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

A cognitively-inspired clustering approach for critique-based recommenders

David Contreras · Maria Salamó

Received: date / Accepted: date

Abstract
Background
The purpose of recommender systems is to support humans in the purchasing decision-making process. Decision-making is a human activity based on cognitive information. In the field of recommender systems, critiquing has been widely applied as an effective approach for obtaining users' feedback on recommended products.
Introduction
In the last decade there have been a large number of proposals in the field of Critique-based Recommenders. These proposals mainly differ in two aspects: in the source of data and in how it is mined to provide the user with recommendations.
Methods
To date, no approach has mined data using an adaptive clustering algorithm to increase the recommender's performance. In this paper we describe how we added a clustering process to a critique-based recommender, thereby adapting the recommendation process and how we defined a cognitive user preference model based on the preferences (i.e., defined by critiques) received by the user. We have developed several proposals based on clustering, whose acronyms are MCP, CUM, CUM-I, and HGR-CUM-I.
Results
We compare our proposals with two well-known state-of-the-art approaches: Incremental Critiquing (IC) and History-Guided Recommendation (HGR). The results of our experiments showed that using clustering in a critique-based recommender leads to an improvement in their recommendation efficiency, since all the proposals outperform the baseline IC algorithm. Moreover, the performance of the best proposal, HGR-CUM-I, is significantly superior to both the IC and HGR algorithms.
Conclusions Our results indicate that introducing clustering into the critique-based recommender is an appealing option since it enhances overall efficiency, especially with a large data set.

Keywords Recommender Systems · Critiquing feedback · User Model · Machine Learning

D. Contreras
Gran via de les Corts Catalanes, 585- 08007 Barcelona, Spain
Tel.: +34-93 4039372
E-mail: dcontrag@maia.ub.es
Conflict of interest: D. Contreras declares that he has no conflict of interest.

M. Salamó
Gran via de les Corts Catalanes, 585- 08007 Barcelona, Spain
Tel.: +34-93 4039372
E-mail: maria.salamo@ub.edu
Conflict of interest: M. Salamó declares that she has no conflict of interest.

Introduction

Although researchers have not yet agreed upon a common definition for cognitive computing, we can say that the term refers to a hardware or software solution that mimics in some way human intelligence capabilities and helps to improve human decision-making [18]. Rather than being explicitly programmed, they learn and reason from their interactions with humans and from their experiences with their environment. The true added value of cognitive computing is the attempt to help people make the best possible decisions in a timely manner.

Recommender Systems (RS) are cognitive computing systems whose aim is to support humans in their decision making processes through convincing, timely product suggestions [37]. There are many and varied

applications of recommender systems. For example, recommending friends in location-based social networks [47]. Recommender systems use different recommendation techniques, from *collaborative filtering* [20,11], which compares other users[1] with active user and recommends items that were liked by users with similar profiles to the active user's profile, to *content-based* techniques [28] that compare users' descriptions of a desired product with item descriptions and recommends the items that match.

Recommender systems research has mostly focused on collaborative filtering techniques. Collaborative filtering suffers from the cold start problem. To overcome the cold start problem, instead of interacting with a recommender system that operates in a *single-shot* fashion (e.g., collaborative filtering), it is more appropriate to engage the user in a *recommendation dialogue* so that incremental user feedback can be used to refine recommendations. The *single-shot* strategy is appropriate to the recommendation of simple products, since users are presented with a single set of recommendations based on an initial query and the recommendation process usually reaches a conclusion.

Critique-based recommenders[2] [27,31,4] guide users through a product space in pursuit of suitable products using a cyclical recommendation process (i.e., a dialogue), alternatively making suggestions and eliciting user feedback, to refine their needs and preferences, based on recent recommendations. Users provide feedback by critiquing features of the currently recommended product. Critiquing is based on the idea that it is easier for a user to critique a product recommendation by saying "*like this but cheaper*" than to construct formal queries [1]. A significant amount of the research carried out on recommender systems has widely recognized the benefits of critique-based recommenders [27,38], since they help customers with ill-defined preferences to both navigate through complex[3] product spaces and to better understand their own buying preferences.

Many different critique-based recommenders have been proposed in the literature. Although previous studies of critique-based recommenders used mechanisms for extracting knowledge from different data sources, to the best of our knowledge, none of them used clustering. Similarly, the majority of prior studies that have dealt with the use of clustering in recommender systems have used a collaborative filtering technique, such as [30,29, 14,21,12]. Instead of promoting an adaptive clustering process, researches have focused mostly on combining clustering with recommendations. However, in recent years many clustering approaches have been proposed that are able to adapt by taking into account changes in the data. One approach to adaptive clustering was a self-tuning p-spectral clustering algorithm based on shared nearest neighbors[17]. Another approach [7] in this vein was an adaptive density data stream clustering algorithm based on an online-offline clustering framework. Lee and Lin [22] developed a hybrid method for data and knowledge-based clustering in gene clustering. In the field of recommender systems, Thanh et al. [43] proposed a novel clustering algorithm in a neutrosophic recommender system for medical diagnosis.

In this work we are concerned with the use of adaptive clustering in critique-based recommenders. We explore the benefits of incorporating adaptive clustering into the recommendation process and in the user preference model. Our objective is to identify clusters of similar products based on their description, learn and reason from the recommender interactions with the user and adapt these clusters in order to use them as a mechanism for making smarter recommendations. We also use clusters to define an adaptive cognitive user preference model. Note that the cognitive user preference model learns from interactions with the user and adapts its content to the user's evolving requirements. Thanks to the enrichment of the critique-based recommender with the adaptive-cognitive user preference model, the recommender is able to focus appropriately on the products the user is interested in and offer quality product recommendations that might otherwise be ignored, thereby, making smarter recommendations and reducing session lengths.

In this paper, our hypothesis is that clustering can improve both the recommendation process and the definition of the user model in critique-based recommenders. With this hypothesis in mind, the contribution of this paper is three-fold:

- First, we propose two new algorithms, called the Maximal Clustering Process (MCP) and the Cognitive User Preference Model (CUM), which are introduced in both an adaptive clustering of the products within the recommendation process and in the user preference model.
- Second, we integrate a number of previous state-of-the-art proposals that are based on a combination of compatibility and weighting scores into the recommendation process of our proposed algorithm

[1] In this work, we use the term *user* to refer to both an e-commerce *customer/shopper* (i.e., a human) and a recommender system *user*.

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

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

(CUM). The new algorithm will be called CUM-I. Additionally, we have integrated CUM-I in a recent well-known history-based recommender called History-Guided Recommendation (HGR) [42]. The resulting algorithm will be called HGR-CUM-I.

– Third, we carried out an exhaustive evaluation of the proposals to demonstrate our initial hypothesis. Our evaluation focuses on a comparison of our proposals (i.e., the MCP, CUM, CUM-I, and HGR-CUM-I) against HGR and the traditional Incremental Critiquing (IC) recommender [33]. The experiments were conducted through an off-line simulator. With these experiments we demonstrated the influence of both the data source (i.e., from the critiques used in IC to the use of history in HGR) and the positive influence of a cognitive user preference model. The results of our in-depth evaluation confirm our hypothesis and indicate that our proposals improve on the efficiency[4] of previous algorithms in critique-based recommenders.

Related Work

Clustering methods partition a set of objects into clusters in such a way that objects in the same cluster are more similar to each other than to those objects in different clusters according to defined criteria. Clustering methods have been applied in many applications. For example, in Single Image Super-Resolution [35] or in Cognitive Wireless Sensor Networks [19]. Researchers in the field of Recommender Systems have shown great interest in combining clustering with recommendations [30]. However, the majority of prior studies that have dealt with the use of clustering in recommender systems have used a collaborative filtering technique. Since collaborative filtering suffers from the data sparsity problem, a clustering approach based on employing users' social information to derive recommendations was presented in [29]. Another proposal [21] used clustering methods to detect similarities among the users and proposed techniques to identify users' profiles.

