Combining Visualization and Feedback for Eyewear Recommendation*

John Doody and **Edwin Costello** and **Lorraine McGinty** and **Barry Smyth**
School of Computer Science and Informatics
University College Dublin,
Belfield, Dublin 4.
{firstname.surname}@ucd.ie

Abstract

The importance of effective customer assistance technologies is imperative in today's online marketplace where users are oftentimes overwhelmed by the product choices available to them. Relating their subjective preferences to the precise product descriptions poses an additional challenge, one which leads us to look at how research from two complimentary research communities (recommender systems and intelligent user interfaces) can be married to improve online recommender systems. In particular, we are interested in content-based recommendation domains that rely heavily on explicit feature-level feedback to narrow the number of relevant products for a user. A user's inability or unwillingness to provide detailed fine-grained information challenges applications in these domains and as such the way in which products are presented to the users and how these products are selected for presentation must adapt to suit this type of domain and user. Here we introduce the *i*CARE System, which provides an combination of product visualization techniques and additions to the current methods of user preference extraction to recommend suitable eyeglasses to individual users.

Keywords: preference elicitation, conversational recommendation, intelligent user interfaces, e-Commerce applications.

INTRODUCTION

Much of the research that has been carried out in the area of recommender systems has focused on the statistical accuracy of the algorithms driving the systems, with little emphasis on the interface issues and the user's perspective (Bergman & Cunningham 2002; Schafer, Konstan, & Riedl 2001). However, recently there has been a surge of interest in developing applications that combine techniques and technologies from recommender systems and intelligent user interfaces (Allen *et al.* 2001). For example, systems like ExpertGuide (Shimazu, Shibata, & Nihei 2002) emphasize interaction and recommendation when it comes to providing intelligent sales support. These systems rely on the user interface to both garner user preference information and display suggestions, and they rely on the recommender engine to narrow down the range of relevant alternatives.

At face value, this integration of technologies is certainly beneficial but, depending on the domain that is being addressed and the information that is available, certain recommender technologies tend to be more suitable than others. For instance, the collaborative filtering approach to the recommendation task has proven to be very effective in e-Commerce domains where detailed content descriptions relating to the recommendation items are unavailable (Herlocker, Konstan, & Riedl 2000). In other recommendation scenarios, detailed feature-rich descriptions of products are available and so these are more suited to the content-based recommendation approach, which relies heavily on feature-level user feedback (see for example, (Burke, Hammond, & Young 1997; Pu & Kumar 2004)).

A key problem with content-based recommenders is the underlying assumption that users are readily able (and willing) to describe their needs and preferences in terms of the content descriptions that are available to the recommender. Related research indicates that this is not always possible for a variety of reasons (Ardissono & Goy 2000; Felix *et al.* 2001; Grenci & Todd 2000; Shimazu, Shibata, & Nihei 2002). Product domain examples such as jewelery, clothing, technology, and art, are especially challenging. For each of these examples, detailed content descriptions of the recommendation items are usually readily available, but users are often unable to understand and map how these relate to their subjective needs. This amounts to a vocabulary problem in content-based recommenders (often due to limited user expertise of domain characteristics), which renders available content descriptions effectively useless. Take, for example, an online user seeking to purchase a diamond engagement ring. While a multitude of distinctive features describe every engagement ring, the stone makeup itself is usually the most important aspect. Features that characterize the stone include, *weight, color, carat, clarity, cut, girdle, fluorescence, rest, polish*, to name but a few (see Figure 1). Importantly, these features are readily available and these are the precise characteristics that influence recommendation for experts in the domain (e.g., jewelers) [1].

*This publication has emanated from research conducted with the financial support of Science Foundation Ireland.

[1] In the 1950's, Gemological Institute of America Inc. (GIA) created the International Diamond Grading/Reporting System. By

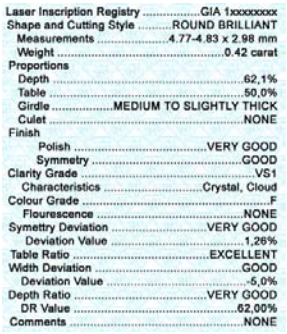

Figure 1: General description of a certified diamond.

