History-Guided Conversational Recommendation

Yasser Salem
School of Electronics, Electrical Engineering and Computer Science
Queen's University Belfast
Belfast BT7 1NN, UK
ysalem01@qub.ac.uk

Jun Hong
School of Electronics, Electrical Engineering and Computer Science
Queen's University Belfast
Belfast BT7 1NN, UK
j.hong@qub.ac.uk

Weiru Liu
School of Electronics, Electrical Engineering and Computer Science
Queen's University Belfast
Belfast BT7 1NN, UK
w.liu@qub.ac.uk

ABSTRACT

Product recommendation is an important aspect of many e-commerce systems. It provides an effective way to help users navigate complex product spaces. In this paper, we focus on critiquing-based recommenders. We present a new critiquing-based approach, History-Guided Recommendation (HGR), which is capable of using the recommendation pairs (item and critique) or critiques only so far in the current recommendation session to predict the most likely product recommendations and therefore short-cut the sometimes protracted recommendation sessions in standard critiquing approaches. The HGR approach shows a significant improvement in the interaction between the user and the recommender. It also enables successfully accepted recommendations to be made much earlier in the session.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrievals**]: On-line Information Services—*Web-based services*

Keywords

Conversational Recommendation, critiquing, Recommender Systems

1. INTRODUCTION

Recommender systems help users select suitable items from a large collection of items with a range of features. Online shopping is one area in which recommender systems are useful for giving shoppers better choices [24]. Recommender systems aim to help users find the item they are looking for as quickly as possible with the fewest number of steps. Today, e-commerce systems rely on recommender systems to help users to navigate complex product spaces. Amazon's use of recommendation technologies is well documented [8]. The recent Netflix competition [1] highlights the value of sophisticated recommendation techniques in the commercial world. Many recommendation techniques have been investigated by the research community, from *collaborative filtering* methods [5, 9], which rely on simple ratings-based profiles to generate recommendations from similar users, to *content-based* methods [23], which use detailed product knowledge to make recommendations. Most recommender systems currently deployed use the *single-shot* strategy. They produce a ranked list of recommendations for the user [7, 19, 21]. The single-shot strategy is suitable for the recommendation of simple products and services, such as ringtones, movies, books, etc., but it is not so well suited for recommending items with complex attributes. In the latter case, it is more effective to offer the user with an opportunity to provide feedback, and to refine their needs and preferences. This has motivated researchers to develop so called *conversational recommendation* approaches [2, 13, 14, 22]. In these approaches, users participate in a *recommendation dialogue*, receiving recommendations and providing feedback in order to receive a new set of improved recommendations. Different approaches to conversational recommendation use different types of feedback. One type of feedback that forms the basis of this work is *critiquing*. Critiquing is a simple form of feedback which allows a *"show me more like this but ..."* type of response. Recently there has been renewed interest in critiquing [3, 4, 10, 11, 15, 16, 18] because of its advantages: it is simple to implement for a variety of interface types, and it can be appropriate for users who are not experts in a given product domain. Critiquing-based recommender systems have proven to be an effective approach to conversational recommendation. However they have a tendency to produce protracted recommendation sessions, due in large part to the limited feedback that critiques can provide. So researchers have recently concentrated on improving the performance of critiquing-based recommender systems [11, 12, 20, 26]. In this paper, we are interested in improving the efficiency of critiquing-based recommender systems without introducing additional critiquing complexity. The starting point is that critiquing histories, or experiences of past users, carry important information about feature preferences and trade-offs, and these experiences can be used to bias the recommendation process.

