The Power of Suggestion*

Barry Smyth and Lorraine McGinty
Smart Media Institute, Department of Computer Science,
University College Dublin, Belfield, Dublin 4, Ireland.
Email: {barry.smyth, lorraine.mcginty}@ucd.ie

Abstract

User feedback is vital in many recommender systems to help guide the search for good recommendations. Preference-based feedback (e.g. *"Show me more like item A* ") is an inherently ambiguous form of feedback with a limited ability to guide the recommendation process, and for this reason it is usually avoided. Nevertheless we believe that certain domains demand the use of preference-based feedback. As such, we describe and evaluate a flexible recommendation strategy that has the potential to improve the performance of case-based recommenders that rely on preference-based feedback.

1 Introduction

At last year's European Case-Based Reasoning conference (ECCBR 2002) in Scotland, delegates were treated to a whiskey tasting. A range of different whiskeys were sampled, each accompanied by a feature-based description (e.g. *distillery, age, alcohol content, cask, peatiness, sweetness, palette, etc.).* However, the value of these descriptions was limited as most of us had little or no understanding of the features, nor could we reliably reconcile individual features with the taste of a particular whiskey. Nevertheless virtually everyone could chose a favourite whiskey by the end of the session (after a lot of sampling it has to be said!). This example corresponds to a challenging problem for developers of recommender systems, for reasons that will become clear.

Normally the success of case-based recommenders depends on two important domain properties: (1) the items need to be described using well-defined features; and (2) users must have some understanding of these features and how they relate to their requirements. For example, restaurants are readily described using features such as *cuisine, location* and *price,* and most people have a reasonable understanding of their needs in relation to these features. When the above characteristics are present content-based recommendation strategies can be combined with different forms of user feedback, such as value elicitation (e.g. "I want an *expensive, Asian* restaurant") [Bridge, 20011, or critiquing (e.g. "Show me more like *Holly's Bistro* but *less formal"),* to good effect [Burke, 20001. However, if these properties are not present then other strategies are needed. For example, if item descriptions are not available, collaborative filtering can be used to recommend items that have been rated positively by similar users [Burke, 2000; Shardanand and Maes, 1995]. The whiskey recommendation scenario above is different. The availability of item descriptions suggests case-based recommendation, but the lack of user expertise limits the use of feedback strategies such as value elicitation or critiquing.

Indeed the whiskey domain is not unique in this respect. In many domains users are able to express a preference without necessarily understanding individual item properties. This is especially true in domains where a specialised v :abulary already exists to describe items (e.g. fashion, jewellery, art, music, etc.). Consider a recommender system designed to help a bride-to-be to choose her ideal dress, individual dresses are described using specific features (e.g. *coul-back, scoop-cut, georgian-style,* etc.) but most of these terms are meaningless to the average bride, and yet the typical bride-to-be is capable of recognising *what* she likes *when* she sees it. This problem is more than the user being unaware of the correct vocabulary to use during recommendation. The point is that even if they were presented with the appropriate vocabulary terms, they would not be in a position to use or appreciate these terms.

One form of feedback that could be used is preference-based feedback; the user simply expresses a preference for an item (e.g. "Show me more like dress 2"). This has been largely avoided by researchers - it is assumed to be too limited to efficiently guide the recommendation process. Nevertheless in our research we arc interested in looking at how to make this a practical form of feedback in these challenging domains. Incidentally, unlike other feedback types, preference-based feedback also benefits from minimal interface needs, and so is particularly relevant for recommenders designed for use with the current generation of mobile devices, such as WAP phones and PDAs.

In this work we describe a way to improve the performance of recommenders that use preference-based feedback. Our novel *adaptive selection* method mirrors how real-life sales assistants adapt their recommendation strategies in response to user feedback by balancing the importance of similarity and diversity during each recommendation cycle. Dramatic improvements in recommendation performance have

"The support of the Informatics Research Initiative of Enterprise Ireland is gratefully acknowledged been observed (e.g. reductions in dialog length of up to 80%).