Many of the approaches to collaborative filtering studied so far incorporate partition-based clustering algorithms. One of these algorithms is k-means. In [14], the data was taken from registering forms and k-means was used to obtain clusters of users sharing similar interests in a tourism mobile recommender. The k-means clustering algorithm is well known for its efficiency in clustering large data sets and for being a simple and fast clustering method. The K-Medoids clustering algorithm has been used to segment the users based on their degree of preference in diversity [12]. K-Medoids is another partition-based clustering which, like k-means, uses the pair-wise distance of data points but instead of computing the mean centroids to minimize the within cluster sum of distances it uses the data points themselves as centroids.

To the best of our knowledge, this paper is the first attempt to introduce clustering in a critique-based recommender. Next, we review critique-based literature.

Critique-based recommenders have been broadly recognized as an effective preference-based search option that employs a feedback mechanism called *critiquing* [27, 31, 4]. The basic idea of critiquing can be traced back to the seminal FindMe system [1]. In particular, a critique is a directional preference over a feature of the current recommended product. For example, "cheaper" is a critique over the price feature in a Mobile Recommender and if the current recommended product has a price of \$300, it implies a critique [price $<$ \$300]. Note that in a complex[3] 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 and 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 [25, 36], whereas in compound critiques, each critique can be a combination of multiple unit critiques [34, 27, 31]. In the literature there are several studies of compound critique approaches, such as dynamic critiquing [32], MAUT-based compound critiques [46], preference-based organization [3], and Example Critiquing [44]. Given that most of critique-based approaches use unit critiques, we focused our work on the evaluation of unit critique proposals.

Concretely, the FindMe system [1] was the first standard unit critiquing recommender (STD) described in the literature. It only uses the current critique introduced by the user to guide the user in the product search space. Later, Incremental Critiquing (IC) [33] increased recommendation efficiency by means of gathering a list that contains the critiques made by the user during the session.

Other studies focused on improving the IC approach. Improvements to IC come from the different aspects that influence the quality measure used to recommend relevant products. One of these aspects is the compatibility score, which was defined originally in [33] as the percentage of critiques in the user model that a case satisfies. Highest Compatibility Selection (HCS) [40,

[4] Analyzed by means of the Average Session Length, which measures the number of cycles that a user must work through before being presented with their ideal target product.

39], and Exponential Hit-Loss (EHL) [38] focused on improving the compatibility score by using Reinforcement Learning reward functions to highlight relevant products, taking into account both the critique and the precise moment at which the critique was made. Another of the aspects that plays an important role in the quality measure is the similarity function, which is used to recommend the most similar product in each cycle. Instead of treating all the features equally in terms of their similarity, it has been demonstrated in [38] that weighting features according to the evolving preferences of the user during the session results in a shorter session length. The two weighting approximations proposed, called Global Users Preference Weighting (GW) and Local User Preference Weighting (LW), were analyzed in isolation and in combination with the Reinforcement Learning compatibility proposals. Among all of these proposals, in the evaluation conducted in [38], LW-EHL and GW-EHL proved to perform significantly better than IC.

More recently, there has been a growing interest in employing user-generated content as another source of data for recommending products [9,8]. Reviewer Assistant (RA) [9] harnesses experiential cases automatically mined from user-generated product reviews. Similarly, Sentimental product recommendation [8] used experimental cases and proposed a recommendation ranking strategy that combined similarity and sentiment to suggest products that are similar but superior to a query product according to the opinion of reviewers. None of these approaches is cognitive. That is, they do not take into account user preferences in the recommendation session. In contrast, other research has focused on using users' opinions to enrich the product descriptions (CB-per) and to define a weight (W-per) for each feature of the product in the recommendation process with the aim of improving their efficiency [5].

In the last few years, novel approaches have mainly focused on using collaboration among users or on using past critique sessions made by other users. 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 [6]. This recommender is called Collaborative Conversational Recommender (CCR). 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. Algorithms that use a history of critiques belong to the family of history-based recommenders. The first algorithm was Experience-Based Critiquing (EBC) [26]. Next, it was proposed Nearest Neighbor Compatibility critiquing (NNC) [24] and History-Aware Critiquing-based conversational recommendation (HAC) [41]. Later, an improvement of HAC, called the History-Guided Recommendation (HGR) [42] technique was presented, which uses the pair recommended product and critique for obtaining similar sessions, from the history, to the current recommendation session. Among all the proposed history-based recommenders, HGR outperforms the aforementioned recommenders, as denoted in [42].

As we have described above, many proposals have been defined in the literature. By their very nature, critique-based recommenders are cognitive systems since they learn and reason from their interactions with the user and from their experiences with the environment. In particular, in our previous works, we improved IC with weighting and compatibility measures based on Reinforcement Learning [38]. We believe that these measures may improve current proposals and we will evaluate their integration in our analysis. Additionally, we designed a Collaborative Conversational Recommender (CCR) [6] system. It is out of the scope of this work to analyze methods that enable online collaboration between users, since several feedback mechanisms can be used and we are only evaluating our proposals with regard to those that exclusively use critiquing. Nonetheless it would be interesting to do so in future work. Finally, we proposed critique-based recommenders that employed user-generated content as an additional source of data. In particular we proposed CB-per and W-per [5]. We have not included CB-per and W-per in our analysis here because they are only applicable to domains with technical and perceptive[5] features and this is unfeasible in many domains. A lesson we have learned from previous works is the importance of understanding the unique requirements of the user. In this paper, we propose to address this issue from a clustering point of view, which to date has not been used and has not yet been evaluated in the field of critique-based recommenders.

Clustering in the Critique-based Recommender

We decided to concentrate on the evaluation of unit critique proposals. Formally, 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 type of the critique c_i (i.e., typically $<$, $>$, $<>$), and v_i is the current value

[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).

of f_i. Critiques typically take the form of directional or replacement preferences [27]. 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".

It is important to note that all critique-based recommenders share a common basis, which is standard or incremental critiquing, and they differ in the source of data and how this data is mined to provide recommendations. The proposals that we are presenting are based on incremental critiquing. Later, we detail how to include one of them in HGR.

Next, we describe the clustering process and we present in depth the new critique-based recommenders that use clustering.

Clustering the case base

We use the k-prototype [16] method for partitioning the case base. We use the k-prototype method because it allows us to deal with very large and complex data sets and it integrates k-means [23] and k-modes [15] algorithms for clustering objects described by mixed numeric and categorical product features: concretely, the k-prototype method searches for a partition of the data set based on between-product dissimilarity measure. On the one hand, the measure is an Euclidean distance for numeric product features (i.e., the k-means method). While on the other hand, for categorical product features, the dissimilarity measure is the total mismatches of the corresponding feature for the evaluated products (i.e., the k-modes method).

In particular, we use k-prototype in the first cycle of the recommendation process to obtain the initial clusters of products with similar features. The input for k-prototype is the complete case base and a predefined k value, which was defined by applying the Elbow method and NbClust mechanism [2].

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. Each product is represented as a pair (F, cl) that contains the set of features and the cluster the product p_i belongs to. The set of features that describes each product is defined as $F = \{f_1, ..., f_m\}$. Note that each product p_i is assigned to only one cluster, which is the range of 1 to k clusters, and it is denoted as $p_{i\,cl}$ (see Figure 1(a)). Additionally, 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.

