The Needs of the Many: A Case-Based Group Recommender System*

Kevin McCarthy, Lorraine McGinty, Barry Smyth, and Maria Salamó

Adaptive Information Cluster, School of Computer Science & Informatics,
University College Dublin, Belfield, Dublin 4, Ireland
{kevin.mccarthy, lorraine.mcginty, barry.smyth, maria}@ucd.ie

Abstract. While much of the research in the area of recommender systems has focused on making recommendations to the individual, many recommendation scenarios involve groups of inter-related users. In this paper we consider the challenges presented by the latter scenario. We introduce a (case-based) group recommender designed to meet these challenges through a variety of recommendation features, including the generation of reactive and proactive suggestions based on user feedback in the form of critiques, and demonstrate its effectiveness through a live-user case-study.

1 Introduction

Recently one of the authors of this paper was trying to book a skiing holiday for a group of 4 friends. This turned out to be far more complex than first imagined, despite the prevalence of many sophisticated search and recommender systems in this domain (e.g., TripMatcher, provided by Triplehop Technologies [3], vacationcoach.com and TripAdvisor.com and also DieToRecs [13]). To begin with it was difficult to capture the many and varied preferences of the group participants. For example, most people only revealed a few of their more salient preferences at the outset and then further requirements were disclosed in the face of certain holiday suggestions later on. In addition, all of the available recommender systems in this domain really were designed for single-user usage. While they obviously provided facilities for a user to search for a package that would accommodate a group of people, it was not possible to introduce the individual sets of preferences of those involved. Instead the responsibility of combining these (often competing) preferences into a single coherent query, fell to the lead searcher [13]. All of this led to a very unnatural, not to mention extremely inefficient, search process. For example, early recommendations were passed on to the group to get individual feedback, and then this feedback needed to be integrated into a new query by our lead searcher in order to generate another batch of suggestions. This process continued for many cycles and the lead searcher had to regularly justify to others why particular options were appropriate, explaining them in the light of the preferences of others. Eventually a holiday was booked, and everyone

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

had a great time, but surely better recommendation support could have been provided.

In this paper we consider a group recommendation scenario just like the one outlined above and we describe a conversational recommendation framework that has been implemented to provide the type of support that we feel is critical. Briefly, the recommender system implements an asynchronous model of group recommendation, allowing a group of users to engage in a collaborative recommendation session via a Web-based interface[1]. This framework provides for a variety of recommendation features including the generation of reactive and proactive suggestions based on user feedback in the form of critiques. One of the critical challenges of group-based recommendation scenarios involves the development of a reliable group-preference model. We will describe how such a model is constructed by analyzing individual user feedback, and how the model works to complement the individual preference models that are maintained for each user during the selection of recommendations. In addition, it is also critically important for a group recommender to help individual users to understand the evolving preferences of the group such that they can better appreciate the compromises that may be required if a satisfactory conclusion is to be reached [6]. To this end we describe a number of innovative interfacing features that are designed to act as *consensus barometers* in order to help the group develop a shared mutual awareness of each others' preferences.

2 Background

One of the key issues that has guided the development of our group recommender concerns the type of feedback that can be solicited from individual users. In our work we are especially interested in *critiquing* (see, [2,4,8,9,14]) as a form of feedback as it strikes a useful balance between the information content of the feedback and the level of user effort or domain expertise that is required. For example, in a travel vacation recommender, a user might indicate that they are interested in a vacation that is *longer* than the one week offered by the currently recommended option; in this instance, *[duration, >, 1wk]* is a critique over the *duration* feature that can be used to filter out certain cases from consideration (i.e., those that have shorter durations) in the next recommendation cycle. Thus, the key advantage of critiquing is that it is a relatively low-cost form of feedback, in the sense that the user does not need to provide specific feature values.

Recently there has been renewed interest in critiquing, especially in product recommendation scenarios, where users have limited domain knowledge, but where they can readily provide feedback on some product features. As a result, the basic approach has been enhanced in a number of ways [1,10,11,12]. In this work we use the *incremental critiquing* technique [12] as a way to effectively leverage a user's critiquing history during recommendation. Incremental

[1] Our recommendation framework has been designed to operate with different types of interaction modalities and later we will discuss an alternative interface that is based on the Mitsubishi DiamondTouch interactive tabletop.

critiquing uses a preference model for user U that is made up of the set of critiques $\{I_1, .., I_n\}$ that have been applied by a user in a given session. As new critiques are made by the user, their preference model is updated. This may involve removing past critiques if they conflict with, or are subsumed by the most recent critique. For example, if a user had previously indicated a *Price* $<$ *\$600* critique and a new *Price* $<$ *\$500* critique is later applied then the earlier critique will be removed to reflect the users refined *Price* preference. Similarly, if a user had previously indicated a *Price* $<$ *\$600* critique but the new critique is for *Price* $>$ *\$650*, then the earlier conflicting critique is deleted. In this way the user's preference model remains a consistent reflection of their most recent preferences.

