Noname manuscript No.
(will be inserted by the editor)

Data-driven decision making in Critique-based Recommenders: from a critique to social media data

David Contreras · Maria Salamó

the date of receipt and acceptance should be inserted later

Abstract In the last decade there have been a large number of proposals in Critique-based Recommenders. They are data-driven in their nature since they use a conversational cyclical recommendation process that elicits user feedback. In the literature, the proposals mainly differ in two aspects: (1) in the source of data and how it is mined for providing recommendations to the user; and (2) they hardly integrate previous enhancements developed in the field. In this paper, we propose new algorithms that integrate both several sources of data such as current user preferences (i.e., a critique), product descriptions, past critiquing sessions of other users, and users' opinions expressed on social media web sites and previous enhancements on the recommendation process such as approaches based on compatibility and weighting scores. The comparison of our proposals with well-known state-of-the-art approaches shows that the new recommendation algorithms significantly outperform previous proposals.

1 Introduction

Data-driven decision making refers to the practice of basing decisions on the analysis of data [2,23]. By their nature, recommender systems have the ability to mine data and use it to provide even the most tentative user[1] with compelling and timely product suggestions [33,15]. These systems have different recommendation techniques, from *collaborative filtering* technique [16,

D. Contreras
Facultad de Ingeniería y Arquitectura,
Universidad Arturo Prat, Chile E-mail: david.contreras@unap.cl

M. Salamó
Matemàtiques i Informàtica,
Universitat de Barcelona, Spain E-mail: maria@maia.ub.es

[1] In this work, we use the term *user* to refer to: an e-commerce *customer/shopper* or a recommender system *user*.

13], which compares other users with the active user and recommends items that were liked by users with similar profiles to the active user's profile, to *content-based* technique [22], which compares the users descriptions about a desired product against the item descriptions and recommends items that match. Moreover, most of these recommenders systems operate in a *single-shot* fashion as they present users with a single set of recommendations based on some initial query and the process usually ends; thus the user is engaged in a single (short-lived) interaction with the system. The *single-shot* strategy is appropriate to the recommendation of simple products, but it is not so well suited for high-risk product domains[2] or to recommending in complex products spaces[3]. In these scenarios, it is more appropriate to engage the user in a recommendation dialogue so that incremental user feedback can be used to refine recommendations. A significant amount of the research carried out on recommender systems has demonstrated the benefits of critique-based recommenders[4] [28, 20, 27, 6], as they help customers with ill-defined preferences to both navigate through complex product spaces and to better understand their own buying preferences.

In particular, critique-based recommenders are data-driven in their nature as they guide users through a product space in pursuit of suitable products using a cyclical recommendation process, alternatively making suggestions and eliciting user feedback, to refine their needs and preferences, based on recent recommendations. Critiquing is based on the idea that for a user is easier to critique a product recommendation by saying "*like this but cheaper*" than to construct formal queries [21]. Many different critique-based recommenders have been proposed in the literature. One way to categorize these proposals is according to the source of data and how it is mined to provide recommendations. Additionally, most of them never take advantage of the enhancements (i.e., the source of data used or the process to mine the data to recommend a product) of previous algorithms.

In this paper, our hypothesis is that some of the previous proposals may improve their efficiency if their algorithms include new sources of data and state-of-the-art enhancements in critique-based recommendations. With this hypothesis in mind, the contribution of this paper is three-fold:

- First, we propose a new algorithm, called History and Opinion Recommender (HOR), which integrates in the recommendation process several sources of data such as user preferences, product descriptions, past critiquing sessions of other users, and users' opinions expressed on social media web sites.
- Second, we integrate some previous state-of-the-art proposals which are based on a combination of compatibility and weighting scores in the rec-

[2] In high-risk product domains, the task of locating a desired choice among a large set of options is indeed becoming intimidating for the average customer [6] because there are many available alternatives.

[3] In complex product spaces, users require a good knowledge of the large amount of characteristics of the products and the relationship with the different available options.

[4] It is also referred to as critiquing-based recommendation in the literature.

ommendation process of both our proposed algorithm (HOR) and a recent well-known history-based recommender called History-Guided Recommendation (HGR). The new algorithms will be called HOR-I and HGR-I, respectively.
- Third, we carried out an exhaustive evaluation of the proposals to demonstrate our initial hypothesis. First, our evaluation focuses on a comparison of our proposals (i.e., the HOR, HGR-I, and HOR-I) against HGR and the traditional Incremental Critiquing (IC) recommender. The experiments were conducted through an off-line simulator. With these experiments it is noted the influence of both the different sources of data (i.e., from the critiques used in IC to the use of social media data in HOR) and the data-driven process to recommend products. Second, we evaluate HOR-I and HGR-I, which are the best ones in the simulator, against IC with a real-user study in order to demonstrate our initial hypothesis with real users. To the best of our knowledge, this is the first real-user study of history-based recommenders. The results of our in-depth evaluation confirm our hypothesis and indicate that our proposals significantly improve on the efficiency, efficacy, and user satisfaction of previous algorithms in critique-based recommenders.

The rest of this paper is organized as follows: Section 2 presents related work on critique-based recommenders; Section 3 describes the History and Opinion Recommender in depth; Section 4 describes the integration of previous state-of-the-art approaches; Section 5 evaluates the efficiency of our proposals through a simulator; Section 6 evaluates the efficiency, efficacy, and user satisfaction of our proposals with real users; Finally, Section 7 presents the conclusions and future directions of our work.

2 Related Work

Critique-based recommenders have been broadly recognized as an effective preference-based search option using a feedback mechanism called *critiquing* [28, 20, 27, 6]. The basic idea of critiquing can be traced back to the seminal work of the FindMe system [3]. In particular, a critique is a directional preference over a feature of the current recommended product. An example of a critique is "a cheaper camera", when the price of the current recommendation is \$300 implies a critique [price $<$ \$300]. Note that, in a complex decision situation, a critique is a form of user feedback that strikes a useful balance between the information content of the feedback and the required level of user effort or domain expertise.

In critique-based recommenders, the most common feedback mechanisms are *unit* and *compound* critiques. In the former, users are allowed to critique a single feature of a product at a time [18, 32], whereas in compound critiques, each critique can be a combination of multiple unit critiques [40, 31, 25, 20, 27]. In the literature there are several studies of compound critique approaches, such as dynamic critiquing [28], MAUT-based compound critiques [40], preference-based organization [5,25], and Example Critiquing [26,4,38]. Considering that most of the proposed algorithms in critique-based recommenders use unit critiques, we focus our work on analyzing and evaluating critique-based recommenders that use a feedback mechanism based on unit critiques.

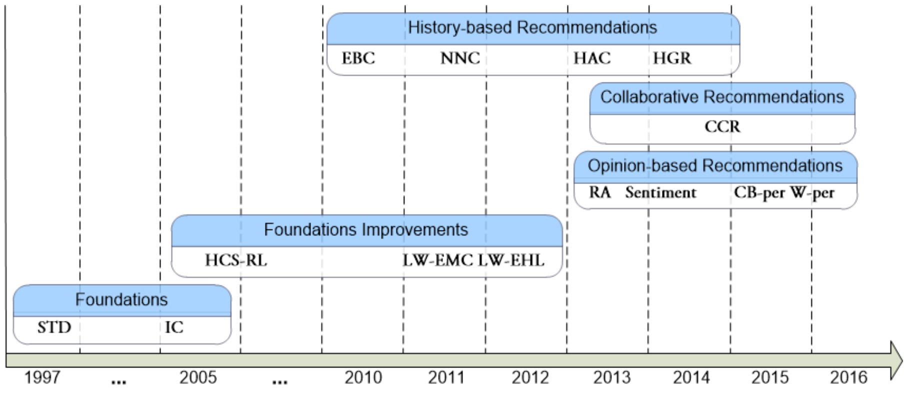


Fig. 1 Timeline of critique-based recommenders based on unit critiques.