The crucial point is that the majority of users may be unable to provide a recommender with such precise feature-specific feedback; feedback that it *needs* to influence retrieval. This severely limits the recommender's ability to narrow down the range of suitable alternatives; thus compromising system efficiency. Keeping these points in mind we limit our focus to one such domain, and discuss how we integrate ideas from conversational recommender systems and product visualization through the interface of an intelligent customer assistant. We introduce the *i*CARE system which focuses on a domain where the number of recommendation items outweighs the users ability to survey them all. Importantly, structured content descriptions are available for all recommendation items, in the form of feature-value representations. However, few of these features are likely to be meaningful/highly influential to the user, and so they are unable to provide crucial feedback to the recommender system in terms of these specific features. It is also a domain where a users subjective appreciation of the *effect* brought about by a recommendation has an enormous influence. Specifically we are looking at recommending eye-wear (i.e., frames for eye glasses), the choice of which lends itself well to the illustration of how product visualization combined with a users high-level preferential feedback can be very beneficial. In addition to the visualization element we look at other ways that users feedback can be used to inform the retrieval and help users find what they want more quickly.

THE *i*CARE SYSTEM

The *i*Care System (i.e., Intelligent Customer Assistance for Recommending Eyewear) is an online system that allows users to shop for suitable glasses (i.e., frames). The overview of its current functionality is as follows: (1) a user can upload their picture to the system, (2) the *i*Care system processes the image using effective feature-detection algorithms in order to pin-point the precise location and dimensions of the users eyes, (3) the user enters into a conversational dialogue with the system, where the system recommends frame options over a series of recommendation cycles. The user can see these frames on their uploaded picture and provide feedback (see Figure 2) , (4) the *i*Care system uses their feedback to adapt the behavior of the recommendation component and influence subsequent retrievals. In addition, the user can interact with *i*Care in a variety of other ways. The following subsections provide an overview of the basic *i*Care system architecture. More precise technical details relating to the components at the heart of the *i*Care application will be discussed in later sections.

this system the characteristics of every diamond are precisely described by a lengthy set of technical feature-value pairs and a quality grading. Quality certified diamonds are the result. This both protects the consumer and the jeweler from fraud and helps determine the price of any particular diamond.

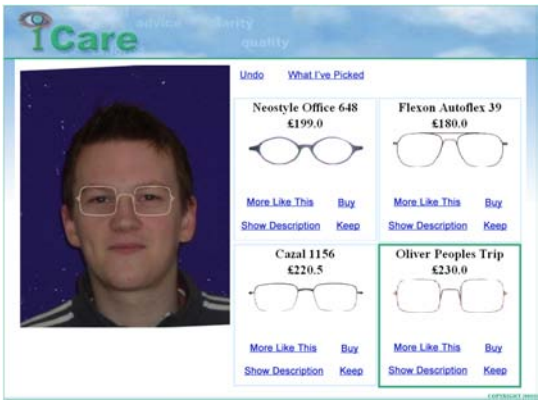

Figure 2: The *i*Care System Interface.

Overview of the Basic Architecture

The *i*Care system (see Figure 3) has three key component layers; (1) the dataset layer, (2) the application layer, and (3) the user-interfacing layer.

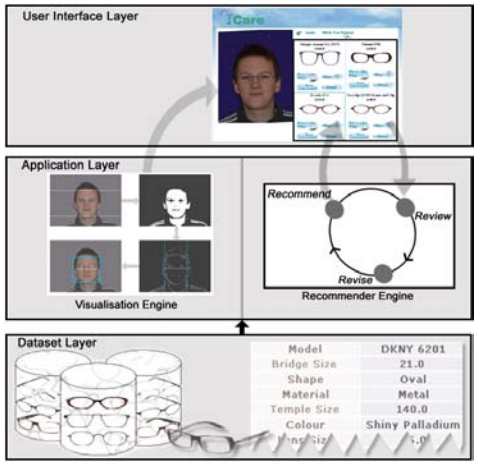

Figure 3: Basic *i*Care System Architecture.