2. RELATED WORK

Recommender systems are a common way to promote products or services that may be of interest to a user, usually based on some profile of interests. The single-shot approach, which produces a ranked list of recommendations, is limited by design. It works well when a user's needs are

WWW'14 Companion, April 7–11, 2014, Seoul, Korea.
ACM 978-1-4503-2745-9/14/04.

clear, but it is less suitable when a user's needs are not well known, or where they are likely to evolve during the course of a session. In these scenarios it is more appropriate to engage the user in a recommendation dialogue so that incremental feedback can be used to refine recommendations. This type of conversational recommender systems are much better suited to help users navigate more complex product spaces. Various forms of feedback are used in conversational recommender systems: value elicitation, ratings-based feedback, preference-based feedback and critiquing [25]. A recent approach to critiquing, *incremental critiquing* (IC)[17], maintains a user model made up of the user's critiques so far in a recommendation session. The user model is then used to guide the selection of new recommendations in the session. For example, suppose a camera shopper has already provided a number of critiques. The user model may indicate that the shopper is looking for a *DSLR* camera that is *cheaper than $500*, *manufactured by Nikon*. [17] describes how a user model is generated from such a sequence of critiques. When the user provides a new critique, perhaps looking for a *resolution higher than 5 mega-pixels*, instead of just selecting a new recommendation based on its similarity to the current recommendation and its compatibility with its critique, the user model is further used to rank all candidate recommendations. The end result is that the new recommendation is guaranteed to maximally match the user's critiques so far (via the user model). It has been shown that this approach produces more focused recommendations that lead to significantly shorter recommendation sessions. A more recent approach to critiquing [20], *History-Aware Critiquing* (HAC), is in spirit similar to incremental critiquing. Instead of allowing the previous critiques of the current user to influence the selection of the new recommendation, it uses the critiques of past users.

3. HISTORY-GUIDED RECOMMENDATION

Inspired by ideas in case-based reasoning, our proposed History-Guided Recommendation technique attempts to harness a new source of knowledge during the critiquing process, namely the critiquing experiences of other users. We will focus on past critiquing sessions that have been successful – in the sense that they have led to an accepted recommendation. Our basic assumption is that these successful experiences must encode useful patterns of recommendations and critiques, which may help us to short-cut the standard critiquing process for future users. In what follows we will describe the HGR technique to leverage these experiences as part of a conventional critiquing-based recommender system and we will go on to demonstrate the potential of these experiences to significantly improve the efficiency on incremental critiquing (IC) and History-Aware Critiquing (HAC).

3.1 Recommendation Sessions

In a typical critiquing session the user will start with a high-level understanding of their needs. For example, when choosing a restaurant they might start by indicating a price range and a location. During the course of a session this will be refined, as the user critiques the features of recommended restaurants, perhaps indicating that they are looking for somewhere that is less formal but more expensive than earlier recommendations. Thus, during a particular critiquing session a user may provide feedback on a range of different features.

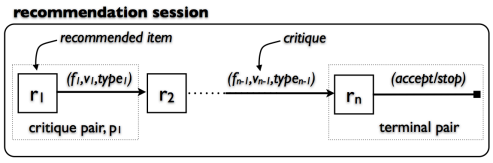

Figure 1: A recommendation session made up of a sequence of recommendation-critique pairs.

We can model each recommendation session, s_i, as a sequence of *recommendation-critique pairs*, as shown in Figure 1 and Equations 1-2, where each r_i represents a recommendation and c_i is the critique that is applied by the user to the recommendation. Each c_i is represented as a triple, $(f_i, v_i, type_i)$ as shown in Equation 4, where f_i refers to the feature $f_i \epsilon r_i$ that the critique applies to, v_i is the value of f_i in r_i ($r_i.f_i$), and $type_i$ is the type of critique that is applied (typically, $type_i \epsilon \{<, >, =, <>\}$). For now we can assume that each session terminates (as shown in Equation 3)when the user chooses to *accept* a recommendation, indicating that they are satisfied with the recommendation, or when they choose to *stop* a given session, presumably because they have grown frustrated with the recommendations received. Thus we can add *accept* and *stop* to the set of permissible critique types such that every session terminates with one or other of these types.

