBR. In: Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S. (eds.): Case-Based Reasoning Technology. Springer-Verlag, Berlin Heidelberg New York (1998) 91-113
2. McSherry, D.: Similarity and Compromise. In: Ashley, K.D., Bridge, D.G. (eds.): Case-Based Reasoning Research and Development. LNAI, Vol. 2689. Springer-Verlag, Berlin Heidelberg New York (2003) 291-305
3. McSherry, D.: On the Role of Default Preferences in Compromise-Driven Retrieval. Proceedings of the 10th UK Workshop on Case-Based Reasoning (2005) 11-19
4. McSherry, D.: Coverage-Optimized Retrieval. Proceedings of the 18th International Joint Conference on Artificial Intelligence (2003) 1349-1350
5. McSherry, D.: Balancing User Satisfaction and Cognitive Load in Coverage-Optimised Retrieval. Knowledge-Based Systems **17** (2004) 113-119
6. Bridge, D., Ferguson, A.: Diverse Product Recommendations using an Expressive Language for Case Retrieval. In: Craw, S., Preece, A. (eds.): Advances in Case-Based Reasoning. LNAI, Vol. 2416. Springer-Verlag, Berlin Heidelberg New York (2002) 43-57
7. McSherry, D.: Diversity-Conscious Retrieval. In: Craw, S., Preece, A. (eds.): Advances in Case-Based Reasoning. LNAI, Vol. 2416. Springer-Verlag, Berlin Heidelberg New York (2002) 219-233
8. Smyth, B., McClave, P.: Similarity vs. Diversity. In: Aha, D.W., Watson, I. (eds.): Case-Based Reasoning Research and Development. LNAI, Vol. 2080. Springer-Verlag, Berlin Heidelberg New York (2001) 347-361
9. Branting, L.K.: Acquiring Customer Preferences from Return-Set Selections. In: Aha, D.W., Watson, I. (eds.): Case-Based Reasoning Research and Development. LNAI, Vol. 2080. Springer-Verlag, Berlin Heidelberg New York (2001) 59-73
10. Bridge, D., Ferguson, A.: An Expressive Query Language for Product Recommender Systems. Artificial Intelligence Review **18** (2002) 269-307
11. Burkhard, H.-D.: Extending Some Concepts of CBR - Foundations of Case Retrieval Nets. In: Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S. (eds.): Case-Based Reasoning Technology. Springer-Verlag, Berlin Heidelberg New York (1998) 17-50
12. Ferguson, A., Bridge, D.: Partial Orders and Indifference Relations: Being Purposefully Vague in Case-Based Retrieval. In: Blanzieri, E., Portinale, L. (eds.): Advances in Case-Based Reasoning. LNAI, Vol. 1898. Springer-Verlag, Berlin Heidelberg New York (2000) 74-85
13. McSherry, D.: A Generalised Approach to Similarity-Based Retrieval in Recommender Systems. Artificial Intelligence Review **18** (2002) 309-341
14. McSherry, D.: Recommendation Engineering. Proceedings of the 15th European Conference on Artificial Intelligence. IOS Press, Amsterdam (2002) 86-90
15. McSherry, D.: The Inseparability Problem in Interactive Case-Based Reasoning. Knowledge-Based Systems **15** (2002) 293-300
16. McSherry, D., Stretch, C.: Automating the Discovery of Recommendation Knowledge. Proceedings of the 19th International Joint Conference on Artificial Intelligence (2005) 9-14
17. Kießling, W.: Foundations of Preferences in Database Systems. Proceedings of the 28th International Conference on Very Large Databases (2002) 311-322

18. Balke, W.-T., Günzer, U.: Efficient Skyline Queries under Weak Pareto Dominance. IJCAI-05 Workshop on Advances in Preference Handling (2005) 1-6
19. Hong, I., Vogel, D.: Data and Model Management in a Generalised MCDM-DSS. Decision Sciences **22** (1991) 1-25
20. Linden, G., Hanks, S., Lesh, N.: Interactive Assessment of User Preference Models: The Automated Travel Assistant. Proceedings of the 6th International Conference on User Modeling (1997) 67-78
21. Aha, D.W., Breslow, L.A., Muñoz-Avila, H.: Conversational Case-Based Reasoning. Applied Intelligence **14** (2001) 9-32
22. Doyle, M., Cunningham, P.: A Dynamic Approach to Reducing Dialog in On-Line Decision Guides. In: Blanzieri, E., Portinale, L. (eds.): Advances in Case-Based Reasoning. LNAI, Vol. 1898. Springer-Verlag, Berlin Heidelberg New York (2000) 49-60
