Minimally Complete Recommendations

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

Abstract

Recent research has highlighted the benefits of completeness as a retrieval criterion in recommender systems. In complete retrieval, any subset of the constraints in a given query that can be satisfied must be satisfied by at least one of the retrieved products. Minimal completeness (i.e., always retrieving the smallest set of products needed for completeness) is also beginning to attract research interest as a way to minimize cognitive load in the approach. Other important features of a retrieval algorithm's behavior include the diversity of the retrieved products and the order in which they are presented to the user. In this paper, we present a new algorithm for minimally complete retrieval (MCR) in which the ranking of retrieved products is primarily based on the number of constraints that they satisfy, but other measures such as similarity or utility can also be used to inform the retrieval process. We also present theoretical and empirical results that demonstrate our algorithm's ability to minimize cognitive load while ensuring the completeness and diversity of the retrieved products.

1 Introduction

Recommender systems use various criteria (e.g., similarity, diversity, utility) to guide the retrieval of recommended products from a list of available products [Burke, 2002; Linden *et al*., 1997; Pu and Chen, 2008; Smyth and McClave, 2001; Viappiani *et al*., 2006]. We focus in this paper on content-based recommender systems in which a target query is represented as a set of unary constraints on the values of product attributes (e.g., PC type = laptop, screen size = 15, price ≤ 500). In this context, recent research has highlighted the benefits of *completeness* as a retrieval criterion [McSherry, 2006]. Informally, a set of products retrieved for a given query Q is complete if any subset of the constraints in Q that is satisfied by one or more available products is satisfied by at least one of the retrieved products. A retrieval algorithm is complete if the retrieval set it produces for every query is complete.

An incomplete retrieval algorithm may fail to retrieve a product that satisfies the query constraints that are most important to the user even if such a product exists. Moreover, it is only in a complete retrieval algorithm that failure to retrieve a given product can always be explained / justified on the basis that another product that has been retrieved satisfies at least the same constraints. Another important benefit of completeness is the user's ability to infer that none of the available products is acceptable if no product in the retrieval set satisfies a constraint, or set of constraints, that must be satisfied.

Despite the benefits of complete retrieval, most retrieval algorithms used in recommender systems are incomplete. For example, case-based recommender systems sometimes use a retrieval approach adapted from the k-NN algorithm widely used in case-based reasoning (CBR) and elsewhere [Bridge *et al*., 2005; Smyth, 2007]. In this approach, the system recommends the k products that are most similar to the target query. However, retrieval based on k-NN is incomplete for all $k \geq 1$ [McSherry, 2006]. In Section 2, we generalize this result to show that *any* algorithm that always retrieves the same number of products is incomplete.

One example of a complete retrieval algorithm is compromise-driven retrieval (CDR) [McSherry, 2006]. In CDR, the available products are initially ranked in order of decreasing similarity. Any product that satisfies only a subset of the constraints satisfied by another product that precedes it in the product ranking is then eliminated, and the remaining products are retrieved. The stronger condition of *minimal* completeness is also beginning to attract research interest as a retrieval criterion [McSherry, 2008]. A retrieval algorithm is minimally complete if it always produces a retrieval set of the smallest possible size required for completeness, thus helping to reduce cognitive load. However, the order in which retrieved products are presented to the user is another important feature of a recommender system's behavior [Branting, 2004; Coyle and Cunningham, 2004].

In this paper, we present a new algorithm for minimally complete retrieval called MCR^+ in which the ranking of retrieved products is primarily based on the number of query constraints that they satisfy, although other measures such as similarity or utility can be used as additional ranking criteria. Our approach is motivated by the view that while it is natural for products that satisfy most constraints to be

ranked highest, more discriminating measures may be needed to rank products that satisfy the same number of constraints. One of the secondary ranking criteria that we use to illustrate the approach is the similarity measure described below.

Alternatively, the secondary ranking criterion used in MCR$^+$ may be as simple as product price, making the approach easy to implement in any recommender system. Another important benefit of the approach is that ranking the available products according to the number of constraints they satisfy results in a much simpler retrieval process than in previous work [McSherry, 2008]. In MCR$^+$, a minimally complete retrieval set can be constructed simply by eliminating any product that satisfies only a subset of the constraints satisfied by another product that precedes it in the product ranking.