$$s_i = \{p_1, ..., p_n\} \tag{1}$$

$$p_i = (r_i, c_i) \tag{2}$$

$$terminal(s_i) = p_n = (r_n, c_n) \tag{3}$$

$$c_i = (f_i, v_i, type) \tag{4}$$

In general, a critiquing-based recommender system has many users who will produce a large collection of critiquing sessions ($S = \{s_1, ..., s_k\}$) as they engage with the recommender system. The sessions reflect the *experience* of these users and capture potentially useful knowledge about their preferences and the different trade-offs they tend to make. In this paper, we are interested in the potential for these experiences to inform the recommendation process itself. In other words, these critiquing sessions are the cases in a case base of critiquing experiences. For the remainder of this paper we will assume that only *successful sessions* — that is, those sessions where the user *accepts* a recommendation — are stored in the case base. We can then treat the accepted recommendation as the *solution* of the case and the critique pairs that proceed it as the *specification* of the case. We will describe how to harness these critiquing experiences to improve the efficiency of the recommendation process by using a new critique-based recommendation approach, called History-Guided Recommendation (HGR), which differs from the traditional approach to critiquing in the manner in which new recommendations are generated; see Figure 2 for a brief overview.

3.2 Conventional Critiquing

In a conventional critiquing-based recommender system, when a user applies a critique c_i to an item r_i, the recom-

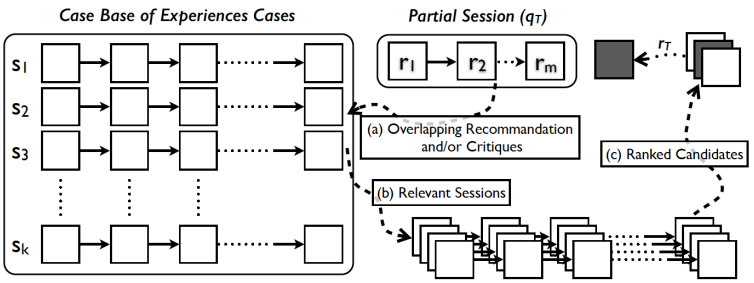

Figure 2: History-Guided Recommendation (HGR) reuses past successful sessions.

mender responds by retrieving an item, r_T, which is compatible with c_i, in the sense that the item satisfies the critique c_i, and which is maximally similar to r_i, as in Equations 5-6. Note that $r.f$ represents the value of feature f in recommended item r. $apply(type, u, v)$ is true if and only if the predicate denoted by $type$ is satisfied by the values u and v; for example, $apply(<, 25, 40)$ is $true$ whereas $apply(=, casual, formal)$ is not.

$$r_T = \underset{\forall\, r_j \epsilon\ items\ \wedge\ satisfies(c_i, r_j)}{argmax} \Big(sim(r_i, r_j) \Big) \tag{5}$$

$$satisfies(c_i, r_j) \leftrightarrow apply(type_i, r_j.f_i, r_i.f_i) \tag{6}$$

3.3 Harnessing History-Guided Recommendation

History-Guided Recommendation extends conventional critiquing by reusing past sessions to guide the critiquing process. Instead of retrieving a new item that is maximally similar to the current recommendation, and compatible with the user's critique, we recommend one of the items that past users have ended up purchasing/accepting in similar critiquing sessions. This can be best understood in terms of three basic steps: (1) identifying past critiquing sessions that are similar (i.e., relevant) to the current session; (2) ranking recommendation candidates from the terminal items of these similar sessions; (3) filtering the ranked candidates to eliminate those that do not satisfy the current user's own critiques.

3.3.1 Identifying Relevant Critiquing Sessions:

When a user applies a critique c_i to a recommended item r_m we will use the user's current (partial) critique session, $(r_1, c_1), ...(r_m, c_m)$, as a query (q_T), over the case base of past critique sessions, in order to identify a set of *relevant sessions*; see (a) and (b) in Figure 2. Briefly, a relevant session is one which has at least some overlap with the current query (see Equation 9, where typically $t = 0$), based on a particular overlap metric. In this case, we propose to use the simple overlap score defined in Equation 7, which computes the number of recommendation pairs (item and critique) in q_T that are also present in the current session. If there is no relevant session, and thus there is no candidate to recommend, then we use Equation 8, which computes the number of critiques only in q_T that are also present in the current session. Finally we use Equation 9 to get the relevant sessions. Note that in the case that there is no relevant session and hence no candidate to recommend, we revert to incremental critiquing (IC) and retrieve a new item that is maximally similar to the current recommendation and compatible with the critique.

