User Preferences Elicitation and Exploitation in a Push-Delivery Mobile Recommender System

Quang Nhat Nguyen, Thuan Minh Hoang, Lan Quynh Thi Ta, Cuong Van Ta, and Phai Minh Hoang

Hanoi University of Science and Technology, Hanoi, Vietnam
`quangnn-fit@mail.hut.edu.vn`

Abstract. Most existing recommender systems follow the pull-delivery approach, where the user must explicitly make request before receiving some product or service recommendations. However, in application domains where the availability of items changes quickly and often (e.g., recommendation of relevant promotions, events, etc.), the pull-delivery recommendation approach seems not effective in helping users keep track of their desired and interested items. In this paper, we present our proposed push-delivery mobile recommendation methodology that is capable of proactively delivering personalized recommendations to mobile users at appropriate context. The proposed recommendation methodology has been implemented in *Prom4U* - a push mobile recommender system that helps users timely receive their interested promotions of commercial products from supermarkets and stores. We present here the experimental results of a live-user evaluation of *Prom4U* that show the appropriateness of the proposed recommendation approach and the effectiveness of the system *Prom4U*.

Keywords: mobile recommender system, push delivery, user preferences elicitation, critique-based recommendation, live-user evaluation.

1 Introduction

Recommender systems (RSs) aims at solving the information overload problem by providing product and service recommendations personalized to a given user's needs and preferences [1], [2]. Most existing RSs follow the pull-delivery approach, where the user must explicitly make request for some product or service recommendations. However, in some application domains (e.g., the problem of providing interested product promotions to a given user), the availability of items changes quickly and often. In such application domains, the pull-delivery approach seems less effective in helping users keep track of their interested items, i.e., at the time of a user's request some of his interested items are not available, but when they are available (often in short durations) the user does not know.

In this paper, we present our proposed mobile push-delivery recommendation methodology that is capable of proactively (i.e., automatically) providing relevant recommendations to users at right contexts. To provide push-delivery recommendations to users, the system must decide: *what recommendations should be* *pushed to a given user*, and *when the system should push these recommendations to the user*. To tackle the first problem, our proposed recommendation methodology integrates both long-term and session-specific user preferences and exploits a critique-based conversational approach [3]. The long-term user preferences are inferred from past recommendation sessions, whereas the session-specific user preferences are derived from the user's critiques to the provided recommendations in the current session. To deal with the second problem, the system models a push context as a case, and uses the Case-Based Reasoning (CBR) problem-solving strategy [4], i.e., a machine learning approach, to exploit (i.e., reuse) the knowledge contained in the past push cases to determine the right push context for the current case.

Our proposed recommendation methodology has been implemented in *Prom4U* - a mobile push recommender system that helps users timely receive their interested product promotions. We conducted an evaluation of the system *Prom4U* with real test users. This live-user evaluation aims at testing the appropriateness of the proposed recommendation approach and the effectiveness of the implemented system. We present in this paper the experimental results of this live-user evaluation.

The remainder of the paper is organized as follows. In Section 2, we discuss some related work on recommender systems and push-delivery information systems. In Section 3, we introduce the formal representations of product promotions, the user profile and the user query. In Section 4, we present our proposed mobile push-delivery recommendation methodology. In Section 5, we report the experimental results of the live-user evaluation. Finally, the conclusion and future work are given in Section 6.

2 Related Work

Recommender Systems (RSs) are intelligent decision support tools that help users find and select their desired products and services when there are too many options to consider or when users lack the domain-specific knowledge to make selection decisions by themselves. Traditional recommendation approaches include: collaborative, content-based, and knowledge-based [1], [2]. RSs have been very effective and popular tools in well-known commercial websites, such as Amazon.com, Barnes&Noble.com, eBay.com, Yahoo! news, iTunes Genius, TripAdvisor.com, MyProductAdvisor.com, etc.