2 Comparison-Based Recommendation

Comparison-based recommendation is a content-based (or case-based) recommendation strategy [Burke, 2000; McGinty and Smyth, 2002], as opposed to a collaborative recommendation strategy [Burke, 2000; Konstan *etal.,* 1997; Shardanand and Maes, 1995]. The notion of comparison-based recommendation was introduced by [McGinty and Smyth, 2002] to emphasise the role of feedback in content-based recommenders, and to allow for an analysis of preference-based feedback in particular. The focus was on how query modification methods might be used with preference-based feedback to produce efficient recommendation dialogs without richer forms of feedback like value elicitation or critiquing. We briefly review this earlier work, describing the basic comparison-based recommendation framework and some successful query modification strategies.

2.1 The Basic Algorithm

Comparison-based recommendation (Figure 1) is an iterative recommendation algorithm that presents the user with a selection of *k* items as recommendations during each of a sequence of *n* recommendation cycles. During each cycle the user expresses a *preference* for one of the suggestions and this *feedback* is used to *revise* the query for the next cycle; in the vocabulary of Shimazu, it is a type of *navigation by proposing* [Shimazu, 2001]. The *recommendation dialog* terminates when the user is presented with an acceptable suggestion.

2.2 Simple Selection: More Like This

The simplest approach to comparison-based recommendation is to recommend the *k* most similar items to the current query, and use the newly preferred item as the query for the recommendation next cycle. This *more like this* (MLT) method is a facility often provided by Web search engines; although it should be noted that search engines rarely rely upon this method, primarily focusing instead on query elicitation and term-based search. The MLT method is flawed by a tendancy to follow so-called *false-leads* as it overfits to the current preference, and this results in protracted recommendation dialogs. Using each preferred item as the next query is problematic if some of its features are not relevant to the user. For example, a user may indicate a preference for a whiskey that is *oak-aged, sweet,* and *peaty.* But if the preference was largely a result of the peatiness and independent of, or even at odds with, the sweetness or the aging, then blindly focusing the next selection on *sweet* and *oak-aged* whiskeys, as well as *peaty* ones, may result in poor recommendations.

Figure 1 shows the scope of this problem by graphing the similarity of the preference to the target in a sample recommendation dialog. Instead of an increasing similarity profile, we find sustained decreases in similarity as MLT follows false-leads. Between cycle 3 and 11, target similarity falls from 0.46 to as low as 0.26 as the user is forced to accept poor recommendations, and again between cycle 11 and 40 there is a noticeable similarity trough. Indeed it is only after cycle 38 that MLT finally begins to focus on the target region and similarity rises.

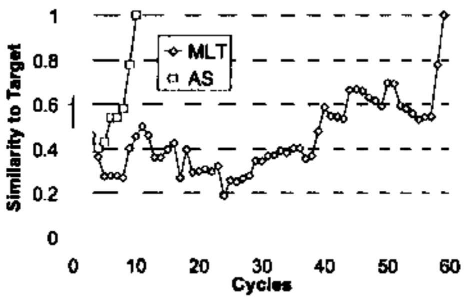

Figure 1: Profiling the similarity of the preference case to the target case in a typical recommendation session

2.3 Query Modification Strategies

Query modification techniques can relieve the false-lead problem, and have been shown to improve recommendation efficiency, reducing MLT dialogs by 30% on average [McGinty and Smyth, 2002]. The central idea is to identify features in the preference case that *are* likely to be the reason for the user's choice, and those features that are likely to be irrelevant. Only relevant features are transferred to the new query. Returning to the whiskey example above, if the rejected cases are also very sweet then sweetness may irrelevant to the user. Hence, one query modification strategy, *partial MLT* (pMLT) only includes preference features in the query that are not found in any of the rejected cases.

[McGinty and Smyth, 2002] also propose more sophisticated query modification strategies that weight preference features in the query according to how likely they are to be reliable. One strategy computes a weight for a preference feature based on the proportion of rejected items that also contain that feature. If 3 cases are recommended to the user, and if the preference and just one of the rejected items are *oak-aged* and then this feature is transferred to the new query with a weight of 0.5 (50% of the rejected items are *oak-aged).*

3 Adaptive Case Selection

So far we have assumed the use of similarity-based selection during each cycle. However, in the real-world two selection strategies can be observed in the dialogs that take place between customers and sales assistants. If a customer is unsure of their exact needs the sales assistant will tend to present a diverse range of alternatives, based on a preliminary set of requirements, to *focus* in on a promising region of a product space. A good sales assistant can recognise when the customer sees something that they genuinely like and uses this as a cue to switch their strategy to one that tries to *refine* subsequent recommendations in the region of the preference by selecting similar items.