23. Kohlmaier, A., Schmitt, S., Bergmann, R.: A Similarity-Based Approach to Attribute Selection in User-Adaptive Sales Dialogues. In: Aha, D.W., Watson, I. (eds.): Case-Based Reasoning Research and Development. LNAI, Vol. 2080. Springer-Verlag, Berlin Heidelberg New York (2001) 306-320
24. McSherry, D.: Minimizing Dialog Length in Interactive Case-Based Reasoning. Proceedings of the 17th International Joint Conference on Artificial Intelligence (2001) 993-998
25. McSherry, D.: Increasing Dialogue Efficiency in Case-Based Reasoning without Loss of Solution Quality. Proceedings of the 18th International Joint Conference on Artificial Intelligence (2003) 121-126
26. McSherry, D.: Conversational CBR in Multi-Agent Recommendation. IJCAI-05 Workshop on Multi-Agent Information Retrieval and Recommender Systems (2005) 331-345
27. McSherry, D.: Explanation in Recommender Systems. Artificial Intelligence Review **24** (2005) 179-197
28. McSherry, D.: Incremental Nearest Neighbour with Default Preferences. Proceedings of the 16th Irish Conference on Artificial Intelligence and Cognitive Science (2005) 9-18
29. Shimazu, H.: ExpertClerk: A Conversational Case-Based Reasoning Tool for Developing Salesclerk Agents in E-Commerce Webshops. Artificial Intelligence Review **18** (2002) 223-244
30. Thompson, C.A., Göker, M.H., Langley, P.: A Personalized System for Conversational Recommendations. Journal of Artificial Intelligence Research **21** (2004) 393-428
31. Burke, R., Hammond, K.J., Young, B.: The FindMe Approach to Assisted Browsing. IEEE Expert **12** (1997) 32-40
32. Hammond, K.J., Burke, R., Schmitt, K.: A Case-Based Approach to Knowledge Navigation. In: Leake, D.B. (ed.): Case-Based Reasoning: Experiences, Lessons & Future Directions. AAAI Press/MIT Press, Menlo Park, CA (1996) 125-136
33. McSherry, D.: Explanation of Retrieval Mismatches in Recommender System Dialogues. ICCBR-03 Workshop on Mixed-Initiative Case-Based Reasoning (2003) 191-199
34. McSherry, D.: Incremental Relaxation of Unsuccessful Queries. In: González-Calero, P., Funk, P. (eds.): Advances in Case-Based Reasoning. LNAI, Vol. 3155. Springer-Verlag, Berlin Heidelberg New York (2004) 331-345
35. McSherry, D.: Maximally Successful Relaxations of Unsuccessful Queries. Proceedings of the 15th Conference on Artificial Intelligence and Cognitive Science (2004) 127-136

36. McSherry, D.: Retrieval Failure and Recovery in Recommender Systems. Artificial Intelligence Review **24** (2005) 319-338
37. Ricci, F., Arslan, B., Mirzadeh, N., Venturini, A.: ITR: A Case-Based Travel Advisory System. In: Craw, S., Preece, A. (eds.): Advances in Case-Based Reasoning. LNAI, Vol. 2416. Springer-Verlag, Berlin Heidelberg New York (2002) 613-627
38. Gaasterland, T., Godfrey, P., Minker, J.: An Overview of Cooperative Answering. Journal of Intelligent Information Systems **1** (1992) 123-157
39. Godfrey, P.: Minimisation in Cooperative Response to Failing Database Queries. International Journal of Cooperative Information Systems **6** (1997) 95-149
40. Kaplan, S.J.: Cooperative Responses from a Portable Natural Language Query System. Artificial Intelligence **19** (1982) 165-187
41. McCarthy, K., Reilly, J., McGinty, L., Smyth, B.: Experiments in Dynamic Critiquing. Proceedings of the 10th International Conference on Intelligent User Interfaces (2005) 175-182
42. Lieber, J., Napoli, A.: Correct and Complete Retrieval for Case-Based Problem Solving. Proceedings of the 13th European Conference on Artificial Intelligence. Wiley, Chichester (1998) 68-72
43. Muñoz-Avila, H.: Case-base Maintenance by Integrating Case-Index Revision and Case-Retention Policies in a Derivational Replay Framework. Computational Intelligence **17** (2001) 280-294
44. Russell, S., Norvig, S.: Artificial Intelligence: A Modern Approach. Prentice Hall, Upper Saddle River, New Jersey (1995)