Context based adaptation of application icons in mobile computing devices

Rahul Jain, Joy Bose, Tasleem Arif
WMG Group
Samsung R&D Institute
Bangalore, India
{rahul.j, joy.bose, tasleem.arif}@samsung.com

***Abstract*— Finding the desired application among a huge number of installed applications in a mobile computing device, like a smartphone or tablet, is problematic for some users. In this paper we propose a novel method to adapt the display of application icons in mobile devices based on context. In our method, the icons that are predicted to be used are displayed prominently. The prediction is based on the current application context being invoked, incorporating learning from past user actions associated with the current application. The system predicts the next applications, whose icons are displayed in a way to attract the user's attention and thus reduce the access time. This allows the user to perform predicted actions in minimal steps and also reduce the clutter on the screen. We use machine learning algorithms to predict the best possible screen configuration with respect to the context or scenario in which user is present, and can be generalized for any user interface (UI) related aspect rather than just the positioning of icons. We study the average response times for a number of users to access certain applications randomly on a phone, and on that basis predict the utility in terms of time saved by adapting the UI in this way. Our approach assists the users to more quickly access the desired application.**

Keywords- adaptive user interface; application icons positioning; application usage

I. INTRODUCTION

The number of applications installed on portable user devices, such as smartphones and tablets, is increasing day by day. A 2012 study found that on average, users have 41 applications installed on their smartphone [1]. Finding the application of interest out of all the installed applications is cumbersome and takes time as the applications are displayed alphabetically. In some operating systems such as Android, the user has the ability to search particular applications by typing the name in a search box. Also, certain third party tools are available to automatically categorize and group the application icons based on the type of the application, such as game or productivity related applications. However, even in such systems, the user would need several steps to navigate to the application icon of their choice. According to research by Weidenbeck [2], users are more comfortable finding the applications by icons rather than only labels, so categorizing the applications on the device screen based only on labels or alphabets does not provide great user experience. In an ideal scenario, the user should be able to see those applications on their device screen which they want to access in any particular scenario or context.

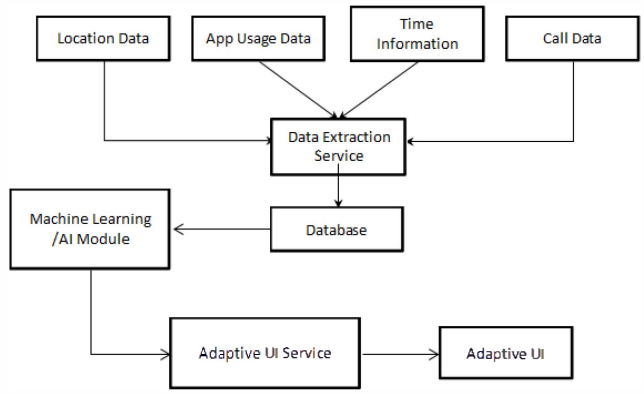

Figure 1. Diagram of a system for a contextual adaptive user interface on a mobile computing device

In this paper, we propose a framework to solve the above problem and provide the best user experience. The framework uses machine learning to learn patterns of user behavior when accessing different applications on mobile devices, including system applications such as calling or messaging. This is then used to make the most likely applications for any given context more accessible to the user, thus saving them time and enhancing the user experience.

The rest of the paper is organized as follows: Section 2 elaborates the different use cases in which an adaptive user interface might be useful. Section 3 surveys related work in the area. Section 4 elaborates the different components of the proposed framework. Section 5 presents possible adaptations to the user interface, section 6 presents the results of a study on response times by users while accessing different applications on a mobile phone. Section 7 concludes the paper.

II. PROBLEM STATEMENT AND USE CASES

Although the usage pattern of mobile devices vary from user to user, we can identify a few factors using which the User Interface (UI) of a typical mobile device might be adapted.

By UI, we primarily refer to the relative positioning and size of different application icons. However this could also mean the content and positioning of menu items, with individual menu items being listed in order of relevance to

the user as per the factors mentioned. Other UI elements such as buttons, pointers, color and resolution settings, and window positioning can also be similarly modified.