This model is then used to influence future recommendations so that they are not only compatible with the current critique (and preference case) but so that they are also compatible with past critiques so far as is possible. In the standard approach to critiquing, the most recent critique is used to *temporarily* filter out incompatible cases and a new recommendation is selected from the remaining cases on the basis of its similarity to the critiqued case (i.e., the preference case). The problem with this approach is that no account is taken of how compatible the remaining cases are with past critiques that have been applied; since the critiques only act as temporary filters over the case-base, cases which are incompatible with past critiques may be reconsidered in the future. One of the advantages of the incremental critiquing approach is that it allows candidate recommendations to be ranked not only because they are similar to the preference case but also on the basis of their compatibility with prior critiques in the form of the user's current preference model. To do this, each candidate recommendation, c', is scored according to its compatibility to the user's current preference model as shown in Equation 1. Essentially, this compatibility score is equal to the percentage of critiques in the user's model that are satisfied by the case; for example, if c_r is a \$1000 vacation case then it will satisfy a *price* critique for less than \$1200 (I_i) and so $satisfies(I_i, c_r)$ will return 1.

$$compatibility(c_r, U) = \frac{\sum_{\forall i} satisfies(I_i, c_r)}{|U|} \tag{1}$$

The *quality* of a case c_r with respect to a preference case c_p, is a weighted sum of preference similarity and critique compatibility. When a user U critiques c_p the next case recommended will be the one with the highest quality score; see Equation 2. By default, for incremental critiquing $\alpha = 0.5$ to give equal weight to preference similarity and critique compatibility.

$$quality(c_p, c_r, U) = \alpha * compatibility(c_r, U) + (1 - \alpha) * similarity(c_p, c_r) \tag{2}$$

2.1 Group Recommendation Challenges

Perhaps the most critical challenge for a group recommender system is how to develop of a comprehensive account of the evolving preferences of the group with a view to using their combined preferences to influence group recommendations. In this work we adapt the incremental critiquing approach for group recommendation. We will describe how critique histories can be combined to produce a group preference model and how future recommendations can be influenced by their compatibility with this group model. In this way we can bias recommendations towards those cases that are likely to be acceptable to the group as a whole as well as the individual participants; see Section 3 for further details.

Ultimately, for this type of recommendation to work effectively we must ensure that individual users come to appreciate their role within the group. It is natural for many users to want to maximise their own preferences, and so if left unchecked, we might expect users to proceed in ignorance of the evolving group preferences as a whole; this is especially true if users are collaborating remotely, thorough a Web interface for example. Hence one of the key challenges in this work has been to look at effective ways for the recommender system to communicate group preferences to all users, in an effort to help individuals develop a mutual awareness of their friends' preferences [5,6], with a view to encouraging compromises across the group as a whole. If users are not willing to compromise then it is unlikely that they will be satisfied with the recommendations they receive, and the only way that we can encourage compromise is by making sure that users come to appreciate the features that are important to others.

In our work this "mutual awareness" goal [6] has translated into a number of interactivity and interface features that are designed to highlight the opinions of other users and the preferences of the group as a whole. These include interactive features such as: (1) the proactive recommendation of cases (separate from the reactive recommendation in response to user critiques) that exceed a certain threshold of acceptability for the group; and (2) a facility that allows users to set aside certain cases that they feel strongly positive about so that these cases may be promoted to other group members. In addition, there are a range of visual interface elements that help each user to see the opinions and preferences of the others so that preferred cases are annotated accordingly.

3 The Collaborative Advisory Travel System (CATS)

In this section we describe how the CATS group recommender, helps a group of users to plan a skiing vacation. The CATS system described here is implemented as a Web-based client-server system with each user interacting with the system through a standard Web browser interface. In Section 5 we will briefly touch on another implementation of the system that uses a very different type of interaction technology with a view to facilitating a more natural form of collaboration between group of up to four members. In both cases, however, the core interface components remain broadly similar. To begin with we will summarize the key components of the CATS interface before describing its core user modeling and recommendation generation techniques.

