Diverse Product Recommendations using an Expressive Language for Case Retrieval

Derek Bridge and Alex Ferguson

Department of Computer Science,
University College, Cork
d.bridge/a.ferguson@cs.ucc.ie

Abstract. We describe Order-Based Retrieval, which is an approach to case retrieval based on the application of partial orders to the case base. We argue that it is well-suited to product recommender applications because, as well as retrieving products that best match customer-specified 'ideal' attribute-values, it also: allows the customer to specify soft constraints; gives a natural semantics and implementation to tweaks; and delivers an inherently diverse result set.

1 Introduction

When CBR has been applied to product recommender systems, similarity is generally the sole retrieval criterion: products whose descriptions in a product catalogue are similar to the customer's 'ideal' product are retrieved and recommended to the customer [10] [11]. But 'pure' similarity-based retrieval may not suffice in this application. In [2], we introduced a new form of retrieval, Order-Based Retrieval (OBR), which we revised and described in more detail in [3]. A major motivation was to allow a query in a recommender system to capture *soft constraints* (e.g. a desired maximum price), as well as 'ideal' values.

Diversity of retrieval may be an additional criterion, again with particular application to product recommender systems [8]. In product recommendation, similarity between the retrieved cases and the customer's 'ideal' values is desirable, but too much similarity *between the retrieved cases themselves* may be less desirable. If the customer is not satisfied with the most similar case in the result set, for example, the chances of him or her being satisfied with an alternative case in the result set are increased if a diverse set is recommended. Diversity is different in nature from similarity: similarity is a property of individual cases in the result set (relative to the query); diversity is a property of the result set as a whole.

There have been a number of attempts to increase retrieval diversity. For example, Smyth & Cotter [8] use a hybrid system, in which similarity-based recommendations are made more diverse by the inclusion of additional recommendations from a collaborative recommender. In other work, Smyth & his co-authors [9] [1] apply an additional selection rule within retrieval to increase result set diversity. We argue in this paper that our new approach, OBR, gives good

result set diversity as a natural consequence of the query language semantics, without recourse to additional mechanisms.

In Section 2, we will summarise OBR's query language. Section 3 introduces some example data that will be used in subsequent sections of the paper. Section 4 illustrates the operation of OBR and argues that it is more expressive than pure similarity-based retrieval. Section 5 report the results of a comparison between OBR and the approach described in [9] & [1].

2 The Order-Based Retrieval Query Language

Order-Based Retrieval is founded on a key observation: the success of similarity-based retrieval rests on its ability to order the cases in the case base. If we call the customer's query q, then case $c_i \in CB$ is lower in the ordering than case $c_j \in CB$ iff $\text{Sim}(c_i, q) < \text{Sim}(c_j, q)$. The numbers that are used to denote the degrees of similarity are less important than the ordering induced on the cases.

Using the relative similarities of cases to a query of 'ideal' values is only one way of ordering a case base. The idea behind OBR is that we can construct orders in other ways too. In product recommendation, the customer will supply a variety of information ('ideal' values, maximum values and minimum values, for example) and we will construct an ordering relation from this information. We will apply this ordering to the case base, retrieve the maxima (or a suitable number of the products at the top of the ordering) and recommend them to the customer.

Accordingly, we have developed a new query language for constructing *partial orders*. Presently, the language comprises six operators. To save space, we present here only the five that we make use of in this paper. Furthermore, the fifth operator listed below (LSPO) is a special case of a more general operator (which we call Prioritisation Ordering, PO). We are only using LSPO in this paper so, again, to save space we do not show the more general definition. The definitions can be found in full in [3].

In the following definitions of the operators, x, y and v may be attribute-values or whole cases. In the way that we list the operators here, their usefulness may not be immediately apparent. But they are exemplified in subsequent sections of this paper. The reader may wish to skim this section on first reading, and then refer back to it as necessary when working through later sections.

Filter-Ordering (FO): Given a unary predicate, p, we can construct an ordering from p as follows:

$$x <_{\text{FO}(p)} y \mathrel{\hat{=}} \neg p(x) \wedge p(y)$$

The definition says that x is lower than y iff y satisfies p but x does not.

