Push-Delivery Personalized Recommendations for Mobile Users

Quang Nhat Nguyen
School of Information and Communication Technology
Hanoi University of Technology, Vietnam
quangnn-fit@mail.hut.edu.vn

Phai Minh Hoang
School of Information and Communication Technology
Hanoi University of Technology, Vietnam
minhphai87@gmail.com

***Abstract*—In application domains where the availability of items changes quickly and often (e.g., the user problem of receiving relevant promotions, events, etc.), users often find it difficult in keeping track of their desired and interested items. Recommender systems are intelligent decision support tools aimed at addressing the information overload problem, suggesting items that best suit a given user's needs and preferences. In this paper, we present our proposed mobile push-delivery recommendation methodology that is capable of proactively providing recommendations relevant to the user's preferences at appropriate context. The proposed methodology is implemented in a mobile push recommender system that helps users timely receive their desired product promotions.**

Keywords- mobile recommender system; push delivery; context-aware mobile application

I. INTRODUCTION

E-commerce sites often provide huge catalogues of diverse products and services. Hence, without the system support, users of e-commerce sites may find it difficult in making product selection decisions. This information overload problem becomes even harder for mobile users who interact with the system using mobile devices, due to the limitations of mobile devices and mobile users' limited spent time and effort. 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]. 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 (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 [2]. 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 [3], i.e., a machine learning approach, to exploit (reuse) the knowledge contained in the past push cases to determine the right push context for the current case. Our proposed methodology has been implemented in a mobile push recommender system that helps users timely receive their interested product promotions.

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. Finally, the conclusion and future work are given in Section 5.

II. RELATED WORK

Recommender Systems (RSs) are 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. Traditional recommendation approaches include: *collaborative*, *content-based*, and *knowledge-based* [1]. RSs have been very effective and popular tools in well-known commercial websites, such as Barnes&Noble.com, eBay.com, Amazon.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 contexts (time and location)*. In some previous approaches, the system just pushes all objects (or items) that locate near the user's position, without regarding

 ISBN: 978-1-61208-100-7

his preferences [4], [5]. 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 [6], [7]. Ciaramella et al. [8] 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 [9] 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 [2]. Such critiques enable the system to better understand the user's preferences, and hence to provide more suitable recommendations to the user.

III. FORMAL REPRESENTATIONS

A. Product Promotion Representation

In our recommendation problem, the system's recommendation aims at promotions, whereas the information of promoted products and gifts is supplemental. In particular, 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. (Due to the paper's space limit, we present here only the first and second levels of the hierarchical representation.)

PROMOTION = (*PROMOTION-INFO*, *PROMOTED-PRODUCTS*, *GIFTS*)

The component *PROMOTION-INFO* stores the information of the promotion:

PROMOTION-INFO = (*Prom-Type*, *CONDITION*, *DURATION*, *PROVIDER*);

where the feature *Prom-Type* represents the type of the promotion, the sub-components *CONDITION*, *DURATION* and *PROVIDER* represent the promotion's condition, available duration and provider, respectively.

The component *PROMOTED-PRODUCTS* represents the set of the promoted products and their quantity (i.e., required to buy in order to get the promotion):

PROMOTED-PRODUCTS = {(*PRODUCT*, *Quantity*)};

where the sub-component *PRODUCT* is represented by the three features: the promoted product's category (e.g., laptop, TV, etc.), identifier and price.

The component *GIFTS* represents the set of the gifts of the promotion:

GIFTS = {(*Gift-Type*, *GIFT*)};

where the value of the feature *Gift-Type* defines the (structured) content of the sub-component *GIFT*.

B. 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 = (*PROMOTION-PREF*, *PRODUCT-PREF*, *GIFT-PREF*);

where the component *PROMOTION-PREF* stores the user's long-term preferences on promotions types, condition types and providers; the component *PRODUCT-PREF* stores the user's long-term preferences on category and price of promoted products; and the component *GIFT-PREF* stores the user's long-term preferences on gift types.

C. User Query Representation

The user query (*Q*) representation encodes the system's understanding (i.e., guess) of the user's session-specific preferences. In a session, at every recommendation cycle the system uses the query *Q* to compute the promotions recommendation list that is then shown to the user.

In our approach, 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.

IV. RECOMMENDATION METHODOLOGY

In our approach, a recommendation session starts when the system's promotions catalogue is updated (with new promotions) or when the user is close to a promotion provider's store, and ends when the user quits the session. The overview of the recommendation process is shown in Fig. 1.