Before proceeding it is worth highlighting one important point: we are not proposing the CATS system as the optimal way to offer group recommendations per se, but rather as a framework for experimenting with, and evaluating, different types of feedback, preference communication and recommendation strategies. It is important to bear this in mind when reading the following sections because many design decisions have been made in order to evaluate particular design features and recommendation strategies, rather than on the basis of a strong commitment to one particular design or strategy.

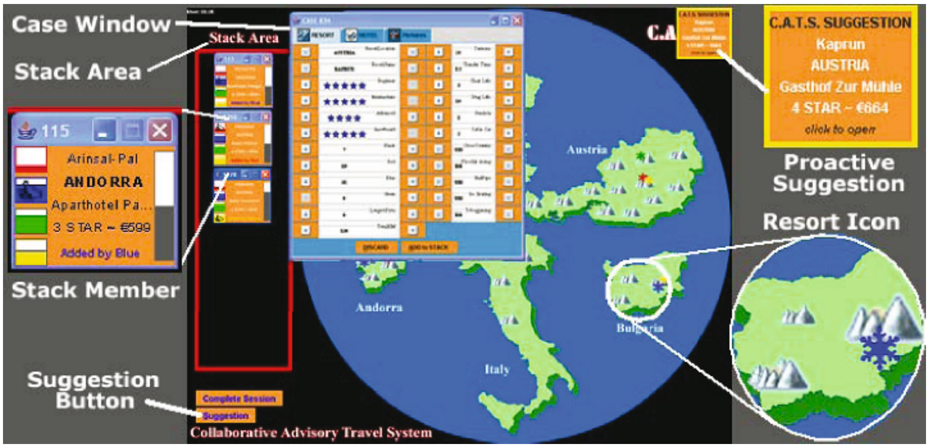


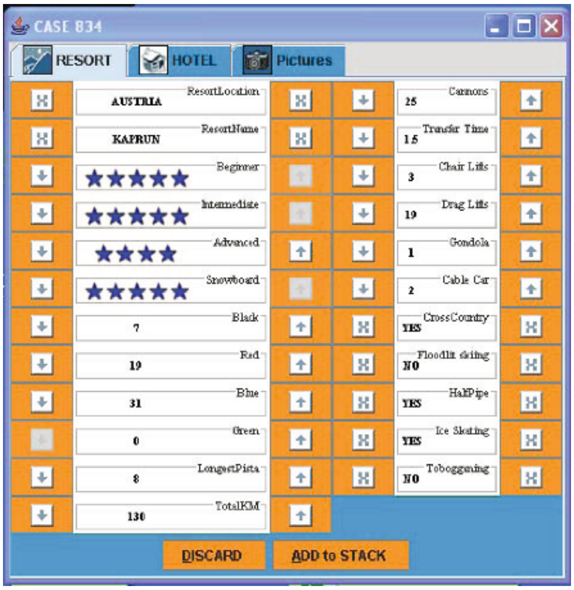
Fig. 1. The main CATS interface

3.1 The CATS Interface

Ultimately a recommender system is a way to translate the preferences of a user (or, in this case, a group of users) into a set of product suggestions. With this in mind, the CATS interface is the primary tool for capturing the preferences of individual users, communicating these preferences to the group as a whole, and then presenting the recommendations that are derived from these preferences to group members. It does this through a combination of interface elements (see Fig. 1) and recommendation techniques. In the remainder of this section we describe each of these elements in detail.

The Case Window. The most familiar element of the CATS interface, the *case window*, presents the user with a case description and some possible critiques. In CATS, each case relates to a ski package, and consists of more than 40 features (attribute-value pairs) describing various aspects of the resort and accommodation. For example, Fig. 2 shows resort information for *case 834*, describing features about the its location, ski runs/lift system, and its appropriateness for different levels of skier. The *hotel* features can be viewed from the *hotel tab* and resort photographs are also available. From this window the user has three basic options—she can *discard* the case; she can add it to the *stack area*, or she can *critique* one of its features to initiate a new recommendation—as follows:

1. Critiquing allows the user to request cases that are like the one displayed but different in terms of at least one feature. For example, our user might request a new recommendation that is *"like the one shown but with more green runs"*. The user can perform such a critique, by clicking on the relevant critique icon beside the feature, see Fig. 2. We will describe how the next recommendation is generated in more detail later.
2. Alternatively our user can decide to add the case to the stack area to indicate that she is interested in this vacation and wants to draw it to the attention of the other group members. The stack area is visible to all group members and is an important way to communicate emerging group preferences to users. We will return to this feature below.
3. Finally, the user can chose to discard the case if she is confident that she would not be at all satisfied with this vacation. Discarding the case means that this particular case will not be used as a suggestion to any of the users for the remainder of the recommendation session.



