A Study on Proactive Delivery of Restaurant Recommendations for Android Smartphones

Daniel Gallego Vico
Universidad Politécnica de Madrid
Avenida Complutense 30,
28040, Madrid, Spain
dgallego@dit.upm.es

Wolfgang Woerndl
Technische Universitaet Muenchen
Boltzmannstr. 3,
85748 Garching, Germany
woerndl@in.tum.de

Roland Bader
BMW Group Research and Technology
Hanauerstraße 46,
80992 Munich, Germany
roland.bader@bmw.de

ABSTRACT

A proactive recommender system pushes recommendations to the user when the current situation seems appropriate, without explicit user request. Important research questions include whether users would accept proactive recommendations, how to present recommended items and possibly notify users. Our scenario is a context-aware restaurant recommender for Android smartphones. We have designed two options for the user interaction with a proactive recommender: a widget- and a notification-based solution. In addition, our user interface includes a visualization of recommended items and allows for user feedback. The approach was evaluated in a survey among 58 users with good results regarding usefulness and effectiveness. The results also showed that test users preferred the widget-based solution.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; H.3.3 [**Information Search and Retrieval**]: Information filtering; H.5.2 [**User Interfaces**]: Evaluation, Graphical user interfaces (GUI)

General Terms

Design, Experimentation, Human Factors

Keywords

Recommender System, Mobile, Proactivity, Evaluation, Personalization, Context-aware, User Interface, Usability

1. INTRODUCTION

Traditional recommender systems usually follow a request-response pattern, i.e. these systems only return item suggestions when a user makes an explicit request. In mobile recommender systems, users cannot browse easily through many search results and suffer from other restrictions in the user experience. This is so because of limitations in the user interface such as small display sizes or missing keyboards. In mobile environments, user experience could possibly be improved by delivering recommendations without any user request or query. Consider the following scenario: A mobile restaurant guide running on a smartphone suggests a restaurant to the user when she is walking near the restaurant that fits her preferences very well, while also factoring in the time of the day and other context attributes.

Proactivity means that the system pushes recommendations to the user when the current situation seems appropriate. Important questions in this scenario are whether users would accept proactive recommendations, how to present recommended items and possibly notify users. To investigate these questions, we have designed and implemented the user interface for a mobile application that proactively delivers recommended restaurants to users on Android smartphones. The mobile recommender system is based on a general model for proactivity in mobile, context-aware recommender systems [12]. We have conducted a user survey to test and compare two options for notifying users about recommended restaurants.

The rest of this paper is organized as follows. First, the next section reviews related work. Section 3 summarizes the proactivity model for mobile recommender systems. In Section 4 we describe the design and implementation of our mobile application for the restaurant scenario that has been evaluated in a survey among users. Section 5 presents the results achieved from the survey carried out. After that, in Section 6 we discuss which of the two proactive solutions analyzed is the best and how the result visualization is accomplished. Finally, the last section provides some concluding remarks and directions for future research.

2. RELATED WORK

A large amount of research and practical applications exist on recommender systems, mobile computing, context-awareness (see e.g. [4]) or location-based services, as well as any combination of the above areas. For example, Kenteris et al. recently surveyed the field of mobile guides [8]. However, proactivity has not gained much attention in personalization and recommender system research. Most systems require the user to perform some kind of action to trigger the generation or retrieval of recommended items.

As an example of proactivity in an existing system, Hong et al. [7] proposed an agent-based framework for proactive personalization services. This approach proposes a model