A common approach to similarity assessment in case-based recommender systems is to measure each product's similarity to an "ideal" product with attribute values that represent the user's known or assumed preferences. Preferred values for some of the product attributes may be given as equality constraints in the target query (e.g., PC type = laptop, screen size = 15). Preferred values for some attributes can be reasonably assumed, whether or not they are mentioned in the target query (e.g., the lowest available price, or the highest available processor speed).

For any product x and ideal product y we define:

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

where A is the set of product attributes for which preferred values are given or assumed, $sim_a(x, y)$ is a local measure of the similarity between the values of a in x and y, and w_a is an importance weight assigned to a for each $a \in A$. In the experiments reported in Section 5, we assign a weight of one to each $a \in A$ other than price, and an integer weight of one or more to price.

For any product x, ideal product y, and numeric attribute a, we define:

$$sim_a(x, y) = 1 - \frac{|v_1 - v_2|}{max(a) - min(a)} \quad (2)$$

where v_1 and v_2 are the values of a in x and y, and $min(a)$ and $max(a)$ are the minimum and maximum values of a in the product dataset. For a nominal attribute (e.g., PC type) we may use domain knowledge to assign a local similarity score from 0 to 1 to its values in x and y, or simply 0 for unequal values and 1 for equal values.

In Section 2, we use an example dataset to compare the behavior of k-NN, CDR, and MCR$^+$. A formal definition of MCR$^+$ is provided in Section 3. In Section 4, we formally demonstrate the minimal completeness of MCR$^+$ and its ability to ensure the diversity of the retrieved products. We also compare upper bounds for retrieval-set sizes in MCR$^+$ and CDR that we show to be achievable, or almost achievable, in the respective algorithms. Our empirical results and conclusions are presented in Sections 5 and 6.

2 Comparison of Retrieval Algorithms

Table 1 shows an example dataset and query in the real estate domain that we use to compare k-NN, CDR, and the MCR$^+$ algorithm that we formally present in Section 3. The attributes of the available properties (P1-P7) are location (A, B), bedrooms (2, 3, 4, 5), reception rooms (2, 3), and price in units of $1,000. The user is looking for a property in location A with at least 4 bedrooms and 3 reception rooms (RRs) and is willing to pay up to $300,000.

The shaded cells in Table 1 indicate constraints that each property fails to satisfy. For example, P3 satisfies the constraints loc = A, beds ≥ 4, and RRs = 3, but not the price constraint. Also shown in Table 1 is the similarity of each property from Eqn. 1 with a weight of one assigned to each attribute except price, a weight of 5 assigned to price, and the preferred values of price and bedrooms assumed to be 250 and 5 respectively. The properties retrieved by 3-NN, CDR, and MCR$^+$, and their ranking in each algorithm (e.g., 1st, 2nd, 3rd) are shown in the last three columns.

	Loc	Beds	RRs	Price ($1,000)	Sim	3-NN	CDR	MCR$^+$
Query:	A	≥ 4	3	≤ 300				
P1	B	3	3	290	0.48	3rd		
P2	A	3	3	320	0.37			
P3	A	5	3	330	0.38		3rd	
P4	B	2	3	270	0.59	2nd	2nd	2nd
P5	B	3	2	250	0.67	1st	1st	
P6	A	4	3	325	0.37			1st
P7	A	4	2	315	0.33			

Table 1. An example dataset and query in the real estate domain, and the properties retrieved by 3-NN, CDR, and MCR$^+$.

A complete retrieval set for the target query Q = {loc = A, beds ≥ 4, RRs = 3, price ≤ 300} must include P1 or P4 as no other property satisfies the RRs and price constraints. It is also essential for completeness to retrieve one of the properties (P3 and P6) that satisfy the constraints loc = A, beds ≥ 4, and RRs = 3. So at least two products must be retrieved for completeness in our example. It can also be seen that any subset of the query constraints that is satisfied by one or more available products is satisfied by one of the products in the MCR$^+$ retrieval set (P4 and P6). It follows that the MCR$^+$ retrieval set is minimally complete.