Similarity-Ordering (SO): Given a similarity measure, Sim, and an 'ideal' value, v, then x is lower than y iff x's similarity to v is lower than y's similarity to v:

$$x <_{\text{SO}(\text{Sim},v)} y \mathrel{\hat{=}} \text{Sim}(x, v) < \text{Sim}(y, v)$$

About-Ordering (AO): Given a partial order, $<$, and an 'ideal' value v, then a new ordering can be defined by 'breaking the back' of the existing ordering at v [7]. Values to either side of v will be ordered by their distance from v in the original ordering:

$$x <_{\mathrm{AO}(<,v)} y \mathrel{\hat{=}} (x < y \leq v) \vee (x > y \geq v)$$

Cross-Product Ordering (CPO): This operator combines two partial orders, $<_1$ and $<_2$, into a single partial order:

$$x <_{\mathrm{CPO}(<_1,<_2)} y \mathrel{\hat{=}} x \leq_1 y \wedge x \leq_2 y \wedge \neg(x =_1 y \wedge x =_2 y)$$

This is simply the conjunction of the two orders: for x to be less than or equal to y, it must be less than or equal to y in both orders (and, to be strictly less than y, it must also not equal y).

Less Strict Prioritisation Ordering(LSPO): Here $<_1$ takes precedence over $<_2$:

$$x <_{\mathrm{LSPO}(<_1,<_2)} y \mathrel{\hat{=}} x <_1 y \vee (x \not<_1 y \wedge x \not>_1 y \wedge x <_2 y)$$

That is, when $<_1$ judges two values to be equal or incomparable, the 'tie' is broken using $<_2$.

This operator yields a partial order only when applied to certain first argument orders $<_1$. In this paper, we apply it only to orders that result from the application of FO. In [3] we prove that if $<_1$ results from an application of FO, then LSPO yields a partial order.

Many more operators could be given, but these form a good starting point for building OBR Recommender Systems.

3 Example Case Base

We will be using a case base of 794 properties that were available for rent in the London area in Summer 2001.[1] Each case in this case base has six descriptive attributes: the price, the number of bedrooms, the number of bathrooms, the location, whether the property is a flat or a house, and whether the property is furnished or not. (There are a further two attributes used only for case identification: a unique number and an address.) Figure 1 shows these attributes and their types (permissible values). In the case of numeric-valued attributes, the ranges of permissible values are based on values currently in the case base.

In places where we use pure similarity-based retrieval in this paper, we will need similarity measures for each of the six attributes. These attribute-specific similarity measures are given in Figure 2. For *location*, only a fragment of an invented similarity measure is shown.[2] *price*, *bdrms* and *bthrms* would use $\mathrm{Sim}_{\mathbb{N}}$ with different values for the *range*: 8373 for *price*, 7 for *bdrms* and 6 for *bthrms*. Sim_{Eq} can be used as the similarity measure for *propType* and *furnished*.

[1] This case base is accessible at: `http://www.cs.ucc.ie/~dgb/research/obr.html`

[2] The similarity measure we use in our software is a bit simpler; it is based on London postal codes.

Attributes with ordered types

Attribute	Type
price	127-8500
bdrms	0-7
bthrms	1-7

Attributes with unordered types

Attribute	Type
location	{*Clapham*, *Chelsea*,...}
propType	{*Flat*, *House*}
furnished	{*Yes*, *No*}

Fig. 1. Attributes and their Types

$$\begin{aligned}
&\mathrm{Sim}_{location}(Battersea, Battersea) = 1.0\\
&\mathrm{Sim}_{location}(Battersea, Clapham) = 0.7\\
&\mathrm{Sim}_{location}(Battersea, Chelsea) = 0.5\\
&\mathrm{Sim}_{location}(Battersea, Hounslow) = 0.3\\
&\mathrm{Sim}_{location}(Battersea, Richmond) = 0.0
\end{aligned}$$

$$\mathrm{Sim}_{\mathbb{N}}(x,y) \;\hat{=}\; 1 - \frac{\mathrm{abs}(x-y)}{range}$$

$$\mathrm{Sim}_{Eq}(x,y) \;\hat{=}\; \begin{cases} 1 \text{ if } x = y \\ 0 \text{ otherwise} \end{cases}$$

Fig. 2. Attribute-Specific Similarity Measures

We distinguish *ordered types* from *unordered types*. Attributes *price*, *bdrms* and *bthrms* have ordered types. Their permissible values have an obvious ordering defined on them (the usual numeric-less-than relation). (Although these examples are all numeric, non-numeric ordered types are also possible [3].) For these attributes, OBR does not require similarity measures.

Attributes *location*, *propType* and *furnished* all have unordered types. Their permissible values have no obvious ordering that would be relevant to product recommendation. For these attributes to participate in either kind of retrieval, we will need the similarity measures in Figure 2.

In our experiments in Section 5, we use all 794 cases and all six attributes. But our examples in Section 4 are rendered more intelligible by confining our attention to just three attributes: *price*, *bdrms* and *location*. Furthermore, in Section 4's simple examples, we use some invented cases, which are shown in Figure 3. (These cases have been invented for the way they demonstrate different aspects of the behaviour of the operators in our query language.)

4 Retrieval Examples

In this section, we explain and exemplify some of Order-Based Retrieval's operators. We will also show that OBR has greater expressiveness than pure similarity-based retrieval by handling some examples that contain soft constraints.