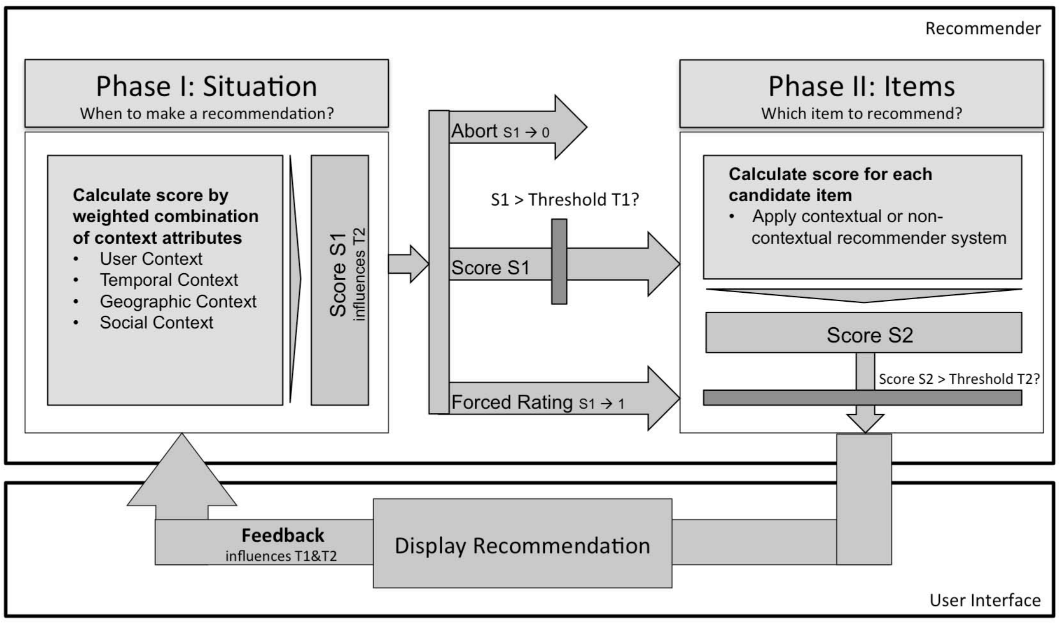


Figure 1: Proactivity Model.

according to which a user profile is deduced from a user's context history. The model enables proactive recommendations in the future. However, training time is very important in the proposed model.

Some studies have investigated the "interruptablity" of persons. [6] presents work on estimating human interuptability using vision-based sensors. Their results indicate that this can be a practical approach. Recent work also covered proactive notications on mobile devices in general. For example, [11] studied the interruptablity of mobile phone users and proposed a model to approximate users' interruptablity costs. This allows for an application to learn when to automatically turn the device's volume on and off.

Ricci discusses proactivity in mobile recommender systems in his survey [10]. Some systems make use of the current user behavior, position and other context information to improve personalization on mobile devices and in ubiquitous computing in general. Ricci concludes that "none of the existing reviewed systems is capable to proactively interrupt the user activity with unsolicited but relevant recommendations" and "[proactive recommendations] can revolutionize the role of recommender systems from topic oriented information seeking and decision making tools to information discovery and entertaining companions" [10].

3. A PROACTIVITY MODEL FOR MOBILE RECOMMENDER SYSTEMS

3.1 Process Overview

To handle proactivity in mobile recommender systems, we propose the following two-phase model [12]. The model analyzes the current context and calculates a score that determines not only the best item(s) in a given situation, but also whether the situation warrants a recommendation at all. Context can be defined as characterizing the situation of entities that are relevant to the interaction between a user and an application [5]. We are utilizing the following four context categories: 1. User context, e.g. the current activity of the user such as "walking" as inferred from sensor data, 2. Temporal context, e.g. current time, 3. Geographic context, e.g. distance of available points of interest, and 4. Social context, e.g. whether the user is alone or not.

Figure 1 summarizes our two-phase proactivity model. In the first phase, the system determines whether or not the current situation warrants a recommendation. The second phase deals with evaluating the candidate items. If one or more items are considered good enough in the current context in the second phase, the recommender system communicates it to the user. The first phase is executed periodically in the background. The second phase is only executed when the first phase indicates a promising situation. Note that the first phase does not take properties of single items into account, but does consider general properties of the set of candidate items, e.g. availability of restaurants in the vicinity as part of the geographic context.

3.2 Phase I: Situation Assessment

In the first phase, the system calculates a score S1 which is a number between 0 and 1. If S1 exceeds a threshold T1, the second phase will be initiated. If S1 = 1, the highest possible value, then a recommendation will be triggered in any case. If the current situation does not warrant a recommendation, no matter how high a particular item would score, S1 is set to 0 and the recommendation process is aborted without considering items for recommendation.

Furthermore, the score S1 has an impact on the threshold T2 of the second phase, i.e. the higher S1 is, the lower T2 is set. This means that when the situation is considered appropriate for a recommendation, S1 is high and it is more likely that at least one item score S2 in the second phase reaches the required threshold T2 and an item will be recommended to the user.

3.3 Phase II: Item Assessment

The second phase evaluates the suitability of particular items. To do so, any recommender algorithm can be used. The result of phase II is a score S2 for each item in the candidate set. S2 corresponds to the predicted rating of collaborative filtering or any other recommendation algorithm. S2 is again a number normalized to [0,1], with S2=1 being the best possible score. An item can be immediately eliminated from the recommendation process (then S2 is set to 0), for example if a restaurant is closed right now. The candidate items will be ranked according to S2 and tested against the threshold T2. If S2 $>$ T2 for an item, then this item is finally considered for recommendation and the user is notified (see next section). Depending on the application scenario, the k best items above the threshold will be displayed. If no item score S2 exceeds the threshold T2, then no item is recommended, the process is aborted and restarted with phase I at the next configured interval.

After the recommended items are communicated to the user, she can optionally give feedback on the recommendation. This is shown in our prototype user interface (cf. Figure 4). The feedback is a rating for an item utilized when assessing the relevance of the item in phase II. In addition, the user can give feedback on the point in time of the recommendation ("not now"). In this case, the feedback influences the thresholds T1 and T2: a negative feedback on the point in time results in higher thresholds and thus decreases the chance of a proactive recommendation in the future.