Product Representation The data representation layer stores detailed content descriptions for 3061 pairs of glasses (i.e., glasses cases). The partial product *case*, shown in Figure 3, is representative of how descriptions are represented by the data-set layer. Each case description is represented

by a list of *attribute-value* instances. Examples of the nominal and numeric descriptive features common to all cases include *price*, *shape*, *lens size*, *bridge-size* and *material*. In total, each case is described by 12 individual feature-value pairs. In addition, a corresponding image of the product option (i.e., frames) that each case describes is also stored.

Application Overview Central to the functioning of the *i*Care system is the application layer. There are two crucial components here: the product recommendation engine and the product visualization engine. The recommender engine is responsible for retrieving relevant product recommendations in view of user feedback, and routing these to the interfacing layer. Importantly, the recommendation approach supports a conversational interaction between a user and the system, based on the comparison-based framework (McGinty & Smyth 2002). The visualization component, on the other hand, is responsible for image processing within the *i*CARE system. It provides for facial feature detection at the image upload stage and precise product placement during the 'Try On' stage of each cycle. Further technical detail relating to both of these component is provided in later sections.

User Interface The primary role of the user interface layer is to handle message passing between the user and the application layer, and display the outputs of the recommender and visualization engines. By design, the *i*Care user interface is clear and intuitive; the left-hand side of the interface is dedicated to visualization interaction and the right-hand side is reserved for the display and review of product recommendations (see Figure 2). Aside from having the opportunity to see the *visual effect* for each recommendation, the user also has the opportunity to directly apply a range of further image adjustments as they feel necessary. Examples include image tilts, zoom-in, zoom-out etc. In addition, the user may review the technical descriptions that relate to each recommendation alternative, or backtrack to an earlier recommendation cycle.

The provision of product visualization as well as product descriptions is useful as it caters for both novice and expert users. Product visualization is useful at the start of the session where a user can get a good idea of the style of glasses that suit them best without having to provide exact values for specific technical features. Later in the recommendation session the user may indeed wish to consult the recommendation descriptions to better appreciate the trade-offs between *neck-n-neck* alternatives (e.g., price differences could be influential in their final purchase decision).

Ultimately, the *i*Care system will be accessible to users online through a retailers website, or in-store through a kiosk/interactive screen. SpecSavers Optical Group Ltd. [2], for example, already offer their in-store customers the kiosk-based opportunity to try on different frame options, while having their picture taken and displayed at the same time. Importantly, the service they provide does not allow users to interact in any other way, nor do they have access to product data or capability of seeing suggestions; they simply take pictures and display these for the customers information. The current version of the *i*Care system requires the user to interface through a standard web browser (e.g., Netscape, MS Explorer, Mozilla Firefox), and upload their own facial image manually. Subsequent user interfacing revisions will involve automating the image capture and upload process (through a web-cam or otherwise), and/or incorporating the system with an in-store display.

[2] A well-known European-based opticians.

IMAGE PROCESSING & VISUALIZATION

Facial analysis and product visualization are key components of *i*CARE. This section focuses on the technical details relating to facial feature detection at the image upload stage, and precise product placement at the 'Try On' stage of each interaction cycle.

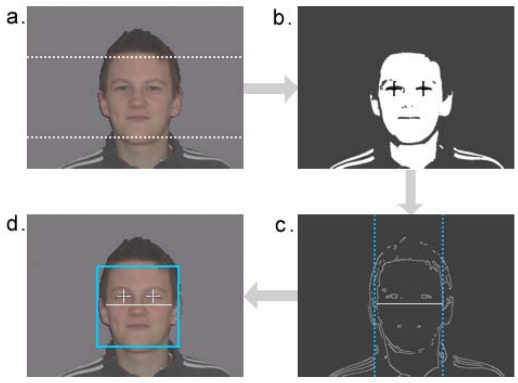

Figure 4: Illustrating the facial detection steps.

