Incremental Critiquing

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

Abstract

Conversational recommender systems guide users through a product space, alternatively making concrete product suggestions and eliciting the user's feedback. Critiquing is a common form of user feedback, where users provide limited feedback at the feature-level by constraining a feature's value-space. For example, a user may request a *cheaper* product, thus critiquing the *price* feature. Usually, when critiquing is used in conversational recommender systems, there is little or no attempt to monitor successive critiques within a given recommendation session. In our experience this can lead to inefficiencies, on the part of the recommender system, and confusion on the part of the user. In this paper we describe an approach to critiquing that attempts to consider a user's critiquing history, as well as their current critique, when making new recommendations. We provide experimental evidence to show that this has the potential to significantly improve recommendation efficiency.

1 Introduction

Recommender systems are designed to alleviate the information overload problem [13] by assisting users to make choices that guide and inform the decision making process. They combine ideas from information retrieval, machine learning and user profiling, among others, to offer the user a more efficient search environment, one that is better suited to their needs and preferences, and one that helps them to locate what they are looking for more quickly and easily. Conversational case-based recommender systems guide users through a *recommendation session* by adopting a *recommend-review-revise* strategy [1, 6]. In short, during each cycle of a recommendation session a user is presented with a new recommendation and is offered an opportunity to provide feedback on this suggestion. On the basis of this feedback, the recommender system will revise its evolving model of the user's current needs in order to make further recommendations.

Considerable research effort has been invested in developing and evaluating different forms of feedback for conversational recommender systems and a variety of feedback alternatives have become commonplace. For example, *value elicitation* approaches ask the user specific questions about specific features (e.g.

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

"what is your target price?") while *preference-based feedback* and *ratings-based* methods simply ask the user to indicate which product they prefer when presented with a small set of alternatives [6, 7, 15], or to provide ratings for these alternatives [13]. It is well known that different forms of feedback introduce different types of trade-offs when it comes to recommendation efficiency and user-effort. For instance, value elicitation is a very informative form of feedback, but it requires the user to provide detailed feature-level information. In contrast, preference-based feedback is a far more ambiguous form of feedback but it requires only minimal user effort. One form of feedback that strikes a useful balance, in this regard, is critiquing [2, 3]. The user expresses a *directional preference* over the value-space for a particular product feature. For example in a PC recommender, a user might indicate that they are looking for a PC that has a *"faster processor"* than the current recommendation, *"faster processor"* being a critique over the *processor speed* feature of the PC case.

In our recent work we have explored a variety of ways of improving the efficiency of conversational recommenders that employ different forms of feedback, with our main focus on preference-based feedback and critiquing [6, 8, 14, 15]. In this paper we continue in this vein by proposing a new approach to critiquing called *incremental critiquing*. The motivation for this new approach stems from our observation, that when critiquing is used in conversational recommender systems, there is usually little or no attempt to monitor successive critiques within a given recommendation session. In our experience this can lead to a number of problems. For instance, users may, during the course of a recommendation session, change their mind about certain feature preferences (maybe temporarily), and so contradict earlier critiques that they may have applied. Our incremental critiquing strategy attempts to remedy this by maintaining a history of successive critiques, during a recommendation session, and by allowing these past critiques to influence future recommendation cycles. In this way new recommendations are chosen based not only on their ability to satisfy the current critique, but also on their compatibility with previous critiques. We describe a flexible approach for incorporating past critiques in a way that facilitates the type of feedback inconsistencies that tend to occur during real recommendation sessions. We further show that this incremental form of critiquing has the potential to deliver significant improvements in recommendation efficiency as well as providing users with more intuitive recommendations.

2 A Review of Critiquing

Critiquing, as a form of feedback, is perhaps best known by association with the FindMe recommender systems [2], and specifically the Entrée restaurant recommender. The original motivation for critiquing included the need for a type of feedback that was simple for users to understand and apply, and yet informative enough to focus the recommender system. For instance, the Entrée system presents users with a fixed set of directional critiques in each recommendation cycle. In this way users can easily request to see further suggestions that are