Note that the user can infer from the MCR$^+$ retrieval set that there is no available property in location A that satisfies her price constraint. Likewise, there can be no available property with beds ≥ 4 and price ≤ 300 as there is no such property in the MCR$^+$ retrieval set. However, these important inferences can only be made if the retrieval algorithm is known to be complete.

Any product in a minimally complete retrieval set can be replaced by another product that satisfies the same constraints without affecting the minimal completeness of the

retrieval set. For example, P4 can be replaced by P1 and/or P6 by P3, so the MCR$^+$ retrieval set is only one of 4 possible retrieval sets that are minimally complete for the example query. By default, MCR$^+$ constructs a retrieval set that minimizes the average price of the retrieved products over all minimally complete retrieval sets. In Section 3, we show how other measures (e.g., similarity) can be used to inform the choice between minimally complete retrieval sets.

The incompleteness of 3-NN can be seen from its failure to retrieve a property that satisfies the location, bedrooms, and RRs constraints even though there are two such properties in the dataset. As we now show, *any* algorithm that always retrieves the same number of products is incomplete.

Theorem 1. *Any retrieval algorithm that always retrieves k products, where* $k \geq 1$, *is incomplete.*

Proof. For any integer $k \geq 1$, we can construct a product dataset with $k + 1$ binary attributes a_1, ..., a_{k+1} and $k + 1$ product descriptions x_1, ..., x_{k+1} such that the value of a_i in x_i is 0 for $1 \leq i \leq k + 1$ and the value of a_i in x_j is 1 for $1 \leq i, j \leq k + 1$ such that $i \neq j$. Now consider the query $Q = \{a_1 = 1, ..., a_{k+1} = 1\}$ in which all attributes are required to have non-zero values. Any retrieval set of size k must fail to include a product that satisfies the constraint $a_i = 1$, where x_i is the product that is not retrieved. As this constraint is satisfied by one of the available products, namely x_i, it follows that any retrieval set of size k is incomplete. □

While CDR is complete, its lack of minimal completeness can be seen from its retrieval of one more product (P5) than needed for completeness. In common with k-NN, CDR always retrieves the most similar product, which can sometimes be useful as a way of drawing the user's attention to an otherwise competitive product that narrowly fails to satisfy one or more constraints [McSherry, 2006]. However, the most similar property in Table 1 (P5) seems unlikely to be acceptable as it satisfies only one of the 4 constraints in the user's query. As we show in Sections 4 and 5, the CDR retrieval set for a given query may be considerably larger than the MCR$^+$ retrieval set.

A feature that MCR$^+$ shares with CDR is that no two products in the retrieval set can satisfy the same constraints, thus ensuring the diversity of the retrieved products (if more than one). As we show in Section 4, MCR$^+$ enforces an even stronger condition with respect to the constraints that two retrieved products may satisfy. That the products retrieved by k-NN may be lacking in diversity with respect to the constraints that they satisfy can be seen from the fact that two of the products in the 3-NN retrieval set (P1 and P4) satisfy the same constraints.

3 Algorithm Definition

A formal definition of MCR$^+$, our algorithm for minimally complete retrieval, is presented in Figure 1. The algorithm constructs a retrieval set for a given query from a list of available products, in order of decreasing number of satisfied constraints, by eliminating any product that satisfies only a subset of the constraints satisfied by another product that precedes it in the given list of products. For any product x and query Q, we denote by *satisfied*(x, Q) the set of constraints in Q that are satisfied by x. A formal proof of the minimal completeness of MCR$^+$ is provided in Section 4.

Algorithm: MCR$^+$

Input:

Q: A query (i.e., a set of constraints)
L: A list of available products in order of (1) decreasing number of satisfied constraints, and (2) increasing price

Output:

R: A complete retrieval set of the smallest possible size

```
1  begin
2      R1 ← [ ]
3          while | L | > 0 do
4          begin
5              x1 ← FIRST(L)
6              D ← [x1]
7              R1 ← [x1 | R1]
8              for all x2 ∈ REST(L) do
9                  if satisfied(x2, Q) ⊆ satisfied(x1, Q)
10                 then D ← [x2 | D]
11             for all x3 ∈ D do L ← REMOVE(x3, L)
12         end
13         R ← REVERSE(R1)
14         return R
15 end
```

Figure 1. Minimally complete retrieval in MCR$^+$ with product price as a secondary ranking criterion.