4. DESIGN AND IMPLEMENTATION OF THE ANDROID APPLICATION

Section 3 explained how the recommendation engine works; this section now focuses on the mobile client application developed for this research. Nevertheless, before starting to describe the different user interfaces involved on it, it is important to explain some design details. After that, we will describe first how the proactivity is achieved in terms of user interface through two possible solutions (widget-based and notification-based), and then we will explain how the result visualization is carried out.

4.1 Technological Decisions

First of all, we want to stress that we chose Android because it allowed us to test two different new ideas of having proactivity in mobile systems that were not available in other platforms during the elaboration of this research. Nevertheless, Apple has launched recently the new iOS 5 allowing to implement also these new ideas [2].

As we will see in the next subsections, the first solution is based on using widgets, and the second one is based on using notifications in the status bar. Both will be explained in detail.

Furthermore, to make the user interface implementation independent from the underlying recommendation engine, an Android service is in charge of the communication with it. In Android, a service is defined as an application component that can perform long-running operations in the background and does not provide a user interface [1]. Therefore, this service is started when the application is installed and it will continue to run in the background even if the user switches to another application. The aim of this service is to wait for push messages from the recommendation engine

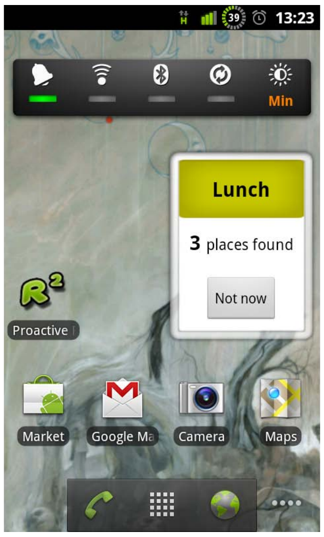


Figure 2: Widget-based proactive user interface.

in order to pass then the received data to the user interface. The push messages contain the recommended items (in our case, restaurants). The items were selected by the recommendation engine to be relevant in the current context, as explained above in Section 3. The user can give feedback in two ways:

- Feedback on the time of the proactive recommendation ("not now").
- Feedback on the recommended items ("like" or "dislike").

Consequently, the communication between the mobile application and the recommendation engine is simple in order to avoid complex communication protocols. The business logic is executed in the engine side while the mobile client is only in charge of informing the user about the available recommendations and presenting the results.

4.2 Widget-based User Interface for Proactive Recommendations

The first solution for proactive recommendation in our application is based on Android widgets. A widget is a miniature application view that can be embedded in other applications such as the Home screen of the mobile device (this is our case) and receive periodic updates. These updates are provided by a service running in the background waiting for recommendations, as explained above. Figure 2 shows a screenshot for a lunch recommendation.

The widget is always active in the Android device's desktop in order to inform the user anytime she unlocks her smartphone. Therefore, the user can notice if any recommendation has been generated by the system. It also informs the user about the category of restaurant recommendation. To do this we have defined four categories (breakfast, lunch, snack/tea or dinner) corresponding to the different restaurant recommendations that can be generated during a day. In addition, the system provides the number of places that are recommended at this time. Then, with one click the user can reject the proactive recommendation by pressing the

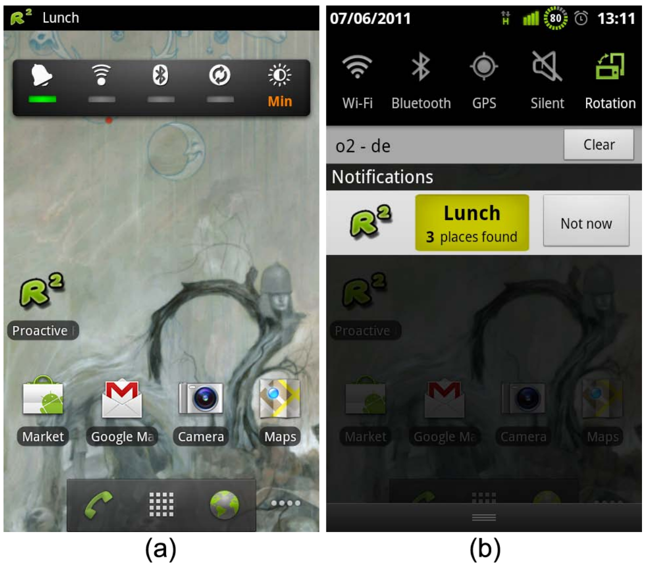


Figure 3: Notification-based proactive user interface. Notification icon and message in the status bar (a), and expanded notification (b).