A push-delivery information system is a system that automatically delivers (i.e., pushes) the information to users without their request. The push-delivery model appears to be effective in application domains where the availability of items changes often and quickly, because it helps users timely receive their interested information. However, if the system pushes uninterested information to a user, or even pushes interested information to the user but at inappropriate contexts, there is a high risk that this push-based delivery will annoy the user (i.e., considered as a spam). Hence, for push-delivery RSs, to provide personalized recommendations and reduce the spamming issue, the system must push *only relevant and targeted* information to the user *at right time*. In some previous approaches, the system just pushes all objects (or items) that locate near the user's position, without regarding his preferences [5], [6]. In other previous approaches, the system, though takes into account the user's preferences, but does not estimate right contexts to push, i.e., the system always pushes advertisements to the user when he is close to (or inside) the store [7], [8]. Ciaramella et al. [9] presented a mobile services RS that uses a rules table to determine a user's situation, but the system pushes all services associated with the determined situation to the user without regarding his preferences. The information service system presented in [10] determines the push time based on a decision table that is the same for all users.

In our proposed approach, the pushed recommendations are personalized for each user (i.e., suitable for his preferences), and the push context is determined based on the system's learning from past push cases. Hence, the system's push-context determination is personalized for each user. Moreover, all the push-delivery information systems mentioned above follow the single-shot strategy, where the system computes and pushes to the user the information, and the session ends. In our proposed approach, a push session, after the user accepts to view the pushed recommendations, evolves in a dialogue where the system's recommendations interleave with the user's critiques to these recommendations [3]. Such critiques enable the system to better understand the user's preferences, and hence to provide more suitable recommendations to the user.

3 Formal Representations

3.1 Promotion Representation

In our recommendation problem, a promotion, *represented hierarchically*, consists of the three main components: the promotion's information, the promotion's promoted product(s) and the promotion's gift(s). In this hierarchical representation, each component is represented by its own sub-components and features. Because of the limited paper space, we elaborate here only the first and second levels of the hierarchical representation of a promotion.

$$X = (PROM\_INFO, PROM\_PRODUCTS, GIFTS)$$

The component $PROM\_INFO$ stores the information of the promotion:

$$PROM\_INFO = (Prom\_Types, DURATION, PROVIDER);$$

where the feature $Prom\_Types$ represents the types of the promotion, the sub-components $DURATION$ and $PROVIDER$ represent the promotion's available duration and provider, respectively.

The component $PROM\_PRODUCTS$ represents the set of the promoted products, where each promoted product is represented by the product identifier, its category and price.

$$PROM\_PRODUCTS = \{(Product\_Id, Category, Price)\}$$

The component $GIFTS$ represents the set of the gifts of the promotion, where each gift is represented by the gift identifier and its type.

$$GIFTS = \{(Gift\_Id, Gift\_Type)\}$$

3.2 User Profile Representation

The user profile stores *the user's long-term preferences* that are exploited by the system to build the initial representation of the user query. The user profile, *hierarchically represented*, consists of the three components that represent the user's long-term preferences on promotions, promoted products and gifts.

$$U = (PROM\_PREF, PRODUCT\_PREF, GIFT\_PREF);$$

where the component $PROM\_PREF$ stores the user's long-term preferences on promotions types and providers; the component $PRODUCT\_PREF$ stores his long-term preferences on category and price of promoted products; and the component $GIFT\_PREF$ stores his long-term preferences on gift types.

3.3 User Query Representation

The user query (Q) representation encodes the system's understanding of *the user's session-specific preferences.* In a recommendation session, at every recommendation cycle the system uses this query Q to compute the promotions recommendation list for the user. The user query Q consists of the two structured components: the favorite pattern (FP) and the component and feature importance weights (W).

$$Q = (FP, W)$$

The favorite pattern FP, hierarchically represented, consists of the three components that represent the user's session-specific preferences on promotions, promoted products and gifts. The structure of FP is similar to the structure of the user profile (U) representation, except that FP includes additionally the sub-component $DURATION$ (i.e., to represent the user's session-specific preference on promotion available duration) and the feature $Distance$ (i.e., to represent the user's session-specific preference on distance to promotion provider).