Figure 1 shows a timeline of critique-based recommenders that use unit critiques. Concretely, the FindMe system [3] was the first unit critiquing recommender, which is known as the *standard critiquing (STD)* (see the Foundations square in Figure 1). STD only uses the current critique introduced by the user to guide the user in the product search space. Later, Incremental Critiquing (IC) [29] appeared, which increases the recommendation efficiency by means of recording –as an additional source of data– a list of the user's critiques during the session. Other studies have presented weighting techniques and a set of compatibility measures for improving the recommendation process with regard to the foundations approaches: HCS-RL, LW-EMC, and LW-EHL [35,34] (see the Foundations Improvements square in Figure 1). More recently, researches have been focused on using a mechanism (called Reviewer Assistant, RA) for mining product features [12,11] and sentiments [10] from users' reviews with the aim to recommend products only based on this information, but without considering the user preferences in the recommendation (see RA and Sentiment in the Opinions-based Recommendation square in Figure 1). In contrast to these, others use a new source of knowledge (users' opinions) to enrich the product descriptions and to define a weight for each feature of the product in the recommendation process with the aim of improving their efficiency [7] (see CB-per and W-per techniques in the Opinions-based Recommendations square in Figure 1). We have not included RA, Sentiment, CB-per and W-per in our analysis for two main reasons. First, RA and Sentiment use data that does not bear in mind the user preferences and all the methods we will evaluate do. In this work, we analyze critique-based recommenders that drive data for user behavior modeling. Second, CB-per and W-per can be applied only to domains with technical and perceptive[5] features. It is important to mention that the new proposal –called HOR– presented in this work addresses this drawback.

In the last few years, novel approaches have mainly focused on using collaboration among users (see the Collaborative Recommendations square in Figure 1) or on using past critique sessions made by other users (see the History-based Recommendations square in Figure 1).

In the former, users may collaborate *online* in the search for a desired product as they are immersed in a 3D virtual environment that enhances their buying experience as well as increasing the interaction elements for eliciting user feedback [8]. This recommender is called Collaborative Conversational Recommendations (CCR). It is out of the scope of this work to analyze methods that enable online collaboration between users as several feedback mechanisms can be used and we are only evaluating those that use critiquing. Nonetheless it would be interesting to do in some future work.

In the second approach, researches focused on reusing past critiquing sessions from other users as a source of data called *history* to improve recommendation efficiency. For instance, Experience-Based Critiquing (EBC) [19] uses the critiquing experiences of other users as a new source of knowledge during the critiquing process. EBC selects from these critiquing histories the final accepted products as recommendation candidates (i.e., similar sessions to the current recommendation session). In EBC, a past critiquing session can be selected if its critiques sufficiently overlap with the critiques so far in the current session. Another approach, called Nearest Neighbor Compatibility critiquing (NNC) [17] adopts the approach of EBC but the recommendation candidates can be the intermediate cases of a session and not necessarily the final purchase decision. Another approach proposed is History-Aware Critiquing-based conversational recommendation (HAC) [36], in which a past critiquing session can be selected if it contains similar recommended items to the ones in the current session and its critiques sufficiently overlap with the critiques so far in the current session. Later, an improvement of HAC, called the History-Guided Recommendation (HGR) [37] technique was presented, which uses the pair recommended product and critique for obtaining similar sessions to the current recommendation session. In HGR, a past critiquing session can be selected if its recommendation-critique pairs sufficiently overlap with the ones so far in the current session. Among all the proposed History-based recommenders, HGR outperforms the aforementioned recommenders, as denoted in [37]. In fact, we also did a complete evaluation of all the history-based recommenders and our results corroborate that HGR in all datasets performed the best.

[5] A *perceptive* feature is a feature that provides an immediate and intuitive recognition or appreciation of the qualities of a product. For example, in the SMARTPHONE domain, *performance* is a perceptive feature that intuitively includes more than one of the *technical* features of a product (e.g., storage, RAM, or CPU).

3 History and Opinion Recommender

We present here a new recommender called History and Opinion Recommender (HOR), which apart from using other users' histories of sessions it also integrates as a source of data opinions extracted from e-commerce social media sites. Next, we define formally the data structures, we present the conceptual recommendation process and we define in depth the algorithm.

3.1 Definitions

We define all data structures involved in HOR: the case base, the session base, and the opinion base. In addition, we formally define the user model and what a critique is in our algorithm.

Definition 1 Case Base. The case base, CB, is a set of products for recommendation, described as $CB = \{p_1, ..., p_n\}$, where p_i is the ith product and the set of features that describes each product is defined as $F = \{f_1, ..., f_m\}$.

We denote a particular feature f_s of a case p_i as $p_i^{f_s}$. For example, $p_i^{f_1}$ and $p_i^{f_2}$ refer to the first and the second features in case p_i, respectively.

Definition 2 Session Base. The session base, SB, is a data set of past critiquing sessions from other users defined as $SB = \{s_1, ..., s_l\}$, where s_i is a sequence of recommendation-critique pairs (r_i, c_i) and each session finishes in a terminal product, noted by $term(s_i)$.

Definition 3 Opinion Base. The opinion base, OB, is a set of product opinions defined as $OB = \{op_1, ..., op_n\}$, where op_i is a set of users' opinions u_i about the i^{th} product obtained from the users through several methods (e.g., explicit ratings, users reviews, or textual opinions, among others). In this paper, we concentrate on explicit ratings. Accordinly, each u_i is an explicit rating and the term o_i is the average of the opinions' rating for each product. Nonetheless, we will consider the use of other sources of social media data in a future work.

Definition 4 User Model. The user model, U, is a set of recommendation-critique pairs, defined as $U = \{u_1, ..., u_k\}$, where $u_i = (r_i, c_i)$ is a particular recommendation-critique pair with r_i representing the recommended product and c_i representing the critique (see Definition 5) applied to r_i.

Definition 5 Critique. A critique c_i, is represented as a triple $(f_i, type_i, v_i)$, where f_i refers to a feature of the recommended product, r_i, $type_i$ is the type of the critique c_i (i.e., typically $<$, $>$, $<>$), and v_i is the current value of f_i.

Critiques typically take the form of directional or replacement preferences [20]. A directional critique effects an increase or decrease in the value of a numerical feature (i.e., greater or lower than current value). For example, "a cheaper camera", when the price of the current recommendation is \$300, implies a critique $(price, <, \$300)$. On the other hand, a replacement critique substitutes any value (i.e., aside from the critiqued value) for a non-numeric feature. For example, in a camera domain, $(manufacturer, <>, Sony)$ represents the user critique "I do not like Sony cameras".

3.2 HOR Conceptual Recommendation Process

Our proposal uses different sources of data to mine user's preferences in order to recommend products. On the one hand, HOR stores a history of past sessions in the session base (see SB in Figure 2), and maintains the current recommendation session of the user in the user model, shown as U in Figure 2. On the other hand, HOR maintains the opinion base, OB, which contains explicit ratings described by the users in an e-commerce social media site (see OB in Figure 2), and uses the set of products for providing recommendations, described as CB in Figure 2. Next, we describe the HOR recommendation process, which is illustrated on the right side of Figure 2.

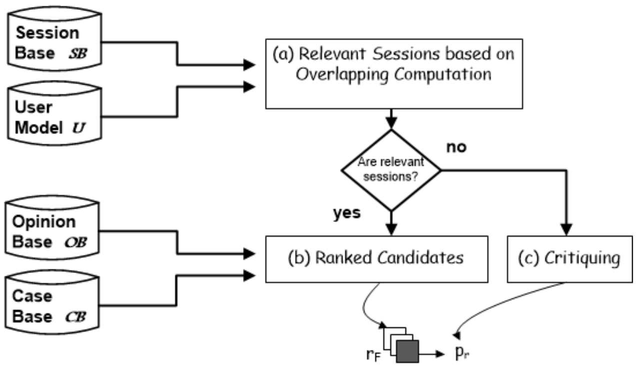


Fig. 2 HOR conceptual recommendation

First, the recommender identifies a set of relevant sessions based on overlapping computation (see letter a in Figure 2), which are history sessions in the SB that overlap with the user's current partial critique session, U. To do so, we first compute the overlap score. The relevant sessions are those that contain an overlap score greater than zero. In particular, we use the overlap score defined in [37], described in Equations 1 and 3. In Equation 1, we compute the number of recommendation pairs, r_i, c_i (recommended product and critique) in the user model U, that are also present in each past session s_i. The $matchPair$ measure is computed as shown in Equation 2.

$$OverlapPair(U, s_i) = \sum_{\forall(r_i,c_i)\in U} \sum_{\forall(r_j,c_j)\in s_i} matchPair((r_i, c_i), (r_j, c_j)) \quad (1)$$

$$matchPair((r_i, c_i), (r_j, c_j)) = \begin{cases} 1 \text{ if } (r_i = r_j) \text{ and } (c_i = c_j), \\ 0 \text{ otherwise} \end{cases} \tag{2}$$

When there are no overlapped pairs using Equation 1 we compute, using Equation 3, the number of critiques (i.e., without considering recommended products) in the user model U, which are also present in each past critiquing session, s_i. This method has been used in previous studies with satisfactory results, such as [19]. The *matchCritique* measure is computed as shown in Equation 4.

$$OverlapCritique(U, s_i) = \sum_{\forall (c_i) \in U} \sum_{\forall (c_j) \in s_i} matchCritique(c_i, c_j) \tag{3}$$

$$matchCritique(c_i, c_j) = \begin{cases} 1 \text{ if } c_i = c_j, \\ 0 \text{ otherwise} \end{cases} \tag{4}$$

Second, we analyze the set of relevant sessions obtained in the initial step, depicted with letter a in Figure 2. On the one hand, if there are relevant sessions, we rank candidates for the next recommendation (see letter b in Figure 2). Concretely, each relevant session terminates with an accepted recommendation, $term(s_i)$ (i.e., a recommendation candidate product in HOR) that may be associated with more than one relevant session. We store candidates in a list, r_F, and compute a score for each of them. We compute the candidate score based on both the score of the relevant session where it is an accepted recommendation, and the average opinion, o_i, rating over the candidate product stored in the Opinion Base, OB. On the other hand, if there are not relevant sessions we revert to the Incremental Critiquing recommender [29], see letter c in Figure 2. Finally, the best ranked product, p_r, which is obtained using r_F or a *critiquing* algorithm, is recommended to the user. The full information of the recommended product is obtained from the case base, CB.