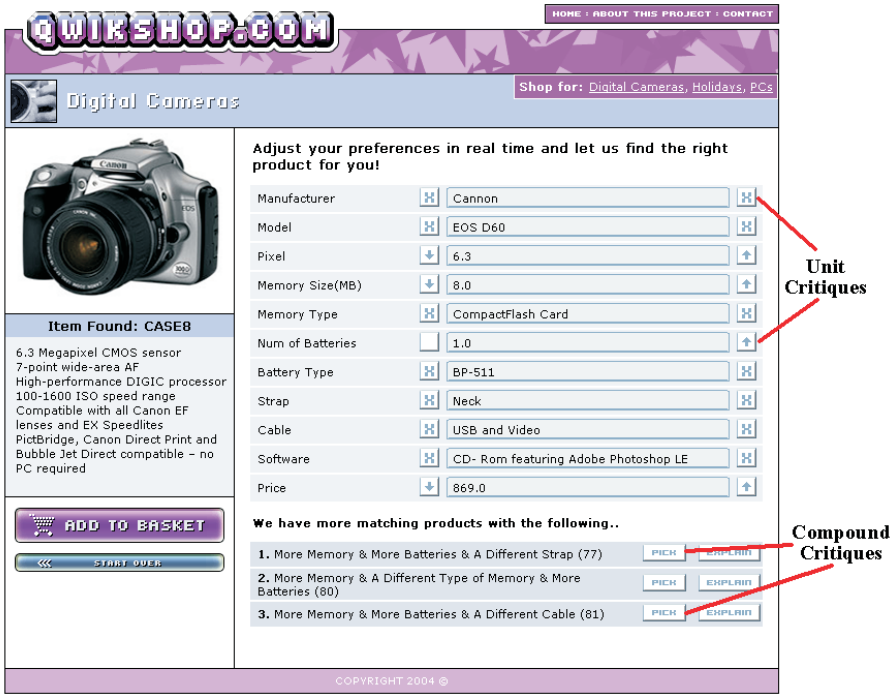


Figure 1: Unit critiques and compound critiques in a sample recommender.

different in terms of some specific feature. For example, the user may request another restaurant that is *cheaper* or *more formal*, for instance, by critiquing its *price* and *style* features.

Recently researchers have begun to look more closely at critiquing and have suggested a number of ways that this form of feedback can be improved. For example, [7, 8, 10] show how the efficiency of critiquing can be improved by incorporating diversity into the recommendation process and in Section 2.1 we will review further recent work. However, a number of challenges remain when it comes to the practicalities of critiquing. Certainly, the lack of any continuity-analysis of the critiques provided in successive cycles means that recommender systems may be misled by the inconsistent feedback provided by an uncertain user, especially during early recommendation cycles.

2.1 Unit vs Compound Critiques

The critiques described so far are all examples of what we term *unit* critiques. That is, they express preferences over a single feature; *cheaper* critiques a *price* feature, *more formal* critiques a *style* feature, for example. This ultimately limits the ability of the recommender to narrow its focus, because it is guided

by only single-feature preferences from cycle to cycle.

An alternative strategy is to consider the use of what we call *compound* critiques [12]. These are critiques that operate over multiple features. This idea of compound critiques is not novel. In fact the seminal work of Burke *et al.* [2] 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 *high performance* compound critique in a PC recommender to simultaneously increase *processor speed, RAM, hard-disk capacity* and *price* features. Obviously compound critiques have the potential to improve recommendation efficiency because they allow the recommender system to focus on multiple feature constraints within a single cycle. In addition, it has also been argued that they carry considerable explanatory power because they help the user to understand common feature interactions [5, 12]; in the PC example above the user can easily understand that improved CPU and memory comes at a price.

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 example above the *sportier* critique would continue to be presented as an option to the user despite the fact that the user may have already seen and declined all the relevant car options. Recently, we have argued the need for a more dynamic approach to critiquing in which compound critiques are generated on-the-fly, during each recommendation cycle, by mining commonly occurring patterns of feature differences that exist in the remaining cases [12]. Figure 1 shows a Digital Camera recommender system that we have developed to explore usability and efficiency issues in a real-world setting. The screenshot shows two types of critiques. Standard unit critiques are presented alongside each individual feature, while k (where k=3) dynamically generated compound critiques appear below the current case description.

2.2 Consistency & Continuity

Regardless of the type of critiquing used (unit versus compound, or a mixture of both), or the manner in which the critiques have been generated (fixed versus dynamic), there are a number of important issues that need to be kept in mind from an application deployment perspective. This is especially important when it comes to anticipating how users are likely to interact with the recommender. In particular, users cannot be relied upon to provide consistent feedback over the course of a recommendation session. For example, many users are unlikely to have a clear understanding of their requirements at the beginning of a recommendation session. In our experience many users rely on the recommender as a means to educate themselves about the features of a product-space. As a result users may select apparently conflicting critiques during a session as they explore different areas of the product space in order to build up a clearer picture

of what is available. For example, in a given cycle we may find a prospective digital camera owner looking for a camera that is *cheaper* than the current 500 euro recommendation, but later on may ask for a camera that is *more expensive* than another 500 euro recommendation. There are a number of reasons for this: perhaps the user has made a mistake; perhaps they are just interested in seeing what is available at the higher price; or perhaps their preferences have changed from the start of the session as they recognise the serious compromises that are associated with lower priced cameras [11].