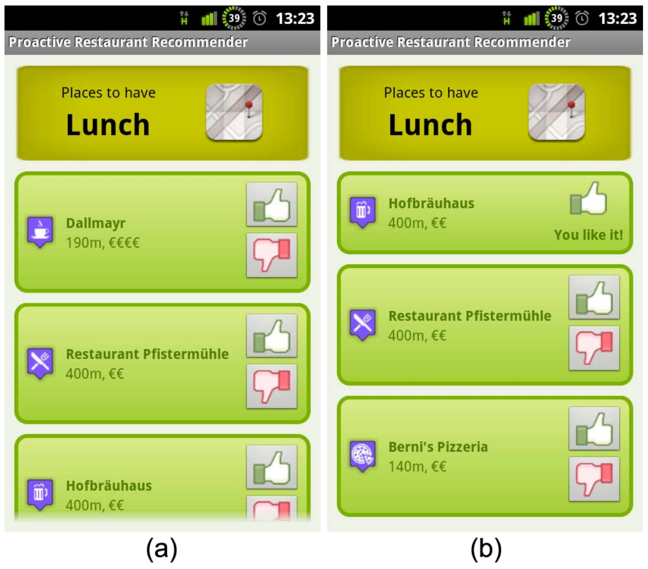


Figure 4: Ranking user interface. Initial view (a) and after giving feedback (b).

"Not now" button, or accepting it by pressing the button of the category ("Lunch" in the Figure 2). With these actions the system collects user feedback to discover if the point of time to generate recommendations has been appropriate or not. This information is sent to the recommendation engine to be taken into account for future recommendations.

To improve the application usability, a color code is used to identify every category in order to let the user be aware of the recommendation without reading anything. We use orange (breakfast), green (lunch), blue (snack/tea) and violet (dinner) colors. This makes it easier for the users to associate every color to every category recommendation, saving in this way time for them.

Finally, if the user does not interact with the widget and the context changes (e.g. the lunch time is over), the widget does not notify any recommendation until a new one is considered suitable by the system.

4.3 Notification-based User Interface for Proactive Recommendations

The second solution to offer proactivity in our restaurant recommender is based on Android notifications. A notification adds an icon to the Android system's status bar with a ticker-text message (Figure 3a) informing the user about an event generated by one of the applications installed. When the notification is expanded by a drag down action, a detailed message in the notifications window is shown (Figure 3b) with more information about the event. Furthermore, in order to help the user noticing the new recommendations available, we added sound and vibration to the recommendation notification event, although it can be configured by the user (e.g. for silent mode). We also followed the same color code described before for the widget-based solution.

As the screenshot of our application illustrates in the Figure 3b, we used the same buttons as in the widget approach to design the expanded notification information, allowing the user to see the results from the recommendation by pressing the category button or rejecting the recommendation by pressing the "Not now" button to inform the system that the time chosen for the proactive recommendation is not desirable for her. Again, this feedback information is sent to the recommendation engine to be considered for future recommendations.

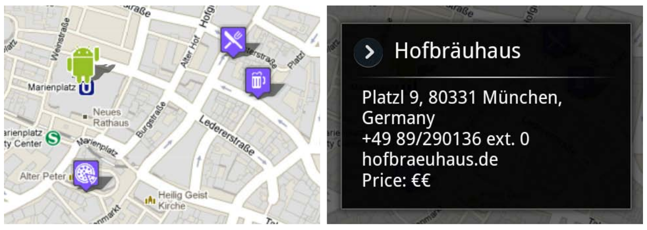
Finally, as in the widget-based solution, the notifications not expanded can disappear or change due to new context conditions.

4.4 Recommendations Visualization: Ranking and Map

The last two sections were focused on detailing the proactive interfaces. Now we are going to describe the result visualization of the places shown when a user wants to see the recommendation made by the system.

The first view the user sees when she presses the "Lunch" button is the one shown by the Figure 4a. This list view shows the items ordered by a ranking criterion that is based on a combination of the different places attributes (e.g. cuisine, distance, price, etc.) and the current context. This is done by the recommendation engine in phase II of the process explained in Section 3. Every element of the list has a category map icon that represents his cuisine attribute. This icon is then used to represent the place in the map view (Figure 5) in order to make it easier for the user to localize every restaurant. Additionally, every element has a brief description of the most important details related to it (i.e. name, distance and average price).

Furthermore, to complete the user feedback related to the place recommendation process, every item has a "Like" and "Dislike" button. When the user presses any of these two buttons, his/her explicit rating is stored in the recommendation engine to be taken into account for future recommendations. By pressing the "Like" button the corresponding item may rise in the ranking, whereas by pressing the "Dislike" button the place disappears from the list and a new item is added, if there is another recommended item in the



Hofbräuhaus

Platzl 9, 80331 München, Germany
+49 89/290136 ext. 0
hofbraeuhaus.de
Price: €€

Figure 5: Map user interface. Map visualization (a) and Place details (b).

list generated by the engine. An example of this situation is illustrated by Figure 4. In the Figure 4a the user likes the Hofbräuhaus bar and dislikes the Dallmayr café for having lunch. As a result, in Figure 4b the Hofbräuhaus raises to the first place of the ranking and a new place (Berni's Pizzeria) appears to replace the disliked one.