The factors using which the UI can be adapted are listed below:

A. *Adapting the UI based on the location of the user*

For example the type of applications a user is expected to access when at home is different from the type of applications accessed when the user is at work. Also, when the user is on the move, in a car or using public transport, they may access different types of applications than when the user is stationary.

B. *Adapting the UI based on the identity or category of a caller during a call or chat conversation, or when the user is composing or reading a text message or email*

Some of the caller categories might be colleagues, family and friends. In case of a call or email or message coming from a colleague at work, one may need to access certain kind of applications during the conversation, e.g., productivity applications. In case of calls or texts from friends, one may need to access other applications like maps or calendar. Hence it would be useful if there is a way to modify what applications appear on the home screen and other similar elements of the user interface based on the identity of the caller.

One issue in categorizing contacts is that the goal of the user when having the conversation can be different and hence, the UI adaptations will not be appropriate. For example, a user can have a friendly or social chat with a work colleague. However, in such cases the user is free to ignore the suggested application icons or other modified UI elements, and manually invoke whichever application he/she seems suitable for the occasion.

C. *Adapting the UI based on the day and time.*

The type of applications accessed on weekdays might be different from the applications accessed on a weekend or on holidays. Similarly, the type of applications accessed in the morning (such as news) might be different from the applications accessed in the evening, and so on. Also, the UI could be adapted based on the calendar and appointments of the user for that day or time. If the user is supposed to be in a business meeting, different modifications could be invoked than if they were in a meeting with friends.

It should be noted here that the UI customizations mentioned are specific for the individual user and based on learning from their specific usage patterns. For example, if a user regularly works on weekends, their UI adaptations would take this into account. However, some general rules can be initially applied that are valid for most users and if the individual user's patterns vary significantly then those settings would override the general settings.

D. *Adapting the UI based on user activity and environmental variables.*

This could use data from different sensors available on the device, such as pressure or humidity sensors, and motion detecting orientation or accelerometer sensors, to decide on the user activity or preference and adapt the UI accordingly.

The above knowledge is obtained from the sensors based on the user's device and other sources including the cloud server. However, in case one or more of the mentioned sensors are not available on the device or are not enabled, the UI will be adapted taking into consideration only those sensors that are actually available and enabled. Also, it is possible for the user to manually state the required information rather than relying on sensors. In this case, an appropriate interface will be provided to enable the user to configure and manage all the settings such as their location etc. Also, when part of the user's information resides in an external cloud server, appropriate measures, including encryption techniques need to be enabled to ensure the security and privacy for the user.

At the moment, the mobile device user interfaces available commercially do not incorporate learning from past user actions, nor (with few exceptions), do they incorporate contextual information, such as location and time, while deciding the UI to be displayed. It would be much more convenient for the user if the interface could learn and thus automate certain actions repeated frequently. An ideal user interface would be able to incorporate knowledge from the past user actions and on that basis predict what actions the user is expected to do or what applications they are expected to invoke. This prediction can be used to save the user time by pre-emptively accomplishing the task (such as invoking an application or opening a website) or prominently displaying the application icons on the home screen.

In this paper we limit ourselves to application icons only. This approach, of only changing the selection and positioning of icons and other UI elements such as menus adaptively, is less invasive and annoying to the user. More invasive approaches, such as pre-emptively invoking the predicted applications, might cause annoyance and would be counter-productive to our goal of improving the overall user experience.

In the next section, we analyze some related work in the area of adaptive user interfaces.

III. RELATED WORK

There are certain efforts described in the prior state-of-the-art literature where adaptive user interfaces have been proposed [3]. A number of patents are also available regarding the same efforts [4-6]. Kamisaka [7] performed a feasibility study for context aware interfaces for mobile phones and found that it is feasible to have machine learning to optimize the interface.