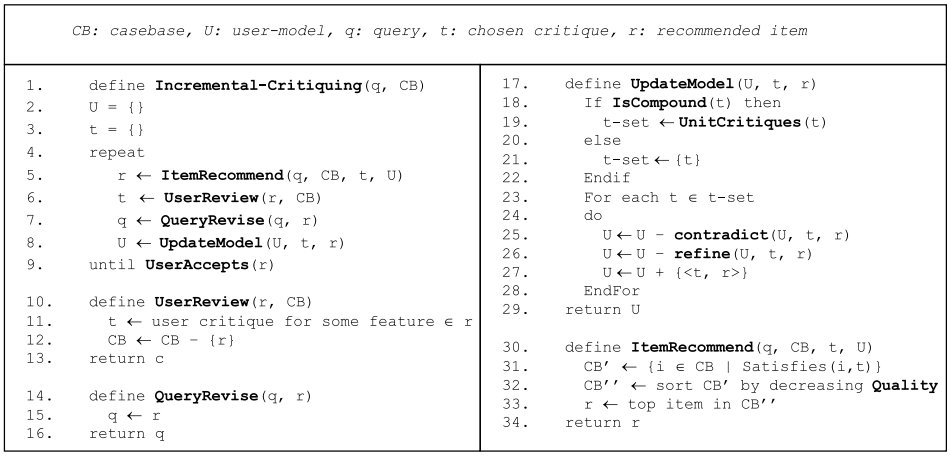
Current recommender systems that employ critiquing tend to focus on the current critique and the current case, without considering the critiques that have been applied in the past. This, we argue, can lead to serious problems. For instance, if the recommender uses each critique to permanently filter-out incompatible product-cases, then a user may find that there are no remaining cases when they come to change their mind; having indicated a preference for sub-500 euro cameras early-on, the user will find the recommender unable to make recommendations for more expensive cameras in future recommendations.

As a result of problems like this, such a strict filtering policy is usually not employed by conversational recommender systems in practice. Instead of permanently filtering-out incompatible cases, irrelevant cases for a particular cycle tend to be temporarily removed from consideration, but may come to be reconsidered during future cycles as appropriate. Of course this strategy introduces the related problem of how past critiques should influence future recommendations, especially if they conflict or strengthen the current critique.

Current implementations of critiquing tend to ignore these issues in the blind hope that users will either behave themselves — that they will responsibly select a sequence of compatible critiques in pursuit of their target product — or that they will have the patience to backtrack over their past critiques in order to try alternatives. In our experience this approach is unlikely to prove successful. In real user trials common complaints have included the lack of consistency between successive recommendation cycles that arise because of these issues.

3 An Incremental Critiquing Strategy

Our response to the issues outlined above is to give due consideration to past critiques during future recommendation cycles using a variation of comparison-based recommendation [6] that we call *incremental critiquing* (See Figure 2). We do this by maintaining a critique-based user model which is made up of those critiques that have been chosen by the user so far. This model is used during recommendation to influence the choice of a new product case, along with the current critique. Our basic intuition is that the critiques that a user has applied so far provide a representation of their evolving requirements. Thus the set of critiques that the user has applied constitutes a type of user model ($U = \{U_1, ..., U_n\}$, where U_i is a single unit critique) that reflects their current preferences. At the end of each cycle, after the user has selected a new critique, we add this critique to the user model.



```
CB: casebase, U: user-model, q: query, t: chosen critique, r: recommended item

1.    define Incremental-Critiquing(q, CB)
2.    U = {}
3.    t = {}
4.    repeat
5.       r ← ItemRecommend(q, CB, t, U)
6.       t ← UserReview(r, CB)
7.       q ← QueryRevise(q, r)
8.       U ← UpdateModel(U, t, r)
9.    until UserAccepts(r)

10.   define UserReview(r, CB)
11.     t ← user critique for some feature ∈ r
12.     CB ← CB - {r}
13.   return c

14.   define QueryRevise(q, r)
15.     q ← r
16.   return q

17.   define UpdateModel(U, t, r)
18.     If IsCompound(t) then
19.       t-set ← UnitCritiques(t)
20.     else
21.       t-set ← {t}
22.     Endif
23.     For each t ∈ t-set
24.     do
25.       U ← U - contradict(U, t, r)
26.       U ← U - refine(U, t, r)
27.       U ← U + {<t, r>}
28.     EndFor
29.   return U

30.   define ItemRecommend(q, CB, t, U)
31.     CB' ← {i ∈ CB | Satisfies(i,t)}
32.     CB'' ← sort CB' by decreasing Quality
33.     r ← top item in CB''
34.   return r
```