The weights vector W is *represented hierarchically* corresponding to the representation of FP. For each representation level, the weight of a sub-component (or a feature) models how much important the sub-component (or the feature) is for the user with respect to the others.

4 Proposed Recommendation Methodology

In our approach, a recommendation session starts when the system shows a notification screen (i.e., of new interesting promotions) on the user's mobile device, and ends when he quits the session. The overview of the recommendation process is shown in Figure 1.

When the session starts, the system builds the initial query representation (Q^0) exploiting the user's contextual information and long-term preferences stored in the user profile. In particular, the values of the features of FP^0 are set

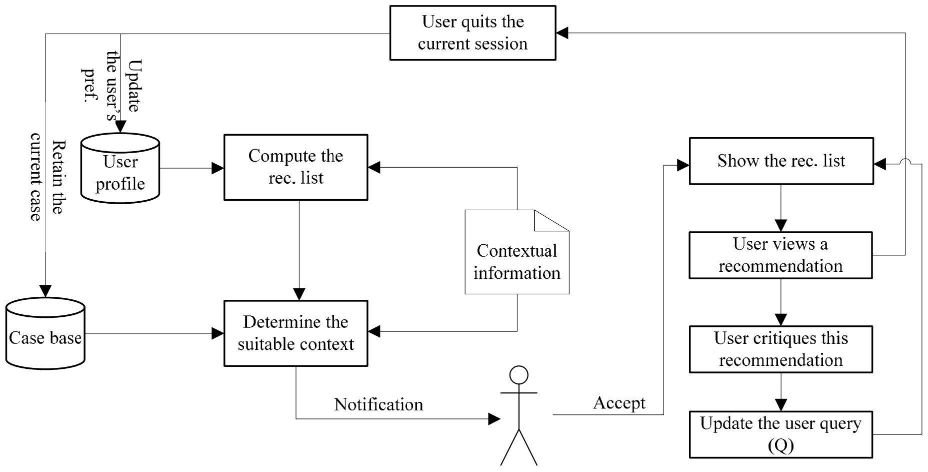

User quits the current session
Update the user's pref.
Retain the current case
User profile
Compute the rec. list
Contextual information
Case base
Determine the suitable context
Notification
Accept
Show the rec. list
User views a recommendation
User critiques this recommendation
Update the user query (Q)

Fig. 1. The overview of the recommendation process

by the values of the corresponding features in the user profile (U). In addition, the values of $DURATION$ and $Distance$ are set to unknown to indicate that the system, at the beginning of the session, does not know about the user's session-specific preferences on promotion available period and distance to provider.

The importance weights vector W is initialized by exploiting the history of user critiques. The intuitive idea is that a feature (or sub-component)'s initial importance weight is proportional to the frequency of the user critiques expressed on that feature (or sub-component) [11]. We note that in our current approach, *at a recommendation cycle the user can make critique to a number of (i.e., more than one) features* of the item; whereas in our previous work [11] the system allows the user to make critique to only one feature per cycle.

The system uses this initial query Q^0 to compute the initial recommendation list for the user, by ranking the available promotions to their similarity to (FP^0, W^0). The ranking is done, using a similarity function computed over the hierarchical representation described in Section 3, so that the more similar to (FP^0, W^0) a promotion is the higher it appears in the ranked list. In case of ties, the promotion provided by the provider closer to the user's position is ranked higher. Only k (i.e., a predefined cut-off parameter) best promotions in the ranked list, i.e., those most similar to (FP^0, W^0), are included in the recommendation list.

After computing the recommendation list, the system must determine when it should push this list to the user. In our proposed approach, this push-context determination is done based on the Case-Based Reasoning (CBR) problem-solving strategy [4]. The CBR is used to exploit (i.e., reuse) the knowledge contained in the past push cases. In our recommendation methodology, each push case is modeled by two parts: *problem description* and *solution*. In particular, the problem description of a case contains information of: 1) the time-slot of the push, 2) the list of providers that provide promotions contained in the recommendation list,

3) the user's distances to those providers, and 4) the user's long-term preferences to those providers. The solution of a case indicates the decision of the user, i.e., either the user accepts to receive (i.e., view) the recommendation list or the user rejects to receive.

