Quality Enhancement Based on Reinforcement Learning and Feature Weighting for a Critiquing-Based Recommender

Maria Salamó[1], Sergio Escalera[1,2], and Petia Radeva[1,2]

[1] Dept. Matemàtica Aplicada i Anàlisi, Facultat de Matemàtiques, Universitat de Barcelona, Gran Via de les Corts Catalanes 585, 08007, Barcelona, Spain
{maria,sergio,petia}@maia.ub.es
[2] Computer Vision Center, Dept. of Computer Science, Universitat Autònoma de Barcelona, 08193, Bellaterra, Spain

Abstract. Personalizing the product recommendation task is a major focus of research in the area of conversational recommender systems. Conversational case-based recommender systems help users to navigate through product spaces, alternatively making product suggestions and eliciting users feedback. Critiquing is a common form of feedback and incremental critiquing-based recommender system has shown its efficiency to personalize products based primarily on a quality measure. This quality measure influences the recommendation process and it is obtained by the combination of compatibility and similarity scores. In this paper, we describe new compatibility strategies whose basis is on reinforcement learning and a new feature weighting technique which is based on the user's history of critiques. Moreover, we show that our methodology can significantly improve recommendation efficiency in comparison with the state-of-the-art approaches.

1 Introduction

Conversational case-based recommender systems guide user through a product space, alternatively making product suggestions and eliciting user feedback [2,9,3,21]. Recommender systems can be distinguished by the type of feedback they support; examples include *value elicitation*, *ratings-based feedback* and *preference-based feedback* [22]. In this paper, we are especially interested in a form of user feedback called *critiquing* [5,14], where a user indicates a directional feature preference in relation to the current recommendation. For example, in a travel/vacation recommender, a user might indicate that she is interested in a vacation that is *longer* than the currently recommended option; in this instance, *longer* is a critique over the *duration* feature.

As part of the recommendation process, conversational systems aim to retrieve products that satisfy user preferences at each cycle. It is expected that over the course of a recommendation session, the recommender learns about user preferences, and therefore, the system could better prioritize products [15,5,13,20].

L. McGinty and D.C. Wilson (Eds.): ICCBR 2009, LNAI 5650, pp. 298–312, 2009.

In this sense, we focus on incremental critiquing [17], which has shown to enhance the recommendation efficiency prioritizing products based on a quality measure. This quality measure is obtained by the combination of compatibility and similarity scores.

In this paper, we consider that compatibility and similarity may improve quality measure taking into account user preferences. In the literature, the compatibility score [17] is essentially computed as the percentage of critiques in the user model that a case satisfies. We argue that the moment in which a critique was made is important enough to influence the compatibility score, and thus, in the final quality measure. Note that the user increases her knowledge of the domain along cycles and her preferences are more accurate over time. In particular, previous work on this direction showed that using a simple Monte Carlo reinforcement learning strategy to compute the compatibility score obtains better case quality results [18].

Reinforcement learning (RL) [24,11] is concerned with how an agent ought to take actions in an environment. Common applications of RL techniques are related to robotics and game theory [16]. In the context of recommenders, an initial attempt to include RL techniques has been performed on web-recommendation systems where a possible analysis of finite-state Markov decision process based on pages links is possible [10]. However, in the content-based recommendation domain, it is quite difficult to infer which are possible good future actions since the environment changes with the decisions of the user. For instance, suppose that the user initially is looking for a particular video camera. The initial expectations may change with the learning process of the user while navigating in the recommendation system. While navigating, the user notices that she needs to spend more money to obtain the required product performance, and thus, critiques change. Because of this reason, instead of looking for RL techniques that predict based on how an action affects future actions, in this paper, we are going to focus on RL techniques which are based on the user specialization. This specialization is grounded on past actions, where the time of the action is closely related to the user critiquing compatibility. We review different state-of-the-art RL techniques based on Monte Carlo and Time Dynamics, and also propose two new RL techniques adapted to conversational recommender systems.

Moreover, we also argue that quality is influenced by similarity. Conversational case-based recommender systems use a similarity function (usually based on nearest neighbor rules) to recommend the most similar product at each cycle [15]. Nevertheless, similarity functions are sensitive to irrelevant, interacting, and also most preferred features [1]. This problem is well-known in Case-Based Reasoning (CBR) systems because it can degrade considerably the system performance. In order to avoid it, many similarity functions weight the relevance of features [25,12]. Previously, in the work of [19], a local user preference weighting (LW) was presented, which was shown to reduce the number of critiquing cycles. In this paper, we present a global user preference weighting (GW). This method basis on the satisfied critiques from the whole set of cases. We show how the new

weighting strategy enhances the quality, and results in a shorter session length than using the local user preference weighting.

Summarizing, this paper describes new strategies for compatibility and weighting based on user's critiquing history for enhancing quality. The paper is organized as follows: Section 2 overviews the incremental critiquing approach as the baseline to present our methodology. Section 3 describes state-of-the-art RL approaches applied to conversational CBR and presents two new approaches adapted to critiquing. Section 4 introduces the new methodology to weight the similarity component of quality, and Section 5 presents the experimental evaluation of the presented strategies. Finally, Section 6 concludes the paper.