3.3 HOR Algorithm

The History and Opinion Recommender (HOR) is described in Algorithm 1 (see lines 1-12). The algorithm receives the initial product query p_q, from a user and the data for recommending products to the user: the case base, the session base, and the opinion base, denoted as CB, SB, and OB, respectively. Note that at the beginning, no critique has yet been provided by the user and the user model is just initialized (see lines 3 and 4). Next, HOR starts a cycle that contains 4 phases (see lines 7-10): *Recommend Item*, *User Review*, *Query Revise*, and *Update Model*. This cycle is repeated until the user accepts a product recommendation or explicitly abandons the process. Next, we describe in depth each phase of Algorithm 1 in Algorithm 2.

In detail, the first phase **RecommendItem** (see lines 1-15 in Algorithm 2) is devoted to recommending a new product p_r to the user from the case base CB. This recommendation is based on: the current query p_q, past critiquing

```
Input: $p_q$: product query, $CB$: case base, $SB$: session base, $OB$: opinion base
define HistoryOpinionRecommender ($p_q$, $CB$, $SB$, $OB$)
begin
    $U_{cq} \leftarrow null$ // current critique
    $U \leftarrow null$ // user model
    repeat
        // $p_r$: product recommendation
        $p_r \leftarrow$ RecommendItem($p_q$, $CB$, $U$, $SB$, $OB$)
        $U_{cq}$, $CB \leftarrow$ UserReview($p_r$, $CB$)
        $p_q \leftarrow$ QueryRevise($p_r$)
        $U \leftarrow$ UpdateModel($U$, $U_{cq}$, $p_r$)
    until UserAccepts($p_r$) or UserAbandon()
end
```

Algorithm 1: History Opinion Recommender Algorithm

sessions of other users SB, other users opinions OB, and considering previous recommendation-critiques pairs, if there are any stored in the user model U.

The *RecommendItem* procedure uses the user's current (partial) critique session, $(r_1, c_1), ...(r_m, c_m)$, stored in the user model U, over the past critiquing sessions stored in SB, in order to identify a set of *relevant sessions*, S^{REL} as shown in the *RelevantSessions* procedure in lines 16-29. A relevant session is a past session into SB that at least has some overlap (an overlap score > 0) with the current user model based on a particular overlap metric. To do so, we evaluate first of all the overlap using the recommendation-critique pairs. That is, product and critique in SB that are also present in the user model (see line 19 and Equation 1). If the **OverlapPair** score results in an empty set (i.e., there are not relevant sessions), we evaluate the overlap score only considering the critiques (see line 21 and Equation 3). Each relevant session terminates with a terminal and accepted product recommendation $term(s_i)$ and it also contains its overlap score, previously computed by means of Equation 1 or Equation 3. If we obtain relevant sessions from **OverlapPair** or **OverlapCritique** (see line 23), we then filter out incompatible recommendation candidates, $term(s_i)$. That is, for each relevant session we eliminate those products $term(s_i)$ that fail to satisfy the user model U (see lines 24-26).

Then, the *RelevantSessions* procedure returns a set of past relevant sessions, S^{REL} (see line 3). When S^{REL} is not empty, the HOR algorithm recommends an accepted product from S^{REL} (see lines 5-7). Specifically, in line 5 relevant sessions are sorted out in decreasing *RecScore* (see Equation 5). Next, we extract the terminal product from each relevant session and it is added to r_F, which is a ranked list of recommendation candidates (see line 6). Finally, the new recommendation product p_r is the top of r_F (see line 7).

$$RecScore(U, s_i, o_j) = \alpha \cdot OverlapScore(U, s_i) + (1 - \alpha) \cdot o_j \tag{5}$$

Equation 5 averages, with the α parameter, the *OverlapScore* (computed using Equations 1 or 3) and the opinion rating $o_j \in OB$ of the terminal product p_j (i.e., the accepted recommendation in s_i, $p_j = term(s_i)$). Note that α is a parameter that controls the relative weight of the overlap score of the products and the users' opinions.

```
1  define **RecommendItem**($p_q$, $CB$, $U$, $SB$, $OB$)
2  begin
3      $S^{REL} \leftarrow$ RelevantSessions($U$, $SB$)
4      if $S^{REL} \neq null$ then
5          $S^{REL} \leftarrow$ sort $s_i \in S^{REL}$ in decreasing $RecScore(U, s_i, o_j)$ //see Equation 5
6          $r_F \leftarrow$ store $p_j = term(s_i)/s_i \in S^{REL}$
7          $p_r \leftarrow$ getTopRanked($r_F$)
8      else
9          $U_{cq} \leftarrow$ getLastCritique($U$)
10         $CB' \leftarrow \{p \in CB | \delta(p, U_{cq})\}$
11         $CB' \leftarrow$ sort all $p_i \in CB'$ in decreasing $Q(p_i, p_q, U)$ //see Equation 6
12         $p_r \leftarrow$ most quality product in $CB'$
13     end
14     return $p_r$
15 end
16 define RelevantSessions ($U$, $SB$)
17 begin
18     $S^{REL} \leftarrow null$
19     $S^{REL} \leftarrow \{s_i \in SB$ : OverlapPair($U$, $s_i$)>0$\}$ //see Equation 1
20     if $S^{REL} == null$ then
21         $S^{REL} \leftarrow \{s_i \in SB$ : OverlapCritique($U$, $s_i$)>0$\}$ //see Equation 3
22     end
23     if $S^{REL} \neq null$ then
24         for $s_i \in S^{REL}$ do
25             $S^{REL} \leftarrow S^{REL} -$ contradict $(term(s_i), U)$
26         end
27     end
28     return $S^{REL}$
29 end
30 define UserReview ($p_r$, $CB$)
31 begin
32     $U_{cq} \leftarrow$ user critique for some $f \in p_r$
33     $CB \leftarrow CB - p_r$
34     return $U_{cq}$, $CB$
35 end
36 define QueryRevise ($p_r$)
37 begin
38     $p_q \leftarrow p_r$
39     return $p_q$
40 end
41 define UpdateModel ($U$, $U_{cq}$, $p_r$)
42 begin
43     $U \leftarrow U -$ contradict ($U$, $U_{cq}$, $p_r$)
44     $U \leftarrow U -$ refine ($U$, $U_{cq}$, $p_r$)
45     $U \leftarrow U \cup (< p_r, U_{cq} >)$
46     return $U$
47 end
```

Algorithm 2: Procedures of HOR

When S^{REL} is empty (i.e., there are no relevant sessions and, hence, no candidates to recommend, see line 8), we revert to the incremental critiquing recommendation. To do so, we first of all obtain the last critique, U_{cq}, with the *getLastCritique* procedure from the user model U (see line 9). Once obtained the last critique, we filter out in CB' all the products from CB that satisfy this critique (see line 10). In particular, the satisfaction measure $\delta(p, U_{cq})$ (see line 10) returns 1 if case p_i satisfies the critique U_{cq} or 0 otherwise. If no critique has been performed, all the case base is considered (i.e., $CB' \leftarrow CB$). Next, we sort out all candidate products from CB' in decreasing order, based on the quality score Q (shown in line 11). The *quality* score combines the *compatibility* and *similarity* measure as shown in Equation 6:

$$Q(p_i, p_q, U) = \beta \cdot C_{p_i}(U) + (1 - \beta) \cdot S(p_i, p_q) \tag{6}$$

where p_i is the *ith* candidate product, p_q is the current recommended product, C_{p_i} is the compatibility score that essentially represents the percentage of critiques in the user model that the candidate product p_i satisfies, S is the similarity function based on an Euclidean distance (i.e., the similarity between the candidate product p_i and the current recommended product p_q), and β is a parameter that prioritizes the compatibility or the similarity measure. In particular, the compatibility score is defined as $C_{p_i}(U) = \frac{\sum_{j=1}^{|U|} \delta(p_i,u_j)}{|U|}$, where p_i is the candidate product given an individual user model (U), δ is the satisfaction function, and $|U|$ is the number of recommendation-critique pairs in U. Finally, the new recommendation product p_r is the candidate product with the most quality (see line 12).