Once the user has given feedback that may influence the ranking that comes from the engine (the score S2 explained in Section 3), by pressing the map button shown in the Figure 4, the map view illustrated by Figure 5a appears. It is in charge of representing the different restaurants selected by the user in the ranking view. To do this, we have integrated Google Maps in our application to localize the places in a map and to have also the possibility of creating routes from user's current location to any of the places. The map icons are the same as in the ranking view to represent the restaurants quickly. The current user location is represented by an Android icon that can be personalized using other images. Finally, the Figure 5b shows the detailed information that is shown when a user presses one of the restaurants.

We have integrated this kind of result representation bearing in mind that Averjanova et al. [3] demonstrated with a real user study, that a map-based interface is more effective than a list-based interface (that is typically used in recommender systems) to select a specific recommendation. We have included also the ranking visualization (list-based) as it is a clearer way of allowing users giving feedback, separating in this way the feedback process from the item selection process, and as a result, taking the advantages of both solutions.

5. EVALUATION AND RESULTS

We have created a survey with two parts to evaluate the implemented user interface. The first one was focused on comparing the two proactivity solutions, whereas the second one was focused on analyzing the result visualization and the recommendation feedback process.

This was done by collecting subjective measures of the user experience given by participants in an online survey that presented some scenarios. These scenarios describe situations in which the users could need a restaurant and for this reason a recommendation is generated by the system. In the survey, the scenarios illustrated by Table 1 had also screenshots and enough previous information about the prototype to understand them properly.

Furthermore, we split up randomly the users into two test groups: the first (called α) evaluated first the notification-based solution and then the widget-based solution. While in the second one (called β) the evaluation order was inverse (first widget, then notification). This was done to compensate any learning effect, i.e. users' opinion about the notification-based solution could be better or worst depending on if they have evaluated before the widget-based solution or not.

Table 1: Survey scenarios

ID	Description
S1	On the way from home to work in the morning. You have not had breakfast and a recommendation is available.
S2	Like the scenario S1, but you have had breakfast at home.
S3	On the way from work to home during the evening. You are not in a hurry and a recommendation is available.
S4	Like scenario S3, but you are in a hurry to arrive home.
S5	At your hotel during a business trip. It is lunch time, the day is rainy and you have not had lunch when a recommendation is available.
S6	On a tourist weekend walking during the lunch time along the street. You have not had lunch and a recommendation is available.

Therefore, in this section we present the results of the survey using figures and tables to show the statistical data, and then, Section 6 will discuss and interpret these findings.

5.1 Demographic Data

58 test users participated in our survey. They were recruited from two different contexts. The first one was composed by people (i.e. professors, PhDs and PhD students) from computer science departments at the Universidad Politécnica de Madrid and the Technische Universitaet Muenchen. The second group was composed by people not related to the university or technological issues in order to achieve different ways of thinking. Both groups did not have any experience with the scenarios and the research topic in advance.

The majority (72%) was male, and the age distribution of the subjects was from 22 to 58 years old, with the majority (84%) in the 22 to 35 years range. In addition, 76% of them owned a smartphone (e.g. Samsung Galaxy S, Nexus One, iPhone, etc.) and asked about how often did they check the device, 23% said *"several times per hour"*, 40% said *"every hour"*, 28% said *"several times per day"* and 9% said *"few times per day"*.

5.2 Notification-based Solution

First of all, we are going to present the results of to the notification-based solution. The scenarios in Table 1 are used for evaluating both options (notification and widget). Hence, the descriptions related to this solution were adapted considering that the users were aware of the notifications by a vibration and a sound produced by their smartphones, as we explained in the Section 4.2. In Table 2 we can see the subjects' answers taking into account the two groups we mentioned before (α and β). Moreover, the possible responses for these scenarios were: Ignore (*ignore the notification*), Not now (*expand the notification and push the "Not now" button*) and Expand (*expand the notification and consult the recommendations*).

Table 2: Notification-based scenarios responses.

ID	Group	Ignore(%)	Not now(%)	Expand(%)
S1	α	29	33	38
	β	24	41	35
S2	α	26	61	13
	β	44	47	9
S3	α	8	38	54
	β	20	15	65
S4	α	50	42	8
	β	62	29	9
S5	α	0	4	96
	β	9	9	82
S6	α	4	13	83
	β	6	9	85

Table 3: Widget-based scenarios responses.