It is important to note that each product is assigned to one cluster at first recommendation cycle based on their similarity with other products. Next, we describe in depth our proposals to adapt the clustering during the session in the critique-based recommender.

Maximal Clustering Process

In this section we describe a new algorithm called Maximal Clustering Process (MCP), which is based on the incremental critiquing algorithm [33].

The incremental critiquing-based recommendation process consists of four phases: (1) a new product is **recommended** to the user from the set of products; (2) the user **reviews** the current recommendation and applies a directional feature critique; (3) the query product is **revised** for the next cycle; (4) the user model is updated (i.e. **modeling** phase) according to the feedback provided in the review phase. The recommendation process terminates either when the user retrieves a suitable product or when she explicitly finishes it.

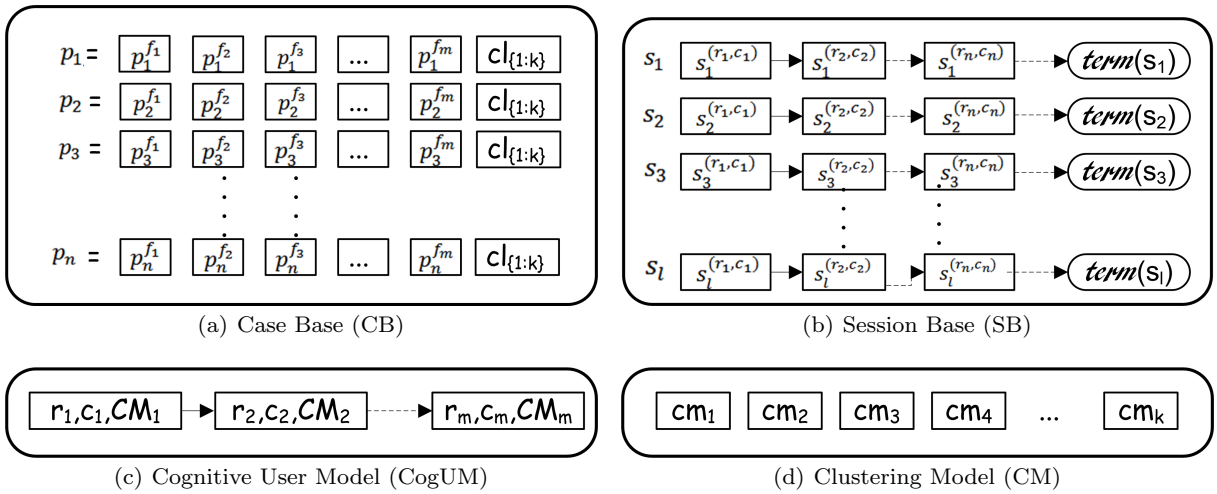
In particular, the goal of the MCP recommender is to introduce the clustering information into the recommendation process. To this end, we propose to modify the quality measure that is used for ranking the candidates products for a new recommendation in the **recommendation phase**. Our *quality* score combines the *compatibility* measure, the *similarity* measure, and a *clValue* as shown in Equation 1:

$$Q(p_i, p_q, UM) = \beta \cdot C_{p_i}(UM) + \alpha \cdot S(p_i, p_q) + (1 - \beta - \alpha) \cdot clValue(p_{i\,cl}) \quad (1)$$

where p_i is the ith candidate product, p_q is the current recommended product, and UM is the user model. The user model UM is defined as $UM = \{c_1, ..., c_m\}$, where c_i describes a particular critique in the recommendation cycle i. Initially, the user model is empty and it is filled with the critiques during the current session of the user to buy a product.

Additionally, in Equation 1, C_{p_i} is the compatibility score, S is the similarity function, and $clValue(p_{i\,cl})$ is the clustering measure. The parameters β, α are used to prioritize the compatibility, the similarity and the clustering measure.

In detail, the C_{p_i} is the compatibility score, which represents the percentage of critiques in the user model that the candidate product p_i satisfies. In particular, the compatibility score is defined as:



(a) Case Base (CB) (b) Session Base (SB)

(c) Cognitive User Model (CogUM) (d) Clustering Model (CM)

Fig. 1 Data structures in the proposed Critique-based Recommenders.

$$C_{p_i}(UM) = \frac{\sum_{j=1}^{|UM|} \delta(p_i, c_j)}{|UM|}, \tag{2}$$

where p_i is the candidate product given an individual user model (UM), $\delta(p_i, c_j)$ is the satisfaction function, and $|UM|$ is the number of elements (critique, recommendation and cluster model) in UM. In particular, the satisfaction measure $\delta(p_i, c_j)$ returns 1 if case p_i satisfies the critique c_j or 0 otherwise.

The similarity function S represents the similarity between the candidate product p_i and the current recommended product p_q and is defined as follows:

$$S(p_i, p_q) = \sum_{s=1}^{|F|} d(p_i^{f_s}, p_q^{f_s}) \tag{3}$$

where $p_i^{f_s}$ is a particular feature of the candidate product, p_i, $p_q^{f_s}$ is a particular feature of the current recommended product, p_q (the product query), 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. In particular, the distance measure, $d(p_i^{f_s}, p_q^{f_s})$ used depends on the type of the feature, as detailed in Equation 4. When the attribute is numeric we use Equation 5 and when it is nominal we apply Equation 6.

$$d(p_i^{f_s}, p_q^{f_s}) = \begin{cases} num(p_i^{f_s}, p_q^{f_s}) \text{ if } f_s \text{ is numeric} \\ nom(p_i^{f_s}, p_q^{f_s}) \text{ if } f_s \text{ is nominal} \end{cases} \tag{4}$$

$$num(p_i^{f_s}, p_q^{f_s}) = 1 - \frac{|p_i^{f_s} - p_q^{f_s}|}{max(p_i^{f_s}, p_q^{f_s})} \tag{5}$$

$$nom(p_i^{f_s}, p_q^{f_s}) = \begin{cases} 1 \text{ if } f_s \text{ is nom. and } p_i^{f_s} = p_q^{f_s} \\ 0 \text{ if } f_s \text{ is nom. and } p_i^{f_s} \neq p_q^{f_s} \end{cases} \tag{6}$$

Finally, the $clValue(p_{i\,cl})$ function computes the percentage of cases in the cluster cl that the product p_i belongs to, with respect to the total of products in the case base. It is important to remark that the $clValue(p_{i\,cl})$ clustering measure is different in each cycle during the recommendation session, as the case base is reduced according to the critiques performed by the user. For example, if the user makes a critique (price, $<$, \$300), all the products that do not satisfy this critique in the current recommendation cycle will be temporarily removed from the case base. Note that the number of cases present at each cluster varies from cycle to cycle in the recommendation session. Concretely, at each cycle, we compute the number of cases in the temporal case base that are present at each cluster. In the next cycle, we again use the case base, but we remove temporarily those cases that satisfy the new critique and therefore the cluster size may increase or decrease with respect to the previous cycle.

Cognitive User Model for recommendation

Here we describe a new algorithm called the Cognitive User preference Model (CUM) for recommendation. The CUM algorithm incorporates product clustering into the cognitive user preference model, called $CogUM$.

The cognitive user model depicted in Figure 1(c) is defined as $CogUM = \{u_1, ..., u_m\}$. Each $u_i = (r_i, c_i, CM_i)$ describes a particular recommendation cycle with

r_i representing the recommended product, and c_i representing the critique applied to r_i, while CM_i is a clustering model that represents the number of cases in each cluster when the user performs the critique.

Figure 1(d) also shows the clustering model CM_i. Each CM_i represents the ith recommendation cycle, which is defined as $CM_i = \{cm_1, ..., cm_k\}$, where cm_j represents the number of cases in a particular cluster j. Note that CM_i stores the information relating to all clusters in a particular cycle, with k being the number of clusters present in the case base. It is important to note that the clustering model is updated during each recommendation cycle. In the first cycle, the case base contains the complete data set and during the recommendation cycles the products in each cluster are maintained if they satisfy the latest critique.