In this paper, instead of using query modification methods to improve recommendation efficiency, we propose an alternative called *adaptive selection.* It adapts the way that new items are selected for recommendation and is motivated by the above observation that different selection strategies are appropriate at different times in the recommendation process.

3.1 Similarity vs Diversity

Our observation about real-life recommendation scenarios is partly echoed by recent research that has questioned the apparent over-emphasis on similarity in content-based recommenders, with diversity forwarded as an important additional selection constraint. A number of strategies for improving recommendation diversity have been suggested. [Shimazu, 2001] proposes the recommendation of three items or cases, i_1, *in* and 73, per recommendation cycle: i_1 is the most similar case to the query; i_2 is maximally dissimilar from i_1; and $i_1$3 is maximally dissimilar from i_1 and i_2. By design these items are maximally diverse, but similarity to the query may be compromised as there are no guarantees that *in* and i_3 will be very similar to the query.

Other researchers have proposed alternative diversity enhancing mechanisms with a more manageable balance between item diversity and query similarity [Bridge, 2001; McSherry, 2002; Smyth and McClave, 20011. Although all of these techniques introduce diversity into selection in a uniform way, they do not fully reflect the strategy switching seen in real-world scenarios. Moreover, while increasing diversity may improve the efficiency of of a recommender by allowing it to cover more of the item-space per recommendation cycle, it may also lead to new types of inefficiencies. Diversity enhancing methods pass over items that might otherwise be selected and the target item may be one of these.

A more flexible mechanism for introducing diversity must be adopted based on whether or not the recommender has focussed on the correct region of the item space. If there is evidence that it is correctly focused then a pure similarity-based selection method can be used to refine the recommendations and safe-guard against passing over the target item. If the recommender is not correctly focused, then diversity is introduced in order to maximise the coverage of the item space in the next cycle, and so help to refocus the recommender by offering the user contrasting recommendations.

3.2 Preference Carrying

Unfortunately it is not immediately obvious how to judge whether the recommender is correctly focused; the target item is unknown and user requirements are typically vague. Adaptive selection solves this by evaluating whether the recommendations made in the i^{th} cycle are an improvement on those in the i ~ I^{th} cycle before choosing a selection strategy for the i + $\backslash^{th}$ cycle.

This is achieved by *carrying the preference* item from the previous cycle to the current cycle. If the user selects the carried preference in the current cycle it must mean that they are unhappy with the *k* - I new alternatives in that cycle. These new items must have failed to improve upon the recommendations made during the previous cycle, and so the recommender must not be correctly focussed. If the user ignores the carried preference and selects one of the newly recommended items then the recommender must be correctly focused. Thus, by carrying the preference item, and monitoring whether or not the user reselects it, we can implement a switching mechanism between two alternative selection strategies: a *refine* strategy that emphasises similarity; and a *refocus* strategy that balances similarity and diversity for improved recommendation coverage.

In theory carrying the preference may lead to new inefficiencies because the carried preference takes up a valuable slot in each cycle, thus limiting recommendation coverage. However, this is easily compensated for because carrying the preference helps protect against against false-leads. If none of the *k*: - 1 new cases are relevant then by reselecting the carried preference the user is at least maintaining the previous best recommendation rather than being forced to accept a lower quality false-lead. In practice, this on its own can offer a substantial improvement to recommendation efficiency.

3.3 Refine & Refocus

As mentioned above, the refine strategy makes use of a standard similarity-based selection method, picking *k* — 1 new items that are maximally similar to the current query. The refocus strategy uses the bounded-greedy diversity technique proposed by [Smyth and McClave, 2001]; see Figure 2.