4.1 Pure Similarity-Based Retrieval

We start with an example in which the customer specifies only 'ideal' values. Suppose the customer desires a two bedroom property in Battersea. Suppose also that these two requirements are equally important to the customer.

identifier	price	bdrms	location
A	325	3	*Clapham*
B	330	2	*Hounslow*
C	400	2	*Chelsea*
D	400	3	*Hounslow*
E	500	3	*Chelsea*
F	550	1	*Richmond*
G	600	1	*Richmond*
H	600	4	*Clapham*

Fig. 3. Example Property Case Base

Let q be the customer's query, let $\mathrm{Attr}(q)$ be the attributes mentioned in this query (in this case, *bdrms* and *location*) and let π_a be a projection function that obtains the value of attribute a from a case or a query. Then, pure similarity-based retrieval would retrieve cases c from the case base using a similarity measure such as the following:

$$\mathrm{Sim}(c,q) \doteq \frac{\sum_{a\in\mathrm{Attr}(q)}(w_a \times \mathrm{Sim}_a(\pi_a(c),\pi_a(q)))}{\sum_{a\in\mathrm{Attr}(q)} w_a} \tag{1}$$

This is a weighted average of attribute-specific similarity measures. In our experiments (Section 5), we are going to assume that all requirements are equally important, and so all weights will be one.

If we evaluate the similarity of each case in Figure 3 to the query, we find that the cases are ordered as follows: A (0.78), C (0.75), H (0.71), E (0.68), B (0.65), D (0.58) and F & G (0.43 each). Conventionally, only the best k of these cases would be recommended to the customer.

4.2 Order-Based Retrieval

We do not use Equation 1 to encode our example query using OBR. In particular, for ordered types, we do not need the attribute-specific similarity measures.

The *bdrms* attribute has an ordered type and we use its ordering, rather than using a similarity measure. The customer wants two bedrooms, so we construct a new ordering from $<$ using the AO operator. The ordering $<_{\mathrm{AO}(<,2)}$ is shown as Figure 4. (The diagram is abbreviated for values greater than 5.)

The *location* attribute has an unordered type so in this case we do use its similarity measure, $\mathrm{Sim}_{location}$, from Figure 2. But we construct an order from it using the SO operator. The order $<_{\mathrm{SO}(\mathrm{Sim}_{location},Battersea)}$ applied to some sample data is shown as Figure 5. To combine the customer's requirements equally, we use the CPO operator. The final query expression is:

$$<_{\mathrm{CPO}(<_{\mathrm{AO}(<,2)},<_{\mathrm{SO}(\mathrm{Sim}_{location},Battersea)})}$$

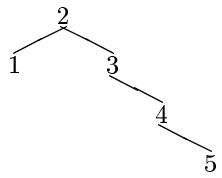

Fig. 4. $<_{\mathrm{AO}(<,2)}$

Fig. 5. $<_{\mathrm{SO}(\mathrm{Sim}_{location},Battersea)}$

Diagrams such as those in Figures 4 and 5 show the effects of the ordering relations that we have constructed when they are applied to a sample of representative data values. OBR software does not have to build data structures corresponding to these diagrams. It uses the query expressions directly to compare pairs of cases in the case base. An algorithm for finding the maxima of the case base for a given query expression is shown as Figure 6. Figure 7 shows how the case base in Figure 3 is ordered by our query expression.

As can be seen from the diagrams, we are dealing with partial orders. In Figure 7, case A, for example, is neither lower in the ordering nor higher in the ordering than case C: they are incomparable. This is because case A comes closer to the query on *location*, but case C does better on *bdrms*. This illustrates that, in our partially-ordered approach, a set of candidates is often retrieved where each is to be preferred on a different respect. This leads naturally to a diverse set, with the size of the set providing an indication of the number of distinct 'directions' in which alternatives lie in the case base. We discuss this inherent diversity in Section 5.

It follows too that the maxima is a set: the set of cases that are equal or incomparable to each other and bettered by no other cases. In the example, the maxima is the set $\{A, C\}$. If more cases are to be recommended to the customer, we can display the second 'rank' of cases: $\{H, E, B\}$. This is the maxima when A and C are taken out of the picture. The next rank is $\{D\}$ and the final rank is $\{F, G\}$. This is a similar outcome to that obtained for pure similarity-based retrieval in Section 4.1, although in general there can be considerable differences. Our experiments in Section 5 reveal some of these differences.