Figure 2: The Incremental Critiquing algorithm

3.1 Maintaining the User Model

Maintaining an accurate user model, however, is not quite as simple as storing a list of previously selected critiques. As we have mentioned above, some critiques may be *inconsistent* with earlier critiques. For example, in the case of a PC recommender, a user selecting a critique for *more memory*, beyond the *512MB* of the recommended case, during one cycle, may later *contradict* themselves by indicating a preference for *less memory* than the *256 MB* offered during a subsequent cycle. In addition, a user may *refine* their requirements over time. They might start, for example, by indicating a preference for more than *128 MB* of RAM (with a *more memory* critique on a current case that offers *128 MB*). Later they might indicate a preference for more than *256 MB* of RAM with a *more memory* critique on a case that offers *256 MB*. In consideration of the above our incremental critiquing strategy updates the user model by adding the latest critique only after pruning previous critiques so as to eliminate these sorts of inconsistencies (see lines 23-26 in Figure 2). Specifically, prior to adding a new critique all existing critiques that are inconsistent with it are removed from the user model. Also, all existing critiques, for which the new critique is a refinement, are removed from the model. Finally, it is worth mentioning how the user model is updated with compound critiques; so far our examples have assumed unit critiques. To keep things simple, we deal with compound critiques by splitting them up into their constituent unit critiques so that the update procedure then involves making a set of unit updates.

3.2 Influencing Recommendation

The basic idea behind the user model is that it should be used to influence the recommendation process, prioritising those product cases that are compatible

with its critiques. The standard approach to recommendation, when using critiquing, is a two step one. First, the remaining cases are filtered by eliminating all of those that fail to satisfy the current critique. Next, these filtered cases are rank ordered according to their similarity to the current recommendation.

We make one important modification to this procedure. Instead of ordering the filtered cases on the basis of their similarity to the recommended case, we also compute a compatibility score for each candidate case, which is essentially the percentage of critiques in the user model that this case satisfies (see Equation 1 and note that $satisfies(U_i, c')$ returns a score of 1 when the critique, U_i. satisfies the filtered case, c, and returns 0 otherwise). Thus a case that satisfies 3 out of the 5 critiques in a user model obtains a compatibility score of 0.6.

$$Compatibility(c', U) = \frac{\sum_{\forall i} satisfies(U_i, c')}{|U|} \tag{1}$$

$$Quality(c', c, U) = Compatibility(c', U) * Similarity(c', c) \tag{2}$$

This compatibility score is then combined with the candidate's (c') similarity to the recommended case, c, in order to obtain an overall quality score; see Equation 2. This quality score is used to rank-order the filtered cases prior to the next recommendation cycle and the case with the highest quality is then chosen as the new recommendation (see lines 30-34 in Figure 2).

Of course there are many ways that we could have combined compatibility and similarity. In this work we give equal weight to both compatibility and similarity. Other weight settings may be considered; for example we might increase the weight given to past critiques as the user model grows. The essential point is, however, that the above formulation allows us to prioritise those candidate cases that: (1) satisfy the current critique; (2) are similar to the previous recommended case; and (3) satisfy many previous critiques. In so doing we are implicitly treating the past critiques in the user model as *soft constraints* for future recommendation cycles; it is not essential for future recommendations to satisfy all of the previous critiques, but the more they satisfy, the better they are regarded as recommendation candidates. Moreover, given two candidates that are equally similar to the previously recommended case, our algorithm will prefer the one that satisfies the greater number of recently applied critiques.

4 Evaluation

One of the key measures of success of any conversational recommender is efficiency in terms of session length. Recommendation sessions that present users with increasingly good recommendations are likely to have less cycles and therefore likely to lead to greater success than long ones [4, 9]. Because incremental critiquing has a much more informed user model than standard critiquing we expect that it should lead to shorter sessions. In this section we describe the

results from an efficiency evaluation of incremental critiquing by comparing its performance to standard forms of critiquing (both unit and compound) on two different datasets. The results are very positive, showing significant reductions in session-length.