Concretely, the goal of the CUM recommender is to use the cognitive user model to make inferences during the recommendation process. Our proposal is to evaluate the behavior of the clustering model during the recommendation cycles and how user feedback (i.e., user's critiques) influences the number of products present at each cluster in the clustering model CM.

Similarly to MCP, the CUM algorithm is based on an incremental critiquing algorithm and we also modify their recommendation phase. Next, we describe the CUM recommendation process, which is illustrated in Figure 2.

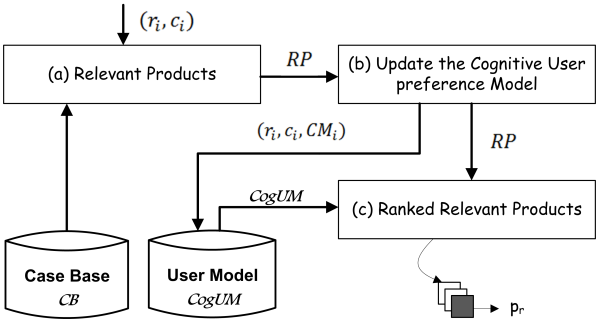


Fig. 2 CUM conceptual recommendation

First, the recommendation phase uses both the current recommended product r_i, and the latest user critique c_i, as input parameters. In particular, the recommender identifies a set of relevant products, RP, that are a subset of the case base, $RP \subseteq CB$. Note that relevant products are those that satisfy the latest critique, c_i (see letter a in Figure 2).

Second, the recommender updates the cognitive user preference model by including the current recommendation, r_i, the latest critique, c_i and the new clustering model, CM_i, which contains the number of products for each cluster only considering the set of relevant products, RP (see letter b in Figure 2). That is, the cognitive user model only contains the number of products in each cluster that satisfy the latest critique, c_i so the CogUM is fully compliant with the user preferences.

Third, we redefine the quality measure and rank the relevant products, RP, according to it (see letter c in Figure 2), as shown in Equation 7.

$$Q(p_i, p_q, CogUM) = \beta \cdot C_{p_i}(CogUM) + \alpha \cdot S(p_i, p_q) + (1 - \beta - \alpha) \cdot clValue(p_i, CogUM) \quad (7)$$

Concretely, we use the compatibility measure, $C_{p_i}(CogUM)$, where $CogUM$ is the input parameter in Equation 2, and the similarity measure $S(p_i, p_q)$ defined in Equation 3, where p_i is the candidate product (i.e., $p_i \in RP$), and p_q is the current recommendation, r_i. Additionally, we define a new $clValue(p_i, CogUM)$ that evaluates how the critiques made by the user produce an increase or decrease in the number of products in each cluster. Note that this information is stored in the cognitive user model. Equation 8 details the new clustering measure.

$$clValue(p_i, CogUM) = \begin{cases} \frac{CM_{j-1}^{cm_l} - CM_j^{cm_l}}{CM_{j-1}^{cm_l}}, & \text{if } CM_{j-1}^{cm_l} > CM_j^{cm_l} \\ 1 - \left(\frac{CM_j^{cm_l} - CM_{j-1}^{cm_l}}{CM_j^{cm_l}} \right), & \text{if } CM_j^{cm_l} > CM_{j-1}^{cm_l} \end{cases} \quad (8)$$

where p_i is the candidate product in the recommendation cycle j. From the $CogUM$ we use CM_j and CM_{j-1}, which correspond to the clustering model for the current cycle j and the previous one $j-1$. In Equation 8, $CM_j^{cm_l}$ is the number of products in the cluster cm_l that product p_i belongs to, while $CM_{j-1}^{cm_l}$ is the number of products for the same cluster in a previous cycle. When the current critique increases the number of products in the cluster that product p_i belongs to, the $clValue$ enhances the *quality* score for the product p_i (see Equation 7). Otherwise, it decreases the quality for this product p_i.

Finally, the best ranked product, p_r is recommended to the user (see letter c in Figure 2).

Combining CUM with previous enhancements

In this section we propose combining the CUM algorithm with previous state-of-the-art approaches. We believe that the CUM may improve recommendation processes by adding previous enhancements made to critique-based recommendations. The resulting new algorithm will be called CUM-I.

Concretely, we focus on the *ranked candidate* phase of the CUM recommender (see letter c in Figure 2). Specifically, we modify the quality measure Q described in Equation 7. To do so, we integrate into the recommendation process a weighting scheme for the similarity measure in Q, called Local User Preference Weighting (LW), and a new compatibility measure in Q, named Exponential Hit-Loss (EHL), based on a previous work in the literature [38]. It has been demonstrated that these two proposals significantly improve the recommendation efficiency of the incremental critiquing algorithm.

First, for the similarity measure, we propose integrating the LW approach whose aim is to prioritize the similarity of the features that have not yet been critiqued. The weight is defined over each feature f_s of candidate case p_i as follows:

$$\omega(p_i^{f_s}) = 1 - \frac{1}{2}\left(\frac{\sum_{j=1}^{|CogUM^{f_s}|}\delta(p_i, CogUM_j^{f_s})}{|CogUM^{f_s}|}\right) \quad (9)$$

where $|CogUM^{f_s}|$ is the number of critiques in the cognitive user model, $CogUM$, that refers to feature f_s and $CogUM_j^{f_s}$ is a critique j over feature f_s . This generates a feature weight vector for each case. 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. As such, the feature weight will be proportional to the number of times a critique over this feature is satisfied by the candidate case. However, as shown in eq. (9), weights never decrease to a zero value.

In particular, we compute the similarity as follows:

$$S(p_i, p_q) = \sum_{s=1}^{|F|} w(p_i^{f_s}) \cdot d(p_i^{f_s}, p_q^{f_s}) \quad (10)$$

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, $d(p_i^{f_s}, p_q^{f_s})$ used depends on the type of the feature, as detailed in Equation 4.

Second, with reference to the compatibility measure, it has been demonstrated in other studies [40,39,38] that the use of Reinforcement Learning (RL) in compatibility enhances retrieval in IC. Specifically, we propose using Exponential Hit-Loss (EHL), which considers that users increase their knowledge over cycles and, accordingly, it is more important to satisfy their latest preferences than their initial ones. Concretely, EHL compatibility is defined as:

$$C_t^{p_i} = \begin{cases} C_{t-1}^{p_i} \cdot (1+\alpha)^{(h_t+t)k}, \text{ if } R_t^{p_i} = 1 \\ C_{t-1}^{p_i} \cdot (1-\alpha)^{(\ell_t+t)k}, \text{ if } R_t^{p_i} = 0 \end{cases} \quad (11)$$

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), α is a constant step-size parameter that fixes the learning rate, and k is a regularization factor. The parameters are fixed at $\alpha = 0.3$ and $k = \frac{1}{2}$ in our experiments, as was described in [38]. 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. Additionally, $R_t^{p_i} = 1$ if case p_i satisfies the current critique and $R_t^{p_i} = 0$ otherwise.

Finally, the best ranked product using the new *quality* score is recommended to the user, p_r.

Combining CUM with history-based recommender

In this section we present a new algorithm called HGR-CUM-I, which combines CUM-I with a well-known history-based recommender named HGR [42]. HGR uses different sources of data to mine users' preferences in order to recommend products.