One objective comparison, however, is that CPO straightforwardly treats the two requirements in the customer's query equally. In weighted averages, setting

```
maxima := {}
for each c ∈ CB
   shouldInsert = true
   for each case in maxima, m
      if c < m
         shouldInsert = false
         break
      else if m < c
         remove m from maxima
      end if
   end for
   if shouldInsert
      insert c into maxima
   end if
end for
```

Fig. 6. Naïve query evaluation algorithm applying $<$ to CB

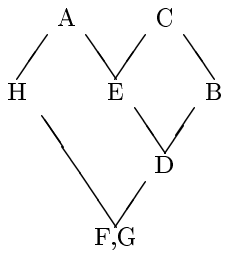

Fig. 7. Ordering the Case Base using the Query

the weights to one is not sufficient for treating requirements equally: the values returned by the similarity measures are sensitive to other factors [4] [3]. For example, suppose we decided that the range of permissible values for *bdrms* was not 7 but 11. Cases C & H would swap places in the similarity ranking that we gave in Section 4.1. This new normalisation factor reweights the similarity measure so that case H's similarity to the query changes from 0.71 to 0.76. Changes to normalisations cannot affect the behaviour of CPO.[3]

4.3 Soft Constraints

Suppose customer Ann wants a 2 bedroom property in Battersea as in the previous two subsections, but the maximum she is prepared to spend is £400. Cus-

[3] Of course, the main use of weighting is to make one requirement more important than another, although here again the interaction with normalisation can make it hard to achieve definite effects. In OBR, the PO operator and its special cases such as LSPO (Section 2) offer a variety of ways of doing the same thing, again without concerns about normalisation [3].

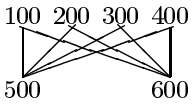

Fig. 8. $<_{\mathrm{FO}(\lambda x[\pi_{price}(x) \leq 400])}$

tomer Ben also wants a 2 bedroom property in Battersea but he can only afford £200. In product recommender systems that use pure similarity-based retrieval, customers can specify only 'ideal' values; they cannot specify constraints, such as a maximum price.

One solution to this, of course, is to use a hybrid system which first filters the case base, eliminating products that do not satisfy the constraints. A result set is retrieved from the remaining products using similarity-based retrieval. For Ann's query, cases E, F, G and H would be eliminated, and she would be recommended the remaining cases in the order A, C, B, D.

The problem is that there is now a risk that the customer's requirements are so demanding that all products are eliminated and so none is recommended to the customer. For Ben's query, for example, no products are recommended at all. This is the problem that motivated the use of similarity-based retrieval in recommender systems; traditional database retrieval, using exact matching, can give empty result sets [10] [11].

In OBR, we would take Ann's maximum value and construct a unary predicate from it: $\lambda x[\pi_{price}(x) \leq 400]$. We do not filter away products using this predicate. Instead, we construct an order from it using the FO operator: $<_{\mathrm{FO}(\lambda x[\pi_{price}(x) \leq 400])}$. This says that values that satisfy the predicate are higher in the ordering than ones that do not. We show this in Figure 8 for a handful of different prices.

We can combine this with the rest of Ann's requirements. It is likely that a constraint (even a soft one) should be treated as more important than other requirements, so we will combine using a form of prioritisation. This is a case where we can use LSPO. The final query expressions is therefore

$$<_{\mathrm{LSPO}(<_{\mathrm{FO}(\lambda x[\pi_{price}(x) \leq 400])}, <_{\mathrm{CPO}(<_{\mathrm{AO}(<,2)}, <_{\mathrm{SO}(\mathrm{Sim}_{location}, Battersea)})})}$$

The set of maxima is now $\{A, C\}$ and the subsequent ranks are $\{B\}$, $\{D\}$, $\{H, E\}$ and $\{F, G\}$. Cases E, F, G and H can still be recommended. But they are now lower in the ordering than products whose prices fall into Ann's budget.

The effect is more stark for Ben. His query is the same as Ann's, except the predicate that he feeds into the FO operator uses a value of £200 instead of £400. No products satisfy this predicate. But, since no products are eliminated, all products remain eligible for recommendation to Ben and will be recommended on the basis of how well they satisfy his other criteria (bedrooms and location).[4]

[4] Ben might prefer it if the recommended products exceed his budget to the smallest extent possible. A more complex query can be used for this.

4.4 Tweaks

The Entrée system offers a natural form of query refinement [6]. The specific mechanism used for this by Entrée-style systems is known as the *tweak* or *critique*. The customer selects from the result set a product that is at least partly acceptable. The next query will, at heart, use similarity-based retrieval to find products that are similar to this selected product while satisfying the tweak (products that are *"like this but..."*). Interaction becomes a process of *navigation* through the product space based on critiques of selected products.

