# An Analysis of Critique Diversity in Case-Based Recommendation *

**Kevin McCarthy** and **James Reilly** and **Lorraine McGinty** and **Barry Smyth**
Adaptive Information Cluster
Department of Computer Science
University College Dublin (UCD)
Belfield, Dublin 4,
Ireland.

## Abstract

Critiquing is a well-known form of user feedback in case-based recommender systems. A *critique* encodes the users preference in relation to a particular feature. For example, in a digital camera recommender a user may be allowed to indicate whether they are interested in cameras with a lower resolution than the one currently presented; so *'lower resolution'* is an example of a critique over the *resolution* feature. Recent research demonstrates how the dynamic generation of *compound critiques* – critiques that operate over multiple features – can deliver significant performance improvements. However user-studies highlight diversity problems that arise during critique generation; for example, one compound critique might constrain *resolution, memory* and *zoom* while another might constrain *resolution, memory* and *price*. In this paper we describe how critique diversity can be improved and demonstrate that this can lead to significant usability and performance benefits.

## Introduction

Recommender systems are emerging as an important technology when it comes to developing e-commerce services that help users to understand and navigate complex product spaces. For example, conversational recommender systems engage users in extended dialog with a view to better understanding the user's product needs and preferences through a series of recommendation cycles. During each cycle the user is typically presented with a suggestion/number of suggestions (for examples see (Cunningham, Doyle, & Loughrey 2003; Burke, Hammond, & Young 1996; Smyth & McGinty 2003; Schafer, Konstan, & Riedl 2001; Shimazu 2001)) and is invited to provide feedback in order to better focus the recommender in the right region of the product space. The appropriate analogy is one where the recommender system is seen as playing the role of the smart sales assistant who makes good suggestions to the user while listening and understanding their comments in order to improve the next batch of suggestions. A good sales assistant should be able to present the user with a satisfactory product in a reasonable amount of time. Likewise a good recommender system should present the user with an acceptable suggestion within a limited number of cycles.

The success of such conversational recommender systems depends critically on the quality and type of feedback that is provided by the user and many different types of recommender systems can be distinguished by the type of feedback that they support; examples include value elicitation, ratings-based feedback, preference-based feedback (see (Smyth & McGinty 2003) for a comparison of these). We are especially interested in a form of feedback called *critiquing* (Burke, Hammond, & Young 1996; 1997; Burke 2002; Faltings *et al.* 2004), where a user indicates a directional preference over a feature of a recommendation. For example, in a digital camera recommender, a user might say that they are interested in a camera with *more memory* than the currently recommended camera; in this case *more memory* is a critique over the *memory* feature.

The traditional critiquing approach has tended to focus on the use of so-called *unit critiques*, which operate to constrain *individual* features. Accordingly, during each recommendation cycle a user can constrain only a single feature at a time, which can limit progress through the product space. Recent research has shown how the standard form of critiquing—where the user can specify a single critique per recommendation cycle (so-called *unit critiques*)—can be extended to accommodate critiques that operate over multiple features (so-called *compound critiques*) (McCarthy *et al.* 2004; Reilly *et al.* 2004). For example, a compound critique in the digital camera domain might offer the user the ability to look for *more memory, greater zoom, and higher resolution*, thereby constrain three features within a given cycle. Importantly, it is possible to automatically identify useful compound critiques dynamically during each recommendation cycle. This kind of *dynamic critiquing* approach can lead to significant reductions in recommendation session length to offer users a much improved recommendation service compared to standard critiquing.

In this paper we build on this related work with a view to solving an important problem that has become apparent as a result of a number of live-user trials. The problem relates to the type of compound critiques that tend to be generated during many recommendation cycles. Real user evaluations have shown that sometimes the critiques that are presented

*This material is based on works supported by Science Foundation Ireland under Grant No. 03/IN.3/I361.

to a user can lack diversity; in other words, they tend to constrain similar feature groups. For example, we might find that one of the compound critiques that is presented to the user during the current session allows the user to opt for a camera with a *higher resolution, less memory, greater price* while another might offer cameras with *higher resolution, less memory, greater zoom*. The point is that both of these critiques constrain two of the same three features thus limiting the scope of the feedback that is on offer to the user. We believe that offering the user a more diverse set of critiques will improve the usability and applicability of the compound critiques so that users are more likely to select these critiques in favour of the standard unit critiques. We present an algorithm for generating compound critiques with a given degree of diversity and we evaluate the ability of these critiques to deliver efficient recommendation sessions.