In particular, HGR-CUM-I stores a history of past sessions in the session base (see SB in Figure 1(b) and Figure 3), and maintains the current recommendation session of the user in the cognitive user preference model, shown as $CogUM$ in Figure 3. It is important to remark that the user model stores the current recommendation, r_i, the latest critique, c_i (i.e., the user feedback for each recommendation cycle), and the clustering model, CM_i.

The session base, SB (see Figure 1(b)), 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 $s_i^{(r_i,c_i)}$ and each session finishes in a terminal product, noted by $term(s_i)$.

Next, we describe the HGR-CUM-I recommendation process, which is illustrated in Figure 3.

First, the recommendation phase in the HGR-CUM-I algorithm uses both the current recommended product r_i, and the latest user critique c_i, as input parameters. Specifically, the HGR-CUM-I recommender identifies a set of relevant products, RP, that are a subset of the case base. Note that relevant products are those that satisfy the last critique, c_i (see letter a in Figure 3).

Second, the recommender updates the cognitive user preference model by including the current recommendation, r_i, the last critique, c_i and the new clustering

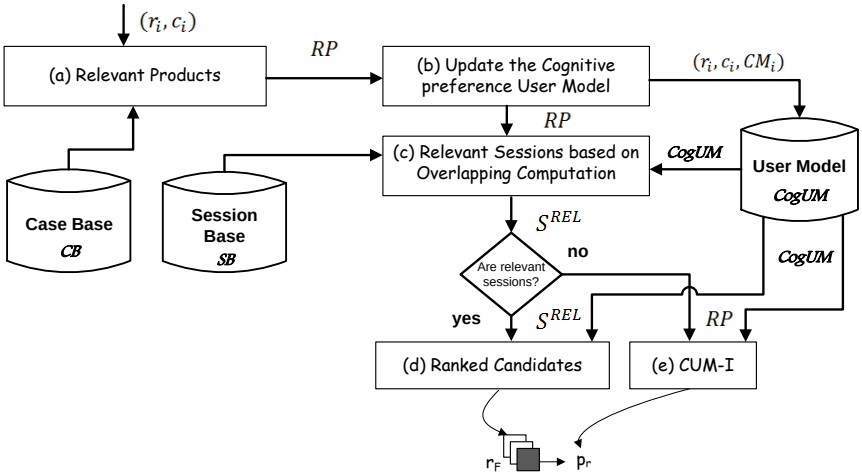


Fig. 3 HGR-CUM-I conceptual recommendation

model, CM_i, which contains the number of products in each cluster considering only the set of relevant products, RP (see letter b in Figure 3). That is, the cognitive user model only contains the number of products in each cluster that satisfy the last critique and so the CogUM is fully compliant with user preferences.

Third, the recommender identifies a set of relevant sessions (S^{REL}) based on an overlapping computation (see letter c in Figure 3), which are historical sessions in the SB that overlap with the user's current partial critique session, $CogUM$. To do so, we first compute the overlap score, $OS(CogUM, s_i)$, for each past critiquing session. The relevant sessions in the history are those that contain an overlap score greater than zero. In particular, we use the overlap score defined in [42]. We define the overlap score, $OS(CogUM, s_i)$, by means of Equations 12 and 14. In Equation 12, we compute the number of recommendation pairs, r_j, c_j (recommended product and critique) in the user model $CogUM$, that are also present in each past session s_i. The $matchPair$ measure is computed as shown in Equation 13.

$$OverlapPair(CogUM, s_i) = \sum_{\forall (r_j,c_j) \in CogUM} \sum_{\forall (r_i,c_i) \in s_i} matchPair((r_i, c_i), (r_j, c_j)) \quad (12)$$

$$matchPair((r_i, c_i), (r_j, c_j)) = \begin{cases} 1 \text{ if } (r_i = r_j) \text{ and } (c_i = c_j), \\ 0 \text{ otherwise} \end{cases} \quad (13)$$

When there are no overlapped pairs using Equation 12 we compute the overlap score, $OS(CogUM, s_i)$, using Equation 14, the number of critiques (i.e., without considering recommended products) in the user model $CogUM$, which are also present in each past critiquing session, s_i. This method has been used in previous studies with satisfactory results, such as [26]. The $matchCritique$ measure is computed as shown in Equation 15.

$$OverlapCritique(CogUM, s_i) = \sum_{\forall (c_j) \in CogUM} \sum_{\forall (c_i) \in s_i} matchCritique(c_i, c_j) \quad (14)$$

$$matchCritique(c_i, c_j) = \begin{cases} 1 \text{ if } c_i = c_j, \\ 0 \text{ otherwise} \end{cases} \quad (15)$$

Fourth, we analyzed the set of relevant sessions obtained in the initial step, depicted with the letter c in Figure 3. If there are relevant sessions, we rank candidates for the next recommendation (see letter d in Figure 3). Concretely, each relevant session terminates with an accepted recommendation, a $term(s_i)$ (i.e., a recommendation candidate product in HGR-CUM-I) that may be associated with more than one relevant session. We store candidates in a list and compute a score for each of them. We compute the candidate score based on the score of the relevant session in which it is an accepted recommendation (see Equation 16) and we select the best score candidate in the list for the next recommendation (r_F).

$$RecScore(term(s_i), S^{REL}) = \sum_{\forall (s_i) \in S^{REL} : term(s_i)} OS(CogUM, s_i) \quad (16)$$

where $term(s_i)$ is an accepted recommendation in s_i, S^{REL} are the relevant sessions, and $OS(CogUM, s_i)$ is the overlap score of s_i obtained from Equation 12 or 14. On the other hand, if there are no relevant sessions, we revert to the CUM-I recommender using the relevant products (RP), as shown in letter e in Figure 3.

Finally, the best ranked product, p_r, which is obtained using r_F or a *critiquing* algorithm, is recommended to the user.

Evaluation

In this section we report experiments with simulated users aimed at evaluating the efficiency of our proposals (MCP, CUM, CUM-I and HGR-CUM-I). We compare them to two related baseline algorithms, IC and HGR. Specifically, we concentrate on six different recommenders: (1) Incremental Critiquing (IC); (2) the History-Guided Recommender (HGR); (3) MCP, which uses clustering information in the recommendation process; (4) CUM, which uses the clusters to create a cognitive user preference model; (5) the CUM-I, which integrates in CUM previous enhancements on the recommendation process; and (6) HGR-CUM-I, which also integrates CUM-I in a HGR recommender.

Data sets and Methodology

In our experiments we used two data sets: SMARTPHONE[6] and RESTAURANT[7]. 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	History size
SMARTPHONE	1721	5	9	10000
RESTAURANT	9945	25	14	10000

Like previous history-based works, we adopted the methodology used in [26,41,42] to automatically generate past critiquing sessions based on the behavior of rational users. Specifically, we selected a random product as our target. From this target we automatically created a query by selecting from 3 to 5 features from the target at random, which acted as a starting point for each session. Each session began by the recommender retrieving the best-matching product for the query. From there the artificial user had to select a feature to critique. To elicit this artificial behavior we automatically selected one of the features of the recommended product and critiqued it in the direction of the target product. Each session terminated once the target product had been recommended. This process can be repeated for generating an arbitrary number of past critiquing sessions. Concretely, we used a session base whose size is 10000 (see the column *History size* in Table 1).

In our evaluation, we also used the leave-one-out methodology employed previously in [33,40,38], which takes a set of randomly selected products from the case base (*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 the basis for generating a set of (simulated) initial user queries, hereafter *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 types 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 [33,38].