Using product price as a secondary ranking criterion in MCR$^+$ has the effect of minimizing average price over all minimally complete retrieval sets. Alternatively, the goal might be to maximize the average similarity of the retrieved products over all minimally complete retrieval sets. This can be achieved by replacing *increasing price* by *decreasing similarity* as the secondary ranking criterion in MCR$^+$. For the example query in Table 1, the result will be that P3 is retrieved instead of P6 because of its higher similarity. More generally, any measure of product suitability can be used as a secondary ranking criterion in MCR$^+$ without affecting the minimal completeness of the retrieval set.

4 Analysis of MCR$^+$ and CDR

In this section, we present necessary and sufficient conditions for a complete retrieval set to be minimally complete and show that MCR$^+$ is minimally complete. We also compare upper bounds for retrieval-set sizes in MCR$^+$ and CDR that we show to be achievable, or almost achievable, in the two algorithms. For any query Q and retrieval algorithm A, we denote by $r(Q, A)$ the retrieval set for Q in A. We will write $r(Q)$ instead of $r(Q, A)$ when discussing the retrieval set for a query Q without reference to the retrieval algorithm. It can easily be verified that the following definitions of completeness and minimal completeness are equivalent to the informal definitions provided in Section 1.

Definition 1. *A retrieval set $r(Q)$ is complete if and only if, for all $x \notin r(Q)$, there exists $y \in r(Q)$ such that satisfied$(x, Q) \subseteq$ satisfied(y, Q).*

Definition 2. *A complete retrieval set $r_1(Q)$ is minimally complete if and only if $|r_1(Q)| \leq |r_2(Q)|$ for every complete retrieval set $r_2(Q)$.*

Definition 3. *A retrieval algorithm A is complete / minimally complete if and only if $r(Q, A)$ is complete / minimally complete for every query Q.*

Theorem 2. *A complete retrieval set $r(Q)$ is minimally complete if and only if $x = y$ for all $x, y \in r(Q)$ such that satisfied$(x, Q) \subseteq$ satisfied(y, Q).*

Proof. Let $r_1(Q)$ be a complete retrieval set and suppose there exist distinct $x, y \in r_1(Q)$ such that *satisfied*$(x, Q) \subseteq$ *satisfied*(y, Q). Another possible retrieval set for Q is $r_2(Q) = r_1(Q) - \{x\}$, and we know that $x \notin r_2(Q)$, $y \in r_2(Q)$, and *satisfied*$(x, Q) \subseteq$ *satisfied*(y, Q). Moreover, for any $w \notin r_2(Q)$ such that $w \neq x$, there exists $z \in r_1(Q)$ such that *satisfied*$(w, Q) \subseteq$ *satisfied*(z, Q). If $z = x$, then *satisfied*$(w, Q) \subseteq$ *satisfied*$(x, Q) \subseteq$ *satisfied*(y, Q), while if $z \neq x$, then $z \in r_2(Q)$. We have now established that $r_2(Q)$ is complete. As $|r_1(Q)| > |r_2(Q)|$, $r_1(Q)$ is not minimally complete, thus proving the necessity.

Now suppose that $r_1(Q)$ is a complete retrieval set such that $x = y$ for all $x, y \in r_1(Q)$ such that *satisfied*$(x, Q) \subseteq$ *satisfied*(y, Q). For any complete retrieval set $r_2(Q)$, it is clear that $|r_1(Q)| \leq |r_2(Q)|$ if $r_1(Q) \subseteq r_2(Q)$. If it is not true that $r_1(Q) \subseteq r_2(Q)$, then for any $x \in r_1(Q) - r_2(Q)$, there exists $y \in r_2(Q)$ such that *satisfied*$(x, Q) \subseteq$ *satisfied*(y, Q). As $x \neq y$, it follows that $y \notin r_1(Q)$, and so there exists $z \in r_1(Q)$ such that *satisfied*$(y, Q) \subseteq$ *satisfied*(z, Q). As $x, z \in r_1(Q)$ and *satisfied*$(x, Q) \subseteq$ *satisfied*(z, Q), it must also be true that $x = z$. It can also be seen that *satisfied*$(x, Q) =$ *satisfied*(y, Q). Denoting by $x_1, \ldots, x_k$ the distinct elements of $r_1(Q) - r_2(Q)$, we can similarly demonstrate the existence of $y_1, \ldots, y_k \in r_2(Q) - r_1(Q)$ such that *satisfied*$(x_i, Q) =$ *satisfied*(y_i, Q) for $1 \leq i \leq k$. Moreover, it cannot be true that $y_i = y_j$ for $1 \leq i < j \leq k$ as this would imply that *satisfied*$(x_i, Q) =$ *satisfied*(x_j, Q) and therefore that $x_i = x_j$. It follows that $|r_2(Q)| \geq k + |r_1(Q) \cap r_2(Q)| = |r_1(Q)|$. We have now established that $r_1(Q)$ is minimally complete, thus proving the sufficiency. □