When the session starts, the system builds the initial query representation (Q^0) exploiting the user's long-term preferences stored in the user profile. In this initialization

 ISBN: 978-1-61208-100-7

step, the values of the features of FP^0 are set by the values of the corresponding features in the user profile (U). In addition, the values of the sub-component *DURATION* and the feature *Distance* are set to unknown to indicate that at the beginning of the session the system does not know about the user's session-specific preferences on promotion available period and distance to provider.

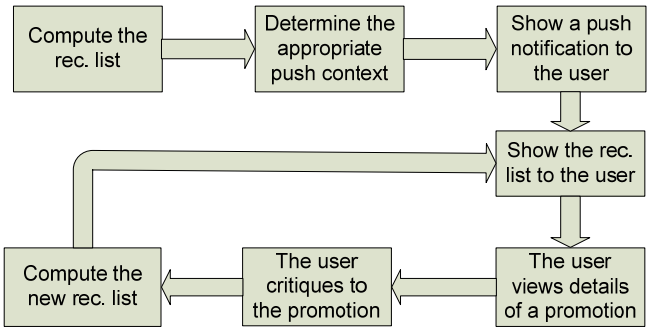


Figure 1. The overview of the recommendation process

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). In Fig. 2, it illustrates an example of a sequence of critiques that a user makes in a recommendation session. A recommendation session evolves in recommendation cycles, where each cycle comprises the stage where the recommended promotions are shown to the user (see Fig. 3-a) and the successive stages where the user browses the details of a promotion and criticizes it (see Fig. 3-b).

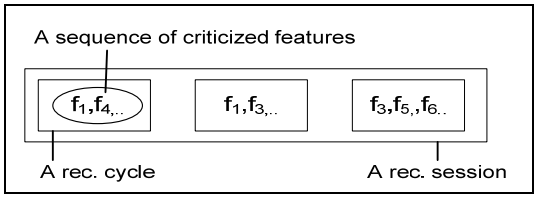


Figure 2. A sequence of critiques in a session

We note that for each representation level (of W) the system computes the importance weights of the features (or sub-components) at that level. For example, for the level of *PROMOTION-INFO*, the system computes the importance weights of the feature *Prom-Type* and the sub-components *CONDITION*, *DURATION* and *PROVIDER*.

First, the system computes the importance weight of feature (or sub-component) f_i, given session s_k of user u_j:

$$w_i(u_j, s_k) = \frac{1}{\lambda_k} \cdot \sum_{l=1}^{\lambda_k} \frac{Ctz(f_i, u_j, c_l)}{\alpha^{(\lambda_k - l)}} \qquad (1)$$

where:

- c_l: a recommendation cycle of session s_k;
- λ_k: the length (i.e., the number of recommendation cycles) of session s_k;
- $Ctz(f_i, u_j, c_l) = 1$, if at cycle c_l user u_j made a critique on feature (or sub-component) f_i ,
  = 0, if otherwise;
- α (>1): a parameter to increase the importance of latest critiques (i.e., those appear later in session s_k).

Next, the system computes the importance weight of feature (or sub-component) f_i over all the sessions of user u_j.

$$w_i(u_j) = \frac{1}{\| S(u_j) \|} \cdot \sum_{k=1}^{\|S(u_j)\|} \frac{w_i(u_j, s_k)}{\beta^{(\|S(u_j)\| - k)}} \qquad (2)$$

where :

- $S(u_j)$: the set of historically ordered sessions of user u_j.
- β (>1): a parameter that shapes how fast the importance of an old session decreases over time.

The system uses the 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 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 approach, this push-context determination is done based on the Case-Based Reasoning (CBR) problem-solving strategy [3]. The CBR is used to exploit (reuse) the knowledge contained in the past push cases. In our methodology, each push *case* is modeled by two parts: the *problem description* and the *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: 1) the user accepts to receive (i.e., view) the recommendation list, or 2) the user rejects to receive.

To estimate (i.e., predict) an appropriate push context, the system identifies: 1) 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 2) 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 *push degree* (i.e., the confidence level to push) and the *not-push degree* (i.e., the confidence level to not push) for the current case.

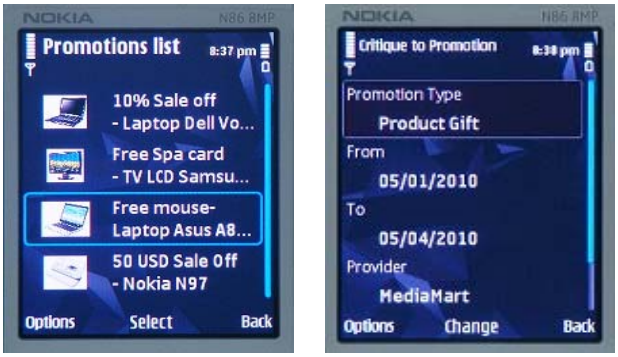
$$push - \deg ree(C^*) = \frac{1}{m} \sum_{C \in C^{Accepted}} Sim(C^*, C) \qquad (3);$$

$$not - push - \deg ree(C^*) = \frac{1}{m} \sum_{C \in C^{Rejected}} Sim(C^*, C) \qquad (4)$$

where C^* is the current case, and C is a past case. If (*push-degree*(C^*) – *not-push-degree*(C^*)) ≥ θ (i.e., θ is a predefined push-confidence threshold), 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 the user (i.e., at the next time-slot, the system re-estimates whether or not to send the push notification to the user).