In the second phase, **UserReview** (see lines 30-35), the user **reviews** the recommendation by introducing a critique. HOR performs two steps: (1) it receives feedback from the user in the form of a critique, see line 32; and (2) it removes current product recommendation, p_r, from the case base, CB, see line 33. At the end, this phase returns the case base and the critique.

The third phase, **QueryRevise** (see lines 36-40), is focused on the **revision** of the product query p_q for the next cycle. Basically, the current product recommendation p_r becomes the new product query, p_q.

Finally, the **UpdateModel** phase (see lines 41-47) stores both the product recommendation p_r and the current user critique U_{cq} in the user model U. Maintaining a user model is not as simple as storing a list of previously selected recommendation-critique pair. Some critiques may be inconsistent with earlier critiques. Hence, HOR assumes that users may refine their requirements over time because they are learning about the product space during the recommendation process. In this way, it is essential to update the user model by adding the latest recommendation-critique pair ($\{p_r, U_{cq}\}$), if there is one, but only after pruning previous critiques so as to eliminate these sorts of inconsistencies. Pruning means removing all existing recommendation-critique pair whose critique *contradict* the latest critique, U_{cq} (see line 43) and those for which the new critique is a *refinement* (see line 44). The *contradict* procedure (see line 43) removes any recommendation-critique pair where the critique contradicts the current critique U_{cq}. For example, in a camera domain, if the user model U contains a recommendation-critique pair ($\{Canon700D, (price, <, \$300)\}$)[6] and the user performs the critique $(price, >, \$400)$ (i.e., the current critique, U_{cq}) over the current recommendation $NikonD3300$ (p_r), the recommendation-critique pair that contains the critique $(price, <, \$300)$ will be removed as their critique contradicts the current critique, U_{cq}. Additionally, the *refine* procedure (see line 44) reformulates those recommendation-critique pairs for which the new critique is a refinement. For example, considering the previous example, when the user model contains a recommendation-critique pair ($\{Canon700D,(price, <, \$300)\}$). Then, the user performs a critique $(price, <, \$200)$ over the current recommendation $NikonD3300$. In this case, the recommendation-critique pair ($\{Canon700D,(price, <, \$300)\}$) will be refined to ($\{NikonD3300,(price, <, \$200)\}$).

[6] To simplify the example, we only show the name of the model of the recommended product r_i, but the pair contains all features of the product.

4 Integrating State-of-the-art Approaches

In this paper we propose to integrate users' opinions with history recommendations. However, we believe that our proposal (HOR) as well as previous history-based recommenders may improve their recommendation process by adding previous advances in critique-based recommendations. We have chosen HGR [37] as it outperforms previous history-based recommenders defined in the literature (see Figure 1). Therefore, we propose to integrate HOR and HGR with previous state-of-the-art approaches. The new algorithms will be called HOR-I and HGR-I, respectively.

Concretely, we focus on the **RecommendItem** phase when relevant sessions are not found in the history session base (see line 8 in Algorithm 2). In this situation, HOR and HGR concretely revert to Incremental Critiquing algorithm, shown in the Foundations square in Figure 1. Instead, we propose to revert to an adapted incremental critiquing algorithm, which modifies the quality measure Q described in Equation 6. This adapted incremental critiquing belongs to the Foundations Improvements square in Figure 1. Specifically, we integrate into the recommendation process a weighting scheme for the similarity measure in Q, called Local User Preference Weighting (LW) [35], and a new compatibility measure in Q, named Exponential Hit-Loss (EHL) [34]. It has been demonstrated that these two proposals (i.e., LW and EHL) significantly improve the recommendation efficiency of the incremental critiquing algorithm [34].

First, for the similarity measure, we propose to integrate the LW approach whose aim is to prioritize the similarity of those features that have not yet been critiqued during a given session. The rationale behind LW is that the most compatible (see Equation 6) cases are quite similar on their critiqued features and their differences mainly belong to those features that have not yet been critiqued. In this way, a feature that has not been critiqued will assume a weight value of 1.0, and a decrease will be applied when a critique is satisfied by the product. In particular, we compute the similarity as follows:

$$S(p_i, p_q) = \sum_{s=1}^{|F|} w(p_i^{f_s}) \cdot d(p_i^{f_s}, p_q^{f_s}) \tag{7}$$

where $w(p_i^{f_s})$ is the weight associated to the feature, f_s, of the candidate product, p_i, and $d(p_i^{f_s}, p_q^{f_s})$ is the distance between the candidate product, p_i, and the current recommended product p_q (the product query). The distance measure, $d(p_i^{f_s}, p_q^{f_s})$ is the Euclidean measure between the current recommendation and the candidate product.

Second, for the compatibility measure, in other studies, it has been demonstrated that the use of Reinforcement Learning (RL) in the compatibility enhances the retrieval in IC. Specifically, we propose to use Exponential Hit-Loss (EHL), which considers that users increase their knowledge over cycles and, accordingly, their last preferences are more important to be satisfied than their initial ones. Concretely, the EHL compatibility, it is defined as:

$$C_t^{p_i} = \begin{cases} C_{t-1}^{p_i} \cdot (1+\alpha)^{(h_t+t)k}, \text{ if } R_t^{p_i} = 1 \\ C_{t-1}^{p_i} \cdot (1-\alpha)^{(\ell_t+t)k}, \text{ if } R_t^{p_i} = 0 \end{cases} \tag{8}$$

where h_t and ℓ_t are the number of times that candidate product p_i has satisfied (hit) or not (loss) the critiques to time t, respectively (for each product 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, as it was described in [34]). The regularization parameter k is utilized to change the influence of the exponent factor depending on the objective of the application and the size of the data set in terms of products and features.

5 Evaluation through a Simulator

In this section we report experiments with simulated users with the aim of evaluating the efficiency of our proposals (HOR, HOR-I, and HGR-I). We compare them to two related baseline algorithms, IC and HGR. Specifically, we concentrate on five different recommenders: (1) Incremental Critiquing (IC); (2) the History-Guided Recommender (HGR); (3) HGR, which integrates previous improvements (HGR-I); (4) the History and Opinion Recommender (HOR); and (5) HOR, which also integrates previous improvements (HOR-I).

5.1 Data sets and Methodology

In our experiments we used two datasets: SMARTPHONE[7] and RESTAURANT[8]. The details of the data sets are shown in Table 1, the smallest being the SMARTPHONE data set with 1721 products. All data sets contain nominal and ordinal features. For example, in the SMARTPHONE data set, the manufacturer is a nominal feature and the price is one of the ordinal features.

Table 1 Data Sets Characteristics

Data set	**Products**	**Nominal features**	**Ordinal features**
SMARTPHONE	1721	5	9
RESTAURANT	9945	25	14

Similarly to previous history-based works, we adopted the methodology used in [19,36,37] to automatically generate past critiquing sessions based on the behavior of rational users. Specifically, we select a random product as our target. From this target we automatically create a query, by selecting from 3 up to 5 features from the target at random, which acts as a starting point for each session. Each sessions begins by the recommender retrieving a best-matching product for the query. From here the artificial user must select a

[7] This data set is available on demand.

[8] This data set has been used in [37] and it was kindly provided by the authors.

feature to critique. To do this artificial behavior we automatically select one of the features of the recommended product and critique it in the direction of the target product. Each session terminates once the target product has been recommended. This process can be repeated for generating an arbitrary number of past critiquing sessions. Concretely, we generated five session bases with different sizes for each domain (see the column *Session Bases* in Table 2). Additionally, we divided both data sets into several subsets to evaluate the recommendation algorithms in search spaces of different sizes, as shown at third column in Table 2.