## Background

In this section, we discuss the standard *unit*-critiquing approach as it relates to conversational case-based recommender systems, before describing how *static* compound critiques can be presented to users as feedback options. The later (and main) focus of this section, however, describes how compound critiques are generated *on-the-fly* by the dynamic critiquing approach.

### A Traditional Approach to Critiquing

The origins of critiquing as a form of feedback can be traced back to the work of Burke et al. (Burke, Hammond, & Young 1996; 1997; Burke 2002) and the FindMe approach to recommender systems development. The work was originally motivated by a form of feedback that was simple for users to use and that still provided sufficient guidance to the recommender system. The Entree recommender (Burke, Hammond, & Young 1996) demonstrated the power of the critiquing approach by helping users to locate restaurants using a set of fixed critiques over features such as *style, cuisine, price* etc. For example, the user may request another restaurant that is *cheaper* or *more formal*, by critiquing its *price* and *style* features, respectively.

The above critiques are all examples of what we call unit critiques. They express preferences over single features; *cheaper* critiques a *price* feature, *more formal* critiques a *style* feature, for example. This ultimately limits the ability of the recommender to make progress through a product space because constraining a single feature per cycle will tend to eliminate only a few cases at a time. Constraining multiple features in a single cycle offers some obvious attractions, leading to the idea of compound critiques that operate over multiple features simultaneously. This idea of compound critiques is not novel; the seminal work of (Burke, Hammond, & Young 1996) refers to critiques for manipulating multiple features. They give the example of the *sportier* critique, in a car recommender, which operates over a number of different car features; *engine size, acceleration and price* are all increased. Similarly we might use a *more professional* compound critique in a digital camera recommender to simultaneously manipulate *resolution, zoom, and memory* features, for example.

In the past when compound critiques have been used they have been hard-coded by the system designer so that the user is presented with a fixed set of compound critiques in each recommendation cycle. These compound critiques may, or may not, be relevant depending on the cases that remain at a given point in time. For instance, in the context of the example above the *more professional* critique might continue to be presented as an option to the user despite the fact that the user may have already seen and declined all cameras from this region of the product space. Far more preferable is the ability to selectively generate such compound critiques as are appropriate given the region of the product space that the user appears to be interested in. Recently such an approach has been proposed (McCarthy *et al.* 2004; Reilly *et al.* 2004) where compound critiques are generated on-the-fly, for each recommendation cycle. In the next section we describe this approach.

### Dynamic Critiquing

A simplified version of the dynamic critiquing algorithm is shown in Figure 1. From a user's perspective dynamic critiquing operates in a manner that is similar to more conventional forms of critiquing. That is, each recommendation session is initiated by an initial user query; each cycle presents the user with a set of recommendations and feedback options (in this case both unit and compound critiques); the user is afforded the opportunity to accept a suggestion or to apply a critique to guide the next cycle. However, the manner in which compound critiques are identified and selected for presentation to the user is novel. First of all, unlike more traditional, static forms of critiquing, the compound critiques are generated on-the-fly as part of each recommendation cycle and with references to the cases that are remaining at that point. Second, a subset of the generated critiques are selected and presented to the user.

Dynamically generating and selecting compound critiques to present to the user is covered by lines 17-21 of the given algorithm and refers to a three-step process: (1) the generation of *critique patterns*, (2) the mining of compound critiques, and (3) the grading of compound critiques. To avoid unnecessary confusion these steps are not expanded in the algorithm provided, but instead are described next.

**From Cases to Critique Patterns** - In order to discover a useful set of compound critiques from a set of product cases we need to first understand how these cases relate to the current recommendation. In other words we need to understand their basic feature differences so that we may then look for frequently occurring sets of differences to act as our compound critiques. The first step is to generate a set of so-called *critique patterns* from the remaining cases. Each remaining case is compared to the current recommended case and the relative feature differences make up the critique pattern.