Given the push notification sent to the user's mobile device, he can decide to accept the push, or postpone it, or reject it. In case the user accepts the push, the system first stores (i.e., records) the current push case in its case base for the future uses, and then shows the recommendation list to him (see Fig. 3-a). In case the user postpones the push, he can specifies the later appropriate time slot (i.e., postponed by time) or the appropriate distance to provider (i.e., postponed by distance) to receive the recommendation list. In case the user rejects to receive the push, the system stores the current push case in its case base.



a) Recommendation list b) User critique

Figure 3. The mobile user interface

When the recommendation list is shown to the user (see Fig. 3-a), for each recommended promotion the system shows an icon corresponding to the promoted product's category, the gift's abstract information and the promoted product's name. The user can select a recommended promotion to see its details. After the user 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 the user is somewhat interested in the promotion, but one (or some) of its features does not completely satisfy him, then he critiques to the promotion to specify (i.e., express) his preferences on these unsatisfactory features (see Fig. 3-b). In Fig. 3-b, for example, the user critiques to the promotion to indicate his preference on the promotion's type. By critiquing, the user at the next recommendation cycle (of the current session) is recommended with other promotions that are "closer" to his preferences. Such critiques help the system adapt its previous user-query representation (i.e., guess) (Q) to the user's new preferences, and re-compute some new recommended promotions based on this adapted user query. The new list of recommended promotions is then shown to the user, and the system proceeds to the next recommendation cycle.

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. This user profile update allows the system to refine its understanding of the user's long-term preferences, and hence better serve the user in the future.

V. CONCLUSION AND FUTURE WORK

Mobile recommender system aims 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 push-context determination helps the system deliver these recommendations to him at right time and location. This mobile push recommendation methodology has been implemented in a recommender system that helps users timely receive their interested product promotions.

We shall run a usability evaluation of the implemented system to test the effectiveness of our proposed methodology and the usability of the implemented system. In addition, we will need to find the best way to visualize the push notification on the user's mobile device.

ACKNOWLEDGMENT

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] R. Burke, "Hybrid web recommender systems", in The Adaptive Web: Methods and Strategies of Web Personalization, P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds. Heidelberg: Springer, 2007, pp. 377–408.

[2] F. Ricci and Q. N. Nguyen, "Acquiring and revising preferences in a critique-based mobile recommender system", IEEE Intelligent Systems, vol. 22, n. 3, pp. 22–29, May-June 2007.

[3] A. Aamodt and E. Plaza, "Case-based reasoning: foundational issues, methodological variations, and system approaches", AI Communications, vol. 7, n. 1, pp. 39–59, March 1994.

[4] N. Hristova and G. O'Hare, "Ad-me: wireless advertising adapted to the user location, device and emotions", in Proc. 37th Annual Hawaii Int. Conf. System Sciences, 2004, pp. 285–294.

[5] L. Aalto, N. Göthlin, J. Korhonen, and T. Ojala, "Bluetooth and WAP push-based location-aware mobile advertising system", in Proc. 2nd Int. Conf. Mobile Systems, Application, and Services, 2004, pp. 49–58.

[6] S. Kurkovsky and K. Harihar, "Using ubiquitous computing in interactive mobile marketing", J. Personal and Ubiquitous Computing, vol. 10, n. 4, pp. 227–240, May 2006.

[7] J. E. de Castro and H. Shimakawa, "Mobile advertisement system utilizing user's contextual information", in Proc. 7th Int. Conf. Mobile Data Management, 2006, pp. 91.

[8] A. Ciaramella, M. G. C. A. Cimino, B. Lazzerini, and F. Marcelloni, "Situation-aware mobile service recommendation with fuzzy logic and semantic web", in Proc. 9th Int. Conf. Intelligent Systems Design and Applications, 2009, pp. 1037–1042.

[9] S. Pinyapong, H. Shoji, A. Ogino, and T. Kato, "A mobile information service adapted to vague and situational requirements of individual", in Proc. 7th Int. Conf. Mobile Data Management, 2006, pp. 20–22.

 ISBN: 978-1-61208-100-7