In [3], we argue that OBR gives a better semantics to tweaks than other approaches. Very simply, tweaks are encoded as filters but then converted into orders using Filter-Ordering, FO. As we saw earlier, this turns a hard constraint into a soft constraint. The system will prefer products that satisfy the filter but will not eliminate products that do not satisfy the filter. For example, suppose the customer likes the look of property E in Figure 3 but desires something cheaper. Since E's price is £500, we encode the tweak as $<_{\mathrm{FO}(\lambda x[\pi_{price}(x)<500])}$. This tweak would be prioritised over the rest of the query, which is based on trying to match E's location and number of bedrooms:

$$<_{\mathrm{LSPO}(<_{\mathrm{FO}(\lambda x[\pi_{price}(x)<500])},<_{\mathrm{CPO}(<_{\mathrm{AO}(<,3)},<_{\mathrm{SO}(\mathrm{Sim}_{location},Chelsea)})})}$$

5 Diversity

As mentioned in Section 1, diversity is a property of a result set as a whole. Approaches to increasing diversity have incorporated a selection criterion additional to similarity [8] [9] & [1]. However, we believe that some of the query operators in Order-Based Retrieval have a semantics which, in the absence of exact matches, automatically selects diverse products.

For example, consider the AO operator, and an example of its use, Figure 4. One can easily imagine an alternative operator, which we will refer to as the Distance-Ordering operator, in which a distance function is used so that items are ordered solely by their distance from the 'ideal' value, v. With Distance-Ordering, one and three bedroom properties would appear *equal* second in the ordering, whereas in AO they are incomparable.

The reason AO can deliver a more diverse maxima can be explained by thinking further about the example. Suppose a case base of properties contains no two or three bedroom properties, but does contain one and four bedroom properties; and suppose that the query consists only of a requirement for a two bedroom property. If we apply Distance-Ordering and return only the maxima, the customer is shown only one bedroom properties. But, if we apply AO and return only the maxima, the customer is shown one and four bedroom properties. Thus, with AO, customers are shown properties to either side of their 'ideal'.

A more fundamental example is the CPO operator. Suppose we combine two orders $<_1$ and $<_2$ using CPO. Suppose case c_1 is better than case c_2 and is indeed the best case with respect to order $<_1$ but that case c_2 is better than case c_1 and indeed is the best case with respect to order $<_2$. In the ordering

given by CPO, they will be incomparable and so, in some sense, they will be as good as each other. If one is in the maxima, the other will also be in the maxima. Similarity-based retrieval will only find these two cases to be indistinguishably good if they have exactly the same degree of similarity to the query, which is not very likely and is hard to design for. We saw an example of exactly this in Figure 7 where cases A and C were both in the maxima because case A does best on *location* and C does best on *bdrms* (and no case does better on both requirements). Hence, as a means of combining requirements, CPO can give inherently more diverse results than weighted average, even if it is used to combine conventional similarity measures.

We now report the results of an investigation into the diversity of result sets.

5.1 Order-Based Retrieval Experiments

We take each of the 794 properties in the whole case base in turn and use each to construct a query. We use all six of its attributes. For the attributes that have ordered types (*price*, *bdrms* and *bthrms*), we take the value from the case and construct an order using AO, e.g. $<_{\mathrm{AO}(<,2)}$. For the attributes that have unordered types (*location*, *propType* and *furnished*), we take the value from the case and construct an ordering using the similarity measure and SO, e.g. $<_{\mathrm{SO}(\mathrm{Sim}_{location},Battersea)}$. We combine all six order expressions using CPO.

For a given query, the remaining 793 cases are the case base, and we retrieve the maxima from the case base for that query. As we explained previously, the maxima is a set and may be different in size and contents for each query. For some queries, the maxima was size 1, for others it was 2, and so on up to 39. The nice thing about judging the quality of only the maxima (rather than some best k) is that the maxima is the set that (according to OBR) best matches the query; its size and contents are not open to arbitrary manipulation.

In Figure 9, for the numbers 1 to 39, we show the number of queries (out of the 794) whose maxima were of that size.

5.2 Bounded Greedy Selection Experiments

We compare OBR with an algorithm from [9] & [1]. Of several algorithms investigated in [9] & [1], the Bounded Greedy Selection algorithm (henceforth BG) gave the most improved retrieval quality while remaining cost-effective. BG retrieves bk cases using pure similarity-based retrieval. From these bk cases, it selects k cases, which will be the result set to be displayed to the customer. It selects the k cases based on their quality relative to the cases selected so far, see Figure 10.