4.1 Setup

Our main objective is to compare our incremental critiquing (IC) approach to the standard version of critiquing (STD). In addition we also take the opportunity to explore a minor variation to incremental critiquing, which we refer to as IC-ENABLE. This approach follows the incremental critiquing method but after each cycle it makes a second attempt to prune the user model by eliminating those critiques that are not compatible with the current recommended case; this involves minor revisions to the UpdateModel method of the algorithm presented in Figure 2. The hope is that this second pruning phase with help to maintain an improved user profile that remains better focused on the region of the product-space in the vicinity of the current recommendation.

We compare our three different recommendation algorithms (STD, IC and IC-ENABLE) by examining recommendation performance on both unit and compound critiques. To be clear, the standard approach to unit critiquing just involves the user being presented with, and selecting, unit critiques. The standard approach to compound critiquing involves the user being presented with both unit and compound critiques. The compound critiques are generated according to the dynamic critiquing strategy described in [12] and discussed briefly in Section 2.1; 5 compound critiques are presented during each cycle along with the standard unit critiques.

The evaluation was performed using two standard datasets. The PC dataset [8] consists of 120 PC cases, each described in terms of 8 features including *manufacturer, processor, memory* etc. The Travel dataset (available from *http://www.ai-cbr.org*) consists of 1024 vacation cases. Each case is described in terms of 9 features including *price, duration, region* etc.

4.2 Methodology

Ideally we would like to have carried out an online evaluation with live-users, but unfortunately this was not possible. As an alternative we chose to opt for an offline evaluation described by [8, 12, 15]. Accordingly, each case (*base*) in the case-base is temporarily removed and used in two ways. First, it serves as a basis for a set of queries by taking random subsets of its features. We focus on subsets of 1, 3 and 5 features to allow us to distinguish between hard, moderate and easy queries respectively. Second, we select the case that is most similar to the original base. These cases are the recommendation targets for the experiments. Thus, the base represents the ideal query for a user, the generated query is the initial query provided by the 'user', and the target is the best available case for the 'user. Each generated query is a test problem for the recommender, and in each recommendation cycle the 'user picks a critique that is compatible with the

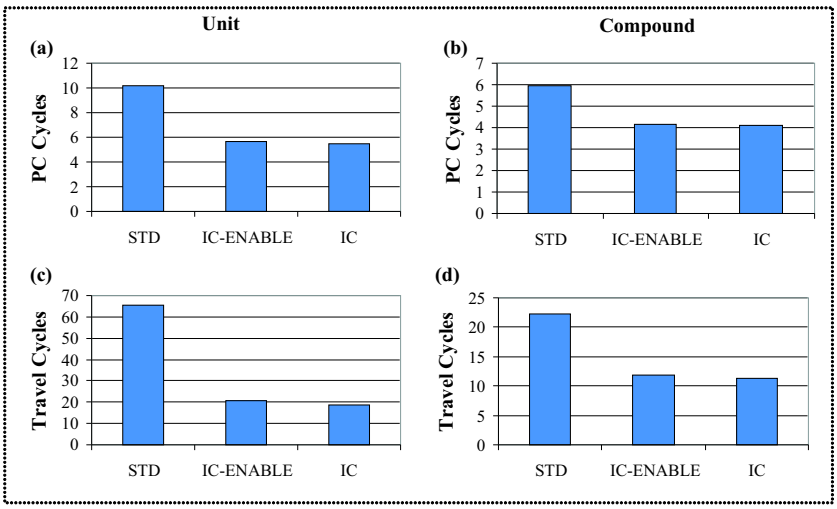


Figure 3: Efficiency results: Incremental critiquing vs standard critiquing.

known target case; that is, a critique that when applied to the remaining cases, results in the target case being left in the filtered set of cases. Each leave-one-out pass through the case-base is repeated 10 times and the recommendation sessions terminate when the target case is returned.

4.3 Recommendation Efficiency

Ultimately, we are interested in how incremental critiquing affects recommendation efficiency; that is, the number of cycles before the target case is returned to the user. In the following sets of experiments we are looking for efficiency improvements over standard approaches to unit and compound critiquing. To evaluate this, we run the leave-one-out test for each dataset (PC and Travel), on both unit and compound critiquing systems, and compare the performance of the two forms of incremental critiquing against the standard approach to critiquing. We measure the average length of the recommendation sessions for each of the different recommendation strategies, with special attention paid to recommendation sessions for different standards of query difficulty.