Theorem 3. MCR$^+$ *is minimally complete.*

Proof. It can be seen from Figure 1 that for any query Q and list of available products L presented to MCR$^+$, a given product $x \in L$ can fail to be included in the MCR$^+$ retrieval set only if there exists $y \in r(Q, \text{MCR}^+)$ such that *satisfied*$(x, Q) \subseteq$ *satisfied*(y, Q). Thus $r(Q, \text{MCR}^+)$ is complete by Definition 1. Now for any distinct $x, y \in r(Q, \text{MCR}^+)$, we can assume without loss of generality that x precedes y in L. As products in L are ranked by decreasing number of satisfied constraints, *satisfied*$(x, Q) \not\subset$ *satisfied*(y, Q). Moreover, it cannot be true that *satisfied*$(x, Q) =$ *satisfied*(y, Q), as this would ensure the exclusion of y from $r(Q, \text{MCR}^+)$. As $r(Q, \text{MCR}^+)$ is complete, it follows from Theorem 2 that $r(Q, \text{MCR}^+)$ is minimally complete, and so MCR$^+$ is minimally complete by Definition 3. □

It follows from Theorems 2 and 3 that no product retrieved by MCR$^+$ can satisfy a subset of the constraints satisfied by another retrieved product. These results confirm our algorithm's ability to minimize cognitive load while ensuring the completeness and diversity of the retrieved products. Moreover, the initial ranking of available products by decreasing number of satisfied constraints is key to the simplicity of the retrieval process in MCR$^+$. For example, consider a product dataset containing only two products x_1 and x_2 and query Q such that *satisfied*$(x_1, Q) \subset$ *satisfied*(x_2, Q). It suffices to retrieve a single product (x_2) for minimal completeness in this example. But unless x_2 is presented to MCR$^+$ before x_1 as specified in the algorithm definition, the retrieval set will contain both x_1 and x_2.

For any product dataset, query Q, and distinct $x, y \in r(Q, \text{MCR}^+)$, we know from Theorems 2 and 3 that *satisfied*(x, Q) and *satisfied*(y, Q) are incomparable subsets of the constraints in Q. It follows from Sperner's [1928] theorem that the size of the MCR$^+$ retrieval set cannot be more than ${}^nC_{\lfloor n/2 \rfloor}$, where $n = |Q|$, the number of constraints in Q. For example, the size of the MCR$^+$ retrieval set cannot be more than ${}^5C_2 = 10$ for queries with up to 5 constraints. As no two products retrieved by CDR for a given query Q can satisfy the same constraints, the size of the CDR retrieval set cannot be more than 2^n, where $n = |Q|$.

An important question is whether these upper bounds for the sizes of the retrieval set in MCR$^+$ and CDR are achievable. In Theorems 4 and 5, we show that they are achievable, or almost achievable, in both algorithms. For example, it follows from Theorem 4 that the number of products retrieved for a query containing 5 constraints can be as high as 31 in CDR. However, as shown by our empirical results in Section 5, retrieval-set sizes in both algorithms tend to be much smaller than the sizes we now show to be possible.