Given query q, the quality of case c relative to the result set so far, R, is defined as follows [1]:

$$\mathit{Quality}(c, q, R) \mathrel{\hat{=}} (1 - \alpha) \times \mathrm{Sim}(c, q) + \alpha \times \mathit{RelDiversity}(c, R) \qquad (2)$$

α is a factor that allows the importance of similarity and diversity to be changed. In line with [9] & [1], we use $\alpha = 0.5$. In Equation 2, similarity, Sim, is measured

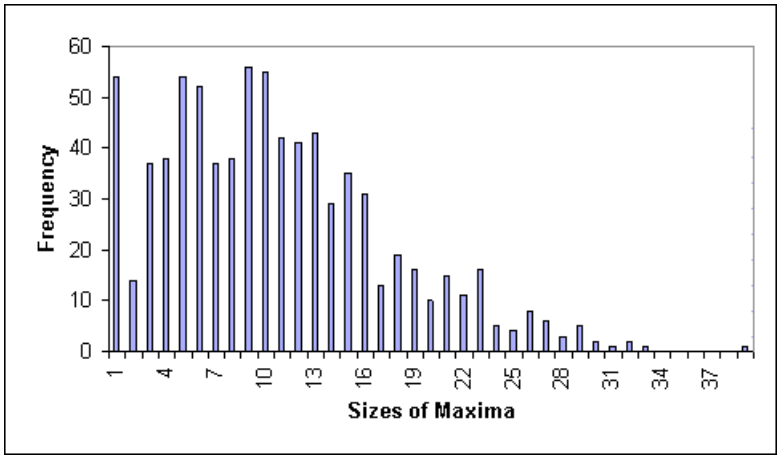

Fig. 9. Size of Maxima for OBR

```
Candidates := bk cases in CB that are most similar to q
R := {}
for i = 1 to k
   best := the c ∈ Candidates for which Quality(c, q, R) is highest
   insert best into R
   remove best from Candidates
end for
return R
```

Fig. 10. The Bounded Greedy (BG) Selection Algorithm

as per Equation 1. Diversity relative to the result set so far is defined as follows:

$$RelDiv(c,R) \hat{=} \begin{cases} 1 & \text{if } R = \{\} \\ \frac{\sum_{i=1\ldots|R|}(1-\text{Sim}(c,r_i))}{|R|} & \text{otherwise} \end{cases} \quad (3)$$

In the experiments, we use $b = 2$. The values we use for k are the sizes of the maxima found using OBR. For example, if some case c is taken to be the query and in OBR the maxima is of size 5, then when we do our experiments using BG, $k = 5$.

5.3 Evaluation Criteria

Having retrieved a result set for a given query q using OBR or using BG, we must evaluate the result set. We want to know whether the result set contains cases that are close to q, but we also want the set to be diverse.

Following [9] & [1], we will measure the average similarity of the result set to query q:

$$AvSim(R,q) \hat{=} \frac{\sum_{i=1...|R|} \mathrm{Sim}(r_i,q)}{|\,R\,|} \tag{4}$$

There is a sense in which this is unreasonably helpful to BG. BG selects its cases in part using Sim and now we are evaluating its selection using Sim; as in [9] & [1], we are measuring what we have already partly optimised for. Ideally a different measure (some kind of measure of how satisfied the customer is from a similarity point of view) should be used. But this is not available.

This advantage is given only to BG. OBR's result set will be the set of maxima; they are therefore, in some sense, the best matches to the query, but Sim will not necessarily judge them to be so. In particular, as highlighted in Section 4.2, due to the effects of normalisation, Sim, unlike OBR, is not treating all requirements as equally important.

We will also measure the diversity of the result set. The measure used here is the average dissimilarity between all pairs of cases in the result set [9] & [1] (taking the diversity of a singleton set to be 1):

$$Div(R) \hat{=} \begin{cases} 1 & \text{if } |\,R\,| = 1 \\ \frac{\sum_{i=1...|R|} \sum_{j=i...|R|} (1 - \mathrm{Sim}(r_i,r_j))}{\frac{n}{2}(n-1)} & \text{otherwise} \end{cases} \tag{5}$$

For similar reasons, there is again an experimental bias towards BG.

5.4 Discussion of Results

The results are shown in Figures 11 and 12. For each maxima size, we take the *AvSim* and *Div* values of the queries that gave rise to maxima of that size and plot the average values.[5]

We also plot on the graphs some figures for comparison. In Figure 11, we plot the optimal average similarity. For query q, this is obtained by taking the k cases that are most similar (using Sim) to q (where, again, k is the size of OBR's maxima for q) and computing their average similarity. This is, of course, the familiar k-NN retrieval algorithm.

In Figure 12, we plot the diversity for k-NN but this time this is not the optimal value. So we also plot an estimated optimal diversity (where 'optimal' here simply means 'highest'). Our optimal diversity figure is computed differently from the way that Bradley & Smyth compute theirs [1]. They take the bk cases that are most similar to q, they form all subsets of size k from these cases, they compute the diversity of these subsets and they report the highest. We found that the diversity figure given by OBR frequently outperformed Bradley & Smyth's optimum. Their optimum is, of course, an optimum relative only to the bk most

[5] Some queries gave rise to maxima of sizes 1-33. No queries gave rise to maxima of sizes 34-38. One query gave rise to a maxima of size 39. To avoid a 'gap' in the graphs, the *AvSim* and *Div* for this final query are not included in Figures 11 and 12. These unplotted values do not contradict the rest of the graphs.