Figure 3 presents summary results for the two data sets on unit and compound critiquing. Each graph represents the average session length (in terms of the number of cycles) for each of the three recommendation strategies. The results show a clear and compelling benefit for the incremental critiquing variations (IC and IC-ENABLE). For example, in the PC domain, the average length of an IC session using unit critiques is 5.4 cycles, compared to 10 cycles for STD (see Figure 3(a); a 45% reduction in cycle length for IC). The results are even more striking for the larger Travel dataset. STD delivers session lengths of 65.7 cycles, on average for unit critiquing, compared to only 18.8 cycles for IC (see

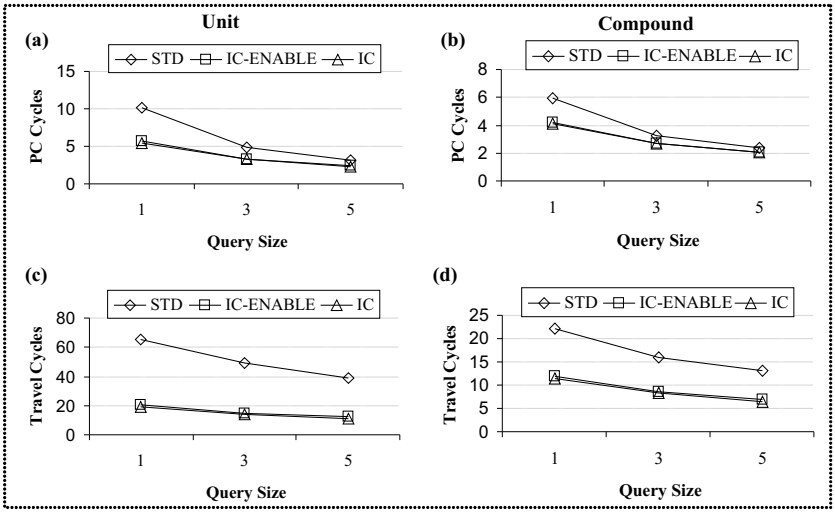


Figure 4: Efficiency results: Incremental critiquing for different query lengths.

Figure 3(c) — a 71% session length reduction). Similar benefits are observed when we look at session lengths in compound critiquing scenarios. For instance, in the PC dataset IC delivers a 52% reduction in average session length compared to STD (see Figure 3(b)) and a 48% improvement is noted in the Travel domain; see Figure 3(d). It is interesting to note that there is little difference between the performance of IC and IC-ENABLE. If anything, IC-ENABLE appears to be performing marginally worse that IC across the board, suggesting that its second pass at pruning is not contributing in a positive way to recommendation quality.

It is also worth investigating how the recommendation efficiency is influenced by the initial query length, as this is a reasonable measure of query difficulty; short queries are more ambiguous as recommendation starting points and therefore represent more of a challenge than longer, more complete queries. To do this we recomputed the session length averages above by separating out the queries of various sizes. As expected, session length decreases as the query size increases; see Figure 4. Once again, the results point to a significant advantage due to incremental critiquing but it is interesting to note that the benefit decreases as query size increases; that is, incremental critiquing has a greater advantage when the initial query is more difficult. For example in the PC domain when using unit critiques (Figure 4(a)), IC reduces session length by 46% for the most difficult queries (where only 1 feature is specified initially) but this then falls to 38% for the easiest queries (where 5 features are specified initially). For compound critiques (Figure 4(b)) this benefit drops from 37% to 15%.

This reduction in benefit is likely due to the fact that shorter sessions to begin with mean that there is less opportunity for incremental critiquing to make a

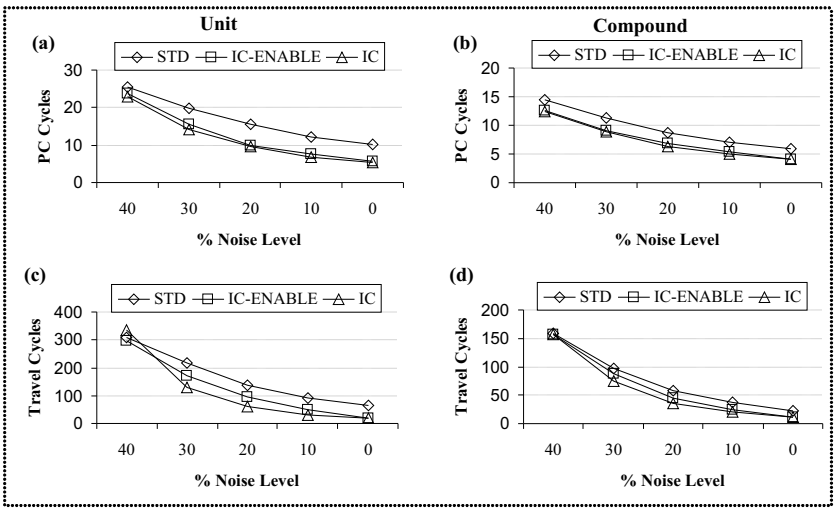


Figure 5: Critique selection noise results.