Theorem 4. *For any integer $n \geq 1$, it is possible to construct a product dataset and target query Q such that $|Q| = n$ and $|r(Q, \text{CDR})| = 2^n - 1$.*

Proof. For any integer $n \geq 1$, suppose we are given n product attributes $a_1, \ldots, a_n$, each with integer values in the range from 0 to $2^n - 1$, and that the user can be assumed to prefer the highest possible value of each attribute. Thus the ideal product (y) is such that $a_i = 2^n - 1$ for $i = 1$ to n. Now consider the query $Q = \{a_1 > 0, \ldots, a_n > 0\}$ in which all attributes are required to have non-zero values. For $r = 1$ to n, there are ${}^nC_r$ distinct product descriptions $x = \{a_1 = v_1, \ldots, a_n = v_n\}$ such that $|satisfied(x, Q)| = r$ and $v_i \in \{0, 2^{n-r+1} - 1\}$ for $1 \leq i \leq n$. We can thus construct a product dataset containing ${}^nC_1 + {}^nC_2 + \ldots + {}^nC_n = 2^n - 1$ product descriptions such that the number of constraints satisfied by each product ranges from 1 to n and no two products satisfy the same constraints. Moreover, with all attributes assigned equal weights in Eqn. 1, the similarity of each product x in the dataset to the ideal product y is $Sim(x, y) = \frac{r(2^{n-r+1} - 1)}{n(2^n - 1)}$, where $r = |satisfied(x, Q)|$. It can also be verified that

$\frac{r(2^{n-r+1}-1)}{n(2^n-1)}$ is strictly decreasing as r increases from 1 to n.

In CDR, a product can fail to be retrieved only if there exists a more (or equally) similar product that satisfies at least the same constraints. As this is not the case for any product in the dataset, the CDR retrieval set must include all the $2^n - 1$ products in the dataset. □

Theorem 5. *For any integer $n \geq 1$, it is possible to construct a product dataset and target query Q such that $|Q| = n$ and $|r(Q, \text{MCR}^+)| = {}^nC_{\lfloor n/2 \rfloor}$.*

Proof. That the result is true for $n = 1$ can be seen from the fact that $|r(Q, \text{MCR}^+)| = 1$ for any product dataset and query Q such that $|Q| = 1$. For any integer $n \geq 2$, consider again the dataset and target query constructed in the proof of Theorem 4. It can be seen from Theorems 2 and 3 that only a single product will be retrieved by MCR$^+$, namely the one that satisfies all the query constraints. However, if all products are deleted from the dataset except the ${}^nC_{\lfloor n/2 \rfloor}$ products that satisfy $\lfloor n/2 \rfloor$ constraints in the target query, then all the remaining products must be retrieved by MCR$^+$. □

5 Empirical Study

In this section, we compare the performance of MCR$^+$ and CDR on two benchmark datasets in terms of average retrieval-set size for queries of different sizes. While CDR can never retrieve fewer products than MCR$^+$, we expect average retrieval-set size to increase with query size (i.e., number of constraints) in both algorithms. Also of interest in our analysis is the extent to which average retrieval-set size in CDR depends on the weights assigned to product attributes in the underlying similarity measure. We will refer to a version of CDR with an integer weight of $w \geq 1$ assigned to price, and a weight of one to each of the other attributes, as CDR(w). Our first experiment applies MCR$^+$, CDR(1), and CDR(10) to the PC dataset [McGinty and Smyth, 2002]. The dataset contains descriptions of 120 PCs in terms of attributes such as price, memory, and screen size.

Using a *leave-one-out* approach to evaluation, we temporarily remove each PC from the dataset and use its description to generate a total of 127 queries of sizes n = 2, 4, 6, and 8. For example, the 8 attributes in a PC's description provide ${}^8C_2 = 28$ different queries involving 2 of the 8 attributes (e.g., type = laptop, price ≤ 750). We use a left-out PC's price to create a constraint that specifies the maximum amount that the user would like to pay. We use its speed, memory, and hard disk size to create constraints that specify the minimum values that the user is willing to consider. Finally, we use the make, processor, type, and screen size (which we round to the nearest inch in the experiment) to create equality constraints.

We submit each query to recommender systems based on MCR$^+$, CDR(1), and CDR(10) and observe the retrieval-set size for each algorithm. Similarity assessment is based on Eqn. 1, for example with the ideal price / memory assumed to be the lowest / highest in the dataset. This process is repeated for all 120 products in the dataset, and the results are summarized in Figure 2.