Figure 2 illustrates what happens in the **GenerateCP** method (line 18 of Figure 1) with the aid of an example from the digital cameras (DC) domain. It shows the current case that has been selected for recommendation to the user as part of the current cycle, and a random case, $c$, from the case-base. The current case describes a Canon digital cam-

era with 5.1M Pixel resolution, 3X optical zoom and 32Mb of storage for 443 Euro. Case $c$ describes a Sony camera with 5.0 M Pixel Resolution, 4X optical zoom and 16Mb of storage for 455 Euro. The resulting critique pattern reflects how case $c$ differs from current case in terms of individual feature critiques. For example, the critique pattern shown includes a "$<$" critique for Storage Included— we will refer to this as $[StorageIncluded <]$—because the comparison case has a less memory storage than the current recommended case. Similarly, the pattern includes the critique $[Price >]$ because the comparison case is more expensive than the current case.

**Generating Compound Critiques** - To identify compound critiques we must look for recurring patterns within a set of critique patterns. Each product case has a number of features and each numeric feature can have a "$<$" or a "$>$" critique and each nominal feature can have a "$=$" or a "$! =$" critique, so there are $2n$ possible critiques in a case-base where the cases are made up of $n$ individual features. The Apriori algorithm (R. Agrawal & Verkamo. 1996) used to characterise recurring item-subsets as association rules of the form $A \rightarrow B$: from the presence of a certain set of critiques (A) one can infer the presence of certain other critiques (B). For example, when buying a Digital Camera you might expect to pay more money for more Resolution, so there is a relationship between these two critiques: $[Resolution >]$ infers $[Price >]$. This part of the algorithm returns compound critiques of the form $\{[Resolution >], [Price >]\}$ along with a measure of their *support*. The support value refers to the percentage of critique patterns for which the compound critique is valid. During each recommendation cycle Apriori is used to generate a collection of compound critiques (frequent itemsets over the critique patterns, CPs), see line 19 of Figure 1. Then, during the next step, a subset of the *best* of these compound critiques is chosen for presentation to the user to complement the standard unit critiques.

**Grading & Selecting Compound Critiques** - From a user interface and availability viewpoint it is not practical to present large numbers of different compound critiques, as feedback options, to the user during each cycle. However, large numbers of compound critiques are likely to be discovered. A filtering strategy is needed so that the most useful critiques can be selected, say the top 5, for presentation purposes. One option is to use the support value of the critique as the basis for filtering. Related work (Reilly *et al.* 2004) has evaluated many different strategies with respect to their ability to improve recommendation performance, finding that the so-called *low-support* (LS) strategy offers superior benefits, as they are capable of eliminating many cases. In the algorithm (Figure 1) the compound critiques are ranked by **RateCC** (see lines 22-27, Note: $div$ is set to off in this standard approach). Figure 3(a) illustrates the final result from a digital camera recommender, where a recommendation for a specific camera is presented to the user. Standard unit critiques are presented alongside each individual feature, while 3 dynamically generated compound critiques appear below the current case description.

```
CB: Casebase, q: query, k: # results,
s: support, div: on/off, r = reccomendation,
CP: Critique Patterns, CC: Compound Critiques,
P: Presented Compound Critiques.
1.   define DynamicCritiquing (q,CB,k,s,div)
2.   do
3.     r ← ItemRecommend (q, CB)
4.     P ← GenerateCC (r, CB, k, s, div)
5.     q ← UserReview (P, CB, r)
6.   until UserAccepts (r)

7.   define UserReview (P, CB, r)
8.     c ← user critique for (f ∈ r || p ∈ P)
9.     CB ← CB - r
10.    q ← c
11.  return q

12.  define ItemRecommend (q, CB)
13.    CB' ← (i ∈ CB | Satisfies(i,q))
14.    Sort CB' by decreasing similarity to q
15.    R ← top item in CB'
16.  return R

17.   define GenerateCC(r, CB, k, s, div)
18.       CP ← GenerateCP(r, CB)
19.       CC ← MineCC(CP, CB, s)
20.       P ← RateCC(CC, CB, k, div)
21.     return P


22.     define RateCC(CC, CB, k, div)
23.       if(div = on)
24.         P ← DiverseSelection(CC, CB, k)
25.       else
26.         P ← top k of CC
27.     return P


28.     define DiverseSelection (CC, CB, k)
29.     P ← {}
30.     For i ← 1 to k
31.       Sort CC by Quality(∀c ∈ C, P)
32.       P ← P + top c in CC
33.       CC ← CC - top c in CC
34.     EndFor
35.     return P
```