difference. Thus, as query difficulty is reduced so is the IC benefit. As we move from unit critiques to compound critiques we find a reduction in average session length, and a similar reduction in IC benefit. The IC benefit found in the Travel domain (Figures 4(c&d)) is less sensitive to query difficulty, or the change from unit to compound critiques, mainly because the Travel sessions length remain fairly long so there is plenty of opportunity for IC to exert its influence.

4.4 Critique Selection Noise

In the experiments so far we have assumed that the our artificial user will always pick a critique that is compatible with the target case. Of course this assumption is unlikely to hold up in reality. Often users will make mistakes or choose sub-optimal critique selections. The question is: how do our incremental critiquing approaches perform under such conditions. To test this, we introduce different levels of noise into the critique-selection process. For example a critique selection noise level of 10% means that at every cycle there is a 10% chance that the 'user' will choose an incorrect critique, one that is incompatible with their target. We examine noise levels of 10%—40% following the standard methodology above to produce an average session length per noise level.

The results are presented in Figure 5(a-d) as graphs of average session length (numbers of cycles) versus different noise levels. Each graph charts the performance of the three algorithmic variations—STD, IC and IC-ENABLE—and, as before, we look at both datasets in terms of unit and compound critiques. As expected, the results show general increases in session length as the noise level increases; unreliable critique-selections inevitably lead to less efficient recom-

mendation sessions as it is harder for the recommender system to focus in on the right area of the product-space. However, we do see that IC maintains a significant advantage over the standard approach in both domains for unit and compound critiques.

In the PC dataset using unit critiques only (Figure 5(a)), IC produces session lengths that vary from 23 (at the 40% noise-level) to about 6 cycles (at the 0% noise-level). The STD algorithm produces session lengths that vary from 26 cycles to just over 10 cycles across the same noise-levels. In other words, incremental critiquing reduces session lengths by between 45% (at the 0% noise-level) and 10% (at the 40% noise-level) when compared to STD. Once again the IC-ENABLE variant is seen to perform slightly worse (by about 7%) than IC in the PC dataset. On the much larger Travel dataset we notice some differences in results when compared to PC. For example, at the 40% noise-level IC actually performs marginally worse than STD for unit critiques; see Figure 5(c). However, as the noise-level drops to more reasonable levels we find that IC rapidly and significantly outperforms STD. It appears that at the highest noise level the IC user model has become swamped with many inconsistent critiques and so fails to perform as well as STD, which is not influenced by any user model. The fact that IC-ENABLE actually performs better than IC at this 40% noise-level supports this hypothesis; IC-ENABLE benefits from a user model whose inconsistent critiques have been eliminated by a second pruning stage.

This study of noise-sensitivity shows that the incremental critiquing approach still offers significant reductions in session-length even if users make many unreliable critique selections. However it is also clear that the benefits of incremental critiquing decrease as the noise level increases. In reality, it seems unlikely that users would behave so erratically as to select spurious critiques 40% of the time and so we are confident that the more significant benefits available to IC at more reasonable (10%-30%) noise-levels are likely to be realised in practice. There are of course other ways that we may be able to cope with noise too. For example, weighting functions could be used to limit the influence of less reliable critiques in the user model or to limit the influence of older critiques on the grounds that they are more likely to be unreliable.

5 Conclusions

Critiquing is an important form of user feedback that is ideally suited to many recommendation scenarios. It is straightforward to implement, easy for users to understand and use, and it has been shown to be effective at guiding conversational recommender systems. Recently we have been interested in ways of improving the efficiency of critiquing in conversational recommender systems. We have pioneered the use of compound critiques that are dynamically generated during each recommendation cycle, and we were able to show that these compound critiques have the ability to reduce session length significantly.