ResortLocation	AUSTRIA
ResortName	KAPRUN
Beginner	★★★★★
Intermediate	★★★★★
Advanced	★★★★
Snowboard	★★★★★
Black	7
Red	19
Blue	31
Green	0
LongestPista	8
TotalKM	130
Cannons	25
Transfer Time	1.5
Chair Lifts	3
Drag Lifts	19
Gondola	1
Cable Car	2
CrossCountry	YES
Floodlit skiing	NO
HalfPipe	YES
Ice Skating	YES
Tobogganing	NO

Fig. 2. The *case window* presents the user with a complete description of a case and is used as the starting point for collecting critiquing-based feedback from each user

The Map Window: The map window displays a graphical representation of the resorts covered by the cases in the case-base and is the initial screen users see when they begin a session. This window provides a way for users to browse through the various resort cases. Each resort is marked by a *mountain range* icon and by selecting a resort icon the user will receive a summary of the resort and a list of its cases. The user can view any case and interact with it in the normal way as described above. The map window also displays important information about the activity of group members and how well particular resorts match evolving group preferences. For instance, if a user is currently accessing a case from a particular resort, then the resort is annotated with a colour-coded *snowflake* icon; in Fig. 1 we see that a user in our group is currently accessing one of the Bulgarian resorts, for example. In addition, the size of the resort icon reflects the compatibility rating of its most group-compatible case; that is, the resort case that satisfies the most critiques contained in the group model appears larger. Thus, the map window is a vital tool for communicating the focus of group activity and preferences.

The Stack Area: Every so often a user will come across a case that they really like or that they imagine may be of interest to other group members. The user can communicate this to the group by adding the case to the stack area where it can be evaluated by other users. In a sense, this allows an individual user to play the role of *recommender* and the stack area serves as a user-based recommendation list. When a case is added to the stack it becomes a *stack member* and is displayed in summary form as shown in Fig. 1. In addition, each stack member is annotated on its left-hand side with a set of colour-coded *compatibility barometers*, each reflecting how compatible the case in question is with respect to the critiques contained in each user's individual model. For example, the stack member highlighted in Fig. 1 for 3-star accommodation in Andorra is annotated to indicate that it is very compatible with the preferences of the blue and green users, but not so compatible with the yellow user, and only marginally compatible with the red user; note that we also indicate which user added the case to the stack with the *thumbs-up* icon overlayed on their compatibility barometer. An overall group-level compatibility barometer is displayed to the right of the stack member to indicate overall compatibility with the group preference model; in this case we see that the stack member is about 50% compatible with the overall group model. These compatibility barometers are dynamically updated during the session to reflect current compatibility levels and provide another important source of preference feedback for the users. At any time any user can view and critique a stack member that has been added by someone else, but currently only the user who originally added the case can remove it from the stack. Finally, the stack area is important when it comes to delivering the final recommendation to the group at the end of the session since this recommendation will be drawn from the current stack members.

Proactive Suggestions: So far we have described two types of recommendations: those that are generated in response to user critiques and, those that are generated by the users themselves as they add cases to the stack for others to evaluate. There is a third type of recommendation: *proactive recommendation.* The CATS system is constantly comparing the group preference model to the remaining cases available for consideration; that is, cases that have not been previously viewed or discarded by any of the group members. Occasionally, one or more of these cases exceeds a certain critical *compatibility threshold* with respect to the group preference model and when this happens the most compatible case is proactively recommended by CATS to all users. For example, one such case (for a 4-star hotel in Austria) has been proactively recommended in Fig. 1 and will appear on the map window for all users where they can interact with it in the usual way. Once again this provides users with direct feedback on the evolving group preference model in an attempt to draw their collective attention towards cases that appear to maximally satisfy their preferences; we will revisit this form of recommendation in the following sections.

Completing the Session: At any time a user can request CATS to recommend another case by selecting the *Suggestion* button; we will discuss the precise mechanism for this form of recommendation below. Also, a user can terminate their session at any time by selecting the *Session Complete* button and once all users have completed their sessions the system reverts with final ranking of the stack cases according to their compatibility with the group preference model and returns the most compatible case.