Figure 1: The dynamic critiquing algorithm.

| | Current Case | Case $c$ from CB | Critique Pattern |
|---|---|---|---|
| **Manufacturer** | Canon | Sony | != |
| **Model** | Powershot S500 | DSC-V1 | != |
| **Format** | Ultra Compact | Ultra Compact | = |
| **Resolution (M Pixels)** | 5.1 | 5.0 | < |
| **Optical Zoom (X)** | 3 | 4 | > |
| **Digital Zoom (X)** | 4.1 | 4 | < |
| **Weight (grams)** | 215 | 298 | > |
| **Storage Type** | Compact Flash | Memory Stick | != |
| **Storage Included (MB)** | 32 | 16 | < |
| **Price (Euro)** | 443.00 | 455.00 | > |

Figure 2: Generating a critique pattern.

## Compound Critique Diversity

There are a number of problems with the proposed approach to dynamic compound critique generation. In a recent real-user study on dynamic critiquing, a number of trialists complained that their options were frequently limited by compound critiques that lacked diversity among their feature constraints. Figure 3(a) shows a typical example in which a camera is presented to the user along with three compound critiques. The problem is that these three different critiques

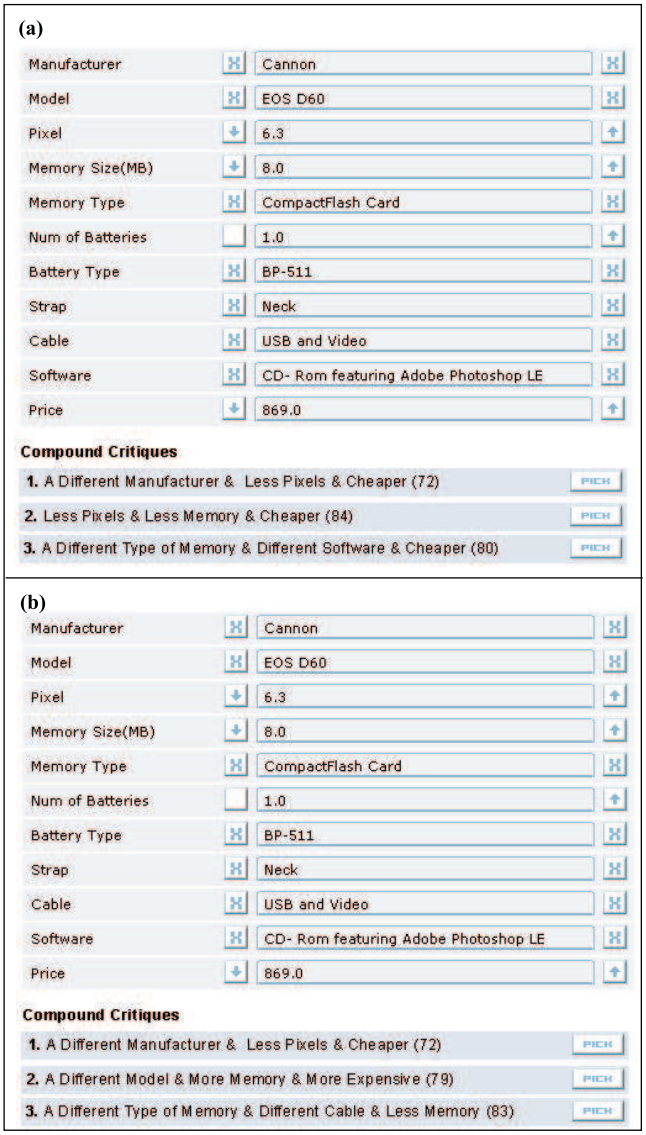


Figure 3: Recommendation cycles for digital cameras with both unit and compound critiques; (a) standard compound critiques, (b) diverse compound critiques.

overlap considerably in terms of their individual critiques thus limiting their applicability. During trials this problem manifested itself by a reluctance of users to select compound critiques over unit critiques, and for this reason the expected performance gains (previously reported by (Reilly *et al.* 2004)) in relation to recommendation efficiency were compromised.