ID	Group	Ignore(%)	Not now(%)	Expand(%)
S1	α	21	29	50
	β	24	41	35
S2	α	29	54	17
	β	41	32	27
S3	α	8	33	59
	β	23	21	56
S4	α	54	38	8
	β	65	29	6
S5	α	8	13	79
	β	3	9	88
S6	α	8	13	79
	β	3	6	91

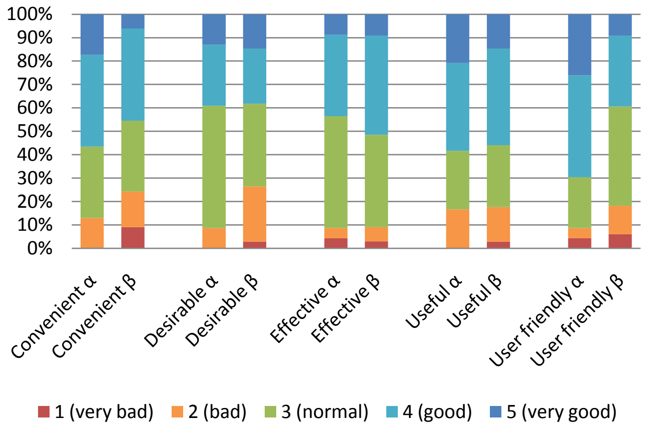


Figure 6: Evaluation results of notification-based solution.

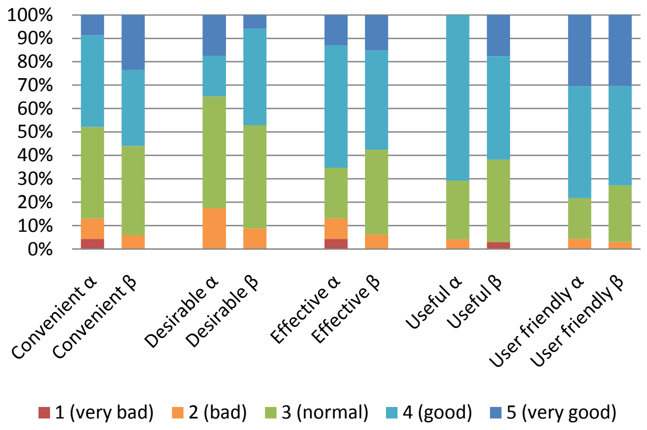


Figure 7: Evaluation results of widget-based solution.

Lastly, the test users were asked to judge some properties related to the solution provided (i.e. convenient, desirable, efficient, useful and user friendly), using a 5-point scale, where 1 means *"very bad"* and 5 *"very good"*. The results are shown in the Figure 6, separate for α and β groups.

5.3 Widget-based Solution

Now, using the same scenarios described in the Table 1 (adapted to the widget-solution in which the users notice the recommendations by seeing the widget when they are checking their email or calling someone), we can see in the Table 3 the subject's responses. In this case, the possible answers were: Ignore (*ignore the widget*), Not now (*press the "Not now" button*) and Expand (*press the category button to expand the recommendations*).

Finally, the test users were asked to judge some properties related to the solution provided using the same 5-point scale as in the previous section. The results are illustrated in the Figure 7.

5.4 User's Assessments in Comparing Both Solutions

In addition to the individual evaluation of both solutions described in the previous sections, we included a part focused on comparing them. To do this, the test users were asked to judge some statements related to compare both solutions using a 5-point scale, where 1 means *"totally disagree"* and 5 *"totally agree"*. In this case, and taking into account that the results from α and β groups were very similar, we have aggregated them in only one bar chart (Figure 8) for clarity.

5.5 Recommendations Visualization

For the second part of the survey, we assumed that the users wanted to see a "Lunch" recommendation (like the one shown by Figures 2 and 3). Bearing this in mind, the subjects were asked about their user experience related to the ranking (Figure 4) and map (Figure 5) visualization interfaces.

On the one hand, in the ranking view (Figure 4), we let the users first assess the use of the "Like" and "Dislike" buttons. 86% of participants answered that their functionality is intuitive and also, it is a good way to give feedback to the application. While 11% preferred to have only the "Like" button and 3% preferred to have only the "Dislike" button.

In regard to use the map category icons, 86% of users supported that is a good way to know quickly which kind of cuisine has every place, making easy to find the place in the map view. 9% preferred the use of text format instead of icons and 5% did not like to know about the cuisine information in this view.

Furthermore, the users were asked about the information provided in every recommended item in the ranking view. 60% liked the current way this is achieved. However, 14%

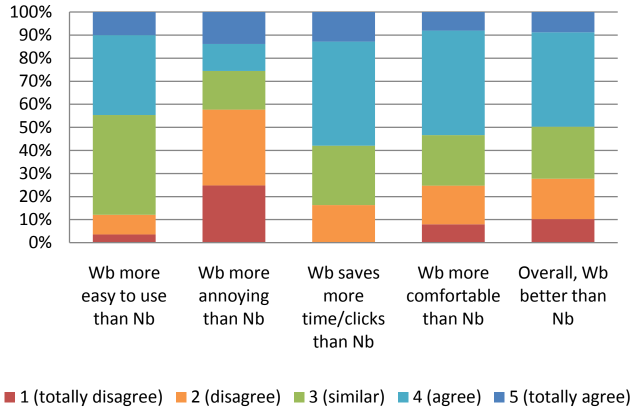


Figure 8: Comparison between Widget-based (Wb) and Notification-based (Nb) solutions.

thought that it was too much and 26% thought that is was too few.