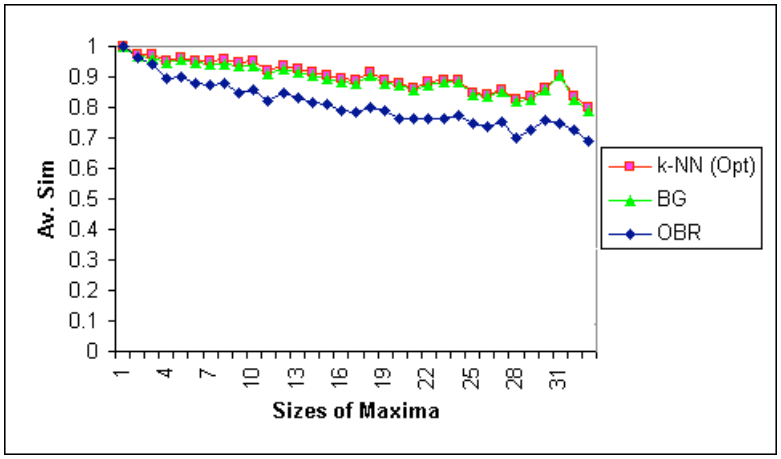

Fig. 11. Average Similarity

similar cases that they have retrieved for query q. It is not an optimum with respect to the case base as a whole. Since our diversity performance was so often higher than their optimum, we thought it better to plot a truer optimum in Figure 12. Ideally, we would obtain this by taking all subsets of size k from the entire case base, computing their diversity and plotting the highest. This is computationally very expensive, so we estimated it with a greedy algorithm. We take the most dissimilar pair of cases, then we add the next case that will most increase diversity. We keep doing this until we have a set of size k. Its diversity is our estimated optimum.

Figure 11 shows that BG generally outperforms OBR for average similarity and does extremely well compared to the optimum (k-NN). Indeed, on average over the 794 queries, the difference between the optimum and BG average similarities is only 0.009; the differences are so small that the two lines virtually coincide on the graph. This may be explained by the fact that in this case base there will often be regions comprising many properties having much the same location, much the same price and much the same facilities. When BG retrieves bk candidate cases ($b = 2$), these regions are often of a sufficient size that BG suffers only a small decrease in average similarity to the query.

The difference between the optimum and OBR is, on average, 0.081; the figures for OBR are, on average, 0.073 lower than those for BG. These results are perhaps unsurprising given the way the experiments favour BG.

As can be seen in Figure 12, despite the experimental bias, OBR outperforms BG quite considerably when it comes to diversity. The difference between the estimated optimum diversity and OBR's is, on average, 0.386, whereas for BG is it 0.479. OBR's diversity is, on average, 0.12 higher than k-NNs, whereas for BG it is 0.02. The difference between OBR's diversity and BG's is, on average, 0.093. The regions of similar cases that keep BG's average similarity figures high

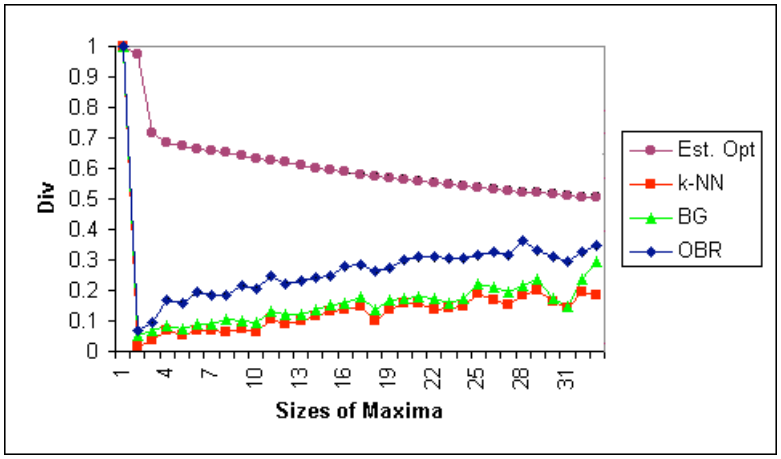

Fig. 12. Diversity

make it harder for BG to increase diversity; it does not decisively outperform k-NN. This is especially interesting because OBR is getting its diversity as a natural consequence of the way the operators are defined, rather than through some additional selection criterion.