The motivation for the present work came from the observation that critiquing-

based recommender systems, whether based on unit or compound critiques, never appeared to explicitly consider the sequence of critiques chosen by a user during recommendation. Our studies suggest that by maintaining a record of a user's critiques we can recognise 'changes of mind', inconsistencies and refinements as the user's requirements evolve during a session, and that by doing so can better influence future recommendations within a session. The incremental critiquing strategy described in this paper attempts to capture this idea. It maintains a model of the user during a recommendation session by storing a history of their selected critiques. We have described how this type of user model can be used to provide further guidance for a conversational recommender system by considering the degree of compatibility between a recommendation candidate and the user's past critiques as a factor during recommendation.

Our evaluation studies have shown that this new incremental critiquing approach can deliver significant performance benefits, by reducing session lengths by up to 70%, under a variety of experimental conditions. These benefits have been observed in multiple domains and regardless of whether unit or compound critiquing is used. By combining our new incremental critiquing approach with our dynamic critiquing strategy [12], we have achieved overall session-length reductions (over standard unit critiquing) of about 60% in the PC domain (with session lengths falling from over 10 to 4 cycles on average) and over 80% in the Travel domain (with session lengths falling from about 65 to 12 cycles).

We believe that these session-length reductions will dramatically improve the appeal of critiquing as a standard form of feedback in a wide variety of recommendation scenarios. The improvements mean that the efficiency of critiquing is brought more into line with the efficiency of value elicitation forms of feedback, but with none of the disadvantages of value elicitation when it comes to increased user effort and additional domain expertise requirements.

References

[1] D.W. Aha, L.A. Breslow, and H. Muoz-Avila. Conversational case-based reasoning. *Applied Intelligence*, 14:9–32, 2000.

[2] R. Burke, K. Hammond, and B. Young. Knowledge-based navigation of complex information spaces. In *Proceedings of the 13th National Conference on Artificial Intelligence*, pages 462–468. AAAI Press/MIT Press, 1996. Portland, OR.

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

[4] M. Doyle and P. Cunningham. A Dynamic Approach to Reducing Dialog in On-Line Decision Guides. In E. Blanzieri and L. Portinale, editors, *Proceedings of the 5th European Workshop on Case-Based Reasoning, (EWCBR-00)*, pages 49–60. Springer, 2000. Trento, Italy.

[5] K. McCarthy, J. Reilly, L. McGinty, and B. Smyth. Thinking Positively - Explanatory Feedback for Conversational Recommender Systems. In *Proceedings of the European Conference on Case-Based Reasoning (ECCBR-04) Explanation Workshop*, 2004. Madrid, Spain.

[6] L. McGinty and B. Smyth. Comparison-Based Recommendation. In Susan Craw, editor, *Proceedings of the 6th European Conference on Case-Based Reasoning (ECCBR-02)*, pages 575–589. Springer, 2002. Aberdeen, Scotland.

[7] L. McGinty and B. Smyth. The Role of Diversity in Conversational Recommender Systems. In D. Bridge and K. Ashley, editors, *Proceedings of the 5th International Conference on Case-Based Reasoning (ICCBR-03)*, pages 276–290. Springer, 2003. Troindheim, Norway.

[8] L. McGinty and B. Smyth. Tweaking Critiquing. In *Proceedings of the Workshop on Personalization and Web Techniques at the International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 20–27. Morgan-Kaufmann, 2003. Acapulco, Mexico.

[9] D. McSherry. Minimizing dialog length in interactive case-based reasoning. In Bernhard Nebel, editor, *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 993–998. Morgan Kaufmann, 2001. Seattle, Washington.

[10] D. McSherry. Diversity-Conscious Retrieval. In Susan Craw, editor, *Proceedings of the 6th European Conference on Case-Based Reasoning (ECCBR-02)*, pages 219–233. Springer, 2002. Aberdeen, Scotland.

[11] D. McSherry. Similarity and Compromise. In D. Bridge and K. Ashley, editors, *Proceedings of the 5th International Conference on Case-Based Reasoning (ICCBR-03)*, pages 291–305. Springer-Verlag, 2003. Trondheim, Norway.

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

[13] B. Smyth and P. Cotter. A Personalized TV Listings Service for the Digital TV Age. *Journal of Knowledge-Based Systems*, 13(2-3):53–59, 2000.

[14] B. Smyth and L. McGinty. An Analysis of Feedback Strategies in Conversational Recommender Systems. In P. Cunningham, editor, *Proceedings of the 14th National Conference on Artificial Intelligence and Cognitive Science (AICS-2003)*, pages 211–216, 2003. Dublin, Ireland.

[15] B. Smyth and L. McGinty. The Power of Suggestion. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 127-138. Morgan-Kaufmann, 2003. Acapulco, Mexico.