Very briefly, the bounded-greedy technique involves two basic phases. First, the *bk* most similar items to the query are selected (where *b* is typically an integer between 2 and 5). During the second phase, the set (R) of selected items is built incrementally. During each step of this build the remainder of the *bk* items are ordered according to their *quality* and the highest quality item added to *R*. The quality of a item *i* is proportional to the similarity between *i* and the current query *q,* and to the diversity of *i relative* to those items so far selected, $R = \{r_1, \ldots, r_m\}$; see Equations 1 & 2.

$$Quali1y(q,i,R) = a^* \ Sim(q,i) + (1 - a)^* \ Div(i, R) \quad (1)$$

$$Div(i,R) = 1 \text{ if } R = \{\};$$

$$— \quad \frac{—}{m} \ otherwise \ (2)$$

The first case to be selected is always the most similar to the query and in subsequent iterations, the chosen case has the highest quality relative to the query and diversity and cases selected so far. Note, we set $b = 3$ and $a = 0.5$ to balance similarity and diversity during refocusing.

4 Evaluation

In this section we evaluate the performance of adaptive selection focusing on the average number of cycles and unique items that must be presented to a user in a typical recommendation dialog. The shorter the dialog, the more successful the recommender is likely to be, and we demonstrate that adaptive selection delivers dramatic reductions in dialog length.

4.1 Setup

A case-base of 585 unique Scottish whiskey cases (Figure 3) is used to compare two comparison-based recommenders using preference-based feedback: MLT uses the standard MLT strategy; MLT+AS implements adaptive selection. We use a leave-one-out methodology for testing using 200 randomly selected cases as test cases. Each test case *(base)* is removed from the case-base and used in two ways.

```
define Comparison-Based-Recommend(q, CB, k)
  $i_{p-1}$ , $i_p$ ← null
  do
    R ← ItemRecommend(q, CB, k, $i_p$, $i_{p-1}$)
    $i_p$ ← UserReview(R, CB)
    Q ← QueryRevise(q, $i_p$, R)
    $i_{p-1}$ ← $i_p$
  until UserAccepts($i_p$)

define QueryRevise(q, $i_p$, R)
  R' ← R - {$i_p$}
  q ← $i_p$
return q

define UserReview(R, CB)
  $i_p$ ← user's preferred case from R
  CB ← CB - R
  return $i_p$

define ItemRecommend(q, CB, k, $i_p$, $i_{p-1}$)
  if($i_p$ != null) && ($i_p$ == $i_{p-1}$)
     R ← ReFocus(q, CB, k)
  else
     R ← ReFine(q, CB, k)
  return R

define ReFine(q, CB, k)
  CB' <- sort CB in decreasing order of their sim to q
  R ← top k items in CB'
  return R

define ReFocus(q, CB, k, $i_p$, $i_{p-1}$)
  return BoundedGreedySelection(q, CB, k, b)

define BoundedGreedySelection (q, CB, k, b)
  CB' := bk cases in CB that are most similar to q
  R := {}
  For j := 1 to k
    Sort CB' by Quality(q,i,R) for each case i in CB'
    R  := R + First(CB')
    CB' := CB' - First(CB')
  EndFor
  return R
```

Figure 2: The Comparison-Based Recommendation algorithm with adaptive selection.

CASE ID	DISTILLERY	AGE	PROOF	SWEETNESS	PEATINESS	AVAILABILITY	COLOR	NOSE	FLAVOR:PALATTE	FINISH
Case 500	The Glenlivet	13	40	7	4	5	gold	sweet	medium-peat	full-body

Figure 3: The features of a sample whiskey case include nominal and continuous features.