2 Background

The incremental critiquing [17] implementation assumes a conversational recommender system in the style of Entrée [4]. Each recommendation session starts with an initial user query resulting in the retrieval of a product p (also known as case) with the highest *quality*. The user will have the opportunity to accept this case, thereby ending the recommendation session, or to critique it as a means to influence the next cycle. The incremental critiquing algorithm consists of four main steps: **(1)** a new case p is *recommended* to the user based on the current query q and previous critiques; **(2)** the user *reviews* the recommendation and applies a directional feature critique, cq; **(3)** the query, q, is *revised* for the next cycle; **(4)** the user model, $U = \{U_1, .., U_i\}$, $i \leq t$ is updated by adding the last critique cq and pruning all the critiques that are inconsistent with it. Finally, the recommendation process terminates either when the user retrieves a suitable case, or when she explicitly finishes the recommendation process.

This recommendation process is highly influenced by the user model U containing previous consistent critiques, which is incrementally updated at each cycle. Incremental critiquing modifies the basic critiquing algorithm. Instead of ordering the filtered cases on the basis of their similarity to the recommend case, it also computes a *compatibility* score C as follows:

$$C_t^{p'}(U) = \frac{\sum_{\forall_i:(1\leq i\leq t)} \delta(p', U_i)}{|U|} \tag{1}$$

where $C_t^{p'}(U)$ is the *compatibility* score of candidate case p' at time t given an user model U. The satisfaction function δ returns 1 if case p' satisfies the critique U_i or 0 otherwise, and $|U|$ stands for the total number of critiques in the user model U. Thus, the compatibility score is essentially the percentage of critiques in the user model that case p' satisfies. Then, the compatibility score and the similarity of a candidate case p' to the current recommendation p are combined in order to obtain an overall *quality* score Q:

$$Q(p', p, U) = \beta \cdot C_t^{p'}(U) + (1 - \beta) \cdot S(p', p) \tag{2}$$

where S is the similarity function, and β is set to 0.75 by default. The quality score Q is used to rank the filtered cases prior to the next cycle, and the case with the highest quality is then chosen as the new recommendation.

3 Compatibility Using Reinforcement Learning

In this section, we analyze the compatibility component of the quality measure. As shown in [18], RL techniques can enhance compatibility efficiency. Thus, we review and propose RL techniques that are used as new compatibility scores to conversational CBR systems.

Among the different classes of RL families that exists in literature, we find Dynamic Programming Methods. These strategies are difficult to adapt to our problem since a complete and accurate model of the environment is required, and we are not able to predict future behavior of the user in the recommendation system [24]. On the other hand, Monte Carlo methods do not require a model, and are conceptually simple. Finally, temporal-difference methods (TD) also do not require a model, and are fully incremental, though they are more complex to analyze. Thus, both TD and Monte Carlo methods seem to be useful to use the user experience in order to solve the prediction problem, and retrieve the optimal product to the user reducing the number of critiquing cycles. In our case, we want to model the current compatibility $C_t^{p'}$ of a candidate case p' at instant t based on its corresponding previous compatibility. For this task, the initial RL model for compatibility computation can be a simple Monte Carlo method [18]:

$$C_t^{p'} = C_{t-1}^{p'} + \alpha \cdot \left(R_t^{p'} - C_{t-1}^{p'} \right) \tag{3}$$

This Monte Carlo method is also called *constant-*α MC [24]. The term $R_t^{p'}$ is the satisfaction of case p' at time t (i.e., $R_t^{p'} = 1$ if the candidate case p' satisfies the current critique, or $R_t^{p'} = 0$ otherwise), and α is a constant step-size parameter. With the simple *constant-*α MC of eq. (3) we can update the compatibility of a case p' at time t based on what happens to p' after current critique. Low values of $\alpha \in [0..1]$ makes the compatibility of p' to be increased/reduced slowly, meanwhile using high values of α makes the new results to affect more the compatibility of the case. We could also use incremental dynamic updates of α depending of our problem domain (i.e., we could think that initial critiques of the user should have less influence that last critiques since the user still does not have a high knowledge of the recommendation environment).

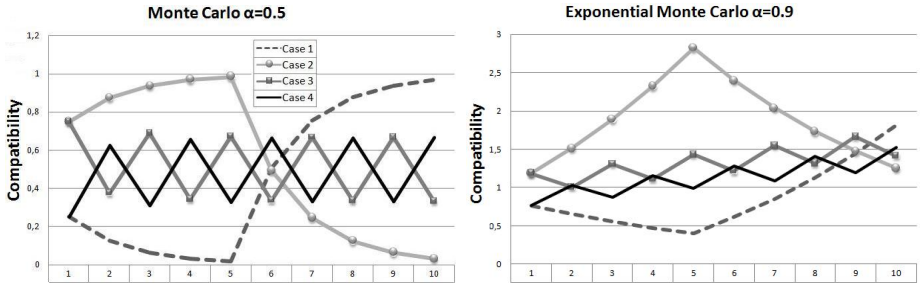
In Figure 1 we show four cases and its corresponding critique satisfaction over ten cycles in an hypothetical recommender. We suppose, for this example, that each cycle $t \in [1, .., 10]$ generates a new critique in our user model. The response R of each case p' at each time t is 1 if the case satisfies the current critique, or 0 otherwise. Note that all cases p' have the same number of 1's and 0's but they differ in the instant they have been satisfied. So, our expectation is that the order