Another related research area is the prediction of patterns of smartphone app usage by analyzing data from multiple users. Xu et al [8] developed a smartphone app usage prediction model, tested it on 35 users over a period of time and used it to optimize the smartphone app responsiveness, particularly in preloading and network prefetching. Tan [9] developed a prediction algorithm for predicting mobile usage patterns of users. Bohmer and Kruger [10] have studied and proposed optimal modes of arrangement of icons on a smartphone based on a number of factors. Another study by

Bohmer and Bauer [11] investigated how the users' manual positioning of icons on a grid based arrangement on a smartphone can provide information about the relevance of the respective applications for the user, with the user positioning more relevant application icons in more prominent positions.

In the above studies, the focus has been on predicting which applications are most used by collating data from multiple users, and using this to optimize the performance of such applications. Our focus in this paper differs in two ways. First, we focus on improving the user interface by positioning the icons, menus and other user interface elements adaptively, thus improving the user experience and reducing the time taken by the user to launch applications. Second, we incorporate learning from the data related to individual users, rather than collating data and generalizing for a group of users. Our proposed model is more local and the UI adaptations specific for each user.

Fukuzawa et al [12] have developed a system to customize menus in mobile phones by ranking the applications by usage; taking as input the operation history of the user and using a support vector machine (SVM) to perform the ranking. Applications are ranked by how recently the application was invoked and the frequency of usage of the application. This ranking is then used to adapt the menus, giving most prominence to the frequently used functions by varying the button size, depth, menu positioning etc. They also performed a usability study. Our goal in this paper is somewhat similar but our scope is more general. We propose to discover rules on which application is invoked in response to which context parameters, and use these rules to adapt the user interface elements including menus, icons and buttons in a variety of ways.

Other researchers including Caminero [17], Stephanidis [18], Verpoorten [19], Balme [20], Clerckx [21], Smith [22], Michalac [23], Hartmann [24] and Zou [25] have also proposed other generic approaches to context sensitive user interfaces. In this paper, our focus is more based on adapting the interfaces present in mobile devices. Moreover, our modifications are specific for the user of the device and not general for all users. In our model, the learning algorithm, whether run on the device or the cloud server, takes input pertaining to specific users while deciding the UI adaptations for those users.

Out of the above, the approach by Zou et al [25] speaks of predicting the next applications the user is expected to use, based on the user's latest used apps. The predicted application icons are then placed on the top or bottom of the screen to enable the user to access them readily. They use simple Bayesian models for learning. In our approach, we do not limit our adaptations to application icons but also consider other interfaces including menus and buttons. Also, the way we modify the application icons to make them easier for the user to access is more extensive, and we use more contextual factors including time and sensor readings of the user's location to make more relevant predictions.

In the following sections, we look at the components of our solution in more detail.

IV. COMPONENTS OF THE PREDICTIVE USER INTERFACE

The aim of our proposed adaptive UI system is to modify the user interface during different actions of the user, as well as for different contexts and parameters. An example is modifying the icons displayed on the screen during an incoming or outgoing call depending on the identity of the caller or the person being called, as well as other related parameters. As mentioned earlier, in this paper we concentrate on positioning of icons, menus and other UI elements, although the same concept might be used with other user interactions such as browsing and application usage.

In practice, when the user is involved in a call, SMS or chat conversation from a colleague in the office, he or she typically would invoke different kinds of applications than say if the call is from a family member or a friend. An adaptable user interface that learns from past user action when calling the same person or someone in a similar group of persons would be useful. However, it may be the case that the same person could be a friend as well as a work colleague. Our system learns and applies the user interface adaptations based on the particular user and so this information would be reflected in the type of modifications chosen. Besides, other factors including time of day or day of week, and location of the users (home or office) can be used to make more specific UI customizations, even when the same user is involved.

Figure 1 shows the block diagram for the system, showing the relation between the different modules/ components of the proposed framework. It is desirable to implement all the components of the system at the device level. However, parts of the system could be implemented on a remote cloud server instead. This would have the advantage of incorporating learning from multiple data sources, for example if the user has more than one device, and also incorporate knowledge from multiple users when deciding on the user interface adaptations.