Average retrieval-set size can be seen to increase as query size increases in all three algorithms, but is lowest in MCR$^+$ for all query sizes. However, average retrieval-set sizes in MCR$^+$ and CDR are much smaller than the sizes we know to be possible from Theorems 4 and 5. This is particularly noticeable for query sizes n = 6 and n = 8, for which retrieval-set sizes can be as high as 20 and 70 in MCR$^+$ and even higher in CDR. A detail not shown in Figure 2 is that retrieval-set size was never more than 10 in MCR$^+$, while the largest retrieval-set size in CDR (with w = 10) was 14.

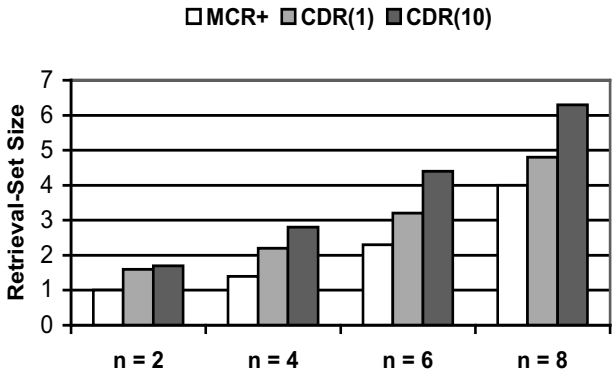

Figure 2. Average sizes of MCR$^+$, CDR(1), and CDR(10) retrieval sets for query sizes n = 2, 4, 6, and 8 on the PC dataset.

Average retrieval-set size in CDR(10) is higher than in CDR(1) for all query sizes, and the observed differences are greatest for query sizes n = 6 and n = 8. Other factors that may influence the size of the CDR retrieval set, though beyond the scope of our analysis, include the attribute types (i.e., numeric and/or nominal) used to describe the available products, and local similarity measures used in the assessment of product similarity.

Figure 3 shows cumulative frequencies of observed retrieval-set sizes in MCR$^+$, CDR(1), CDR(5), and CDR(10) for queries of size n = 8 in a similar experiment on the "Travel" dataset from `http://cbrwiki.fdi.ucm.es/`. This is a larger dataset that contains the descriptions of 1,024 holidays (e.g., price, holiday type, duration). Retrieval-set sizes from 1 to 6 can be seen to account for more than 50% of queries in MCR$^+$. In contrast, only 27% of retrieval sets contain from 1 to 6 products in CDR(10). It can also be seen from the steep rise in cumulative frequency as retrieval-set size increases from 14 to 15+ in CDR(10) that it is not exceptional for 15 or more products to be retrieved in this version of CDR.

As might be expected, average retrieval-set size in MCR$^+$ (6.9) is higher than for queries of size n = 8 on the smaller PC dataset (4.0). However, the largest retrieval-set size in MCR$^+$ for any query on the Travel dataset (19) is much smaller than the maximum size (70) we know to be possible for queries of size n = 8.

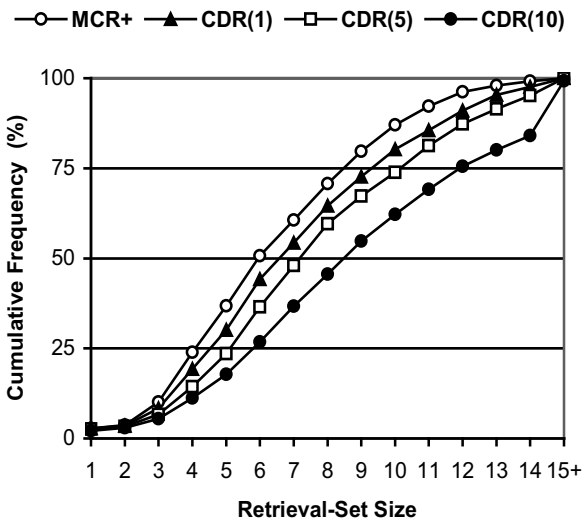

Figure 3. Cumulative frequencies of retrieval-set sizes for queries of size $n = 8$ on the Travel dataset.