of compatibility should be: first case 1, since all satisfied critiques are produced at the last cycles; next, case 4 and case 3, since both alternate 1 and 0 but the case 4 satisfies the last critique; and finally case 2 with the less compatibility since all 1's are produced at the initial cycles.

	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
Case 1	0	0	0	0	0	1	1	1	1	1
Case 2	1	1	1	1	1	0	0	0	0	0
Case 3	1	0	1	0	1	0	1	0	1	0
Case 4	0	1	0	1	0	1	0	1	0	1

Fig. 1. Case base satisfaction of critiques in a toy problem

Figure 2(a) shows the RL values for *constant-*α MC method for the case base shown in Figure 1. Note that the final order of compatibility is the expected based on the previous criterion. We set up the compatibility at time $t = 0$ to 0.5. The graph shows a logarithmic growing of the compatibility when satisfying critiques, and the same influence decreasing the compatibility for non satisfied cases.



(a) (b)

Fig. 2. (a) *constant-*α MC and (b) EMC numerical representation for the case base shown in Figure 1

On the other hand, we could require that changes between different results R_t modify the compatibility of p' in a different magnitude. For example, we could think that a wrong result for the case p' defined as $R_t^{p'} = 0$ to have less influence than a good result $R_t^{p'} = 1$. Then, we propose the Exponential Monte Carlo (EMC) as follows:

$$C_t^{p'} = \begin{cases} C_{t-1}^{p'} + \alpha \cdot \left(R_t^{p'} + C_{t-1}^{p'} \right) & \text{if } R_t^{p'} = 1 \\ C_{t-1}^{p'} - \alpha \cdot C_{t-1}^{p'} & \text{if } R_t^{p'} = 0 \end{cases} \tag{4}$$

Note that the Monte Carlo variant EMC defined in eq. (4) varies the logarithmic increasing of the compatibility in eq. (3) by an exponential tendency for an input sequence of satisfied critiques $R^{p'}_{[1,..,t]} = 1$, as shown in Figure 2(b). With the exponential tendency the compatibility score is more significant when more critiques are satisfied in the last cycles of the recommendation process.

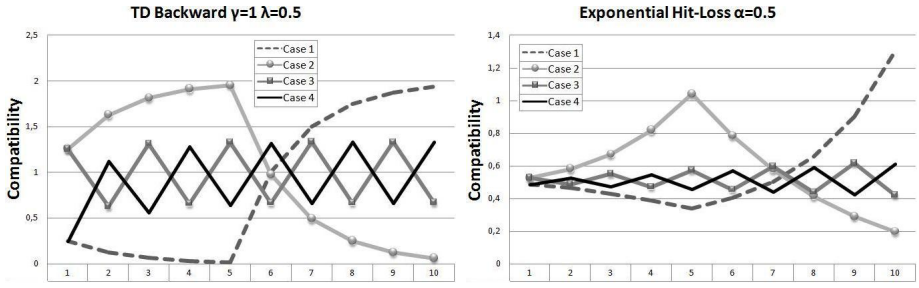
Concerning to TD methods, the Backward TD(λ) used in literature [24] considers an internal variable $e^{p'}_t$ defined as the *elegibility trace* of case p' at instant t. This variable is defined as follows:

$$e^s_t = \begin{cases} \gamma \cdot \lambda \cdot e^s_{t-1} & \text{if } s \notin s_t \\ \gamma \cdot \lambda \cdot e^s_{t-1} + 1 & \text{if } s \in s_t \end{cases} \tag{5}$$

where s is the state being analyzed, s_t is the set of valid states at time t, γ is the discount rate, and λ is related to the *eligibility trace* influence. In our case, considering the state s as a candidate case p', we can express eq.(5) as follows:

$$C^{p'}_t = \gamma \cdot \lambda \cdot C^{p'}_{t-1} + R^{p'}_t \tag{6}$$

where $R^{p'}_t = 1$ if the case p' satisfies the current critique and use the eligibility trace as a measure of the compatibility of p'. This method has a similar tendency than the *constant-α* MC, with a logarithmic increasing of the measure for an input sequence of satisfied critiques $R^{p'}_{[1,..,t]} = 1$, as shown in Figure 3(a). In this case, the desired compatibility order of the cases is also maintained. Note that the behavior of this strategy in the case of the figure is very similar to that one shown by the *constant-α* MC method, but working on a different compatibility range.