For instance, an example of a *hard* query in a SMARTPHONE data set might be $[Manufacturer = Samsung]$, a *moderate* query might be $[screensize = 7, RAM = 8, SO = Android]$ and an *easy* query might be $[price = 200, Manufacturer = Samsung, screensize = 7, SO = Android, RAM = 8]$. Finally, we set up the β parameter as $\beta = 0.6$ and the α parameter as $\alpha = 0.25$ in the *Quality* measure (see Equations 1 and 7), based on the best results in our empirical experiments with several α and β values. In addition, we applied the Elbow method and the NbClust mechanism [2] to evaluate the best k for the clustering of each data set. In particular, based on our analysis of data sets, we defined a number

[6] This data set is available on demand.

[7] This data set has been used in [42] and it was kindly provided by the authors.

of clusters $k = 7$ and $k = 9$ for the SMARTPHONE and RESTAURANT data sets, respectively.

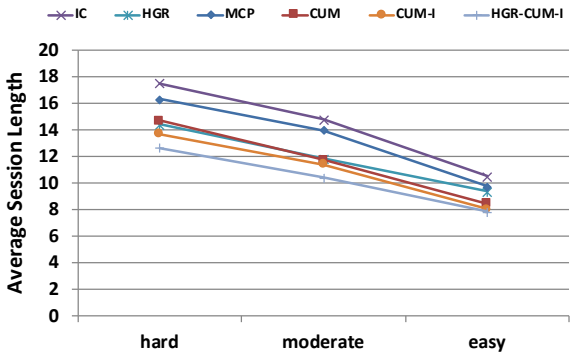
Experimental Results

In this section, we will begin by analyzing the recommendation efficiency of all algorithms and the percentage of benefit. Recommendation efficiency is measured through the number of recommendation cycles required to reach a desired product and is also known as *average session length* (from now on ASL). Note that the percentage of benefit is expressed as:

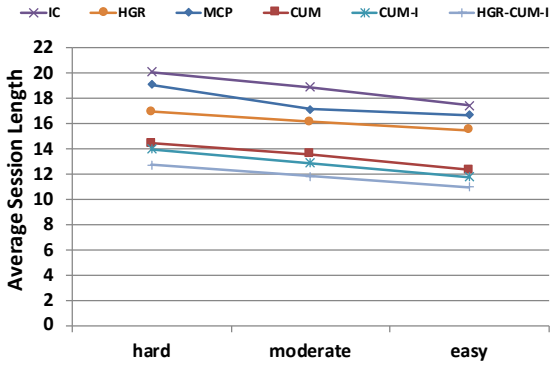
$$Benefit(x, y) = (1 - \frac{y}{x}) \cdot 100 \quad (17)$$

where y and x represent the ASL of the compared algorithm and the baseline, respectively. The baseline in this article corresponds to the IC algorithm. It is important to remark that the ASL measure has been widely used in evaluations of critique-based recommenders, such as [33,38,41,45].

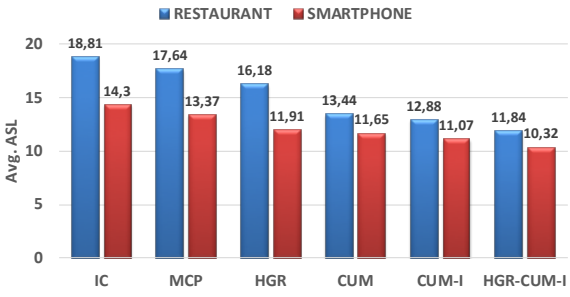
First, in Figure 4 we present three figures with the results of analyzing the ASL needed to reach a target for all algorithms in both the SMARTPHONE and the RESTAURANT domains. In Figures 4(a) and 4(b), each of the lines shows the average of ASL obtained in the three different initial query lengths (*hard*, *moderate*, and *easy*) for each particular algorithm described above. Note that in Figure 4(a) the lines are steep. For example, as shown in Figure 4(a), the IC algorithm achieves an ASL of 17.53, 14.82, and 10.54 for hard, moderate, and easy queries, respectively. In the same figure, we can observe that MCP obtains an ASL of 16.34, 14.02, and 9.76 for the different query lengths. In contrast in Figure 4(b) the lines are slightly steep, nearly flattened. This is mainly due to the size of the domain. The larger the domain (e.g., RESTAURANT), the more difficult it is to search for a product and larger sessions are needed for recommending a product, even for an easy query. For example, the IC algorithm achieves an ASL of 20.09, 18.91, and 17.43 for hard, moderate, and easy queries, respectively. Figure 4(c) shows the average ASL of all queries (hard, moderate, and easy). This figure shows that all the strategies proposed consistently reduce average session length when compared to the incremental critiquing approach, demonstrating their potential to improve recommendation efficiency. For example, in the RESTAURANT domain, the incremental recommender results in an average session length of 18.81 cycles while the MCP, CUM, CUM-I, and HGR-CUM-I recommenders results in an average of 17.64, 13.44, 12.88, 11.84 cycles, respectively.



(a) SMARTPHONE domain



(b) RESTAURANT domain



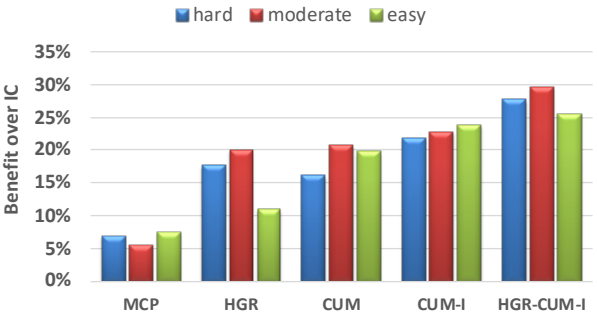
(c) Average ASL

Fig. 4 The ASL of the proposed algorithms compared to IC and HGR algorithms in a SMARTPHONE and RESTAURANT domains

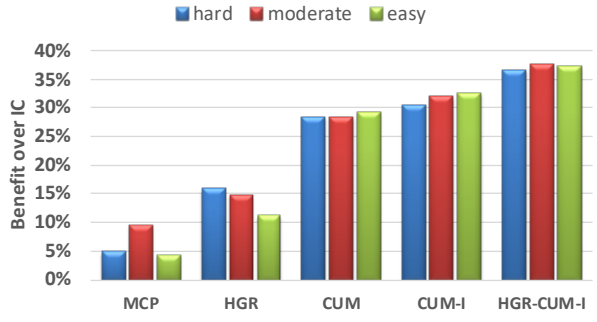
Secondly, we will address attention to the benefits of these proposals in comparison to the baseline IC algorithm. Figure 5 shows the benefit of each recommender (MCP, HGR, and CUM) separately and the combined recommenders (CUM-I and HGR-CUM-I) when compared to the incremental critiquing. We find that all recommenders separately result in a relative session length reduction of between 6.2% and 28.6% (see Figure 5(c)), with some variation in the relative benefit due to the MCP, HGR, and CUM approaches. The lowest benefit is for the MCP recommender, which ranges from 5.4% to 7.4% in the SMARTPHONE domain (see Figure 5(a))

and between 4.4% and 9.4% in the RESTAURANT domain (see Figure 5(b)). This may be because it performs the same process as the incremental critiquing approach with one minor modification. It consists of using a quality measure that incorporates the average for products within the clusters in order to determine the set of cases available for recommendation. In HGR the benefit ranges from 10.8% to a 19.9%, see Figures 5(a) and 5(b). Note that, on average (shown in Figure 5(c)), the HGR benefit is lower for large domains. This is largely due to the fact that HGR needs a large history to recommend appropriately. The larger the domain, the larger the history needed in HGR. When HGR does not match a previous historical session, it reverts to IC. On the other hand, the CUM recommender provides the highest benefit ranging on average from 16% to 29.2%, when applied alone (see Figures 5(a) and 5(b)). These results show that introducing clustering in the user preference model and using it in the recommendation process enables the recommender to discover and detect differences between products that are highly compatible with user critiques. Results are particularly excellent in large domains (see Figure 5(b)) where we find a benefit of between 28.3% and 29.2%.