$$OverlapScore(q_T, s_i) = \sum_{(r_i, c_i) \in q_T} \sum_{(r_j, c_j) \in s_i} match((r_i, c_i), (r_j, c_j)) \tag{7}$$

$$OverlapScore(q_T, s_i) = \sum_{c_i \in q_T} \sum_{c_j \in s_i} match(c_i, c_j) \tag{8}$$

$$S^{REL} = RelevantSessions(q_T, S) = \Big\{ s_i \epsilon S : OverlapScore(q_T, s_i) > t \Big\} \tag{9}$$

3.3.2 Ranking Recommendation Candidates:

These relevant sessions (S^{REL}) have previously led a user to a successful outcome. Each relevant session terminates with an accepted recommendation r_F, which forms a candidate for the next recommendation in the current session. Generally speaking, an accepted recommendation may be associated with more than one past session. Intuitively, it makes sense to give preference to a recommended item that is accepted in more relevant sessions. Therefore, we can rank these recommendation candidates based on their score as defined in Equation 10; see (c) in Figure 2.

$$RecScore(r_F, q_T, S^{REL}) = \sum_{\{\forall s_i \epsilon S^{REL} : r_F = terminal(s_i)\}} OverlapScore(q_T, s_i) \tag{10}$$

3.3.3 Filtering Conflicting Candidates:

There is no guarantee that the ranked recommendation candidates are compatible with the user's critiques. Thus, in the final step we eliminate incompatible recommendation candidates. The simplest way to do this is to eliminate those candidates that fail to satisfy *all* of the user's critiques so far, $(r_1, c_1), ...(r_k, c_k)$, in the current session. However, this is not ideal since in many cases, users may change mind during a session, and as a result the critiques in the session may be conflicting [17]. For example, a user might start by looking for a product that is *cheaper than $100* only to later shift towards looking for a product in the $100 - $150 range, once they start to recognise the different tradeoffs that exist in the target product space. So eliminating recommendation candidates in the $100 - $150 range in the

current session, based on the earlier critique, would be inappropriate. Accordingly we edit the current user's critiques by working backwards through the session starting with the most recent critique. If a critique conflicts with a more recent critique (that has already been processed) then it is eliminated. This leaves a set of core critiques which represent the *boundaries* of the user's preferences with respect to the features that have been already critiqued. Items that do not satisfy the core critiques are eliminated from the ranked list of recommendation candidates and the remaining top-ranked candidate is selected, as r_T, and recommended to the user.

4. EVALUATION

In conventional critique-based recommendation systems, new recommendations are influenced by the sequence of critiques in the current session. These critiques help to focus the recommender within the recommendation space. According to the new technique presented in this paper, the critiquing experiences of other users can also play a role in guiding the session. We evaluate this new technique, by comparing it to conventional critiquing, using a number of different performance metrics. To do this we have developed a restaurant recommender system, based on a comprehensive database of tripadvisor restaurants.