Determining Facial Dimensionality

Facial feature detection algorithms are used to detect the location of the eyes in the image as well as the width, height and skin tone of the face. The approach we took was initially proposed by Neeta Parmar (Parmar 2002). The process involves converting the image into a monochromatic format which leaves the face predominately white with black areas around the mouth, nose and eyes (see Figure 4b). The next step is to look for changes in the density of black pixels over each row of the image. The row with the greatest density of black pixels is the row corresponding the the center of the eye.

Estimating the height and width of the face involved the application of a number of separate convolutions to the image. Each convolution applies a kernel over all image pixels. Each kernel provides a matrix of weights that are applied to the target pixel's neighbors in order to compute a new target pixel value, $g(x, y)$ according to Equation 1. For instance a brightnes/contrast convolution is used to highlight edges prior to the application of the canny edge detection convolution (Canny 1986).

$$g(x,y) = \sum_{k=-n2}^{n2} \sum_{j=-m2}^{m2} h(j,k) f(x-j, y-k) \quad (1)$$

Once the facial edges have been computed, the width of the face is estimated by the width of a line, parallel with the line connecting the two eyes, drawn 30 pixels below the eyes. A similar method is used to determine the height of the face (see Figure 4c).

Product Visualization

When the user wishes to 'Try On' a pair of glasses we use the information gathered in the facial detection phase to prepare the frames for positioning on the users face. The initial stage is to determine the angle and distance between the two eyes (see Equations 2 and 3, where $(x1, y1)$ and $(x2, y2)$ are the locations of the two eyes in the image).

$$Distance = \sqrt{(x2 - x1)^2 + (y2 - y1)^2} \quad (2)$$

$$Angle = \tan^{-1}(\frac{y2 - y1}{x2 - x1}) \quad (3)$$

Using these values we resize the glasses to fit the face by ensuring the distance between the center of the lenses and the distance between the eyes are equal. Once this work is done it is a simple matter of superimposing the pixels from the glasses onto the the image of the user in the correct location.

CONVERSATIONAL RECOMMENDATION

The conversational recommender engine we have implemented supports an iterative interaction with the user, providing them with cyclic feedback opportunities to influence retrieval. The assumptions we make here are: (1) users are capable of recognizing *what they like when the see it*, (2) users are willing to provide a minimal preference information for products they like in order to see more suitable recommendation results.

The basic algorithm behind the approach we have implemented is provided in Figure 5, and can be summarized as follows: (1) new items are *recommended* to the user based on the current query; (2) the user *reviews* the recommendations and indicates which option they prefer; (3) information about the difference between the selected item and the remaining alternatives is used to *revise* the query for the next cycle. The recommendation process terminates when the user is presented with a suitable recommendation.

Recommendation and Review

Before the recommender can recommend the user with the k most similar cases to their most recent preference for review, the remaining product cases are ranked in decreasing order of their similarity to the current query, Q, according to the Equation 4.

$$sim(Q, C) = \frac{(\sum_{i=1}^{n} featureSim(F_{Qi}, F_{Ci}))}{n} \quad (4)$$

For nominal features an exact match comparison is carried out, returning the value 1 when the values match, and 0 otherwise. Numeric values, on the other hand, use their relative difference as a basis for similarity calculation. The equation for this is shown in Equation 5 where F_Q and F_C are the values for the numeric features being compared.

$$featureSim(F_Q, F_C) = 1 - \frac{\mid F_Q - F_C \mid}{max(F_Q, F_C)} \quad (5)$$

```
1.    define Comparison-Based-Recommend(Q, CB, k)
2.    begin
3.      Do
4.        R ← ItemRecommend(Q, CB, k)
5.        c_p ← UserReview(R, CB)
6.        Q ← QueryRevise(Q, c_p, R)
7.      until UserAccepts(c_p)
8.    end
9.    define ItemRecommend(Q, CB, k)
10.   begin
11.     CB' ← sort cases in CB in decreasing order of their sim to Q
12.     R ← top k cases in CB'
13.     return R
14.   end
15.   define UserReview(R, CB)
16.   begin
17.     c_p ← user selects best case from R
18.     CB ← CB - R
19.   return c_p
20.   define QueryRevise(Q, c_p)
21.   begin
22.     for each f_i ∈ c_p
23.       Q ← add f_i
24.     end for
25.     return Q
26.   end
```

Figure 5: Comparison-Based Recommendation.