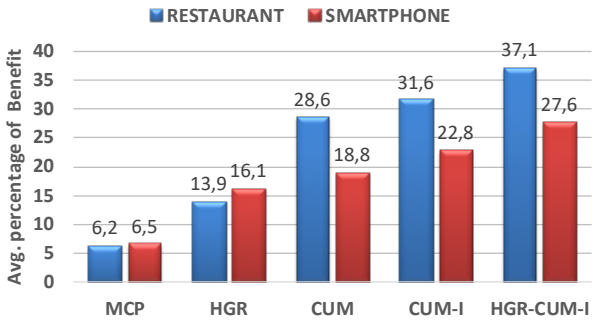
On the other hand, we are also interested in the integration of our proposals, based on using clustering, in existing critique-based recommenders. To this end, we selected CUM because it provides the highest benefit and we combined it with existing improvements to the IC algorithm and to the HGR recommender. These two combined approaches are named CUM-I and HGR-CUM-I. As shown in Figures 5(a) and 5(b), the combined recommenders result in a reduction in session length that ranges from nearly 22% in the SMARTPHONE domain to 37.2% in the RESTAURANT domain. Combining all of the recommendation techniques further enhances recommendation performance, resulting in the discovery of better recommendations for all queries (hard, moderate and easy). It seems that the recommenders ability to learn user preferences is greater when incorporating information from the clustering into the user preference model and into the previous approaches in critique-based recommendation. An important point to note is that nearly all results show a major benefit for easy queries. Easy queries naturally result in shorter sessions and thus there are fewer opportunities to find proper historical sessions in HGR, though with the introduction of our proposal, this drawback was largely alleviated. Analyzing the results in both domains, we observe that the benefit provided by the HGR-CUM-I is less dependent on the size of history-session base than HGR. Note that in Figure 5(c), the average benefit provided by HGR is lower for the



(a) SMARTPHONE domain



(b) RESTAURANT domain



(c) Average benefit over IC

Fig. 5 Benefits of our proposals in comparison to the IC algorithm

largest domain and greater benefit for the smallest one. This is because HGR needs a larger history session base to provide better recommendations. Figure 5(c) shows that the greatest benefit is obtained by HGR-CUM-I, which in our experiments used the same history session base used in HGR, with an averaged benefit of between 27.6% and 37.1%. We can therefore conclude that the introduction of clustering in a user preference model not only enables the recommender to locate more easily the products that are highly compatible with user preferences, but also reduces the need for a large history session base.

To sum up, these results highlight that the benefits in recommendation efficiency of all our proposals (i.e., the MCP and CUM algorithms and the ones that integrate with previous recommenders —the CUM-I and

HGR-CUM-I algorithms) are greater than the IC, in both the SMARTPHONE and RESTAURANT domains. Additionally, we have demonstrated that combining them with previous recommenders, such as HGR, increases their initial benefit. This suggests that our initial hypothesis in this paper is true. Recall that the hypothesis was that clustering may improve both the recommendation process and the definition of the user model in critique-based recommenders.

In order to demonstrate that the hypothesis concerning the introduction of clustering in critique-based recommenders significantly outperforming IC, we applied the Friedman [13] and Bonferroni-Dunn [10] tests to analyze whether the difference between the tested algorithms and the baseline is statistically significant in both data sets.

First of all, we computed the *mean rank* (r) of each algorithm considering all the experiments. In particular, the evaluation considers $k = 6$ algorithms (i.e., IC, HGR, MCP, CUM, CUM-I, and HGR.CUM-I) and $N = 6$ different experiments for each test. The experiments depend on two different case base sizes and three different queries (i.e., hard, moderate, and easy). We ranked alternative algorithms, for each experiment, following the practice of [13]. The one that attains the best performance is ranked 1, the second best ranked 2, and so on. Then, an algorithm's mean rank is obtained by averaging its rank across all experiments.

Secondly, we applied the Friedman and Bonferroni-Dunn tests to analyze whether the difference between algorithms is statistically significant. In particular, we applied the Friedman test in which F_F is distributed according to the F distribution with $(6-1) = 5$ and $(6-1)\cdot(6-1) = 25$ degrees of freedom. The critical value of $F(5, 25)$ is equal to 2.603 at the 0.05 critical value. For our efficiency comparison we obtained the values of X_F= 29.52 and $F_F = 17.60$ for the efficiency rankings. As the value of F_F is higher than 2.603 we rejected the null hypothesis. Once we had checked for the non-randomness of our results, we computed the Bonferroni-Dunn test to find out which algorithms were significantly different. In our case, when comparing six algorithms with a critical value $\alpha = 0.05$, $q_{0.05} = 2.576$ in a two-tailed Bonferroni-Dun test. We obtained a critical difference value of $CD = 1.26$.

The Bonferroni-Dunn test results are illustrated in Figure 6. In this graph, diamonds represent the mean ranks of each algorithm and the vertical lines across diamonds indicate the 'critical difference', CD. Basically, the efficiency of the two algorithms is significantly different if their vertical lines are not overlapping. For example, it can be seen that CUM, CUM-I and HGR-CUM-I performed significantly better than the baseline (IC). Additionally, we can see that the well-known HGR algorithm did not perform significantly better than the baseline IC. Our proposals improved the efficiency of the baseline IC, as their mean rank is shorter than IC but only CUM is significant with regard to IC. Among our proposals, CUM obtained the best mean rank. The graph also shows that the combination of CUM with previous existing critique-based recommenders achieved the shortest mean rank. In fact, the best algorithm (i.e., the one with the shortest mean rank) is HGR-CUM-I. Note that we can see that the results for HGR-CUM-I are also significantly better than HGR, with a confidence of 95%.

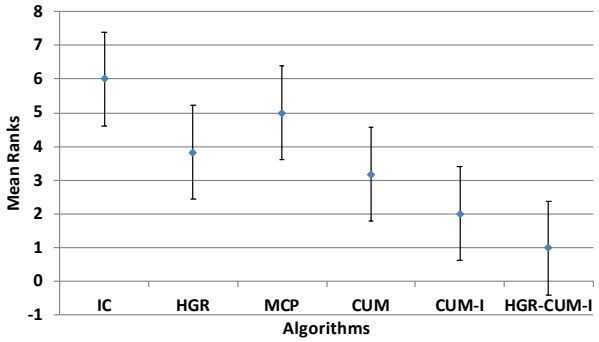


Fig. 6 Application of the Bonferroni-Dunn test to alternative algorithms' mean rank of ASL

In summary, from our significance analysis, we conclude that CUM, CUM-I and HGR-CUM-I algorithms performed significantly better than IC, and that HGR-CUM-I enhanced significantly the recommendation efficiency of well-known state-of-the-art algorithms (IC and HGR), which confirms our initial hypothesis.

Conclusions and Future Work

In this paper we hypothesize that clustering may improve both the recommendation process and the definition of the user model in critique-based recommenders. To this end, we developed several proposals based on adaptive clustering. First, we have presented a novel critique-based recommender, called MCP, which adapts the recommendation process to recommend on the basis of the number of products in each cluster. Second, we have proposed a new cognitive user preference model that integrates the information from the clusters and a new critique-based recommender, called CUM, which modifies the recommendation process to recommend on the basis of the new cognitive user preference model. Next, we have integrated a weighting measure, called LW, and a reinforcement learning compatibility measure, called EHL into the new proposal (CUM). The resulting new algorithm has been named CUM-I. Ad-

ditionally, we have also integrated CUM-I into HGR. This new algorithm is called HGR-CUM-I.