4.1 Datasets

For the purposes of the evaluation, two datasets are used: *restaurants* as recommended items and *experience cases* as recommendation sessions of past users. The restaurant dataset [6], contains a total of 10,000 restaurants that were crawled from an online restaurant database. Each restaurant is represented by 39 different features (e.g. price, quality, etc.), including 3 nominal features, 14 numeric features, and 22 binary features. We divided the 10,000 restaurant into *6,000, 8000 and 10,000 restaurant groups* to test HGR's performance with a few thousands to 10 thousand restaurants. Ideally, we would like to be able to evaluate HGR using real users. However, this is not currently feasible since it would require a major live deployment over an extended period of time. In the alternative, we adopt the approach taken by [11, 16, 12, 20] to automatically generate critiquing sessions based on the behaviour of rational users, using the standard approach to critiquing described in Section 3.2. We do this by selecting a random restaurant as our *target*. From this target we automatically create a *query*, by selecting 2-3 features from the target at random, which acts as a starting point for each session. Each session begins by the recommender retrieving a best-matching restaurant for the query. From here the 'user' must select a feature to critique. To do this we automatically select one of the features of the recommended restaurant and critique it *in the direction* of the target restaurant. For example, if the target restaurant has a price range of \$20-\$30 and the retrieved case has a price range of \$40-\$50, then if the price feature is selected for critiquing we will apply the *cheaper* ($<$) critique. Moreover, features are selected for critiquing based on a probability model that favours nominal and numeric features over binary features to simulate a more realistic critiquing session. Each session terminates once the target case has been recommended. We can repeat this process to generate an arbitrary number of critiquing sessions. In the case of this experiment, we generate between 4-5 different queries for each of the 10,000 restaurant cases to generate a total of a million distinct critiquing sessions. These sessions can then be used as the case base for our HGR technique.

4.2 Algorithms & Methodology

We are interested in comparing the performance of an incremental critiquing (IC) recommendation algorithm [17] and HAC [20] to our HGR algorithm. We generate a separate set of 500 target problems as our *test set* by using the aforementioned technique to generate a query-target pair. Next, we 'solve' each target problem, simulating the actions of a rationale user using: (a) IC; (b) HAC; and (c) the HGR approach. In the case of the latter we leverage different sized case bases and overlap thresholds. A target problem is deemed to be solved once the problem's target restaurant has been recommended, at which point we note the session length.

4.3 Results

The key performance issue to consider is whether the HGR technique leads to earlier target recommendations, when compared to IC or HAC, and thus shorter sessions? If it does then this can lead to some very tangible benefits both for the user and for the recommendation service provider, since all other things being equal, shorter sessions mean less effort for the user and improved conversion rate for the service provider. HGR achieves significant performance than IC and HAC. HGR achieves much more reduction in session length; for the largest case base, the average session length of HGR is reduced to under 9 cycles (compared to 35 for IC) as shown in Figure 3, a relative reduction of more than 74% compared to IC and 37% compared to HAC. Note that the benefits were calculated with 10,000 cases in the case base. The first point to notice is the performance of the IC approach to critiquing. On average its sessions extend to 35 steps. In reality it would be a rare user that would tolerate the need to provide feedback almost 35 times before receiving an acceptable recommendation. In Fig. 4, HGR has much more benefits over IC across various case bases size, ranging from 27% for smaller case bases (500 cases) to over 64% for full case base of 10,000 cases. The benefits of HGR over HAC grows from 8% when there are 500 possible cases in the case bases space up to over 30% when the full case base are available. Clearly, this benefit is not independent of case base size. Fewer cases mean fewer relevant sessions are available as basis for ranking and selection. Figure 5 shows the average session lengths of HGR, IC and HAC with reverting back to IC. HGR achieves better reduction in session length; for the largest case base, the average session length of HGR is reduced to 10 cycles (compared to 24 cycles for IC and 13 cycles for HAC). We looked at the benefits of HGR over IC and HAC as the complexity of the product space increases. In our experiments, the complexity is approximated by the number of restaurants available for recommendation and the benefits are calculated on the basis of reduction in session length, that is, how much reduction HGR achieves over IC and HAC. As shown in Figure 6, HGR has much more benefits over IC across various complexities of the product space, ranging from 58% to about 74% when the full set of 10,000 restaurants are available in the product space. Figure 6 also shows how much reduction HGR achieves over HAC, the benefits of HGR over HAC grows from 21% when there are 6,000 possible restaurants in the product space up to over

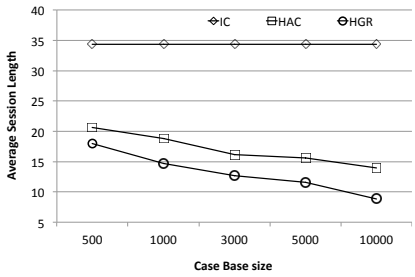

Figure 3: The average session lengths of HGR compared to IC and HAC (using 10,000 restaurants)