(a) (b)

Fig. 3. (a) TD Backward and (b) EHL numerical representations for the case base shown in Figure 1

The only case from the previous methods that consider an exponential tendency for an input sequence of satisfied critiques $R^{p'}_{[1,..,t]} = 1$ corresponds to the EMC RL strategy, the rest of strategies have a logarithmic compatibility

increasing. However, we can also think that in the recommendation process, as the user increases her knowledge along cycles, maybe first matches are finally not relevant meanwhile consecutive or final matches can be more confident to the user preferences. This effect could be modelled by a RL technique which changes the logarithmic increasing to an exponential one. In order to observe if this hypothesis works in conversational recommendation systems, we propose the Exponential Hit-Loss RL technique (EHL) as follows:

$$C_t^{p'} = \begin{cases} h \leftarrow h+1, C_t^{p'} = C_{t-1}^{p'} \cdot (1+\alpha)^{(h^{p'}+t)k} \text{ if } R_t^{p'} = 1 \\ f \leftarrow f+1, C_t^{p'} = C_{t-1}^{p'} \cdot (1+\alpha)^{(f^{p'}+t)k} \text{ if } R_t^{p'} = 0 \end{cases} \quad (7)$$

where $h^{p'}$ and $f^{p'}$ are the number of times that candidate case p' has satisfied (hit) or not (loss or fall) the critiques, respectively (for each case in the data set these values are initialized to zero at time t=0), and k is a regularization factor (fixed to $k = \frac{1}{2}$ in our experiments). This technique has an exponential behavior, which varies based on the amount of hit and losses in the history of each p' and the instant of time, as shown in Figure 3(b). Note that the desired compatibility order is also satisfied.

4 Similarity Using User Preference Weighting

As explained before, the basic idea of the recommender is to present the product that best satisfy user's preferences and we aim to do it by means of enhancing compatibility and similarity. Similarity plays, as in traditional CBR, an important role in the recommender. At each cycle, in the *standard* or the *incremental* recommendation process, the similarity between the candidate case p' to the recommended case p is computed as follows:

$$S(p', p) = \sum_{\forall f} d(p_f^{'}, p_f) \quad (8)$$

where the similarity is the combination of distances d between the candidate p' case and the recommended case p for each feature f.

A common tendency in CBR systems is to use weighting in the similarity measure. In this sense, we propose to change the similarity measure as follows:

$$S(p', p) = \sum_{\forall f} W(p_f^{'}) \cdot d(p_f^{'}, p_f) \quad (9)$$

where $W(p_f^{'})$ is the weight associated to the f feature of the candidate case p'.

Next, we review the local user preference weighting (LW) proposal and propose the global user preference weighting (GW). Both strategies are based on the history of critiques made by the user along the session. This history is the *user model U* defined previously, which specifies the user preferences in the current session.

4.1 Local User Preference Weighting

The local user preference weighting (LW) [19] discovers the relative importance of each feature in each case as a weighting value for computing the similarity, taking into account the user preferences. LW is basically motivated by the fact that most compatible cases are quite similar on their critiqued features and differences mainly belong to those features that have not been yet critiqued. So, the aim of the local weighting method is to prioritize the similarity of those features that have not yet been critiqued. The weight of the local approach is defined over each feature $p_f^{'}$ of candidate case p' as follows:

$$W(p_f^{'}) = 1 - \frac{1}{2}\left(\frac{\sum_{\forall i \in U^f} \delta(p_i^{'}, U_i^f)}{|U^f|}\right) \tag{10}$$

where $|U^f|$ is the number of critiques in U that refer to feature f, U_i^f is a critique over feature f . This generates a feature weight vector for each case. A feature that has not been critiqued will assume a weight value of 1.0, and a decrement will be applied when a critique is satisfied by the case. As such, the feature weight will be proportional to the number of times a critique on this feature is satisfied by the case. However, as shown in eq. (10), weights never decrease to a 0 value. For example, in a travel vacation recommender with a user model that contains two critiques [price, >, 1000] and [price, >, 1500], a case with two features {*duration*, *price*} whose price is 2000 will have as price weight a 0.5 value because it satisfies both critiques whereas the duration weight will be 1.0 because there is no critique on this feature. It is important to recap that the key idea here is to prioritize the similarity of those features that have not yet been critiqued in a given session.

4.2 Global User Preference Weighting

LW computes a feature weight vector for each case depending on the degree of satisfaction of the user critiques for this case. However, considering that the compatibility function is correctly focusing into the product space, the remaining set of cases are similarly satisfying the preferences of the user, so their feature weight vectors will also be similar and a global weighting vector is feasible.

The idea is to compute a vector of weights that will be used for the whole set of candidate cases. This weighting method only enhances the set of features that may produce better recommendation to all the cases. For each case p' in the list of candidate cases, the global weighting is defined as follows:

$$W(f) = 1 - \frac{1}{2}\left(\frac{\sum_{\forall p' \subseteq P'} \delta(p', U_i^f)}{|P'|}\right) \tag{11}$$

where $|P'|$ is the total number of cases in the list of candidate cases. The final weight for each feature f depends on the number of cases that satisfy a critique on this feature. Similarly to LW, the most satisfied critiques will have the lowest

weight for a feature, since the system looks for prioritizing features that have not been previously critiqued. As before, weights never decrease to a 0 value. The maximum decrease is 0.5 which has experimentally shown to obtain the best performance.