To estimate (i.e., predict) an appropriate push context, the system identifies the set of m past push cases most similar to the current one in that the users accepted to receive the recommendation list (denoted as $C^{Accepted}$) and the set of m past push cases most similar to the current one in that the users rejected to receive the recommendation list (denoted as $C^{Rejected}$). Then, the system computes the *acceptance degree* (i.e., the confidence level to push) and the *rejection degree* (i.e., the confidence level to not push) for the current case.

$$acceptance\_degree(C^*) = \frac{1}{m} \sum_{C \in C^{Accepted}} sim(C^*, C); \tag{1}$$

$$rejection\_degree(C^*) = \frac{1}{m} \sum_{C \in C^{Rejected}} sim(C^*, C); \tag{2}$$

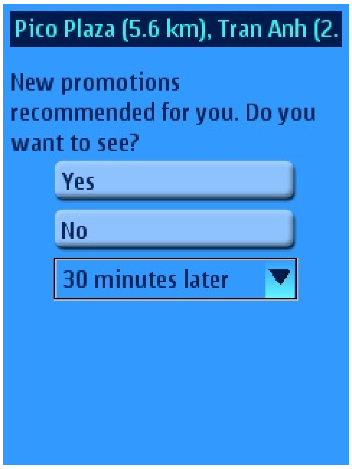
where C^* is the current case, C is a past one, and $sim(C^*, C)$ is the similarity between C^* and C.

If $(acceptance\_degree(C^*) - rejection\_degree(C^*)) \geq \theta$ (i.e., θ is a predefined push-confidence threshold value), then the system sends a push notification to the user. Otherwise, the system does not, and the recommendation list is saved in the pending list for him (i.e., at the next time-slot, the system re-estimates whether or not to send the push notification to him).

We note that in our proposed approach, at the predicted push time the system does not immediately show the recommendation list. Instead, the system just shows a notification on the screen of the user's mobile device and lets him make the final decision (i.e., to accept or reject the push). In this way, we get the two advantages. First, in case the system's predicted time is not (really) appropriate for the user, showing just a notification screen makes him less annoying compared to showing immediately the recommendation list. Second, showing the notification screen allows the system to collect the user's decision (i.e., to accept or reject the push) that is recorded as the solution part of the current push case.

Given the push notification sent to the user's mobile device, he can decide to accept the push, or postpone it, or reject it (see Figure 2-a). If he accepts the push, then the system stores the current push case in its case base (for the future uses) and shows the recommendation list to him (see Figure 2-b). If he rejects to receive the push, then the system stores the current push case in its case base, and the session ends. If he postpones the push, then he specifies a later appropriate time slot (see Figure 2-a). At that indicated (postponed) time the system re-sends the push notification to him. Until the end of the current day, if he has not accepted the postponed push, then the system records the postponed push case as a rejection one in its case base.

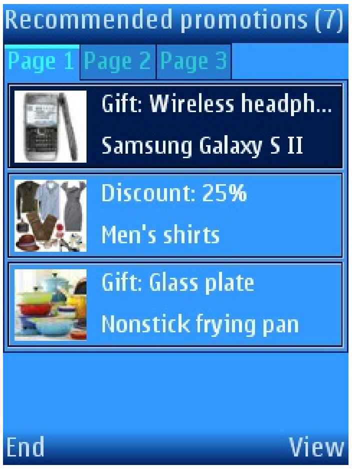
When the recommendation list is shown to the user (see Figure 2-b), he can select a recommended promotion to see its details (see Figure 2-c). After the user



Pico Plaza (5.6 km), Tran Anh (2.

New promotions recommended for you. Do you want to see?

(a) Push notification.