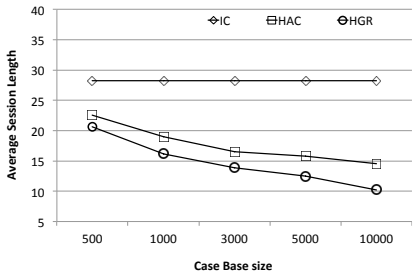

Figure 4: The average session lengths of HGR compared to IC and HAC (using 8,000 restaurants)

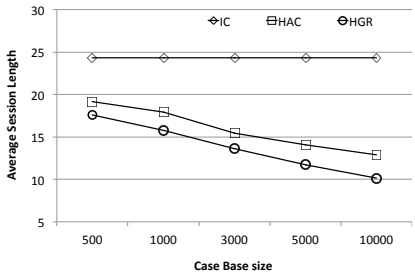

Figure 5: The average session lengths of HGR compared to IC and HAC (using 6,000 restaurants)

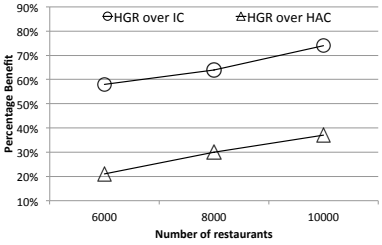

Figure 6: Percentage of reduction of session length of HGR over IC and HAC with different complexities of product spaces

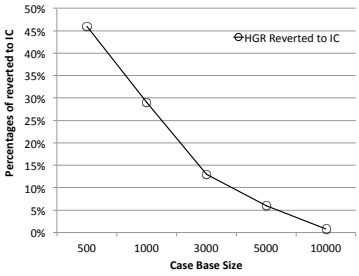

Figure 7: The percentages of sessions when reverted back to IC

37% when the full case base are available. Note that the benefits were calculated with 10,000 cases in the case base. We also examined how often HGR cannot find a relevant recommendation session from the case base and thus needs to revert back to IC. As shown in Figure 7, HGR reverts to IC in 46% of instances with a case base of 500 cases and down to 0.8% at the largest case base size. Note that there is clearly a correlation between the percentage of sessions in which HGR reverts to IC and session length when there are only 500 cases in the case base. It is useful to consider the efficiency of HGR in terms of its ability to deliver shorter sessions than IC, with reverting back to IC only in less than 1% of the sessions when there are 10,000 cases in the case base.

These results show that History-Guided Recommendation has significant benefits compared to conventional critiquing-based recommendation. When dealing with a large set of products the History-Guided Recommendation approach can lead to sessions that are much shorter than those produced by incremental critiquing and History-Aware Critiquing. This means a significant time saving for users, bringing them to the target recommendation in less than half the steps needed by conventional critiquing. This is likely to lead to a significant improvement in successful sessions.

5. CONCLUSIONS

Critiquing-based recommendation techniques are useful when it comes to helping users to navigate complex product spaces. However, they can lead to protracted sessions and a high session-failure rate for users. While conventional approaches have been extended to deliver more efficient recommendation sessions (e.g. [11, 16, 18]), these extensions typically introduce an additional cost to the user, often in the form of a more complex interface and/or feedback options. Our goal in this work has been to improve the efficiency of critiquing-based recommender systems, but without introducing additional interface components and/or costs for the user. To this end we have described the History-Guided Recommendation approach, which reuses a case base of prior critiquing experiences on the ground that these past experiences encode important users preferences and feature trade-offs that may help to improve recommendation efficiency. We have described how these past experiences can be used to influence recommendation generation and the results of a comprehensive offline evaluation demonstrate the potential benefits of this experience-based recommendation approach.

6. REFERENCES

[1] J. Bennett and S. Lanning. The Netflix Prize . In *Proceedings of the KDD Cup and Workshop*, 2007.

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

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

[4] L. Chen and P. Pu. Critiquing-based recommenders: survey and emerging trends. *User Model. User-Adapt. Interact*, 22(1-2):125–150, 2012.