The rationale behind prioritizing with the highest weight values the non critiqued features is based on the idea that they are the most important to denote differences on similarity between two cases. This happens because the compatibility score correctly focuses the product space and thus, the candidate cases are similar in the critiqued features. Consequently, the effort of the similarity is to show where the differences are in the features that the recommender is not able to focus with the compatibility because there are not critiques about them.

5 Empirical Evaluation

In previous sections we described different reinforcement learning measures and two weighting approaches to improve the quality measure of a conversational recommender system. We argue that quality measure may benefit from improvements on compatibility and similarity. As a result, the tendency of the quality measure is to recommend cases that better satisfy user preferences. In this section, we test our proposals using a well-known recommender data set. In particular, we look for the performance of the recommender system when using reinforcement learning techniques and also the combination of both RL and weighting proposals.

5.1 Setup

The evaluation was performed using the standard Travel data set (available from *http://www.ai-cbr.org*) which consists of 1024 vacation cases. Each case is described in terms of nine features. The data set was chosen because it contains numerical and nominal features and a wide search space.

First, we evaluate the different RL measures: Monte Carlo (MC), Exponential Monte Carlo (EMC), BackwardTD (BTD), and Exponential Hit-Loss (EHL). Second, we also test the combination of RL with the weighting strategies in our recommender. The configurations analyzed are LW-MC (which corresponds to a local user preference weighting combined with a Monte Carlo compatibility), LW-BTD, LW-EHL, LW-EMC, GW-MC, GW-BTD, GW-EHL, and GW-EMC. We use incremental critiquing-based recommender (IC) [17] as baseline.

We follow the evaluation methodology similar to that one described in [23]. Accordingly, each case (which is called the 'base') in the case-base is temporarily removed and used in two ways. First, it serves as a basis for a set of queries by taking random subsets of its features. We focus on subsets of one, three, and five features to allow us to distinguish between *hard*, *moderate*, and *easy* queries, respectively. Second, we select the case that is most similar to the original base. These cases are the recommendation targets for the experiments. Thus, the base represents the ideal query for a user, the generated query is the initial

query provided by the user, and the target is the best available case for the user. Each generated query is a test problem for the recommender, and at each recommendation cycle the user picks a critique that is compatible with the known target case. We repeat each leave-one-out ten times and the recommendation sessions terminate when the target case is returned. Different statistics are also used to evaluate the statistical significance of the obtained results.

5.2 Reinforcement Learning Recommendation Efficiency

We analyze the recommendation efficiency —by which we mean average recommendation session length— when comparing our RL measures to incremental critiquing. RL measures need to set up a parameter α that fix the learning rate (in the case of BTD we modify the parameter λ). We run different variations of $\alpha, \lambda \in [0.1, ..., 0.5]$ in order to appreciate the influence of this parameter over each strategy. Before to compute the quality for each product, we normalize the compatibility term C so that it ranges between ϵ and one for the lowest and highest compatibility, respectively. This is computed as $C^{p'} = \frac{C^{p'}}{\max_{\forall a' \subseteq P'}(C^{a'})}$, where $C^{p'}$ is the compatibility of the case p' to be normalized. This normalization makes comparable the results obtained by the different RL strategies. Figure 4 (a) presents a graph with the evolution of the average session length for different values of α and λ. We can see that MC and BTD present a tendency to increase and decrease the average session length when increasing α and λ, respectively. The best configuration for MC is $\alpha = 0.1$ and, although not shown in the graph, the best configuration for BTD is $\lambda = 0.9$. This large value for BTD suggests that high changes on the compatibility value may improve (reduce) session lengths. The EMC and EHL strategies consider an exponential tendency in order to make final critiques more relevant to the user preferences than the initial ones. As shown in Figure 4 (a), the exponential behavior of these strategies, in contrast to the logarithmic one of the remaining, results in shorter session lengths.

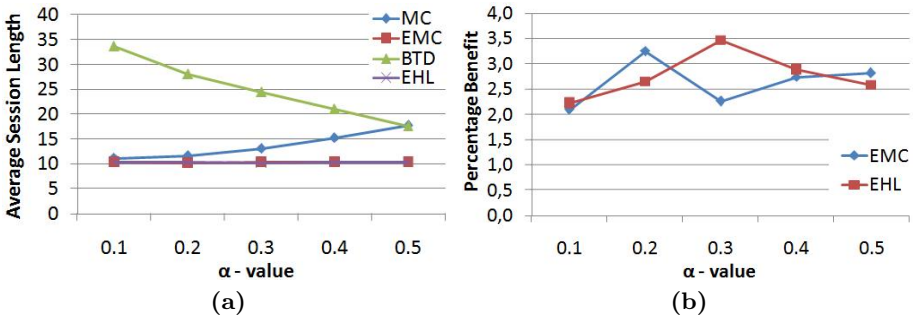
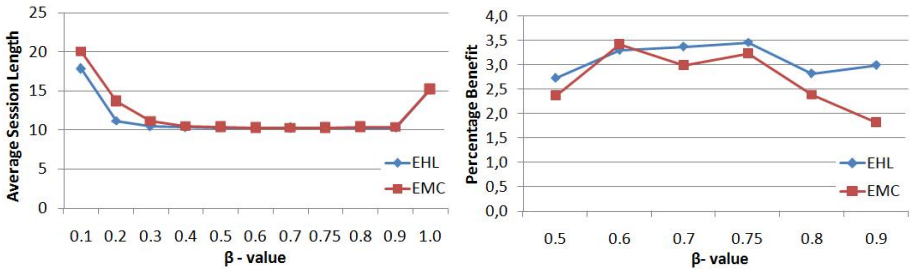