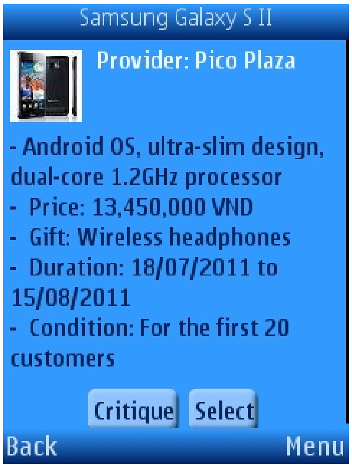
Recommended promotions (7)

Gift: Wireless headph...
Samsung Galaxy S II

Discount: 25%
Men's shirts

Gift: Glass plate
Nonstick frying pan

(b) Recommendation list.

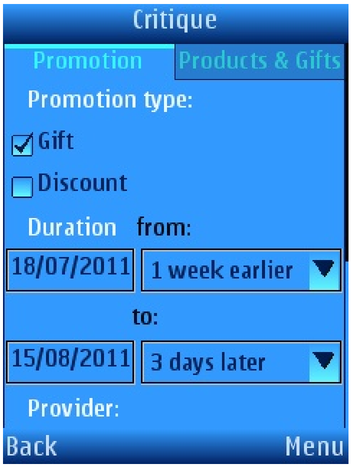


Samsung Galaxy S II

Provider: Pico Plaza

- Android OS, ultra-slim design, dual-core 1.2GHz processor
- Price: 13,450,000 VND
- Gift: Wireless headphones
- Duration: 18/07/2011 to 15/08/2011
- Condition: For the first 20 customers

(c) Promotion details.



Critique

Promotion type:

☑ Gift
☐ Discount

Duration from: 18/07/2011 1 week earlier

to: 15/08/2011 3 days later

Provider:

(d) Critique on promotion's type and duration.

Fig. 2. The user interface of the system *Prom4U*

views a promotion's details, if he accepts the promotion, then this promotion is added to his Selection List, and he can view another recommended one or quit the session. If he is somewhat interested in the promotion, but some of its features do not completely satisfy him, then he critiques the promotion to specify his preferences on these unsatisfactory features (see Figure 2-d). Such critiques help the system adapt its current understanding of the user's preferences (i.e., encoded in Q), and re-compute a new list of recommended promotions shown to the user, and the system proceeds to the next recommendation cycle.

We note that this critiquing mechanism is different from that used in our previous work [3] in two aspects. First, in our previous approach [3], at a recommendation cycle the user can make a critique to only one feature. However, in our current approach, at a recommendation cycle the user is allowed to make a critique to a number of features (i.e., multi features are criticized per cycle). This helps to increase the convenience in critiquing, and to decrease the length (i.e., the number of cycles) of the recommendation session. Second, in our previous approach [3], after the user makes a critique, the user-query representation (Q) is updated exploiting solely the critique (i.e., not exploiting the values of the uncriticized features of the criticized item). However, in our current approach, after the user makes a critique, the user-query representation (Q) is updated exploiting both the critique and the values of the uncriticized features of the criticized item. Our current approach is motivated by the fact that a user makes a critique to an item when: 1)he likes that item, but 2)wants to modify some unsatisfactory features of that item. Hence, in our current approach the system infers implicitly that the user likes those uncriticized features. Also, the system's graphical user interface (GUI) is designed to help the user easily and quickly see all the features of the promotion before making critique (see Figure 2-c).

When the user quits the session, the system exploits the information of his expressed critiques and selected promotions in the current session to update the user profile (U). This user profile update allows the system to refine its understanding of the user's long-term preferences, and hence better serve him in the future.

5 Live-User Evaluation

The proposed recommendation methodology has been implemented in *Prom4U* - a mobile RS that aims at automatically (i.e., proactively) providing relevant promotion recommendations to mobile users at appropriate contexts. In the current version of *Prom4U*, the system provides promotions of five categories (i.e., clothes and shoes, household goods, foods, mobile phones, computers and accessories), and the promotions are provided by five biggest super markets (i.e., promotion providers) in Hanoi, Vietnam.