3.2 Modeling Group and User Preferences

The maintenance of preference models is critical to the operation of the CATS recommender system. As discussed earlier these models are critique-based: a preference model is made up of a set of unit critiques provided by a user. CATS maintains two types of preference model. An *individual model* is maintained for each user and is equivalent to the preference models maintained in the standard form of incremental critiquing as proposed by [12]. Thus each user U is associated with an individual preference model, IM^U, that is made up of the critiques that they have submitted (see Equation 3) with conflicting and redundant critiques removed as summarized in Section 2.

$$IM^U = \{I_1, ..., I_n\} \tag{3}$$

In addition, a group preference model, $GM(U_1, ..., U_k)$, is also maintained by combining the individual user models and associating each unit critique with the user who contributed it as shown in Equation 4 such that G_i^U refers to the i^{th} critique in the preference model for user U.

$$GM^{U_1,...,U_k} = \{I_1^{U_1}, ..., I_n^{U_1}, ..., I_1^{U_k}, ..., I_m^{U_k}\} \tag{4}$$

During recommendation it will sometimes be necessary (as we will see in the next section) to leverage part of the group preference model, usually the model less some individual user's critiques. Thus we will often refer to the *partial group model* or the *members model*, MM^U, to be the group model without the critiques of user U as shown in Equation 5.

$$MM^U = GM^{U_1,...,U_k} - IM^U \tag{5}$$

This means that the group preference model is based on the preference models for individual users after they have been processed to remove inconsistent or redundant critiques. We have chosen not to repeat this processing over the group preference model and therefore it is possible, indeed likely, that the group preference model will contain conflicting preferences, for example. Of course, during recommendation these inconsistencies will have to be minimised by preferring cases that are maximally compatible with the overall group model.

3.3 Recommendation Generation

In Section 3.1 we highlighted how CATS is capable of making a number of different types of recommendations:

1. *Critiquing-based recommendations* are generated when a user critiques a case through the case window.
2. *User-requested suggestions* are generated when the user selects the *suggestion* button.
3. *Proactive recommendations* are generated when CATS locates a case that exceeds a preset compatibility threshold with the group preference model.
4. A *final recommendation* is drawn from the stack when all of the users complete their session.

Each of these recommendations is generated differently by combining individual and group preference models in different ways. This was largely an attempt to experiment with a variety of combination strategies as opposed to making a strong commitment to these specialized strategies. They may work well in practice but we are not proposing them as *best* practice.

Critiquing-Based Recommendations. This is arguably the most important source of recommendations in CATS and involves a two-step procedure. As in the standard model of critiquing [2], the first step is to temporarily filter-out cases that are not compatible with the current critique. This leads to a set of *recommendation candidates*. The standard approach to critiquing ranks these candidates according to their similarity to the critiqued case (c_p), whereas incremental critiquing uses a quality metric that combines similarity to the critiqued case and compatibility with past critiques (IM^U). Our group recommender is based on the latter but adapted to include the preferences of the other group members (MM^U) in the quality metric, as well as the preferences of the user applying the critique, to select a recommendation according to Equation 6.

$$c_{rec} = argmax_{c_r}(quality(c_p, c_r, IM^U, MM^U)) \tag{6}$$

Thus, we compute a new compatibility score, for a recommendation candidate c_r, as shown in Equation 7 and combine this with similarity to the preference case (c_p) as in Equation 8. The β parameter controls how much emphasis is placed on individual versus group compatibility while α controls the emphasis that is placed on compatibility versus preference similarity; by default we set both parameters to 0.5. In this way, the case that is recommended after critiquing c_p will be chosen because it is compatible with the critique, similar to c_p, and compatible with both the user's own past critiques and the critiques of other users. Thus we are implicitly treating past critiques as *soft constraints* for future recommendation cycles [15]. It is not essential for recommendation candidates to satisfy all of the previous critiques (individual or group), but the more they satisfy, the better they are regarded as recommendation candidates. As an aside, the binary features of a case are only considered during the similarity calculation when a user has shown a preference for that feature.

$$\begin{aligned} GCompatibility(c_r, IM^U, MM^U) = {} & \beta * compatibility(c_r, IM^U) + \\ & (1-\beta) * compatibility(c_r, MM^U) \end{aligned} \tag{7}$$

$$\begin{aligned} quality(c_p, c_r, IM^U, MM^U) = {} & \alpha * GCompatibility(c_r, IM^U, MM^U) + \\ & (1-\alpha) * similarity(c_p, c_r) \end{aligned} \tag{8}$$

User-Requested Suggestions. This type of recommendation is generated in the same way as critiquing-based recommendation in the sense that the next highest quality case is chosen. Thus the suggestion button allows the user to move down through the list of ranked cases after a critique has been chosen.