By way of a solution, in this section we describe a technique for increasing the relative diversity of the compound critiques that are selected for presentation to the user. The solution comes in two parts. First, we present the overall diversity enhancement algorithm, which depends on a metric for measuring the relative diversity of a pair of compound critiques. Second, we present two alternative strategies for measuring relative compound critique diversity.

### Diversity Enhancement

The default strategy for selecting a short-list of $k$ compound critiques for recommendation to the user is to select those critiques with the lowest support values. But we are concerned because this strategy does not consider the relationship between the critiques that are chosen, which may lead to diversity problems. This is analogous to a very familiar and related problem in recommender systems, whereby cases selected for retrieval because they are maximally similar to the current query tend also to be very similar to each other. This problem has been addressed by a number of diversity-enhancing retrieval techniques (Bridge 2002; McGinty & Smyth 2003; McSherry 2001; 2002; Shimazu 2001), and we believe that similar methods can be adopted to improve the diversity of our compound critiques.

The diversity enhancing algorithm described by (Smyth & McClave 2001) selects cases on the basis of a quality metric that maximizes similarity to the current query, while minimizing average similarity to cases selected so far. A similar metric can be defined for compound critiquing as shown in E.q. 1 (where $c$ is the current critique and $P$ are the critique options presented to the user so far). Accordingly we give preference to compound critiques that have low support scores, and that are diverse relative to any compound critiques that have been so far selected for presentation for the current cycle. Importantly, this quality metric also allows us to adjust the relative emphasis that is placed on diversity during critique selection.

$$Qual(c,P) = \alpha*(1-Support(c))+(1-\alpha)*Overlap(c,P) \quad (1)$$

### Relative Critique Diversity

Our quality metric above relies on a technique for measuring the relative diversity of a pair of compound critiques. There are two ways that we might do this. The most direct approach is to consider the overlap between the unit critiques that make up a compound critique; see Eq. 2 for (*feature-overlap*). Alternatively, we can measure relative diversity indirectly by computing the overlap between the cases that the critiques satisfy. In other words, two critiques are considered to be diverse if they satisfy different groups of cases; see Eq. 3 for (*case-overlap*).

$$F\_overlap(c,P) = 1 - \frac{|critiques(\{c\}) \cap critiques(P)|}{|critiques(\{c\}) \cup critiques(P)|} \quad (2)$$

$$C\_overlap(c, P) = 1 - \frac{|cases(\{c\}) \cap cases(P)|}{|cases(\{c\}) \cup cases(P)|} \quad (3)$$

In this work we propose to investigate these two approaches to improving the diversity of compound critiques that are presented to the user during a recommendation cycle. Both approaches adopt the standard greedy selection, diversity enhancing algorithm proposed by (Smyth & McClave 2001). Figure 1, lines 28 - 35, shows the **DiverseSelection** function; where **Quality** is calculated by Equation 1 and using different overlap strategies (Equation 2 & 3) according to the experiment being executed. The *feature-overlap* strategy considers the overlap between the unit critiques that make up a compound critique, whereas the *case-overlap* strategy considers the the overlap between the cases that critiques satisfy.

## Evaluation

The motivation for increasing critique diversity is to improve the degree of variation that is presented to end-users. The two diversity approaches above will achieve this and in this section we will evaluate the degree to which critique diversity is improved in practice and how this impacts the general applicability of the resulting critiques. Of course there is a risk to increasing critique diversity. By definition we will be rejecting some compound critiques that are highly rated and favour less highly rated ones that are more diverse. These less highly rated critiques may be less efficient at guiding recommendation and so may degrade recommendation efficiency. We will also investigate this issue in the evaluation that follows.

### Setup & Methodology

In this evaluation our experimental set-up and methodology is the same as that used by (Reilly *et al.* 2004). In our experiments we use a PC dataset, consisting of 120 cases, each describing a single PC or laptop in terms of features such as *price, manufacturer, CPU, Memory* etc. We compare three versions of our algorithm: *standard, case-overlap* and *feature-overlap*. The case and feature-overlap strategies are the diversity enhanced versions of the algorithm described above and we evaluate these approaches with different levels of diversity. The standard strategy refers to the basic low-support selection strategy described by (Reilly *et al.* 2004) in which compound critiques are selected solely on the basis of their support values (i.e., no diversity enhancement). This evaluation was carried out using the standard leave-one-out methodology with an artificial user as described in (Reilly *et al.* 2004).