Fig. 4. Figure (a) corresponds to session lengths evolution for different α, λ-values and Figure (b) represents session length benefit over incremental critiquing

We also found that the exponential EMC and EHL configurations have a more stable average session length than the rest of RL techniques for different values of α, ranging from 10.37 to 10.24. In Figure 4 (b) we present EMC and EHL benefit compared to incremental critiquing. We compute the percentage benefit as $\texttt{Benefit(y,x)} = \left(1 - \frac{\texttt{y}}{\texttt{x}}\right) \cdot 100$, where `y` and `x` stands for the number of cycles of the compared strategy and IC, respectively. The EMC and EHL benefit ranges from 2% to 3.5%. Although the result seems low, we want to point out that the only difference between IC and our EMC and EHL methods is how we compute the compatibility measure. Note that in our previous work [18], we introduced the MC strategy in combination with a product recommendation strategy called *Highest Compatibility Selection* (HCS) [19], whose benefit applied together in the recommender system was around 2% to under 4%. Our new RL measures are able to obtain the same benefit by themselves without introducing the HCS methodology. Thus, we can state that EMC and EHL RL exponential strategies are able to focus on those products that best satisfy the user preferences, obtaining more accurate quality measurements.

Additionally, in Figure 5 (a) we summarize the average session lengths results over all types of queries for different variations of β using EMC (set up with $\alpha = 0.2$) and EHL (set up with $\alpha = 0.3$). Once again, the results are quite similar between EMC and EHL. Session lengths are maintained between $\beta = 0.5$ to $\beta = 0.9$. It is significant that session lengths remain shorter for $\beta = 1.0$ than $\beta = 0.1$. Note that $\beta = 1.0$ means that each recommendation cycle is influenced by the compatibility measure with no similarity and a $\beta = 0.1$ specifies that the most important role for recommendation is the similarity.

Figure 5 (b) presents EMC and EHL benefit over incremental critiquing for the β-values for which RL techniques obtain better results. EMC and EHL reduce the session length from nearly 2% to 3.5%. The best results are obtained for the values of $\beta = 0.6$ and $\beta = 0.75$, respectively. The last value coincides with the best result obtained by the incremental critiquing. Thus, we decided to fix $\beta = 0.75$, $\alpha = 0.1$ for MC, $\alpha = 0.2$ for EMC, $\alpha = 0.9$ for BTD, and $\alpha = 0.3$ for EHL in the next experiments, respectively.



(a) **(b)**

Fig. 5. Figure (a) corresponds to session lengths evolution for different β-values and Figure (b) represents benefit over incremental critiquing

5.3 Quality Recommendation Efficiency

Earlier we mentioned how quality, apart from the compatibility, also uses similarity to optimize recommendations. In this section, we analyze the benefits over incremental critiquing when applying both weighting and reinforcement learning to compute the quality measure.

In Figure 6 we present the average benefit to different levels of query difficulty. Figure 6 (a) depicts the results for the combination of local weighting with RL measures. These combinations result for all algorithms tested in a reduction in session length that ranges from 0.5% up to 8%. On the other hand, see Figure 6(b), global weighting and RL measures gives the highest benefit, ranging from 3.44% to 11.13%. Combining weighting and reinforcement learning compatibility further enhances recommendation performance, resulting in better recommendation for all queries (hard, moderate, and easy).

We also statistically analyze the benefits of using our methodology instead of the standard incremental critiquing. As before, we separately evaluate local and global weighting. The algorithms analyzed are: (1) IC, LW-MC, LW-EMC, LW-BTD, and LW-HLE, and (2) IC, GW-MC, GW-EMC, GW-BTD and GW-HLE. First of all, we compute the *mean rank* (r) of each algorithm considering all the experiments (five algorithms and three different queries for each test). The rankings are obtained estimating each particular ranking r_i^j for each query i and each algorithm j, and computing the mean ranking R for each algorithm as $R_j = \frac{1}{N}\sum_i r_i^j$, where N is the total number of queries. Compared with mean performance values, the mean rank can reduce the susceptibility to outliers which, for instance, allows a classifier's excellent performance on one query to compensate for its overall bad performance [6]. Second, we apply the Friedman and Nemenyi tests to analyze whether the difference between algorithms is statistically significant [7,8].