Table 2 Data sets configuration for the experiments

Data set	Session Bases	Search Spaces
SMARTPHONE	500, 1000, 3000, 5000, and 10000	1000, 1300, and 1721
RESTAURANT	500, 1000, 3000, 5000, and 10000	6000, 8000, and 9945

In our evaluation, we also used the leave-one-out methodology previously used in [29,34], which takes a set of products randomly selected from the search space (*original base*) and uses each of them as a *test case*. Each selected test case is temporarily removed from the data set and used in two ways. First, the *test case* serves as a basis for generating a set of (simulated) initial user queries, *initial queries set*, by taking random subsets of its features. Second, the *test case* is used to select, from the *original base*, the case that is most similar to it. This case represents the recommendation *target product* for the experiments. That is, the product that the simulator's "artificial user" reaches through a series of (random) critiques. A set of random critiques is generated in each recommendation cycle and they are all compatible with the known target case. The "artificial user" randomly selects one of the critiques from this set. Thus, when the remaining set of cases are filtered according to the last critique selected randomly by the "artificial user", it results in the target case being left in the filtered set of cases. In addition, we used three type of *initial queries* (i.e., *hard*, *moderate*, and *easy*), by selecting one, three or five features respectively from a random target product, which act as a starting point for each evaluated session [30,34].

Finally, we set up the α parameter as $\alpha = 0.6$ in the $OverlapScore$ measure (see Equation 5), based on better results in our empirical experiments with several α values. In addition, the β parameter in Equation 6 is set to 0.75 based on previous empirical experiments [29], with the aim to prioritize the candidate product that satisfy the current critique.

5.2 Experimental Results

In this section, we will begin by analyzing the recommendation efficiency of all algorithms by means of the average session length ASL[9] (from now on

[9] The session length measures the number of cycles that a user must work through before being presented with their ideal target product.

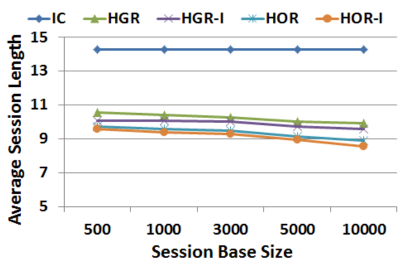
ASL) and then we will evaluate the benefit[10] of the algorithms (i.e., HGR, HGR-I, HOR, and HOR-I) in comparison to the IC algorithm, which is used as baseline. It is important to remark that the ASL measure has been widely used in evaluations of critique-based recommenders, such as [29,34,36,39].

First, in Figure 3 we present several figures with the results of analyzing the ASL to reach a target of algorithms with different size of session base in relation to the size of the search space for both SMARTPHONE and RESTAURANT domains. In these figures, each of the lines shows the average of ASL obtained in all the queries (*hard*, *moderate*, and *easy*) for each particular algorithm described above. It is important to recall that IC does not make use of past critiquing sessions from other users. For this reason, its results are the same for all session bases.

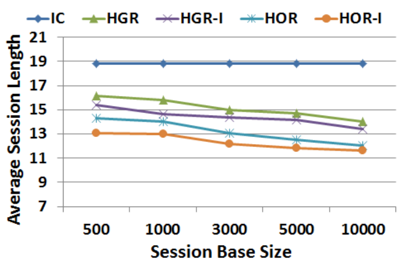
In Figures 3(a), 3(c), and 3(e), we show the results using 1721, 1300, and 1000 smartphones as a case base of products, respectively. This case base is the search space where users are looking for their desired product. The largest the case base, the more difficult to locate a product. Note in Figure 3(a), that is the largest case base, with 1721 smartphones, the IC algorithm reached an ASL of 13.96 whereas HGR, HGR-I, HOR, and HOR-I achieved the best results when increasing the history-session base size (i.e., 10000), obtaining in a history-session base of 10000 an ASL of 10.51, 9.63, 9.18, and 8.64, respectively. Moreover, in Figures 3(c) and 3(e), that we used a case base of 1300 and 1000 smartphones, the best results were also for the largest history-session base. Thus showing that the larger the history-session base, the more chances the recommender for finding sessions that are similar to the current session. Specifically, in Figure 3(c) IC achieves 12.48 cycles while in contrast HGR, HGR-I, HOR, and HOR-I achieve an ASL of 10.42, 9.08, 8.97, and 8.67, respectively. Finally, in Figure 3(e) for the smallest case base, which contains only 1000 products, the ASL of IC was 11.47 and the other algorithms results in an ASL of 8.92 in the case of HGR, 7.98 for HGR-I, 7.88 for HOR, and 7.58 in HOR-I.

In Figures 3(b), 3(d), and 3(f), we show the results using 9945, 8000, and 6000 restaurants as a case base of products. Note that the search space of this data set is larger than the SMARTPHONE data set, in which the largest case base size is 1721 smartphones. In the RESTAURANT data set, IC reaches an ASL of 18.81, 18.10, and 17.57 for each size of the case base (i.e., 9945, 8000, and 6000), respectively. Furthermore, in this data set the best results were also for the largest history-session base as in the SMARTPHONE data set. As we can see in Figure 3(b) HGR, HGR-I, HOR, and HOR-I achieve an averaged ASL of 14.02, 13.55, 12.08, and 11.68, respectively. These results are better than the ones obtained with IC. In Figure 3(d), which shows the results for the medium case base size, HGR, HGR-I, HOR, and HOR-I obtain an ASL of 13.94, 13.44, 12.40, and 11.50. Finally, in Figure 3(f) for the smallest case

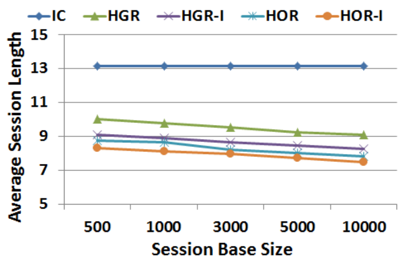
[10] We computed the percentage of benefit as $Benefit(x, y) = (1 - \frac{y}{x}) \cdot 100$, where y and x stand for the ASL of the compared algorithm and the base line, respectively.



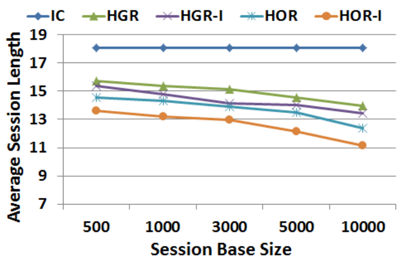
(a) Using 1721 smartphones



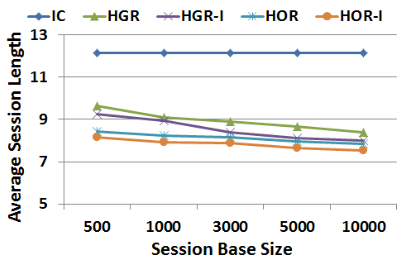
(b) Using 9945 restaurants



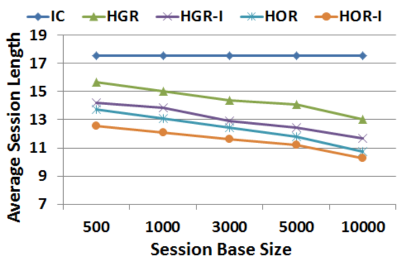
(c) Using 1300 smartphones



(d) Using 8000 restaurants



(e) Using 1000 smartphones



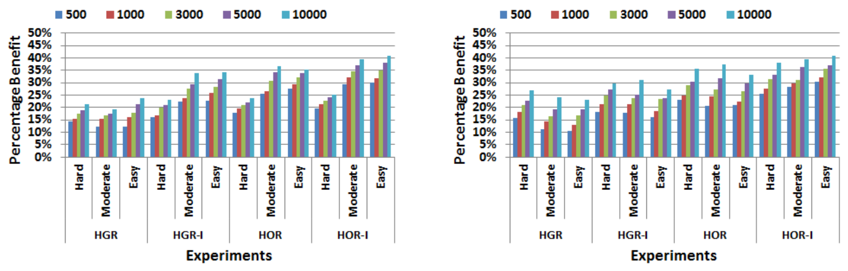
(f) Using 6000 restaurants

Fig. 3 The ASL of the proposed algorithms compared to IC and HGR algorithms in a SMARTPHONE and RESTAURANT domains

base the ASL is 13.03 for HGR, is 11.80 for HGR-I, is 10.72 for HOR, and is 10.66 for HOR-I.

Analyzing the results in both data sets we can observe that the curve of our proposals (i.e., the HOR algorithm and the ones that integrate previous improvements —the HOR-I and the HGR-I algorithms) are less dependent of the size of history-session base than HGR. Note that in all graphs depicted in Figure 3 the ASL is lower in our proposals (HOR, HOR-I, and HGR-I) than in IC and HGR. Moreover, our proposals result in the greatest reduction of the ASL for the largest history-session bases.