6 Conclusions

We presented a new algorithm for minimally complete retrieval called MCR$^+$ and demonstrated its ability to minimize cognitive load while ensuring the completeness and diversity of the retrieved products. While the size of the MCR$^+$ retrieval set can never be more than 10 for queries of size $n \leq 5$, we showed that the number of products retrieved by CDR can be as high as 31 for such queries. Moreover, our empirical results show that retrieval-set size in MCR$^+$, while increasing with query size, tends to be much lower in practice than the maximum we showed to be possible for each query size.

Even for queries of size $n = 8$ on the PC dataset, only 4 products were retrieved on average by MCR$^+$, with retrieval-set sizes from 1 to 10 accounting for 87% of queries of this size on the larger Travel dataset. Retrieved products are ranked by decreasing number of satisfied constraints in MCR$^+$, making the approach easy to implement in any recommender system. Moreover, any measure of product suitability can be used as a secondary ranking criterion in MCR$^+$ without affecting the minimal completeness of the retrieval set.

We focused in this paper on MCR$^+$ as an approach to ensuring the minimal completeness and diversity of an initial set of recommended products. As in CDR [McSherry, 2006], additional recommendations can be provided in our approach without increasing the number of initially recommended products by allowing the user to view other products that satisfy the same constraints as an initially recommended product. We expect this to be most beneficial in situations where recommended items are sought in competition with other users (e.g., jobs, rental apartments). Other approaches to generating additional recommendations when none of the products initially retrieved by MCR$^+$ is acceptable to the user will be investigated in our future research.

References

[Branting, 2004] L.K. Branting. Learning feature weights from customer return-set selections. *Knowledge and Information Systems*, 6:188-202, 2004.

[Bridge *et al*., 2005] D. Bridge, M. Göker, L. McGinty, and B. Smyth. Case-based recommender systems. *Knowledge Engineering Review*, 20: 315–320, 2005.

[Burke, 2002] R. Burke. Interactive critiquing for catalog navigation in e-commerce. *Artificial Intelligence Review*, 18:245–267, 2002.

[Coyle and Cunningham, 2004] L. Coyle and P. Cunningham. Improving recommendation ranking by learning personal feature weights. In *Proceedings of the Seventh European Conference on Case-Based Reasoning*, pages 560–572, Madrid, Spain, August/September 2004. Springer.

[Linden *et al*., 1997] G. Linden, S. Hanks, and N. Lesh. Interactive assessment of user preference models: The Automated Travel Assistant. In *Proceedings of the Sixth International Conference on User Modeling*, pages 67–78, Chia Laguna, Sardinia, 1997. Springer Wien.

[McGinty and Smyth, 2002] L. McGinty and B. Smyth. Comparison-based recommendation. In *Proceedings of the Sixth European Conference on Case-Based Reasoning*, pages 575–589, Aberdeen, Scotland, September 2002. Springer.

[McSherry, 2006] D. McSherry. Completeness criteria for retrieval in recommender systems. In *Proceedings of the Eighth European Conference on Case-Based Reasoning*, pages 9–29, Ölüdeniz/Fethiye, Turkey, September 2006. Springer.

[McSherry, 2008] D. McSherry. Minimally complete retrieval in recommender systems. In *Proceedings of the Nineteenth Irish Conference on Artificial Intelligence and Cognitive Science*, pages 83–92, Cork, Ireland, August 2008.

[Pu and Chen, 2008] P. Pu and L. Chen. User-involved preference elicitation for product search and recommender systems. *AI Magazine*, 29:93–103, 2008.

[Smyth, 2007] B. Smyth. Case-based recommendation. In *The Adaptive Web*, pages 342–376. Springer, Heidelberg, 2007.

[Smyth and McClave, 2001] B. Smyth and P. McClave. Similarity vs. diversity. In *Proceedings of the Third International Conference on Case-Based Reasoning*, pages 347–361, Vancouver, BC, July/August 2001. Springer.

[Sperner, 1928] E. Sperner. Ein Satz über Untermengen einer endlichen Menge. *Mathematische Zeitschrift*, 27: 544-548, 1928.

[Viappiani *et al*., 2006] P. Viappiani, B. Faltings, and P. Pu. Preference-based search using example-critiquing with suggestions. *Journal of Artificial Intelligence Research*, 27:465–503, 2006.