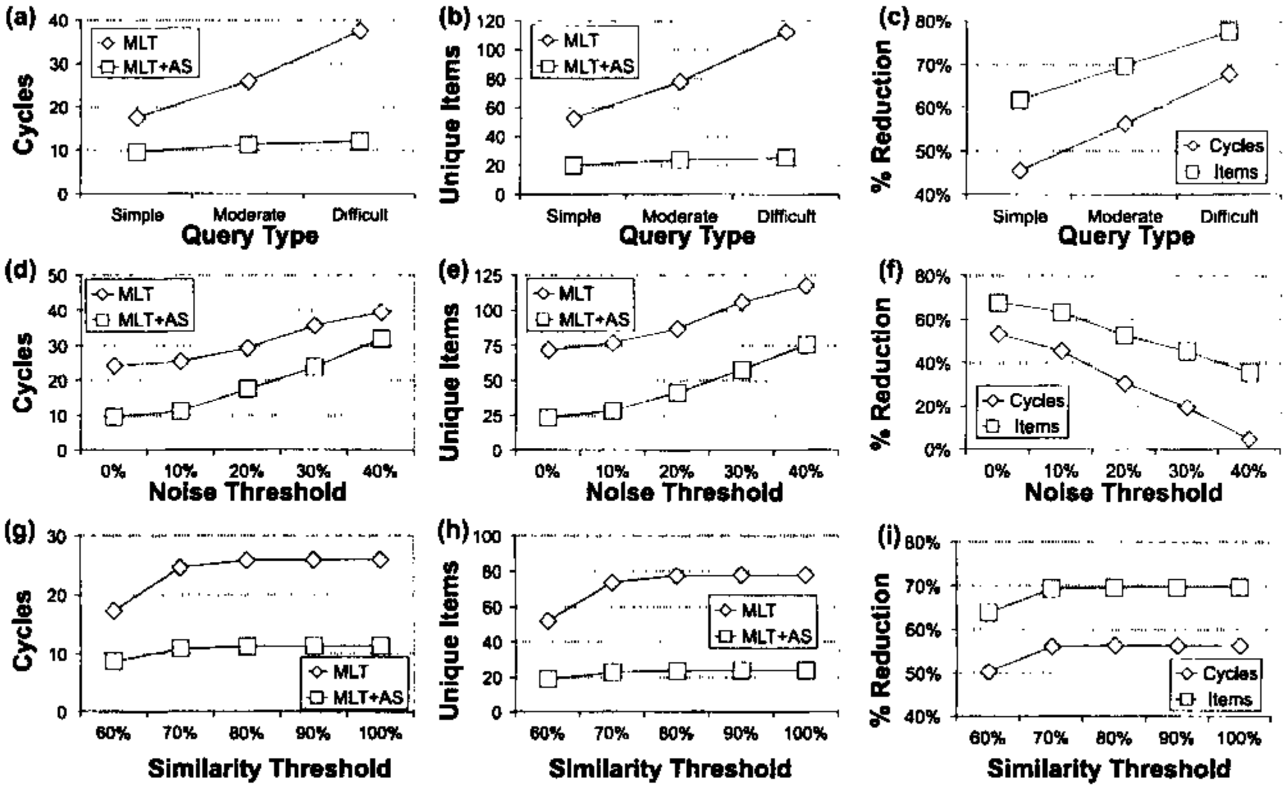

Figure 4: Key experimental results

First it serves as the basis for a set of queries made by taking random subsets of features. Second, it is used to identify a maximally similar *(target)* case. Thus, the base is the ideal query for a user, the generated query is the initial test query given to the recommender, and the target is the best case for the user given their ideal. During each recommendation cycle 3 cases are recommended to the user, and the most similar to the target is chosen as a preference, thus simulating a user that is capable of correctly selecting the best preference item in each cycle; we relax this assumption in Section 4.3. A query is satisfied when the target is returned in a cycle, thereby simulating the situation where the user is seeking a specific case; once again, we relax this condition in Section 4.4. In total 1250 queries are generated and divided into three groups based on their difficulty; where difficulty is based on the number of recommendation cycles required by MLT.

4.2 Recommendation Efficiency

Basic recommendation efficiency can be measured in terms of the average number of cycles (or unique cases or items) that a user must work through before being presented with their ideal target. The leave-one-out method outlined above is used by the two recommenders for each of the 1250 queries and and the mean number of cycles and unique items presented to the user are measured. The results are shown in Figure 4(a-b) as graphs of mean cycles and items for each algorithm and query group. The benefits of MLT+AS should be clear for both measures. For example, we see that MLT requires 17.5 cycles (and 52.5 items) for simple queries, but MLT-AS needs only 9.5 cycles (and 20 items), a relative reduction of 45% in terms of cycles, and 62% in terms of items.