Finally, the test users were asked to judge some properties related to the solution provided using the same 5-point scale used in Figures 6 and 7. Between 40% and 58% of users selected the 4 (good) mark, while almost all participants chose a value between 3 (normal) and 5 (very good).

On the other hand, and related to the map view (Figure 5), 67% of subjects thought that it is a perfect way to understand easily where the places shown by the ranking view are and also, consult their details by just clicking on them. 23% considered it good, but too simple and 10% considered it poor, needing a redesign to improve it.

To conclude, the test users were asked to judge some properties related to it using the 5-point scale used previously. Again, almost all the users chose a mark between 3 (normal) and 5 (very good), being specially concentrated in the 4 (good) qualification with a minimum of 40% of the users for the "user friendly" property and a maximum of 56% the users for the "useful" property).

6. DISCUSSION

6.1 Proactivity

Starting with the results related to the two proactive mobile user interfaces proposed, first of all we are going to analyze the users' responses in the different scenarios. Tables 2 and 3 show that in the scenario 1 the majority of the users (71% in the worst case) would have interacted with the system (i.e. they chose the Not now or Expand response). Although not everyone wanted to know the recommendation available, at least they gave temporal feedback with the "Not now" button. It is an important factor to allow the system to learn about user's habits and improve the proactivity property. In the scenario 2 the ignore rate increases compare to the scenario 1. This confirms the intuitive hypothesis that users ignore the application when they do not need a recommendation.

In the scenario 3, we found that when users were not in a hurry, the majority of them consulted the recommendation (between 54% and 65% of the users). On the contrary, when the users were in a hurry (scenario 4) the majority (50-65%) of the users decided to ignore the application without significant differences between notification and widget results. As a conclusion we can state that the "time pressure" factor is a good indicator to know when a proactive recommendation is reasonable or not, because in these situations users give less feedback.

In the scenarios 5 and 6 the users mainly (between 82% and 91%) checked the recommendation available without differences if the scenario is related to a tourist or a business situation. Therefore, we can state that both proactive recommender methods work in the same way with typical scenarios already tested in traditional non-proactive recommender systems.

Finally, the most important outcome of our research is that the widget-based solution is considered a better way to achieve proactivity compared to the notification-based solution. We are going to explain this result in more detail.

First of all, Figure 6 shows the first indication based on the different results gave by the α and β groups. As can be seen, the users that evaluated first the widget-based solution (group β) gave a lower mark in average to the notification method compare with the users that evaluated it first (α group). This demonstrates that when the users evaluated the notification method after having evaluated the widget method, their perception about the notification solution was worse. In addition, if we analyze Figure 7, the results are similar: the α subjects scored with higher marks the widget-based solution in the "effective", "useful" and "user friendly" properties, being lower in the "convenient" and "desirable" properties.

Secondly, if we study now the results depicted by Figure 8, we can see that when both solutions are compared, the widget method has a better qualification in all the statements that were evaluated. Therefore, this outcome supports the statement we presented previously. To conclude this point, there is another unequivocally result which confirms that the users considered the widget-based a better solution to achieve proactivity compared to the notification-based: the users agree (41%) or totally agree (9%) with the statement *"Overall, the Wb is better than the Nb"*, whereas only 18% disagree and 10% totally disagree.

This statistical outcome can be also supported if we pay attention to some of the comments the test users wrote in the survey during the evaluation process. For example: *"I personally prefer the widget application because I don't like apps that put themselves in the notification area."*, *"The crucial difference between the two interfaces is that a notification can be more annoying."*, *"Notification-based is more proactive but maybe more annoying."* or *"I prefer looking for information when I need it than receive notifications."*. As we can see, despite the notification-based solution is considered a good method to achieve proactivity (as we have demonstrated by comparing the Tables 2 and 3), the problem of being a more annoying solution was crucial when the users had to choose between one of them.

6.2 Recommendations Visualization

Analyzing first the results achieved related to the ranking visualization, we can see that the majority of the users were mainly satisfied with the way the recommended places are presented. The use of the "Like" and "Dislike" buttons, a well know paradigm of user experience in the Social Web, was easy to understand by the users. Besides, their functionality related to a recommender system was also understood intuitively by them. In this way, some of the users' comments emphasized these kinds of social features as an important factor to decide which place to go. Thus, this "social" feature could be integrated as a part of the proactivity model (e.g. ratings of other users can be considered when calculating scores). Specially, one participant commented: *"it would be interesting to have some information about other users' opinions, for example a karma mechanism"*.