Secondly, we will address attention to the benefit of history-based algorithms in comparison to the baseline IC algorithm. Figure 4 shows the average benefit of the defined algorithms in comparison to the IC algorithm using different history-session base sizes –i.e., from 500 to 10000 sessions– for both SMARTPHONE and RESTAURANT data sets as well as the benefit of the defined



(a) SMARTPHONE case base (b) RESTAURANT case base

Fig. 4 Benefits of our proposals in comparison to the IC algorithm

algorithms using each type of query (i.e., *hard*, *moderate*, and *easy*). First of all, analyzing both data sets, it can be seen that the benefit depends on the size of the history-session base (i.e., the benefit is the greatest for the largest history-session base) and it also depends on the type of the query. For example, in most of cases, easy queries obtained the greatest benefit.

Analyzing each of the algorithms in depth, as shown in Figure 4(a), the greatest benefit is obtained by the HOR-I algorithm with a benefit of between 20% and 41%. Moreover, HOR obtained a benefit in the range of 18% and 37% whereas the benefit obtained by HGR-I ranges from 16% to 34% in comparison to IC. Using the RESTAURANT case base we obtained similar results to the SMARTPHONE data set. As we can see in Figure 4(b), HOR obtained good benefits, which are in the range of 23% to 37%. It is worth noting that the benefit in recommendation efficiency is also improved when integrating previous techniques, such as LW and EHL with HGR. That is, HGR-I achieved a percentage of benefit over IC that ranges from 18% to 32%, while in contrast HGR obtained a benefit of between 14% and 27%. The same applies to HOR-I, which obtained a benefit in the range of 25% to 41%.

We also analyzed how often the history-based recommenders are unable to find relevant sessions from the session base (SB) and, thus, revert back to the Incremental Critiquing algorithm. In our experiments, both HGR and HOR algorithms revert to IC the same number of times that HGR-I and HOR-I, respectively. In the SMARTPHONE domain, with a sessions base of size 500, HGR-I reverts with a 52% of times to the improved IC algorithm. Reversions decreases until 1.7% at the largest session base size. In the case of HOR-I algorithm, it reverts to the improved IC algorithm in 45% of occasions with 500 history sessions and this value is reduced to 1.2% with 10000 past critiquing sessions. In the RESTAURANT domain, we obtained similar results, HGR-I reverts to the *critiquing* algorithm in 46% of occasions using 500 history sessions and this percentage is decreased until 0.8% for a history session base of size 10000. In the case of HOR-I algorithm, it reverts in 40.2% and 0.7% of instances using 500 and 10000 past critiquing sessions, respectively.

To sum up, the results highlight that the benefit in recommendation efficiency of all our proposals (i.e., the HOR algorithm and the ones that integrate previous improvements —the HOR-I and HGR-I algorithms) are greater than the IC and HGR algorithms, using both the SMARTPHONE and RESTAURANT case bases. This suggests that our initial hypothesis in this paper is true. Recall that the hypothesis was that the integration of some of the previous proposals in state-of-the-art of critique-based recommendation may improve their performance.

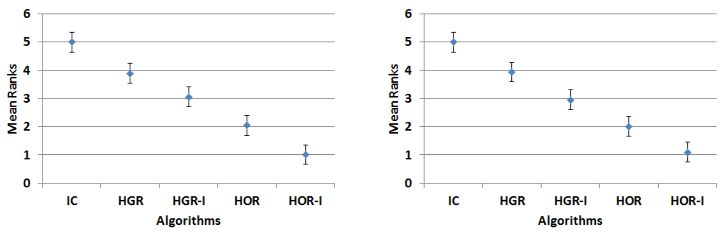
Additionally, in order to demonstrate that the hypothesis about the integration of previous proposals significantly outperforms previous algorithms, we apply the Friedman and Nemenyi tests [14] to analyze whether the difference between the tested algorithms and the baseline is statistically significant in both data sets. These tests, as stated by [9], are specialized procedures for testing the significance of differences between multiple means and they also control the multiple hypothesis testing problem that is usually present in a pair T-TEST. For this reason, given that we are testing multiple[11] recommender systems, the use of Friedman and its corresponding Nemenyi post-hoc test is more appropriate.

First of all, we compute the *mean rank* (r) of each algorithm considering all the experiments. In particular, the evaluation considers $k = 5$ algorithms (i.e., IC, HGR, HOR, HOR-I, and HGR-I) and $N = 45$ different experiments for each test. The experiments depend on three different case base sizes, three different queries (i.e., hard, moderate, and easy), and five different history-session bases. We ranked alternative algorithms, for each experiment, following the practice of [14]. The one that attains the best performance is ranked 1, the second best ranked 2, so on and so forth. Then, an algorithm's mean rank is obtained by averaging its rank across all experiments.

Secondly, we apply the Friedman and Nemenyi tests to analyze whether the difference between algorithms is statistically significant. In particular, we applied the Friedman test, F_F is distributed according to the F distribution with $(5-1) = 4$ and $(5-1) \cdot (45-1) = 176$ degrees of freedom. The critical value of $F(4, 176)$ is equal to 2.42 at the 0.05 critical value. For our efficiency comparison we obtained the values of X_F= 145.65 and $F_F = 186.58$ for the efficiency rankings. As the value of F_F is higher than 2.42 we rejected the null hypothesis. Once we have checked for the non-randomness of the results, we computed the Nemenyi test to find out which algorithms are significantly different. In our case, when comparing five algorithms with a critical value $\alpha = 0.05$, $q_{0.05} = 2.569$ for a two-tailed Nemenyi test. We obtained a critical difference value $CD = 0.699$. The Nemenyi results are shown in Figures 5(a) and 5(b), for the SMARTPHONE and RESTAURANT data sets, respectively. In these graphs, diamonds represent the mean ranks of each algorithm and vertical lines across diamonds indicate the 'critical difference', CD. Basically, the efficiency of two algorithms is significantly different if their vertical lines are not overlapping. For example, it can be seen that in both domains all tested algorithms are significantly better than the baseline (IC). In fact, the best algorithm (i.e., the one with the shortest mean rank) is HOR-I, which is better

[11] It is considered multiple five or more algorithms.

than the others. In addition, we can see that our proposals –HGR-I, HOR, and HOR-I– are also significantly better than HGR with a confidence of 95%.



(a) SMARTPHONE case base (b) RESTAURANT case base

Fig. 5 Application of the Nemenyi test to alternative algorithms' mean rank of ASL

In summary, from our significance analysis, we conclude that: (1) HOR algorithm is significantly better than IC and HGR; and (2) the integration of previous improvements on the field of critique-based recommenders, HGR-I and HOR-I, enhances significantly the recommendation efficiency of well-known state-of-the-art algorithms (IC and HGR), which confirms our initial hypothesis.

6 Real-user evaluation

Considering that the results with the simulator has shown that the best algorithms are respectively HOR-I and HGR-I, in this section, we evaluate the efficiency, efficacy, and user satisfaction of these proposals in comparison to the IC algorithm, with real users. History-based recommenders have been always evaluated with a simulator. To the best of our knowledge, this is the first time history-based recommenders have been evaluated with real users.

6.1 Methodology

We developed a wizard-like online web application[12] that contains all instructions, interfaces and questionnaires so that participants could remotely perform the evaluation. The experiment was designed as a between-subject test. That is, a participant only evaluates a couple of algorithms and for each algorithm the participant is asked to fill in a post-test questionnaire. We recruited 50 participants, diverse in features such as age, gender, and computer skills. The test was performed using the SMARTPHONE data set, which is detailed in Table 1. In addition, we used as *SB* the largest session base with 10000 past critiquing sessions obtained from the simulator.

The online evaluation procedure consists of the following phases:

[12] The application is available in http://164.77.134.92:8081/WebRec/signin.htm

1. *Pre-test:* In this phase participants were asked to input their background information and their previous experience with recommender systems.
2. *Training:* In this phase participants performed a training task where they had to find a predefined target product, which was randomly selected from the case base.
3. *Efficiency Test:* In this phase users performed two tasks with a predefined target product with the aim of evaluating the recommendation efficiency of the algorithms by means of the ASL measure. To do so, the online web application randomly chooses one algorithm for each task between IC, HGR-I, and HOR-I. The randomly selection is to equilibrate any potential bias.
4. *Efficacy Test:* In this phase users performed one task without a predefined target product with the aim of evaluating the decision accuracy. We apply the same methodology described in [4]. Concretely, the decision accuracy was quantitatively measured by the fraction of participants that switched to a different, better option when, once finished the recommendation session, they browsed all alternatives in the data set. A lower switching fraction means that the algorithm allows a higher decision accuracy since most of users are able to find their target choice with it. In this phase we evaluated the baseline IC and our algorithms HOR-I and HGR-I. Since SMARTPHONE data set is too large (1722 products) we selected a subset of 90 products. Note that this reduced size is not a limitation of our system but a way of facilitating users' searching task, being enough for validating decision accuracy.
5. *User Satisfaction:* For each task in the *efficiency test* participants answered a questionnaire, which consists of 5 questions using a seven-point likert scale, where 1 corresponded to "strongly disagree" and 7 to "strongly agree". In addition, in the *efficacy test* participants also answered a questionnaire which consists of 7 questions. It is important to remark that both questionnaires are based in previous works [32, 24].