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 clustering (i.e,. MCP and CUM) improves the recommendation efficiency with respect to the traditional IC algorithm, but they also confirm with an statistical analysis that integrating CUM-I significantly improves the efficiency of IC. HGR-CUM-I performs significantly better in terms of efficiency than HGR and the traditional IC algorithms.

MCP and CUM focused on unit critique proposals. Nonetheless, the two new critique-based recommenders proposed, which perform an adaptive clustering, can be easily applied to any of the recommenders that use compound critiquing. Taking into account that compound critiques are a combination of multiple unit critiques, instead of storing a unit critique, the cognitive user model will store a list of unit critiques. Accordingly, cases in the clustering model are maintained if they satisfy one (for unit critique) or more critiques (for compound critiquing). In this way, the remaining new clustering proposals are the same as those defined for the unit critiquing algorithms.

Although it has been out of the scope of this paper to analyze clustering in compound critiquing algorithms, it would be interesting to do so in future work. Additionally, we will address the analysis of using clustering in a Collaborative and Conversational Recommender that integrates several feedback mechanisms.

Compliance with Ethical Standards

– Funding: This study was supported by Spanish Ministry of Science and Innovation (grant number TIN2015-71147-C2-2) and by Agència de Gestió d'Ajuts Universitaris i de Recerca, Generalitat de Catalunya, AGAUR (grant number SGR-2017-341).
– Ethical approval: This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Burke, R., Hammond, K., Yound, B.: The FindMe approach to assisted browsing. IEEE Expert (1997)
2. Charrad, M., Ghazzali, N., Boiteau, V., Niknafs, A., Charrad, M.M.: Package nbclust. Journal of Statistical Software **61**, 1–36 (2014)
3. Chen, L., Pu, P.: Preference-based organization interfaces: aiding user critiques in recommender systems. In: User Modeling 2007, pp. 77–86. Springer (2007)
4. Chen, L., Pu, P.: Critiquing-based recommenders: survey and emerging trends. User Modeling and User-Adapted Interaction **22**(1-2), 125–150 (2012)
5. 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)
6. 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)
7. Ding, S., Zhang, J., Jia, H., Qian, J.: An adaptive density data stream clustering algorithm. Cognitive Computation **8**(1), 30–38 (2016). DOI 10.1007/s12559-015-9342-z
8. 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)
9. 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)
10. Dunn, O.J.: Multiple comparisons among means. Journal of the American Statistical Association **56**(293), 52–64 (1961)
11. 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)
12. Eskandanian, F., Mobasher, B., Burke, R.: A clustering approach for personalizing diversity in collaborative recommender systems. In: Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization, pp. 280–284. ACM (2017)
13. 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)
14. Gavalas, D., Kenteris, M.: A web-based pervasive recommendation system for mobile tourist guides. Personal Ubiquitous Comput. **15**(7), 759–770 (2011)
15. Huang, Z.: A fast clustering algorithm to cluster very large categorical data sets in data mining. DMKD **3**(8), 34–39 (1997)
16. Huang, Z.: Extensions to the k-means algorithm for clustering large data sets with categorical values. Data Min. Knowl. Discov. **2**(3), 283–304 (1998)
17. Jia, H., Ding, S., Du, M.: Self-tuning p-spectral clustering based on shared nearest neighbors. Cognitive Computation **7**(5), 622–632 (2015). DOI 10.1007/s12559-015-9331-2
18. Kelly III, J.: Computing, cognition and the future of knowing, how humans and machines are forging a new age of understanding. IBM Research: Cognitive Computing. IBM Corporation (2015)
19. Kim, S., McLoone, S.F., Byeon, J., Lee, S., Liu, H.: Cognitively inspired artificial bee colony clustering for cognitive wireless sensor networks. Cognitive Computation **9**(2), 207–224 (2017). DOI 10.1007/s12559-016-9447-z
20. Koren, Y., Bell, R.: Advances in collaborative filtering. In: Recommender Systems Handbook, pp. 145–186. Springer (2011)
21. Kuzelewska, U.: Clustering algorithms in hybrid recommender system on movielens data. Studies in Logic, Grammar and Rhetoric **37**(1), 125–139 (2014)

22. Lee, W., Lin, C.: Combining expression data and knowledge ontology for gene clustering and network reconstruction. Cognitive Computation **8**(2), 217–227 (2016). DOI 10.1007/s12559-015-9349-5
23. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, 14, pp. 281–297. Oakland, CA, USA (1967)
24. Mandl, M., Felfernig, A.: Improving the performance of unit critiquing. In: User Modeling, Adaptation, and Personalization, vol. 7379, pp. 176–187. Springer (2012)
25. 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)
26. 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)
27. McGinty, L., Reilly, J.: On the evolution of critiquing recommenders. In: Recommender Systems Handbook, pp. 419–453. Springer (2011)
28. Pazzani, M., Billsus, D.: The Adaptive Web: Methods and Strategies of Web Personalization, chap. Content-Based Recommendation Systems, pp. 325–341. Springer (2007)
29. Pham, M.C., Cao, Y., Klamma, R., Jarke, M.: A clustering approach for collaborative filtering recommendation using social network analysis. Journal of Universal Computer Science (j-jucs) **17**(4), 583–604 (2011)
30. Pitsilis, G., Zhang, X., Wang, W.: Clustering recommenders in collaborative filtering using explicit trust information. In: I. Wakeman, E. Gudes, C.D. Jensen, J. Crampton (eds.) Trust Management V, pp. 82–97. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
31. 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)
32. 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)
33. Reilly, J., McCarthy, K., McGinty, L., Smyth, B.: Incremental critiquing. Knowledge-Based Systems **18**(4-5), 143–151 (2005)
34. 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)
35. Ren, P., Sun, W., Luo, C., Hussain, A.: Clustering-oriented multiple convolutional neural networks for single image super-resolution. Cognitive Computation **10**(1), 165–178 (2018). DOI 10.1007/s12559-017-9512-2
36. Ricci, F., Nguyen, Q.: Acquiring and Revising Preferences in a Critique-Based Mobile Recommender System. IEEE Intelligent Systems **22**(3), 22–29 (2007)
37. Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.): Recommender Systems Handbook. Springer (2011)
38. Salamó, M., Escalera, S.: Increasing retrieval quality in conversational recommenders. IEEE Transactions on Knowledge and Data Engineering **24**(10), 1–14 (2012)
39. Salamó, M., Reilly, J., McGinty, L., Smyth, B.: Improving Incremental Critiquing. In: Proc. of the 16th Artificial Intelligence and Cognitive Science, pp. 379–388 (2005)
40. Salamó, M., Reilly, J., McGinty, L., Smyth, B.: Knowledge discovery from user preferences in conversational recommendation. In: Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases, pp. 228–239. Springer-Verlag (2005)
41. 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)
42. Salem, Y., Hong, J., Liu, W.: History-guided conversational recommendation. In: Proceedings of the 23rd International Conference on WWW Companion, pp. 999–1004 (2014)
43. Thanh, N.D., Ali, M., Son, L.H.: A novel clustering algorithm in a neutrosophic recommender system for medical diagnosis. Cognitive Computation **9**(4), 526–544 (2017). DOI 10.1007/s12559-017-9462-8
44. Viappiani, P., Faltings, B., Pu, P.: Preference-based search using example-critiquing with suggestions. Journal Artificial Intelligence Research **27**, 465–503 (2006)
45. 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)
46. 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)
47. Zhang, Z., Zhao, X., Wang, G.: FE-ELM: A new friend recommendation model with extreme learning machine. Cognitive Computation **9**(5), 659–670 (2017). DOI 10.1007/s12559-017-9484-2