In each recommendation cycle the user need only provide high-level preference-based feedback, (largely based on visual preference). They do this by clicking the *More Like This* option associated with the recommendation alternative that they feel suits them best. This feedback is provided in the review stage of each recommendation cycle (see lines 15-19 of Figure 5). Importantly, there is a direct mapping between the high-level feedback they provide, and the technical feature-based item descriptions available to the system. Hence, the following section focuses on the technical details of *how* this low-cost feedback is utilized (at the query revise stage) in order to influence retrievals in the next cycle.

Cumulative Query Revision

Lines 20 to 26 of Figure 5 summarize one way the the *i*Care recommender uses to update its understanding of a user's personal requirements (i.e., the evolving query) at the revise stage of each recommendation cycle. Here the preference case (indicated by the user) in each cycle serves as the query for the next set of retrievals. This is the traditional *More-Like-This* approach often used in online conversational recommenders. One restriction of this approach is that it focuses on the most recent recommendation cycle (i.e., feature preferences provided by a user), and does not take into account any preferential information provided over the preceding cycles. As an extension to existing work in this area we describe *two* new alternative strategies that focus on using prior experience to adapt (i.e., revise) the query from cycle to cycle, based on the cumulative feedback collected as the recommendation session proceeds. As you would expect, these strategies involve revisions to the **QueryRevise** procedure of the comparison-based recommendation algorithm, and are described in the following subsections.

Cumulative Feature Occurrence (CFO) As users make their way from cycle to cycle they make a preference-based decision and choose one case from a set of recommendations. As the recommendation process continues two sets of cases are built up, one contains their preference case for each cycle (i.e., *PreferredCase(i)*), and the other contains the

cases they rejected (i.e., *RejectedCases(i)*), for each cycle. Importantly, these sets contain implicit *frequency of occurrence* information that can provide valuable input for subsequent query revisions and retrievals. Ideally, our algorithm will recognize and prioritize feature values that have been repeatedly preferred by the user.

A straightforward method of attempting to extract this information is as follows: as a user makes choices throughout their session, a count is kept of how many times a particular feature value, v, occurs in each of their preference cases. As a user proceeds, a feature will have a value that occurs in their preference frequently and will build up a larger count than others.

$$\textit{Preferred(i,f,v)} = \begin{cases} 1 & \textit{if (f,v) in PreferredCase(i)} \\ 0 & \textit{Otherwise} \end{cases} \quad (6)$$

$$\textit{Rejected(i,f,v)} = \begin{cases} -1 & \textit{if (f,v) in RejectedCases(i)} \\ 0 & \textit{Otherwise} \end{cases} \quad (7)$$

In addition to this, the values contained in the rejected cases are also monitored such that if a feature value occurs in one of these rejected cases, its count value is decremented. Equations 6 and 7 outline how the count for a feature value is affected if it is contained in either a preference or a rejected case. So the count of a value v for the feature f, across all cycles 1 to n will be as follows;

$$Count(f,v) = \sum_{i=1}^{n} Preferred(i,f,v) + \sum_{i=1}^{n} Rejected(i,f,v) \quad (8)$$

The count value kept can become negative if it is rejected more than it is chosen from cycle to cycle. Importantly, the rejection of a feature value is only counted once in a cycle, even if it occurred in all the rejected cases for that cycle. To penalize a value for being involved in 2 rejected cases for example, would mean that it would need to be in the preference case twice again before breaking even, not taking into account any additional rejected occurrences.