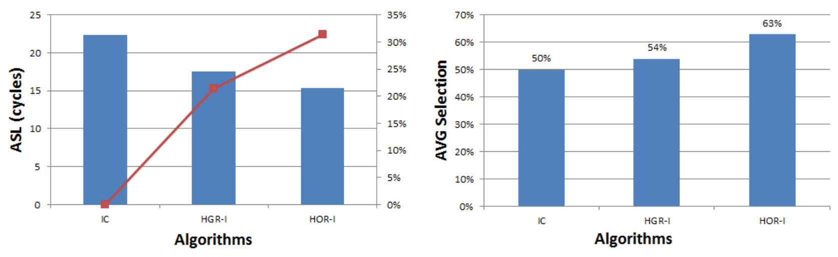
After the test, we collected data from logs and questionnaires. Then, we analyzed these data to extract relevant information concerning test objectives.

6.2 Experimental results

In this section, we show results related to efficiency, efficacy, and user satisfaction.

Figure 6(a) shows that the ASL in both HGR-I (17.58) and HOR-I (15.35) is lower than IC (22.35). In addition, the red line in this figure shows the benefit[10] of both HGR-I and HOR-I over IC. Concretely, HGR-I reaches a benefit over IC of 21.37% and in HOR-I this benefit is 31.35%. Note that the real-user evaluation confirms the simulation results and the best efficiency in these experiments is obtained by HOR-I again.

We apply the ANOVA statistical method to analyze whether the differences in efficiency between HOR-I and HGR-I with respect to the IC are statistically significant. As denoted in [9] when the number of algorithms in the comparison is lower than five, it is more convenient to use T-TEST or ANOVA. Therefore, we apply the ANOVA in three algorithms, $k = 3$, with $k - 1 = 2$ degrees of freedom. The ANOVA results show that the differences are significant among the algorithms, *p-value* of 0.02421, that is lower than the critical value, $\alpha = 0.05$. Additionally, to denote that the efficiencies of HOR-I and HGR-I are significantly better than IC separately, we apply the multiple significance Bonferroni test [1]. We obtained a *p-value* of 0.041 between IC and HGR-I and 0.022 between IC and HOR-I.



(a) Recommendation efficiency (b) Recommendation efficacy

Fig. 6 Efficiency and efficacy results in the real-user evaluation

Figure 6(b) shows the results of the recommendation efficacy (i.e., decision accuracy), where HOR-I obtained a relatively higher decision accuracy than HGR-I (61% and 52% respectively). This measurement means that, in average, in the HOR-I recommender the final accepted product is 61% similar to the final selection when users browsed the full set of products. Moreover, the IC algorithm obtained the lower decision accuracy (50%).

To collect user satisfaction measurements we designed two post-test questionnaires based on previous studies [32, 24], for evaluating the user perception in two different scenarios: (1) when users navigate to locate a predefined target product (see questions in Table 3); (2) when users navigate to locate a desired product without a target (see questions in Table 4).

Results are shown in Figures 7(a) and 7(b). Notice that, in average for our proposals HGR-I and HOR-I, all answers in both figures are higher than 4.62 in a 7 point likert scale with the exception of the Q2 in Table 3 and Q4 in Table 4, which by the own nature of these questions lower values represent better results.

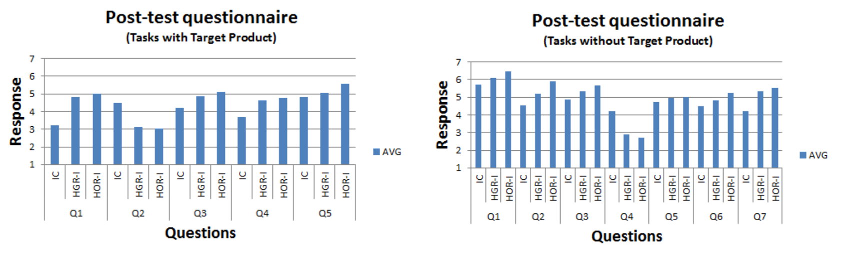
Figure 7(a) shows that participants positively evaluated the efficiency of the proposals (Q1 and Q2). In particular, they evaluated Q1 with an average of 4.83 for HGR-I and 5.01 for HOR-I. Moreover, Q2 responses (with an average of 3.11 for HGR-I and 3.05 for HOR-I) show that our proposals provide adequate recommendations in a short period of time. In addition, participants positively evaluated the privacy of their information (Q3) in the recommender (4.85 in HGR-I and 5.1 in HOR-I). Finally, Q4 responses (with an average

Table 3 Efficiency Post-test questionnaire

Question Number	Statement
Q1	In general, I perceived that the recommender followed my preferences.
Q2	It takes too much time before the system provides adequate recommendations.
Q3	I feel confident that the system respects my privacy.
Q4	Using the system is a pleasant experience.
Q5	I would use this recommendation system for buying products in the future.

Table 4 Efficacy Post-test questionnaire

Question Number	Statement
Q1	I liked the items recommended by the system.
Q2	I believe that the final selected product was the best for me.
Q3	Each of the recommended products was relevant.
Q4	It takes too much time before the system provides adequate recommendations.
Q5	I feel confident that the system respects my privacy.
Q6	Using the system is a pleasant experience.
Q7	I would use this recommendation system for buying products in the future.



(a) Average responses value for efficiency post-test questionnaire.

(b) Average responses value for efficacy post-test questionnaire.

Fig. 7 User perception on the efficiency and efficacy of the proposals

of 4.62 for HGR-I and 4.78 for HOR-I) and Q5 responses (with an average of 5.44 HGR-I and 5.62 HOR-I) show a high level of user satisfaction regarding to the use of the recommender. Notice that our proposals obtained better results than the baseline IC in all questions.

Figure 7(b) depicts that the results with respect to the efficacy of the proposals are very satisfactory as shown in Q1 responses (with a value of 6.10 for HGR-I and 6.45 for HOR-I) and Q2 (with a value of 5.21 for HGR-I and 5.89 for HOR-I). With respect to the efficiency of the proposals, Figure 7(b) shows a high level of user satisfaction in Q3 and Q4. In the former HGR-I and HOR-I reach an average of 5.32 and 5.68, respectively. In Q4, both HGR-I (2.89) and HOR-I (2.70) show that participants perceived that the proposals provide adequate recommendation in a short period of time. Moreover, participants positively evaluated the privacy (Q5) of their information in the recommender (4.98 in HGR-I and 5.02 in HOR-I). Finally, Q6 and Q7 responses show that users had a positive perception about the usefulness of the recommender. Concretely, Q6 obtain an average of 4.83 for HGR-I and 5.23 for HOR-I and Q7 an average of 5.32 HGR-I and 5.54 HOR-I. Moreover, Figure 7(b) shows that the user perceived efficacy of the IC algorithm is lower than our proposals.

Summarizing, our experiments with real-users show satisfactory results when we analyze the behaviour and perception of the users. Concretely, our proposals reduce the ASL in the recommendation and show a high level of recommendation efficacy. Moreover, responses to the post-test questionnaires depict a positive user perception with respect to the efficiency, efficacy, and user satisfaction.

7 Conclusions and Future Work

Critique-based recommenders are data-driven in their nature since they use a conversational cyclical recommendation process that elicits user feedback. Considering that most of the state-of-the-art approaches in the literature differ in the source of data (i.e., from a critique to social media data) and how this data is mined to recommend products, in this paper we concentrate on evaluating these two aspects that influence data-driven decision making in critique-based recommenders.

In this paper we hypothesize that the integration of several sources of data and previous enhancements in the field may improve the performance of both existing proposals and new proposals in the future. To this end, first of all we have presented a novel critique-based recommender, called HOR, which uses critiquing as feedback mechanism, a history of sessions and also integrates the users' opinions extracted from e-commerce social media sites. Next, we have integrated previous enhancements (i.e., a weighting measure, called LW, and a reinforcement learning compatibility measure, called EHL) on the field of critique-based recommenders to the new proposal (HOR) and to one of the best and well-known critique-based algorithm, called HGR, that also uses as source of data a history of sessions. The resulting algorithms that integrate previous state-of-the-art enhancements are: HGR-I and HOR-I. Finally, we have evaluated all these algorithms in two different domains in order to confirm our initial hypothesis. Our results not only support the hypothesis that the integration of different sources of data as well as previous state-of-the-art improvements for mining this data (HGR-I and HOR-I) improves the recommendation efficiency in front of HGR and the traditional IC algorithms, but they also confirm with an statistical analysis that this integration significantly benefits the efficiency and efficacy of previous approaches.