### Diversity Improvements

In the first of the experiments we looked at the average overlap of the critiques being presented to the users for the different levels of diversity. (The overlap is found by performing a pairwise comparison between the critiques and computing the overlap between their unit critiques.) We get different levels of diversity by varying the "$\alpha$" value in Equation 1. If $\alpha$ is set to 1, the compound critiques will be selected on support alone (overlap not considered; no diversity enhancement), whereas if $\alpha$ is set to 0 the compound critiques will be selected based solely on overlap (support not considered; full diversity enhancement).

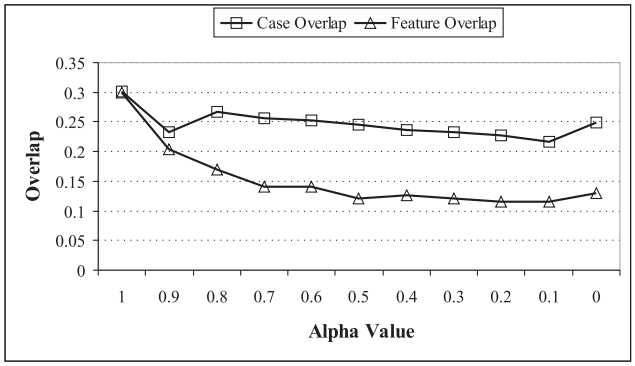


Figure 4: Overlap results for varying $\alpha$.

Figure 4 shows that when there is no diversity enhancement ($\alpha$=1) the overlap is about 0.3. Given that the average compound critique contains about 3 unit critiques this means that we can expect our compound critiques to share one unit critique, on average, with the other presented compound critiques. As the level of diversity enhancement is increased (by lowering $\alpha$) we can see that the overlap of the compound critiques decreases for both case and feature overlap strategies, although more-so for feature overlap strategies. The overlap values keep on decreasing as the diversity enhancement increases as expected. Also as expected, the feature-overlap technique produces more diverse critiques as it directly affects the critiques as it is calculated using the unit critiques within the candidate compound critiques. Whereas the case-overlap strategy affects the critiques in an indirect manner as it is calculated based on the cases in the casebase that satisfy the compound critique. The case and feature overlap strategies can improve the diversity of the compound critiques up to 28% and 62% respectively. The 62% improvement for feature overlap strategies, in particular, indicates that its compound critiques are unlikely to share any unit critiques. Thus compound critiques generated using these methods are unlikely to share unit critiques.

### Recommendation Efficiency

Recommendation efficiency is one of the key motivations of this work. Importantly, introducing diversity to improve the quality of the critiques options presented to the user, should *not* increase recommendation session length. Figure 5 looks at the effect our diversity enhancing algorithms have on efficiency. The results show that the introduction of more diverse critiques actually improves the efficiency results. Although this improvement is small it is nonetheless important as it means that we can safely increase the diversity of the compound critiques presented without affecting the efficiency of our system.

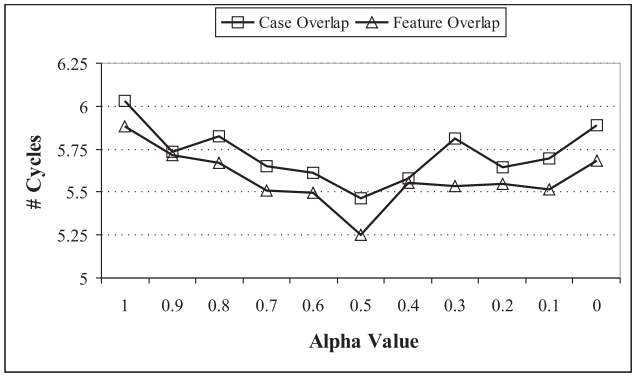


Figure 5: Efficiency results for varying $\alpha$.

## Conclusions

Dynamic critiquing has been previously shown to offer significant recommendation performance benefits by generating compound critiques that operate over multiple case features (McCarthy *et al.* 2004; Reilly *et al.* 2004). In this paper we have addressed a known shortcoming related to the lack of potential diversity between alternative compound critiques. We have described a diversity enhancing approach to compound critique generation and compared two alternative strategies for evaluating critique diversity.