Related to the use of the map category icons and its relation with the map visualization, the majority of the users supported this solution, but some of them said: *"I would like to know what the rank of each place in the map view is. Perhaps, using a color scale, different sizes or attaching a number to the icon could be good solutions"*. This is a good feature that we will take into account for future work. Another significant outcome achieved related to the map visualization is that despite the users considered it a useful way of visualizing recommendations, some of them recommend us to work on a more sophisticated interface with features like the previous one described.

Last but not least, a great majority of test users liked the application look and feel. As Miller said in [9], design can play a critical, and even primary, role in determining which products stand out. Therefore, in our case it is an important factor to avoid losing users after a first contact with the application, because we need from them a long experience with it in order to enhance the proactivity property and also learning their tastes to personalize the recommendations as much as possible.

7. CONCLUSION AND FUTURE WORK

In this paper, we have presented two mobile user interfaces developed for achieving proactivity in context-aware recommender systems. We have shown an innovative way of doing this using current technologies (i.e. Android). The findings of the evaluation carried out have demonstrated that the widget-based solution is better than the notification-based solution. Despite the fact that both options are considered good solutions to achieve proactivity, the second one is considered by the users more annoying. In addition, the results related to the user experience suggest that the methods chosen to visualize the results based on a ranking and a map view are useful and efficient for the users, specially due to its usability and the integration of Social Web paradigms of giving feedback via the "Like" and "Dislike" buttons that are very intuitive for them.

Current and future work includes the implementation of the complete system, and a field study to evaluate it with users really interacting with a mobile device in a realistic scenario in order to have a better feedback related to use this kind of proactive systems in daily life.

In this way, an option is to implement the application on more mobile platforms. During the elaboration of this research, Apple has announced for the next iOS version the addition of notifications and widgets. This would allow us to test our research not only in Android devices, but also in iPhone, covering this way the most extended smartphone platforms.

In regard to enhance the proactivity in our system, we have observed that trying to infer the current activity of the user (e.g. when a user is in a hurry) could enhance the proactive recommender system utilizing this information as user context of phase I for the model described in Section 3. To do so, we could use for example information from sensor data (such as GPS or acceleration) or from the handover among network point access to try to determinate what the user is doing right now. A user study could also investigate resource consumption (e.g. battery power) of the user observation.

An additional open issue to study is the usage of a color encoding or numbers in the map icons to represent the ranking order. But also it is important to try to refine the way we achieve the proactivity through the user interface.

Finally, another interesting topic to research is how to integrate recommendations of different items types (e.g. restaurants and shops during a tourist itinerary). This is very important in a real commercial exploitation of a recommender system since all the data used could be obtained from differences sources (e.g. banking data, tourist companies, etc.).

8. REFERENCES

[1] Android. Developers: Service. In *http://developer.android.com/reference/android/app/Service.html*, 2011.

[2] Apple. Developers: ios 5. In *http://developer.apple.com/technologies/ios5/*, 2011.

[3] O. Averjanova, F. Ricci, and Q. Nguyen. Map-based interaction with a conversational mobile recommender system. In *Mobile Ubiquitous Computing, Systems, Services and Technologies, 2008, UBICOMM '08*, pages 212 –218, 29 oct. 2008.

[4] M. Baldauf, S. Dustdar, and F. Rosenberg. A survey on context-aware systems. *Int. J. Ad Hoc Ubiquitous Comput.*, 2:263–277, June 2007.

[5] A. K. Dey, G. D. Abowd, and D. Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Hum.-Comput. Interact.*, 16:97–166, December 2001.

[6] J. Fogarty, S. E. Hudson, C. G. Atkeson, D. Avrahami, J. Forlizzi, S. Kiesler, J. C. Lee, and J. Yang. Predicting human interruptibility with sensors. *ACM Trans. Comput.-Hum. Interact.*, 12:119–146, March 2005.

[7] J. Hong, E.-H. Suh, J. Kim, and S. Kim. Context-aware system for proactive personalized service based on context history. *Expert Syst. Appl.*, 36:7448–7457, May 2009.

[8] M. Kenteris, D. Gavalas, and D. Economou. Electronic mobile guides: a survey. *Personal Ubiquitous Comput.*, 15:97–111, January 2011.

[9] J. Miller. The user experience [internet]. *Internet Computing, IEEE*, 9(5):90 – 92, 2005.

[10] F. Ricci. Mobile recommender systems. *International Journal of Information Technology and Tourism*, 12:205–231, 2011.

[11] S. Rosenthal, A. K. Dey, and M. M. Veloso. Using decision-theoretic experience sampling to build personalized mobile phone interruption models. In *Pervasive 2011, Springer LNCS 6696*, pages 170–187, 2011.

[12] W. Woerndl, J. Huebner, R. Bader, and D. Gallego-Vico. A model for proactivity in mobile, context-aware recommender systems. In *5th ACM Conference on Recommender Systems*, 23-27 oct. 2011.