The components of our system are described in detail in the following subsections.

A. Data extraction service

This service extracts multiple modes of contextual user data including location data, application usage data, call data and time information, as well as input from various sensors on the device. The extracted data can be in XML or some similar format.

B. Database

The user data collected by the data extraction service is stored in the database. This can be local on the device, or as mentioned previously, this could be on a remote cloud server, which would combine the data from different devices owned by the same user. In case the database is on a server, the XML files from the devices would have to be uploaded to the server periodically.

Figure 2. User interface modified according to the identity of the caller. The modifications are learnt as per the past user actions when the same caller's incoming phone call was received

C. *Machine learning module*

This module uses appropriate machine learning algorithms to learn from past user actions regarding user interaction with the device and input from various sensors.

The task of the machine learning module is to learn from patterns of user behavior and make rules by mapping different patterns of user behavior and interaction with the applications on the device.

The rules are in the format of contextual factor →user interface adaptation or application invoked in response to the contextual factor. Some example rules are as below:

- <Caller id> → <Application invoked during the call>
- <Caller id> → <Application invoked just after the call within a threshold of time T>
- <Id of person being called> → <Application invoked during the call>
- <Time of day, Caller Id> → <Application invoked>
- <Location of user> → <Application invoked>
- <Application being invoked currently> → <Next application predicted to be invoked>

Our model would involve taking a weighted average of all the above rules to predict the most UI appropriate modification. Initially, each of the above rules can be given equal weight, however the user can change the weight of the rules individually if they so wish.

Our focus in this is to predict the user's next action and position the user interface icons in the most optimal way so as to be convenient for the user. The learning and formation of rules, can take place either locally on the device, or on a remote cloud server.

The learning mechanism uses supervised learning algorithms. The past data on which application is invoked in response to user inputs is used for training, and the model learns to predict future applications based on the training data. Support Vector Machines (SVMs), Hidden Markov Models (HMMs) and multilayered perceptrons or RBF Neural Networks are suitable for this purpose. Existing open source tools or libraries, such as ghmm for HMMs [13] or SVMTool for SVMs [14] can also be used for the purpose. In all the cases, regardless of the learning algorithm used, the model is trained from some past user behavior data collected by the data extraction service running on the mobile device.

One issue while deciding which model to use for learning is that the learning algorithm should be able to run in the limited processing resources (including power, memory, processor speed) available on the mobile device. There is no such restriction if the learning takes place on a server.

D. *Adaptive user interface*

The output of the machine learning module is sent to the adaptive UI interface which uses it to adapt the user interface dynamically based on the context and user actions. The main task of the smart user interface engine is to modify the user interface as per the learning rules output from the learning module. This modification can take various forms depending on the user interface element.

In the following section, we look at some of the ways in which the user interface can be modified.

V. USER INTERFACE ADAPTATIONS

There are multiple ways of modifying the user interface elements in a mobile device in response to predicted application usage. In the following subsections we study some of these methods.

A. *Selectively displaying application icons*

In this approach the number of application icons to be displayed on the mobile device screen is reduced, and only the icons of applications predicted to be invoked are displayed.

For example, if after a call to a business contact Y, the user is likely to invoke an office application such as a spreadsheet or word processor, only shortcuts to those applications and no other would appear on the screen as soon as the call is completed.

The advantage of this approach is to reduce clutter and make it easier for the user to access their applications of choice. The disadvantage is that the user might be annoyed if a prediction goes wrong and the application of the user's choice is not displayed.

B. *Changing positioning, sizes and fonts of icons*

In this approach, the application icons that are predicted to be invoked are positioned on the present screen of the user in a more prominent way, depending on the screen size. The increased prominence would make it easier for the user to invoke the applications of choice. Alternatively, the same logic could be used to give decreased prominence to application icons not expected to be used, such as smaller icon sizes or fonts, or just listing them separately in a corner.

In this case for example, after a call to a business contact the shortcuts to office applications (such as spreadsheet) would appear in the center or with increased prominence, while the other application icons would be smaller.