Finally, as future work, we consider that more sophisticated approaches can be proposed. Note that from all the algorithms analyzed in this work, the best one is the proposed HOR-I, which integrates critiquing, past recommendation sessions from other users, user's opinions from social media sites and previous state-of-the-art proposals for improving the recommendation process in critique-based recommenders. Accordingly to our results, we plan to extend the use of social media data such as reviews. In addition, we will address the analysis of using different sources of data into a Collaborative and Conversational Recommender, which integrates several feedback mechanisms.

Acknowledgements This research has also received support from the projects SGR-2014-623 and TIN2015-71147-C2-2 from the Spanish Ministry of Science and Innovation.

References

1. Bland, J.M., Altman, D.G.: Multiple significance tests: the bonferroni method. Bmj **310**(6973), 170 (1995)
2. Brynjolfsson, E., Hitt, L.M., Kim, H.H.: Strength in numbers: How does data-driven decision making affect firm performance? Available at SSRN: https://ssrn.com/abstract=1819486 or http://dx.doi.org/10.2139/ssrn.1819486 (2011)
3. Burke, R., Hammond, K., Yound, B.: The FindMe approach to assisted browsing. IEEE Expert (1997)
4. Chen, L., Pu, P.: Evaluating critiquing-based recommender agents. In: Proceedings of the National Conference on Artificial Intelligence, vol. 21, pp. 157–162 (2006)
5. Chen, L., Pu, P.: Preference-based organization interfaces: aiding user critiques in recommender systems. In: User Modeling 2007, pp. 77–86. Springer (2007)
6. Chen, L., Pu, P.: Critiquing-based recommenders: survey and emerging trends. User Modeling and User-Adapted Interaction **22**(1-2), 125–150 (2012)
7. Contreras, D., Salamó: On the Use of User-generated Content in Critiquing Recommendation. In: Proceedings of the XVIII International Conference of the Catalan Association for Articial Intelligence, pp. (195–204) (2015)
8. Contreras, D., Salamó, M., Rodríguez, I., Puig, A.: A 3d visual interface for critiquing-based recommenders: Architecture and interaction. International Journal of Artificial Intelligence and Interactive Multimedia **3**(3), 7–15 (2015)
9. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research **7**, 1–30 (2006)
10. Dong, R., O'Mahony, M., Schaal, M., McCarthy, K., Smyth, B.: Sentimental product recommendation. In: Proceedings of the 7th ACM Conference on Recommender Systems, pp. 411–414. ACM (2013)
11. Dong, R., Schaal, M., O'Mahony, M., McCarthy, K., Smyth, B.: Opinionated Product Recommendation. In: Case-Based Reasoning Research and Development, *LNCS*, vol. 7969, pp. 44–58. Springer (2013)
12. Dong, R., Schaal, M., OMahony, M., McCarthy, K., Smyth, B.: Mining features and sentiment from review experiences. In: Case-Based Reasoning Research and Development, pp. 59–73. Springer (2013)
13. Elahi, M., Ricci, F., Rubens, N.: Active learning strategies for rating elicitation in collaborative filtering: A system-wide perspective. ACM Trans. Intell. Syst. Technol. **5**(1), 13:1–13:33 (2014)
14. 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)
15. Konstan, J., Riedl, J.: Recommender systems: from algorithms to user experience. User Modeling and User-Adapted Interaction **22**(1-2), 101–123 (2012)
16. Koren, Y., Bell, R.: Advances in collaborative filtering. In: Recommender Systems Handbook, pp. 145–186. Springer (2011)
17. Mandl, M., Felfernig, A.: Improving the performance of unit critiquing. In: User Modeling, Adaptation, and Personalization, vol. 7379, pp. 176–187. Springer (2012)

18. McCarthy, K., Salamó, M., Coyle, L., McGinty, L., Smyth, B., Nixon, P.: Group recommender systems: A critiquing based approach. In: Proceedings of the 11th International Conference on Intelligent User Interfaces, IUI '06, pp. 267–269. ACM Press (2006)
19. McCarthy, K., Salem, Y., Smyth, B.: Experience-based critiquing: Reusing critiquing experiences to improve conversational recommendation. In: Proceedings of the International Conference on Case Base Reasoning, pp. 480–494. Springer (2010)
20. McGinty, L., Reilly, J.: On the evolution of critiquing recommenders. In: Recommender Systems Handbook, pp. 419–453. Springer (2011)
21. McSherry, D., Aha: The Ins and Outs of Critiquing. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 962–967 (2007)
22. Pazzani, M., Billsus, D.: The Adaptive Web: Methods and Strategies of Web Personalization, chap. Content-Based Recommendation Systems, pp. 325–341 (2007)
23. Provost, F., Fawcett, T.: Data Science and its Relationship to Big Data and Data-Driven Decision Making. Big Data **1**(1), 51–59 (2013). DOI 10.1089/big.2013.1508
24. Pu, P., Chen, L., Hu, R.: A user-centric evaluation framework for recommender systems. In: Proceedings of the Fifth ACM Conference on Recommender Systems, pp. 157–164 (2011)
25. Pu, P., Chen, L., Kumar, P.: Evaluating product search and recommender systems for e-commerce environments. Electronic Commerce Research **8**(1-2), 1–27 (2008)
26. Pu, P., Faltings, B.: Decision Tradeoff Using Example-Critiquing and Constraint Programming. Constraints **9**(4), 289–310 (2004)
27. Pu, P., Faltings, B., Chen, L., Zhang, J., Viappiani, P.: Usability guidelines for product recommenders based on example critiquing research. In: Recommender Systems Handbook. Springer (2011)
28. Reilly, J., McCarthy, K., McGinty, L., Smyth, B.: Dynamic critiquing. In: Advances in Case-Based Reasoning, *Lecture Notes in Computer Science*, vol. 3155, pp. 763–777. Springer (2004)
29. Reilly, J., McCarthy, K., McGinty, L., Smyth, B.: Incremental critiquing. Knowledge-Based Systems **18**(4-5), 143–151 (2005)
30. Reilly, J., Smyth, B., McGinty, L., McCarthy, K.: Critiquing with confidence. In: Case-Based Reasoning Research and Development, *Lecture Notes in Computer Science*, vol. 3620, pp. 436–450. Springer (2005)
31. Reilly, J., Zhang, J., McGinty, L., Pu, P., Smyth, B.: A comparison of two compound critiquing systems. In: Proceedings of the 12th Int. Conf. on Intelligent User Interfaces, pp. 317–320. ACM, USA (2007)
32. Ricci, F., Nguyen, Q.: Acquiring and Revising Preferences in a Critique-Based Mobile Recommender System. IEEE Intelligent Systems **22**(3), 22–29 (2007)
33. Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.): Recommender Systems Handbook. Springer (2011). DOI 10.1007/978-0-387-85820-3
34. Salamó, M., Escalera, S.: Increasing retrieval quality in conversational recommenders. IEEE Transactions on Knowledge and Data Engineering **24**(10), 1–14 (2012)
35. Salamó, M., Reilly, J., McGinty, L., Smyth, B.: Improving Incremental Critiquing. In: Proceedings of the 16th Artificial Intelligence and Cognitive Science, pp. 379–388 (2005)
36. Salem, Y., Hong, J.: History-aware critiquing-based conversational recommendation. In: Proceedings of the 22Nd International Conference on WWW Companion, pp. 63–64. Switzerland (2013)
37. Salem, Y., Hong, J., Liu, W.: History-guided conversational recommendation. In: Proceedings of the 23rd International Conference on WWW Companion, pp. 999–1004 (2014)
38. Viappiani, P., Faltings, B., Pu, P.: Preference-based search using example-critiquing with suggestions. Journal Artificial Intelligence Research **27**, 465–503 (2006)
39. Zhang, J., Jones, N., Pu, P.: A Visual Interface for Critiquing-based Recommender Systems. Proceedings of the 9th ACM conference on Electronic commerce pp. 230–239 (2008)
40. Zhang, J., Pu, P.: A comparative study of compound critique generation in conversational recommender systems. In: Adaptive Hypermedia and Adaptive Web-Based Systems, *Lecture Notes in Computer Science*, vol. 4018, pp. 234–243. Springer (2006)