Proactive Recommendations. The idea behind this type of recommendation is as a mechanism to bring certain cases to the attention of all users if they satisfy an unusually high proportion of the group preferences. As mentioned earlier, these cases are drawn from the set of cases that have not yet been critiqued by any user or discarded and a given case is selected according to the following rule:

$$argmax_c(compatibility(c, GM^U)) \text{ iff } compatibility(c, GM^U) > 0.65 \tag{9}$$

The use of the compatibility threshold is important. It limits the frequency of proactive recommendations; after all it is unwise to interrupt users too often during the course of their session. More importantly perhaps is that the threshold also ensures that, when cases are proactively suggested, they are likely to be acceptable to all users.

Final Recommendation. Once all of the users have completed their session the CATS system recommends the case in the stack area that has the highest compatibility with the group preference model; see Equation 10. The stack area is provided for users to share their favourite cases during the session and at the end of the session the final recommendation is the one that satisfies the greatest proportion of group preferences.

$$c_{final} = argmax_{c_{stack}}(compatibility(c_{stack}, GM^{U_1,\ldots,U_n})) \tag{10}$$

4 Experimental Analysis

With CATS as very much a "work in progress" our core evaluation objective was to understand how users would respond in a group recommendation setting to the particular combination of feedback, communication, and recommendation that CATS provided. Given this objective it was clear that there would be little value in performing an off-line or artificial user study. Instead we carried out a small-scale live-user trial, the results of which are summarized in this section.

4.1 Trial Setup

As mentioned in the previous section the CATS system operates over a comprehensive case base of European skiing holidays consisting of 5700 cases, each made up of 43 different features related to the *resort* (25 features such as *country, transfer time, lift system, etc.*) and the accommodation (18 features such as *accommodation rating, price, ski room facilities, restaurant facilities, etc.*). The trialists were 3 groups of 4 computer science graduate students with varying degrees of interest and experience when it came to skiing.[2] Prior to the start of the trial we gathered some preliminary information about the preferences of each user, in order to judge the quality of the cases ultimately recommended by the group recommender. By comparing the cases selected by the members of a given group we were also able to understand the extent to which each group agreed/disagreed on various case features.

Each group of users was provided with a short demonstration of the operation of the system paying particular attention to the core features such as critiquing, the stack area and its compatibility indicators, the map and its activity icons, and proactive recommendations. They were instructed to behave as if they were really trying to plan and book a skiing holiday to go on together. As such, they were reminded that some compromises would likely be required from each user. In each trial user interactions and recommender activity was recorded. At the end of each session the users were asked to complete an extensive questionnaire covering issues such as: their satisfaction with the final case; their evaluation of the recommendations provided; the ease-of-use of the interface etc.

4.2 Behaviour Results

Overall the average session length, measured in terms of the total number of cases each user interacted with, across all three trials was just under 18. Almost two thirds (63%) of case accesses were the result of users critiquing cases (see Fig. 3(a)) with the remainder of accesses arising out of the users interacting directly with the map (14%), accessing a stack member (12%), and opening one of the occasional proactive suggestions that are made by CATS (8%). These results are encouraging in that they demonstrate that users at least made regular use of CATS' secondary case access mechanisms, although the 'suggest' button was rarely used (2% of accesses). In particular, we see that users are frequently attracted to the stack area which plays a vital role in communicating strong group preferences and also provides the source of cases for the final group recommendation. The trial data indicates that the average user places between 3 and 4

[2] All trials were conducted in the computer laboratories at the School of Computer Science & Informatics at University College Dublin, Ireland.

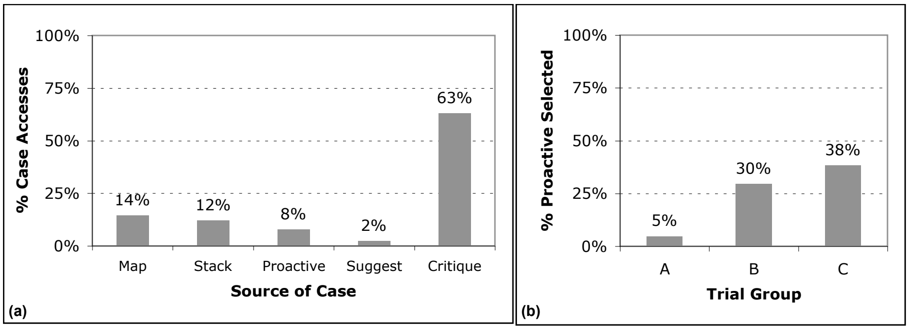


Fig. 3. (a) Source of case accesses; (b) Proactive recommendations selected

cases on the stack which corresponds to about 18% of the cases that they view. In other words, users are presented with cases that satisfy their needs about 18% of the time with other users accessing these cases about 14% of the time; thus there is a strong correlation between stack additions and selections.