Which of the two algorithms is better will depend, to some extent, on the trade-off to be made, in a particular context, between similarity to the query and diversity. No satisfactory way yet exists to measure this trade-off.

Efficiency will also be relevant. In the worst case, the OBR algorithm in Figure 6 carries out $n(n-1)/2$ comparisons. In other work, in the context of a distinct system concerned only with similarity-based retrieval [5], we describe a refinement of this algorithm which gives an improved complexity behaviour: the number of comparisons is bounded above by nm, where m is a constant for each given query, equal to the maximum 'width' of the partial order denoted by the query, e.g., in Figure 4, $m = 2$. In [9] the cost of BG is given as $n + \frac{k^3(b-1)}{4}$ comparisons. In practice, the performance of both algorithms is acceptable.

6 Conclusion

In conclusion, we believe that Order-Based Retrieval is a new, promising approach. It is more expressive than pure similarity-based retrieval; it gives a natural implementation to tweaks; and it can give diverse result sets.

There is much future work to be done, and we mention one major issue here. In OBR, the maxima is the set of best cases; its size is not open to manipulation. But there may be a requirement in a recommender system to present the customer with a minimum or maximum number of cases. If OBR's maxima is too small, subsequent 'ranks' can be displayed. But it will take further research to

investigate ways of proceeding when the set of retrieved cases is too large. Options include: incorporating community-wide or customer-specific default orders to narrow the set; asking the customer a dynamically-chosen question to elicit a further customer requirement to narrow the set; or the use of a presentation strategy that requires less than the full set of cases to be displayed on the screen.

References

1. Bradley, K. & B. Smyth: Improving Recommendation Diversity, In D. O'Donoghue (ed.), *Procs. of the Twelfth Irish Conference on Artificial Intelligence and Cognitive Science*, pp.85–94, 2001.
2. Bridge, D.: Product Recommendation Systems: A New Direction, In R. Weber & C.G. von Wangenheim (eds.), *Procs. of the Workshop Programme at the Fourth International Conference on Case-Based Reasoning*, pp.79–86, 2001.
3. Bridge, D. & A. Ferguson: An Expressive Query Language for Product Recommender Systems, *Artificial Intelligence Review*, to appear, 2002.
4. Ferguson,A. & Bridge,D.G.: Partial Orders and Indifference Relations: Being Purposefully Vague in Case-Based Retrieval, in Blanzieri,E. & Portinale,L. (eds.), *Advances in Case-Based Reasoning (Procs. of the 5th European Workshop on Case-Based Reasoning)*, LNAI 1898, pp.74–85, Springer, 2000
5. Ferguson, A. & D. Bridge: Weight Intervals: Conservatively adding quantified uncertainty to similarity, In D. O'Donoghue (ed.), *Procs. of the Twelfth Irish Conference on Artificial Intelligence & Cognitive Science*, pp.75–84, 2001.
6. Hammond, K.J., R. Burke & K. Schmitt: Case Based Approach to Knowledge Navigation, In D.B. Leake (ed.), *Case-Based Reasoning — Experiences, Lessons and Future Directions*, pp.125–136, MIT Press, 1996.
7. Osborne, H.R. & D.G. Bridge: A Case Base Similarity Framework, In I. Smith & B. Faltings (eds.), *Advances in Case-Based Reasoning (Procs. of the Third European Workshop on Case-Based Reasoning)*, Lecture Notes in Artificial Intelligence 1168, pp.309–323, Springer, 1996.
8. Smyth, B. & P. Cotter: Surfing the Digital Wave: Generating Personalised TV Listings Using Collaborative, Case-Based Recommendation, In K.D. Althoff, R. Bergmann & L.K. Branting (eds.), *Case-Based Reasoning Research and Development (Procs. of the Third International Conference on Case-Based Reasoning)*, Lecture Notes in Artificial Intelligence 1650, pp.5612–571, Springer, 1999.
9. Smyth, B. & P. McClave: Similarity vs. Diversity, In D.W. Aha & I. Watson (eds.), *Case-Based Reasoning Research and Development (Procs. of the Fourth International Conference on Case-Based Reasoning)*, Lecture Notes in Artificial Intelligence 2080, pp.347–361, Springer, 2001.
10. Vollrath, I., W. Wilke & R. Bergmann: Case-Based Reasoning Support for Online Catalog Sales, *IEEE Internet Computing*, vol.2(4), pp.45–54, 1998.
11. Wilke, W., M. Lenz & S. Wess: Intelligent Sales Support with CBR, In Lenz, M., B. Bartsch-Spörl, H.-D. Burkhard & S. Wess (eds), *Case-Based Reasoning Technology: From Foundations to Applications*, Lecture Notes in Artificial Intelligence 1400, pp.91–113 Springer, 1998.