So, when revising the query for the next recommendation cycle the feature values that have the largest count are transferred into the new query. In the event of a tie, one of the values with the joint highest count is chosen randomly, and for features where the count is negative or zero that feature is left empty so it will not influence similarity calculations and retrieval. This means that only features that suggest a definite majority user preference are included. Also, feature values that the user has shown a consistent dislike to are pushed down so that a once off choice where that feature value is included in a chosen case will not have an immediate affect on recommendations.

Proportional CF0 (pCFO) In the basic CFO method only one instance of a rejected feature is accounted for per cycle. To refine this outlook on rejected features, a variation of CFO was implemented. Instead of decreasing the overall score for a feature by 1 if it is rejected, a more fine grained approach is to take into account how many times it is rejected per cycle. So for a cycle where k cases are presented to the user and she chooses a preference case, the number of rejected cases is $k-1$. Therefore for each of the features in these rejected cases, the scores for the values are now decreased by $1/(k-1)$. To reflect this, Equation 7 now changes to the following;

$$\textit{Rejected(i,f,v)} = \sum_{j=1}^{k} RejectedCase(j,f,v) \quad (9)$$

with

$$\textit{RejectedCase(j,f,v)} = \begin{cases} \frac{-1}{k-1} & \textit{if (f,v) in Case(j)} \\ 0 & \textit{Otherwise} \end{cases} \quad (10)$$

and where *Case(j)* in *RejectedCases(i)*. So if a feature is in all the rejected cases and in the preference case its overall score remains unchanged. On the other hand if the feature is only rejected in 2 out of, for example, 3 cases then the features score is increased by 1/3.

Experimental Evaluation

A key criteria in gauging the effectiveness of conversational applications of this kind is recommendation efficiency (i.e., the number of recommendation cycles a user must go through before they reach a target product). In this section we investigate how the cumulative query revision strategies compare against the traditional *More-Like-This* alternative.

Setup and Methodology

For all of our evaluations we used the glasses dataset described earlier. In order to generate a set of test queries, 500 random cases are selected to serve as *seed* cases. For each of these seed cases, random queries of lengths 2 (difficult), 5 (moderate) and 8 (easy) were generated. A specific *target* case (i.e., the most similar to the seed) was then appointed. All queries were solved using a leave-one-out methodology for all the 500 query cases. For clarity, the preference case in each recommendation cycle was deemed to be that which is most similar to the target. The experiments were conducted for varying values of k (i.e., the number of recommendations returned during each recommendation cycle) from 2 to 10 in increments of 2 with the session terminating when the target case was presented.

Recommendation Efficiency Results

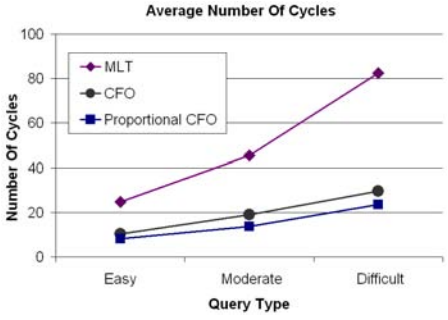

Figure 6: Evaluation results looking recommendation efficiency across query types.

Figure 6 shows how the two algorithms compared over a range of query difficulties, easy, moderate and difficult. We find that the basic CFO approach achieves a significant efficiency advantage over the stand-alone MLT approach recording an overall improvement on MLT of over 61%. The pCFO variation shows even larger benefits recording an overall improvement over MLT of 70%. Importantly the relative benefits (over MLT) enjoyed by the cumulative algorithms seem to increase with query difficulty (see Figure 7). Figure 6 shows that CFO leads to session reductions of between 58%(easy queries) and 64% (difficult queries). pCFO shows a similar pattern of results with further benefits of between 67% and 71% for easy and more difficult queries respectively.