It is also worth commenting on the quality of the proactive recommendations. These recommendations contributed only 8% of the cases to the average user session, but this is more a reflection of the rarity of these suggestions than their quality. Remember that by definition these suggestions are only made once a case has been found that satisfies 65% of the group's current preferences. However, when this condition is met and a proactive suggestion is made, we see that users respond to these suggestions approximately 26% of the time on average; see Fig. 3(b). Groups B and C respond positively to these suggestions 30% and 38% of the time respectively, with group A users responding only 4% of the time; in fact 3 of the 4 group A users completely ignored the proactive suggestions and on the basis of the post-trial questionnaire this seems to have been due a lack of awareness of the feature, as opposed to any negative comment on recommendation quality. Finally, it is worth supporting the above statistics with information from the post-trial questionnaire regarding the perceived quality of the recommendation received during group sessions. In particular, only 2 of the 12 users indicated they were not happy with the general quality of recommendations.

4.3 Group Compromise and User Satisfaction

Ultimately the success of any group recommender system will depend critically on its ability to identify cases that achieve reasonable compromise between the potentially competing requirements of different users. At the start of the trial we asked users to pick an example case that they would be interested in booking. They were also asked to highlight up to 5 features from this case that viewed as especially attractive. We refer to these features as the user's *initial preferences*; remember these features were not explicitly used during the trial, although it seems reasonable to assume that users may have started by looking for cases that satisfied these. During the course of the trial each user's critiques contributed to another set of preferences, which we will call their *trial preferences.*

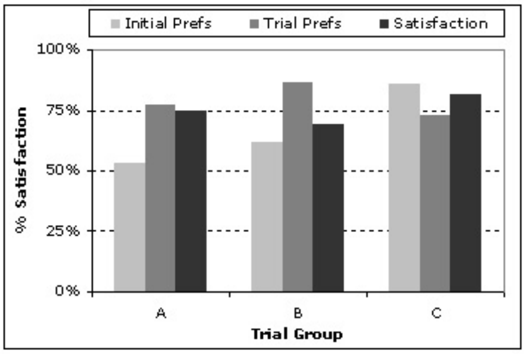


Fig. 4. The level of critique compatibility and user satisfaction with the case that is ultimately recommended to the group

We can use these preferences to evaluate the quality of the final case recommended to each group of users by measuring how many of these preferences are satisfied or contradicted by the the final case. The results (see Fig. 4) show that the final case satisfied just over 66% of the initial preferences across the 3 groups and almost 79% of the trial preferences. During the post-trial questionnaire users we asked how happy they were with the final case by rating it on a scale of 1 ("not happy at all") to 5 ("really happy"). On average the 12 users rated the final case as a 4 ("fairly happy") with 10 out of the 12 users providing a rating of at least 3 ("happy"); we have expressed these results as a percentage value in Fig. 4. These results suggest that the CATS system effectively translates the often competing preferences of a group of individual users into a recommendation that broadly satisfies the whole group.

5 Concluding Remarks

While traditional research has focused on making recommendations to the individual, many recommendation scenarios involve groups of inter-related users. Here the most critical challenge for a group recommender system is how to develop of a comprehensive account of the evolving preferences of the group with a view to using their combined preferences to influence group recommendations.

In this paper we have described an approach to asynchronous, cooperative group recommendation that: (1) uses a variety of interface cues to communicate group, as well as individual, preferences and activity, and (2) constructs a reliable group-preference model by combing critique histories in order to generate recommendations on a proactive and reactive basis. Preliminary evaluation results suggest that our approach to group recommendation effectively generates recommendations that satisfy group needs. Furthermore users responded positively to the various interface elements and recommendation strategies implemented by the CATS prototype group recommendation system.