We conducted a live-user evaluation of *Prom4U* to evaluate the appropriateness of the proposed recommendation approach and the usability of the system. This live-user evaluation lasted in more than two months and involved seven test users[1], i.e., three men and four women, in the ages of from 20 to 30 years old. All of the test users were interested in receiving suitable product promotions from super markets. In this live-user evaluation, we used the Vietnamese user-interface version of *Prom4U*, since all the test users are Vietnamese and for some of them their English are not good. (Note that Figure 2 shows the English user-interface version of *Prom4U*.)

[1] In fact, there were twenty two persons registering to join in this evaluation. But unfortunately, we had to exclude fifteen of them, because their mobile phones do not support running J2ME application and/or do not have a GPS built-in receiver.

After the client side of *Prom4U* (i.e., a J2ME midlet application) was installed on the mobile phone of a test user, he (or she) was introduced by the use guide of *Prom4U* and the test scenario, and then started using the system (in about two months). The test scenario, which every test user followed, consists of four main steps.

- Step 1. Given the push notification shown on the mobile phone's screen, the test user can either accept, postpone or reject to see the recommended promotions list (see Figure 2-a).
- Step 2. If the user accepts to see the recommended promotions list, it is shown on his/her mobile phone's screen. He/she can view the details of any of the recommended promotions.
- Step 3. After viewing a recommended promotion, the user can either selects it (i.e., if it satisfies him/her) or make a critique.
- Step 4. If the user likes a recommended promotion, but is not satisfied with it, then he/she uses the critiquing function to indicate his/her preferences to the promotion's unsatisfactory features. After the user makes a critique (to a number of features), the system exploits the critique to compute a new recommendation list and shows it to him/her.

The live-user evaluation collected both objective measures and subjective comments. The objective measures include the following metrics.

- *Push Acceptance Rate*: The number of the sessions in that the test users accept to see the recommendation list (after the system shows the push notification).
- *Recommendation Success Rate*: The number of the sessions in that the test users select some promotion(s).
- *Average Recommendation Length*: The average number of cycles of a recommendation session.
- *Average Number of Criticized Features Per Cycle*: The average number of features that are criticized per cycle.

In total, we collected 32 recommendation sessions from the test users' use of *Prom4U*. Regarding the push acceptance rate, among 32 sessions there were 30 ones in that the test users accepted to see the recommendation list after receiving the notification screen (i.e., the push acceptance rate of 93.75%). This objective result shows that in many cases the context (i.e., time) when the system automatically pushes the recommendation list to the user is appropriate (or at least acceptable) for him/her.

Regarding the recommendation success rate, among 30 sessions (i.e., those in that the user accepted to see the recommendation list) there were 27 successful recommendation sessions (i.e., 90% of the recommendation sessions in that the user selected some recommended promotions). This objective result is very promising, which shows that the system is capable of providing good recommendations for the users.

The average recommendation length was 1.67, which means that on average the test users could find their desired promotions within 1-2 recommendation cycles. Recall that in our proposed recommendation approach, at each recommendation cycle the user can make critique to a number of features of the criticized item. Given the simple and easy user-system interactions (see Figure 2-c,d), this average recommendation length is really acceptable for mobile users.

The average number of criticized features per cycle was 2.56. This result shows our proposed approach's effectiveness of allowing to make critique to more than one feature per cycle. If the user is limited to make critique to only one feature per cycle (e.g., as in [3]), then certainly the average recommendation length would be (much) longer than the reported value of 1.67.

At the end of this live-user evaluation (i.e., after more than two months of using *Prom4U*), the subjective comments and suggestions were collected from each test user, in form of free-text writing, regarding the effectiveness and usability of *Prom4U*. All the test users found that: 1)the system was effective in helping them receive their interested promotions, and 2)the system was easy to use and fast in the system-user interactions. However, by exploiting the test users' comments and suggestions, we found some possible improvement aspects of *Prom4U*. First, at first some test users did not find how to execute the screen-embedded commands (e.g., the buttons "Critique" and "Select" in Figure 2-c). (To execute such a screen-embedded command, the user has first to navigate through several display objects and then to activate that command.) For them, soft-button or menu commands are more traditional and easier to use. Second, some test users wanted to see again those promotions that they had previously criticized (in their current recommendation session). Third, some test users suggested that visualizing the recommended promotions on an electronic map, i.e., corresponding with their providers' locations, facilitates their promotions selection. In the next improvement version of *Prom4U* we should take into account these comments and suggestions.