The **Friedman test**, recommended by Demšar [6], is effective for comparing multiple algorithms across multiple data sets, in our case, across multiple

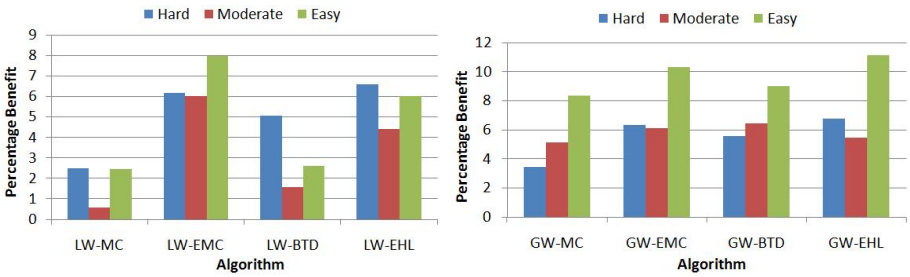


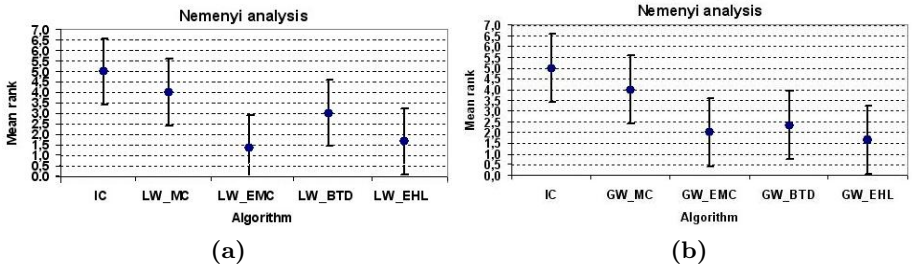
Fig. 6. Average benefit over incremental critiquing. Each figure represents a different benefit weighting where (a) corresponds to local weighting and (b) corresponds to global weighting.

queries. It compares mean ranks of algorithms to decide whether to reject the null hypothesis, which states that all the methods are equivalent and so their ranks should be equal. The Friedman statistic value is computed as $X_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right]$. Since this value is undesirable conservative, Iman and Davenport proposed a corrected statistic, $F_F = \frac{(N-1)X_F^2}{N(k-1)-X_F^2}$.

When we apply the Friedman test in our experimental setup with five algorithms and three different queries, F_F is distributed according to the F distribution with $(5-1) = 4$ and $(5-1)\cdot(3-1) = 8$ degrees of freedom. The critical value of $F(4,8) = 3.83$ at the 0.05 critical level. We obtained the values of $X_F = 11.42$ and $F_F = 40.06$ for the local weighting rankings and $X_F = 9.86$ and $F_F = 9.22$ for the global weighting rankings. As the values of F_F are always higher than 3.83 we can reject the null hypothesis in both cases.

Once we have checked for the non-randomness of the results, we can perform a post hoc test to check if one of the techniques can be singled out. For this purpose we use the Nemenyi test —two techniques are significantly different if the corresponding average ranks differ by at least the critical difference value, $CD = q_\alpha\sqrt{\frac{k(k+1)}{6N}}$, where q_α is based on the Studentized range statistic divided by $\sqrt{2}$. In our case, when comparing five algorithms with a critical value $\alpha = 0.1$, $q_{0.1} = 2.45$ for a two-tailed Nemenyi test. Substituting, we obtain a critical difference value $CD = 3.17$. Thus, for any two pairs of algorithms whose rank difference is higher than 3.17, we can infer —with a confidence of 90%— that there exists a significant difference between them.

The results of the Nemenyi test are illustrated in Figure 7. In the figure, bullets represent the mean ranks of each algorithm. Vertical lines across bullets indicate the 'critical difference'. The performance of two algorithms is significantly different if their corresponding mean ranks differ by at least the critical difference. For instance, Figure 7 (a) reveals that LW-EMC and LW-EHL are significantly better than IC. We cannot say the same with regard to LW-MC and LW-BTD, though. The same behavior occurs in the case of the global weighting analysis of Figure 7 (b). However, note that the global weighting of Figure 6(b)



(a) **(b)**

Fig. 7. Application of the Nemenyi test critical difference to algorithms mean rank considering their session length for (a) local weighting and (b) global weighting

shows more stable results for all combination of strategies than local weighting, obtaining higher benefits in terms of session length.

6 Conclusions

Retrieving the most suitable product for a user during a live customer interaction is one of the key pieces in conversational case-based recommender systems. Specifically, in incremental critiquing the recommendation process is primarily guided by a quality measure. In this paper we have proposed new strategies for compatibility computation and feature weighting that enhance quality. We reviewed the state-of-the-art on reinforcement learning which can be applied to conversational CBRs, and proposed two new compatibility strategies which offer better benefit in terms of session length. Concerning the similarity score, we presented a global weighting strategy, which uses a common weight over all cases based on the number of satisfied critiques. Our experiments show significantly improvements in comparison to the state-of-the-art approaches.