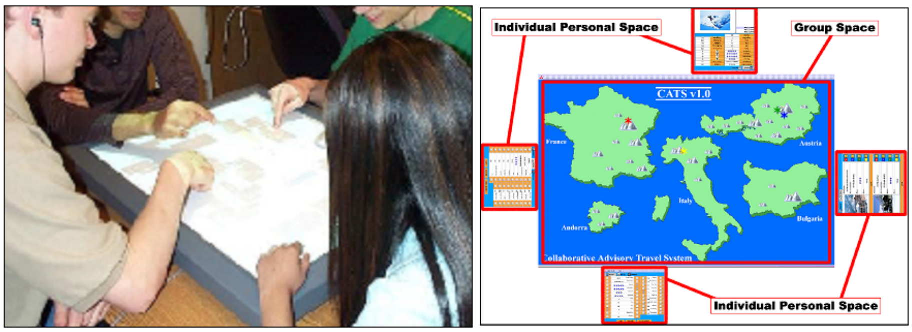

Individual Personal Space
Group Space
CATS v1.0
France
Austria
Italy
Andorra
Bulgaria
ollaborative Advisory Travel System
Individual Personal Space

Fig. 5. Illustrating the CATS interaction with the DiamondTouch

Our future work will focus on exploring a range of different strategies for combining individual preferences with a view to generating improved recommendations. This will of course include extended live-user evaluations. In addition is it worth highlighting recent work [7] that we have carried out that looks at alternative interfacing modalities for this group recommendation approach. While in this paper we have concentrated on our Web-base interface, CATS has also been implemented Mitsubishi DiamondTouch interactive tabletop. The DiamondTouch (see Fig. 5) consists of a touch sensitive tabletop display and supports the interaction of multiple simultaneous users. Its 'coffee table' form factor is ideal for supporting collaborative tasks. The CATS system interface has been adapted to offer users personal interaction spaces and a shared group space as shown in Fig. 5.

References

1. R. Burke. Interactive Critiquing for Catalog Navigation in E-Commerce. *Artificial Intelligence Review*, 18(3-4):245–267, 2002.
2. R. Burke, K. Hammond, and B.C. Young. The FindMe Approach to Assisted Browsing. *Journal of IEEE Expert*, 12(4):32–40, 1997.
3. J. Delgado and R. Davidson. Knowledge Bases and User Profiling in Travel and Hospitality Recommender Systems. In *Proceedings of the ENTER 2002 Conference*, pages 1–16. Springer Verlag, 2002. Innsbruck, Austria.
4. B. Faltings, P. Pu, M. Torrens, and P. Viappiani. Design Example-Critiquing Interaction. In *Proceedings of the International Conference on Intelligent User Interface(IUI-2004)*, pages 22–29. ACM Press, 2004. Funchal, Madeira, Portugal.
5. A. Jameson. More than the sum of its members: Challenges for group recommender systems. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pages 48–54, Gallipoli, Italy, 2004.
6. A. Jameson, S. Baldes, and T. Kleinbauer. Enhancing mutual awareness in group recommender systems. In B. Mobasher and S.S. Anand, editors, *Proceedings of the IJCAI 2003 Workshop on Intelligent Techniques for Web Personalization.* AAAI, Menlo Park, CA, 2003.

7. K. McCarthy, M. Salamó, L. Coyle, L. McGinty, and B. Smyth & P. Nixon. CATS: A Synchronous Approach to Collaborative Group Recommendation. In *Proceedings of the FLAIRS 2006 Conference*, pages 1–16. Springer Verlag, 2006. Florida, USA.
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*. Morgan-Kaufmann, 2003.
9. Q.N. Nguyen, F. Ricci, and D. Cavada. User Preferences Initialization and Integration in Critique-Based Mobile Recommender Systems. In *Proceedings of Artificial Intelligence in Mobile Systems 2004, in conjunction with UbiComp 2004*, pages 71–78. Iniversitat des Saarlandes Press., 2004. Nottingham, UK.
10. P. Pu and B. Faltings. Decision Tradeoff Using Example Critiquing and Constraint Programming. *Special Issue on User-Interaction in Constraint Satisfaction. CONSTRAINTS: an International Journal.*, 9(4), 2004.
11. 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)*, pages 763–777. Springer, 2004. Spain.
12. J. Reilly, K. McCarthy, L. McGinty, and B. Smyth. Incremental Critiquing. In M. Bramer, F. Coenen, and T. Allen, editors, *Research and Development in Intelligent Systems XXI. Proceedings of AI-2004*, pages 101–114. Springer, 2004. UK.
13. F. Ricci, K. Woeber, and A. Zins. Recommendations by Collaborative Browsing. In *Proceedings of the 12th International Conference on Information and Communication Technologies in Travel & Tourism (ENTER 2005)*, pages 172–182. Springer Verlag, 2005. Innsbruck, Austria.
14. S. Sherin and H. Lieberman. Intelligent Profiling by Example. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI 2001)*, pages 145–152. ACM Press, 2001. Santa Fe, NM,US.
15. M. Stolze. Soft Navigation in Electronic Product Catalogs. *International Journal on Digital Libraries*, 3(1):60–66, 2000.