Our evaluation results demonstrate that this technique is capable of generating diverse compound critiques that are likely to be more acceptable to users in practice. The diversity-enhancing approach carries an inherent performance risk because rejecting high quality critiques on the grounds that they lack diversity will lead to the selection of lower quality (albeit more diverse) critiques and this may lead to performance sacrifices. Our results, however, demonstrate that diverse compound critiques can be presented to the user without compromising recommendation efficiency.

## References


Bridge, D. 2002. Diverse Product Recommendations using an Expressive Language for Case Retrieval. In Craw, S., and Pearce, A., eds., *Proceedings of the Sixth European Conference on Case-Based Reasoning (ECCBR 2002)*, 42–57. Springer. Aberdeen, Scotland.

Burke, R.; Hammond, K.; and Young, B. 1996. Knowledge-based navigation of complex information spaces. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 462–468. AAAI Press/MIT Press. Portland, OR.

Burke, R.; Hammond, K.; and Young, B. 1997. The FindMe Approach to Assisted Browsing. *Journal of IEEE Expert* 12(4):32–40.

Burke, R. 2002. Interactive Critiquing for Catalog Navigation in E-Commerce. *Artificial Intelligence Review* 18(3-4):245–267.

Cunningham, P.; Doyle, D.; and Loughrey, J. 2003. An Evaluation of the Usefulness of Case-Based Explanation. In Ashley, K., and Bridge, D., eds., *Case-Based Reasoning Research and Development. LNAI, Vol. 2689.*, 191–199. Springer-Verlag. Berlin.

Faltings, B.; Pu, P.; Torrens, M.; and Viappiani, P. 2004. Design Example-Critiquing Interaction. In *Proceedings of the International Conference on Intelligent User Interface(IUI-2004)*, 22–29. ACM Press. Funchal, Madeira, Portugal.

McCarthy, K.; Reilly, J.; McGinty, L.; and Smyth, B. 2004. On the Dynamic Generation of Compound Critiques in Conversational Recommender Systems. In Bra, P. D., ed., *Proceedings of the Third International Conference on Adaptive Hypermedia and Web-Based Systems (AH-04)*. Springer. Eindhoven, The Netherlands.

McGinty, L., and Smyth, B. 2003. The Role of Diversity in Conversational Systems. In Bridge, D., and Ashley, K., eds., *Proceedings of the Fifth International Conference on Case-Based Reasoning (ICCBR-03)*. Springer. Troindheim, Norway.

McSherry, D. 2001. Increasing Recommendation Diversity Without Loss of Similarity. In *Proceedings of the Sixth UK Workshop on Case-Based Reasoning*, 23–31. Cambridge, UK.

McSherry, D. 2002. Diversity-Conscious Retrieval. In Craw, S., ed., *Proceedings of the Sixth European Conference on Case-Based Reasoning (ECCBR-02)*, 219–233. Springer. Aberdeen, Scotland.

Agrawal, R.; Mannila, H.; and Verkamo., A. I. 1996. Fast Discovery of Association Rules in Large Databases. *Advances in Knowledge Discovery and Data Mining* 307–328.

Reilly, J.; McCarthy, K.; McGinty, L.; and Smyth, B. 2004. Dynamic Critiquing. In Calero, P. A. G., and Funk, P., eds., *Proceedings of the European Conference on Case-Based Reasoning (ECCBR-04)*. Springer. Madrid, Spain.

Schafer, J. B.; Konstan, J. A.; and Riedl, J. 2001. E-Commerce Recommendation Applications. *Data Mining and Knowledge Discovery* 5(1/2):115–153.

Shimazu, H. 2001. ExpertClerk : Navigating Shoppers' Buying Process with the Combination of Asking and Proposing. In Nebel, B., ed., *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, 1443–1448. Morgan Kaufmann. Seattle, Washington, USA.

Smyth, B., and McClave, P. 2001. Similarity v's Diversity. In Aha, D., and Watson, I., eds., *Proceedings of the International Conference on Case-Based Reasoning*, 347–361. Springer.

Smyth, B., and McGinty, L. 2003. An Analysis of Feedback Strategies in Conversational Recommender Systems. In Cunningham, P., ed., *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Cognitive Science (AICS-2003)*. Dublin, Ireland.