We see similar results for the other query sets. In fact, the relative reductions enjoyed by MLT-AS increase with query difficulty. The recommendation dialogs associated with more difficult queries include more false-leads than the dialogs for simpler queries, and so offer adaptive selection an even greater opportunity for dialog reduction. This trend is clearly seen in Figure 4(c), which graphs the percentage reduction in cycles and items due to MLT-AS, relative to MLT. For cycles, the relative reduction grows from 45% (simple) to 68% (difficult) and for unique items it grows from 62% to 78%. These results clearly show a dramatic benefit for adaptive selection. Indeed this benefit is so great that MLT-AS is capable of satisfying the most difficult queries far more efficiently than MLT takes to satisfy even the simplest queries.

4.3 Preference Noise

In this experiment we re-evaluate our recommenders by relaxing the assumption that the user always prefers the most similar item to the target to test whether benefits are found with sub-optimal preferences. The above experiment is repeated except that noise is introduced into the preference selection by perturbing the similarities between each recommended item and the target by some random amount within a set noise limit. A 10% noise means that each similarity value can change by up to +/-10% of its actual value. This will potentially change the internal ordering of recommended items during each cycle resulting in the selection of a preference that may not be the most similar to the target. This approach mimics the situation where users are likely to make preference mistakes more frequently if there is little difference between the target similarities of the recommended items.

Figure 4(d&e) graph the mean number of cycles and items presented to the user versus the noise limit for moderate queries. As expected, introducing preference noise has a negative impact on the ability of each recommender to locate the target item. MLT dialogs increase from 24 cycles at 0% noise to 39 cycles at 40% noise, and MLT-AS dialogs increase from 11 cycles to 37 cycles. The number of items required is also seen to increase in a similar manner. However, the benefits of MLT-AS remain across all levels of noise (see Figure 4(f)) although the magnitude of these benefits is seen to fall as noise increases. For example, the MLT-AS benefit, in terms of unique items, falls from 68% at the 0% noise level to 36% at the 40% level. Nevertheless, adaptive selection once again offers significant improvements in recommendation efficiency even when users make imperfect preference selections. While the results presented here focus only on moderate queries, for reasons of brevity, it is worth noting that qualitatively similar results are found for simple and difficult queries. Once again the MLT-AS benefits increase with query difficulty; for example, in terms of unique items, at the 40% noise level, a 27% MLT-AS benefit is observed for simple queries, and a 43% MLT-AS benefits for difficult queries.

4.4 Target Noise

Our second key assumption is that users are interested in a single target item during recommendation, and that the dialog only terminates when this item is returned. It is perhaps more realistic to assume that the user will be satisfied by any one of a group of items that are similar to the optimal target. We re-evaluate our recommenders under this more relaxed termination condition by repeating the basic efficiency experiment except that we terminate each dialog once an item has been recommended that is within some pre-defined similarity of the target. A similarity threshold of 70% means that the dialog terminates when an item that is at least 70% similar to the targer has been recommended.

Figure 4(g-i) presents the results in a number of ways. First Figures 4(g&h) graph the mean number of cycles and items presented to the user versus the similarity threshold for moderate queries. As expected, relaxing the termination condition results in shorter recommendation dialogs for MLT and MLT-AS. For example, MLT dialogs reduce from just under 26 cycles at the 100% similarity threshold (where the optimal item must be recommended) to just over 17 cycles at 60% similarity. MLT-AS dialogs reduce from 11 cycles to just under 9 cycles across the same similarity range. The number of items required is also seen to reduce in a similar manner. However, positive MLT-AS benefits are maintained across all similarity thresholds (see Figure 4(i)). In fact, as the threshold increases, and the termination condition becomes more rigid, we find increasing benefits for MLT-AS in terms of cycles and items. For example, the MLT-AS benefit, in terms of unique items, grows from 54%; at the 60% similarity threshold to just under 70% at the 100% threshold. Once again, adaptive selection delivers significant improvements in recommendation efficiency across a variety of termination conditions.

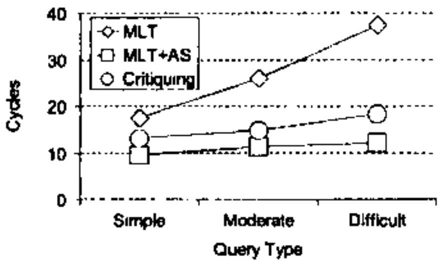

Figure 5: Preference-Based Feedback vs Critiquing.