Acknowledgements

This work has been supported in part by projects TIN2006-15308-C02, FIS PI061290, and CONSOLIDER-INGENIO CSD 2007-00018.

References

1. Aha, D.W.: Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. International Journal of Man-Machine Studies 36(2), 267–287 (1992)
2. Aha, D.W., Breslow, L.A., Muñoz-Avila, H.: Conversational Case-Based Reasoning. Applied Intelligence 14, 9–32 (2000)
3. Burke, R.: Interactive Critiquing for Catalog Navigation in E-Commerce. Artificial Intelligence Review 18(3-4), 245–267 (2002)
4. Burke, R., Hammond, K., Young, B.: Knowledge-Based Navigation of Complex Information Spaces. In: Proceedings of the 13th National Conference on Artificial Intelligence, Portland, OR, pp. 462–468. AAAI Press/MIT Press (1996)
5. Burke, R., Hammond, K., Young, B.C.: The FindMe Approach to Assisted Browsing. Journal of IEEE Expert 12(4), 32–40 (1997)
6. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal Machine Learning Research 7, 1–30 (2006)
7. Friedman, M.: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of the American Statistical Association 32(200), 675–701 (1937)
8. Friedman, M.: A comparison of alternative tests of significance for the problem of m rankings. The Annals of Mathematical Statistics 11(1), 86–92 (1940)
9. Göker, M.H., Thompson, C.A.: Personalized conversational case-based recommendation. In: Blanzieri, E., Portinale, L. (eds.) EWCBR 2000. LNCS, vol. 1898, pp. 99–111. Springer, Heidelberg (2000)

10. Golovin, N., Rahm, E.: Reinforcement learning architecture for web recommendations. In: Proceedings of the International Conference on Information Technology: Coding and Computing, Washington, DC, USA, vol. 2, p. 398. IEEE Computer Society Press, Los Alamitos (2004)
11. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement Learning: A survey. Journal of Artificial Intelligence Research 4, 237–285 (1996)
12. Kohavi, R., Langley, P., Yun, Y.: The utility of feature weighting in nearest-neighbour algorithms. In: van Someren, M., Widmer, G. (eds.) ECML 1997. LNCS, vol. 1224. Springer, Heidelberg (1997)
13. McGinty, L., Smyth, B.: Comparison-Based Recommendation. In: Craw, S. (ed.) ECCBR 2002. LNCS, vol. 2416, pp. 575–589. Springer, Heidelberg (2002)
14. McGinty, L., Smyth, B.: Tweaking Critiquing. In: Proceedings of the Workshop on Personalization and Web Techniques at the International Joint Conference on Artificial Intelligence. Morgan Kaufmann, San Francisco (2003)
15. McSherry, D.: Similarity and Compromise. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS, vol. 2689, pp. 291–305. Springer, Heidelberg (2003)
16. Moon, A., Kang, T., Kim, H., Kim, H.: A service recommendation using reinforcement learning for network-based robots in ubiquitous computing environments. In: EEE International Conference on Robot & Human Interactive Communication (2007)
17. Reilly, J., McCarthy, K., McGinty, L., Smyth, B.: Incremental Critiquing. In: Bramer, M., Coenen, F., Allen, T. (eds.) Research and Development in Intelligent Systems XXI. Proceedings of AI 2004, Cambridge, UK, pp. 101–114. Springer, Heidelberg (2004)
18. Salamó, M., Reilly, J., McGinty, L., Smyth, B.: Improving incremental critiquing. In: 16th Artificial Intelligence and Cognitive Science, pp. 379–388 (2005)
19. Salamó, M., Reilly, J., McGinty, L., Smyth, B.: Knowledge discovery from user preferences in conversational recommendation. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS, vol. 3721, pp. 228–239. Springer, Heidelberg (2005)
20. Shimazu, H.: ExpertClerk: A Conversational Case-Based Reasoning Tool for Developing Salesclerk Agents in E-Commerce Webshops. Artificial Intelligence Review 18(3-4), 223–244 (2002)
21. Shimazu, H., Shibata, A., Nihei, K.: ExpertGuide: A Conversational Case-Based Reasoning Tool for Developing Mentors in Knowledge Spaces. Applied Intelligence 14(1), 33–48 (2002)
22. Smyth, B., McGinty, L.: An Analysis of Feedback Strategies in Conversational Recommender Systems. In: Cunningham, P. (ed.) Proceedings of the 14th National Conference on Artificial Intelligence and Cognitive Science, Dublin, Ireland (2003)
23. Smyth, B., McGinty, L.: The Power of Suggestion. In: Proceedings of the International Joint Conference on Artificial Intelligence. Morgan Kaufmann, San Francisco (2003)
24. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An introduction. MIT Press, Cambridge (1998)
25. Wettschereck, D., Aha, D.W.: Weighting features. In: Aamodt, A., Veloso, M.M. (eds.) ICCBR 1995. LNCS, vol. 1010, pp. 347–358. Springer, Heidelberg (1995)