6 Conclusion and Future Work

Mobile recommender systems aim at providing recommendations to users at anytime and anywhere, exploiting the popularization of mobile devices and their unique features like mobility, high targeting and personality. In this paper, we have presented our proposed methodology for proactively providing personalized recommendations to mobile users at appropriate contexts. The integration of the user's long-term and session-specific preferences enables the system to provide relevant recommendations, and the appropriate push-context prediction helps the system deliver these recommendations to him at right time. This mobile push recommendation methodology has been implemented in a recommender system called *Prom4U* that helps users timely receive their interested product promotions. In this paper, we have also presented the experimental results of the live-user evaluation of *Prom4U*, which show that our proposed approach is a good and promising solution for the mobile push-delivery recommendation problem.

For future work, we plan to do some tasks. First, we would like to run an empirical study to understand if users see all the features of the item before making a critique. This strongly influences the update method of the user query representation (Q). Second, we need to exploit all the test users' comments and suggestions to improve the usability of *Prom4U*. Third, we will need to find the best way to visualize the push notification on the screen of the user's mobile device. In the current design of *Prom4U*, this notification occupies the whole screen (see Figure 2-a), and it should be reduced much smaller (e.g., like the way of visualizing an incoming SMS message); so the notification causes less interruption and becomes more friendly for mobile users.

Acknowledgement. The financial support for this research work from the Vietnam National Foundation for Science and Technology Development (NAFOSTED) under the grant number 102.01.14.09 is gratefully appreciated.

References

1. Burke, R.: Hybrid Web Recommender Systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) The Adaptive Web 2007. LNCS, vol. 4321, pp. 377–408. Springer, Heidelberg (2007)
2. Ricci, F., Rokach, L., Kantor, P.B.: Recommender Systems Handbook. Springer (2010)
3. Ricci, F., Nguyen, Q.N.: Acquiring and Revising Preferences in a Critique-Based Mobile Recommender System. IEEE Intelligent Sys. 22(3), 22–29 (2007)
4. Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications 7(1), 39–59 (1994)
5. Aalto, L., Göthlin, N., Korhonen, J., Ojala, T.: Bluetooth and WAP Push-Based Location-Aware Mobile Advertising System. In: 2nd International Conference on Mobile Systems, Application, and Services, pp. 49–58 (2004)
6. Hristova, N., O'Hare, G.: Ad-me: Wireless Advertising Adapted to the User Location, Device and Emotions. In: 37th Annual Hawaii International Conference on System Sciences, pp. 285–294 (2004)
7. de Castro, J.E., Shimakawa, H.: Mobile Advertisement System Utilizing User´s Contextual Information. In: 7th International Conference on Mobile Data Management, p. 91 (2006)
8. Kurkovsky, S., Harihar, K.: Using Ubiquitous Computing in Interactive Mobile Marketing. Personal and Ubiquitous Computing 10(4), 227–240 (2006)
9. Ciaramella, A., Cimino, M.G.C.A., Lazzerini, B., Marcelloni, F.: Situation-Aware Mobile Service Recommendation with Fuzzy Logic and Semantic Web. In: 9th International Conference on Intelligent Systems Design and Applications, pp. 1037–1042 (2009)
10. Pinyapong, S., Shoji, H., Ogino, A., Kato, T.: A Mobile Information Service Adapted to Vague and Situational Requirements of Individual. In: 7th International Conference on Mobile Data Management, pp. 20–22 (2006)
11. Nguyen, Q.N., Ricci, F.: User Preferences Initialization and Integration in Critique-Based Mobile Recommender Systems. In: 5th International Workshop on Artificial Intelligence in Mobile Systems, pp. 71–78 (2004)