[5] C. Desrosiers and G. Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender Systems Handbook*, pages 107–144. 2011.

[6] KDE-Group. Big-dataset contains a total of 9945 restaurants. available on http://www.qub.ac.uk/research-centres/kde/. 2013.

[7] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. GroupLens: Applying Collaborative Filtering to Usenet News. *Communications of the ACM*, 40(3):77–87, 1997.

[8] G. Linden, S. Hanks, and N. Lesh. Interactive assessment of user preference models: The Automated Travel Assistant. In C. P. A. Jameson and C. Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference*, pages 67–78. Springer Wien, 1997.

[9] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, Jan. 2003.

[10] M. Mandl and A. Felfernig. Improving the performance of unit critiquing. In J. Masthoff, B. Mobasher, M. C. Desmarais, and R. Nkambou, editors, *UMAP*, volume 7379 of *Lecture Notes in Computer Science*, pages 176–187. Springer, 2012.

[11] K. McCarthy, J. Reilly, and B. Smyth. On the generation of diverse compound critiques. In *Proceedings of the 15th Artificial Intelligence and Cognitive Science Conference (AICS-04)*, 2004.

[12] K. McCarthy, Y. Salem, and B. Smyth. Experience-Based Critiquing: Reusing Critiquing Experiences to Improve Conversational Recommendation. In *Proceedings of the 18th International Conference on Case-Based Reasoning, (ICCBR-2010)*, volume 6176 of *Lecture Notes in Computer Science*, pages 480–494. Springer, 2010. Alessandria, Italy.

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

[14] D. McSherry. Incremental Relaxation of Unsuccessful Queries. . In P. A. G. Calero and P. Funk, editors, *Proceedings of the European Conference on Case-Based Reasoning (ECCBR-04)*, pages 331–345. Springer, 2004.

[15] P. Pu and B. Faltings. Decision Tradeoff Using Example-Critiquing and Constraint Programming. *Constraints*, 9(4):289–310, 2004.

[16] J. Reilly, K. McCarthy, L. McGinty, and B. Smyth. Dynamic critiquing. In P. Funk and P. González-Calero, editors, *Proceedings of the Seventh European Conference on Case-Based Reasoning (ECCBR-04)*, volume 3155 of *Lecture Notes in Computer Science*, pages 763–777. Springer, 2004.

[17] J. Reilly, K. McCarthy, L. McGinty, and B. Smyth. Incremental critiquing. *Knowledge-Based Systems*, 18(4-5):143–151, 2005.

[18] J. Reilly, J. Zhang, L. McGinty, P. Pu, and B. Smyth. A comparison of two compound critiquing systems. In *IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces*, pages 317–320. ACM, 2007.

[19] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM Conference on Computer-Supported Cooperative Work (CSCW 94)*, pages 175–186, North Carolina, USA, August 1994. ACM Press.

[20] Y. Salem and J. Hong. History-Aware Critiquing-Based Conversational Recommendation. In *Proceedings of the 22nd international conference companion on World Wide Web WWW2013*, pages 63–64. ACM, 2013. Rio de Janeiro, Brazil.

[21] U. Shardanand and P. Maes. Social Information Filtering: Algorithms for Automating "word of mouth". In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems (CHI '95)*, pages 210–217, Denver, Colorado, United States, 1995. ACM Press/Addison-Wesley Publishing Co.

[22] H. Shimazu. ExpertClerk : Navigating Shoppers' Buying Process with the Combination of Asking and Proposing. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 1443–1448. Morgan Kaufmann, 2001.

[23] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(12):1349–1380, Dec. 2000.

[24] B. Smyth. *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, chapter Case-Based Recommendation, pages 342–376. Springer, 2007.

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

[26] J. Zhang and P. Pu. A comparative study of compound critique generation in conversational recommender systems. In *Proceedings of 4th International Adaptive Hypermedia and Adaptive Web-Based Systems Conference (AH-06)*, pages 234–243, 2006.