4.5 Preference-Based vs Critiquing

Finally, it is worth briefly considering the impressive dialog reductions observed for adaptive selection with preference-based feedback in the context of a much richer form of feedback, namely critiquing [Burke, 2000; McGinty and Smyth, 2003]. Figure 5 presents an efficiency comparison (in terms of average recommendation cycles) between our two preference-based recommenders and an equivalent recommender that implements critiquing; briefly, during each cycle feedback corresponds to a preference case *plus* a critique that constrains a selected feature value.

On the face of it we expect critiquing to produce more efficient dialogs than preference-based feedback. And so it does for all query types when compared to standard preference-based feedback. However, the surprise is that combining preference-based feedback and adaptive selection produces recommendation dialogs that are consistently and significantly shorter than even those produced by critiquing. For example, relative to MLT-AS, critiquing requires 37% more cycles for simple queries and 52% more cycles for the difficult queries.

5 Conclusions

To date preference-based feedback has been largely ignored by recommender systems. It is an inherently ambiguous form of feedback with a limited ability to efficiently guide the recommendation process. Nevertheless, we believe that this form of feedback is useful, and perhaps even vital, in certain recommendation domains where other forms of feedback cannot be used, perhaps because of limited user expertise or even basic device restrictions.

We have described how preference-based feedback can be made more efficient using the adaptive selection technique, which modifies its recommendation strategy depending on whether or not the recommender is correctly focused on the right region of the recommendation space. Using this method we have demonstrated dramatic performance improvements over standard preference-based feedback, across a variety of experimental conditions, with reductions in dialog length of up to nearly 80%. Indeed, adaptive selection is so effective that it is capable of using simple and cheap preference-based feedback to produce recommendation dialogs that are even more efficient than those available with richer forms of feedback such as critiquing.

In this paper we have presented evaluation results from one recommendation domain, Whiskey. This domain is especially interesting because of its dependency on preference-based feedback (as discussed in Section 1). We have also tested our technique on more traditional domain datasets (i.e. Travel, PC) and have found similar results iMcGinty and Smyth, 2003]. We believe that these results have the potential to change the perception of simple preference-based feedback, facilitating its practical use in a variety of recommendation scenarios. Indeed adaptive selection is in no way limited to preference-based feedback. In our future research we intend to investigate if similar performance benefits can be achieved by integrating this technique in recommenders that use other forms of feedback (e.g. *critiquing* and *rating-based).*

References

[Bridge, 2001] D. Bridge. Product Recommendation Systems: A New Direction. In D. Aha and 1. Watson, editors, *Workshop on CBR in Electronic Commerce at The International Conference on Case-Based Reasoning (ICCBR-01),* 2001. Vancouver, Canada.

[Burke, 2000] R. Burke. Knowledge-based Recommender Systems. *Encyclopedia of Library and Information Systems,* 69(32), 2000.

[Konstan et al., 1997] J.A. Konstan, B.N Miller, D. Maltz, J.L. Herlocker, L.R. Gorgan, and J. Riedl. GroupLens: Applying collaborative filtering to Usenet news. *Communications of the ACM,* 40(3):77-87, 1997.

[McGinty and Smyth, 2002] L. McGinty and B. Smyth. Comparison-Based Recommendation. In Susan Craw, editor, *Proceedings of the Sixth European Conference on Case-Based Reasoning (ECCBR-02),* pages 575-589. Springer, 2002. Aberdeen, Scotland.

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

[McSherry, 2002] D. McSherry. Diversity-Conscious Retrieval. In Susan Craw, editor, *Proceedings of the Sixth European Conference on Case-Based Reasoning (ECCBR-02),* pages 219-233. Springer, 2002. Aberdeen, Scotland.

[Shardanand and Maes, 1995] U. Shardanand and P. Maes. Social Information Filtering: Algorithms for Automating "Word of Mouth". In *Proceedings of the Conference on Human Factors in Computing Systems (CHI '95),* pages 210-217. ACM Press, 1995. New York, USA.

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

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