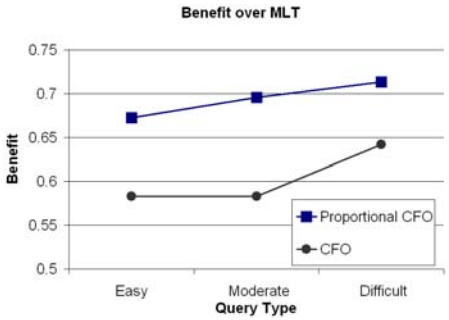

Figure 7: Overall Benefits for each method over MLT and for each query Type

CONCLUSION AND FUTURE WORK

A key challenge for recommender systems research in the area of e-commerce is the accurate modeling of user preferences in the course of a *once-off* recommendation session (Ricci & del Missier 2004). To date, work in this area has focused on implementing query revision strategies that use *only* information collected in an individual recommendation cycle. Importantly, the preference decisions made by the user in the preceding cycles do not influence subsequent case retrievals.

In this paper we propose and evaluate two alternative query revision strategies that revise the recommender system's understanding of what the user is looking for based on the high-level, preference-based *cumulative feedback* they provide. We show how this low-cost feedback can be translated and utilized by a conversational recommender through the interface of an intelligent customer assistant and show how the use of these session-dependent query revision strategies can provide benefits over purely cycle-dependent ones. Our evaluation results indicate that even very basic approaches to cumulative query revision can lead to very significant reductions in terms of the number of cycles a user must engage in before they find their ideal target. In summary, the *i*Care System allows users to shop for suitable glasses by providing the facility for users to visualize and appreciate the *effect* of each recommendation option (i.e., see how different frame options actually look on them). Users need not be relied upon to understand or describe precise features as they relate to their preferences. Instead this is implicity captured by the feature-level item descriptions (i.e., the causes that bring about the effects).

References

Allen, J.; Byron, D.; Dzikovska, M.; Ferguson, G.; Galescu, L.; and Stent, A. 2001. Towards conversational human-computer interaction. In *AI Magazine*, volume 22(4), 27–38.

Ardissono, L., and Goy, A. 2000. Tailoring the interaction with users in web stores. In *User Modelling and User-Adapted Interaction*, volume 0(4), 251–303.

Bergman, R., and Cunningham, P. 2002. Acquiring customers' requirements. *Artificial Intelligence Review* 18(3-4):163–193.

Burke, R.; Hammond, K.; and Young, B. 1997. The findme approach to assisted browsing. *Journal of IEEE Expert* 12(4):32–40.

Canny, J. 1986. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.* 8(6):679–698.

Felix, D.; Niederberger, C.; Steiger, P.; and Stolze, M. 2001. Feature-oriented vs needs-oriented product access for non-expert online shoppers. In *Proceedings of the First IFIP Conference on e-Commerce, e-Business and e-Government*, 399–406.

Grenci, R. T., and Todd, P. 2000. Solutions-driven marketing. *Communications of the ACM* 45(3):209–249.

Herlocker, J. L.; Konstan, J.; and Riedl, J. 2000. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM Sonference on Computer Supported Cooperative Work*, 241–250.

McGinty, L., and Smyth, B. 2002. Comparison-based recommendation. In Ed. Craw, S., ed., *Proceedings of the Sixth European Conference on Case-Based Reasoning (ECCBR-02)*. Aberdeen, Scotland: Springer.

Parmar, N. 2002. Drowsy driver detection system. Master's thesis, Department of Electrical and Computing Engineering, Ryerson University.

Pu, P., and Kumar, P. 2004. Evaluating example-based search tools. In *Proceedings of the ACM Conference on Electronic Commerce (EC'04)*, 208–217.

Ricci, F., and del Missier, F. 2004. Supporting travel decision making through personalized recommendation. *Designing Personalized User Experiences in eCommerce* 231–251.

Schafer, J. B.; Konstan, J.; and Riedl, J. 2001. E-commerce recommendation applications. *Data Mining and Knowledge Discovery* 5(1-2):115–153.

Shimazu, H.; Shibata, A.; and Nihei, K. 2002. Expert-guide: A conversational case-based reasoning tool for developing mentors in knowledge spaces. *Applied Intelligence* 14(1):33–48.

Smyth, B., and McGinty, L. 2003. The power of suggestion. In *Proceedings of IJCAI*, 127–132.