Figure 3. Example of a user interface adaptation : highlighting the maps icon by (a) increasing its size relative to other icons and (b) positioning the icon at the centre of the screen and decreasing the relative size of other icons. In this case the user is expected to use the maps icon next as per the prediction of the model

The advantage of this approach is that even if the prediction goes wrong, the user is able to manually select the icon they would like to invoke. This would also enable the optimal use of the limited screen space.

The changing icon positioning to give increased prominence to predicted application icons can be accomplished by a number of ways (as illustrated in figure 3):

- Positioning the predicted icons at the center of the screen
- Increasing the size of predicted application icons
- Highlighting the predicted icons by special effects or animation, such as flashing them or adding glowing edges to the icons
- Having special gestures for the user (such as swipe from the top right edge of the screen) to automatically invoke the predicted applications for the particular screen.

C. *Overlaying the predicted icons on the current application screen*

In this approach the icons for the predicted application are superimposed on the current screen of the user. In this approach, the previous adaptations are combined in a limited but controlled way.

If, for example, the system determines that the user is most likely to invoke a browser application during or after a call is made to a contact person X, the shortcuts to the browser application can be kept on the screen during the call being made to X. Figure 3 illustrates how the modified adaptive user interface looks like during an incoming phone call.

This method has the advantage of saving the user time, since they do not have to leave the current screen to access the applications predicted to be invoked. But it may not be applicable in cases where the new application invocation takes place after the previous application has completed.

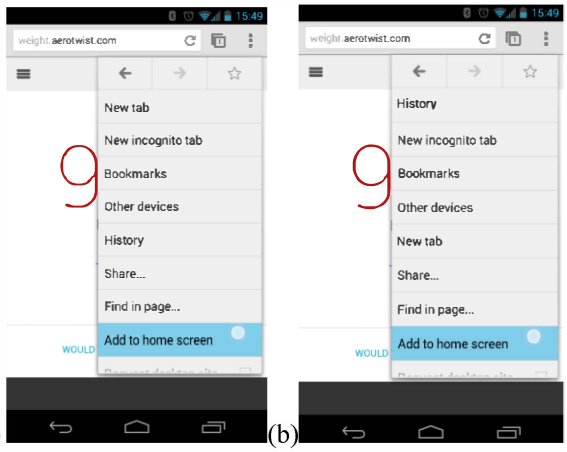

Figure 4. Example of a menu adaptation by changing ordering of menu items as per context (a) normal menu and (b) context sensitive menu, where the history appears at top

D. *Changing the ordering of items in the menus and other UI elements based on context*

The menus, as well as other UI elements such as buttons on the browser or other applications, can also be adapted based on which of the items the user is expected to click next based on the current context. The menu items are arranged in order of likelihood of the user selecting them next, with the most likely elements being on top. The other menu items are kept hidden and accessible with an arrow key.

For example, if the user is more likely to open the history while using a browser, then in the menu that pops up on doing a long press, the history comes at the top as seen in figure 4(b), as opposed to the open new tab that comes in a normal menu of figure 4(a).

This approach can save space on the device screen and lead to shorter and more meaningful menus. Of course, the user always has the option to invoke the full menus by clicking the expand button or similar interfaces.

All the above methods can also be used in combination to provide optimal results and decrease the time taken for the user to access the application of their choice. Also, initially the user interface is kept as default, over time the adaptations become visible to the user as they become familiar and more intuitive. This would partially solve the problem of the user being unfamiliar with the adaptations.

In the next section we present the results of a small study to analyze application access times and thus have some idea of the time saved in accessing such applications.

VI. ANALYSIS OF ACCESS TIMES

We performed a study of access times of a number of users to invoke a random application installed on a mobile device. For this study, we gave a mobile device with 50 Android applications installed to a number of healthy male participants from the 20-30 age group who are already own and use mobile phones, asked them to invoke some specific applications from different categories, and logged the time it took to click the correct icon and invoke the application. The

users always started from the home screen, and had to navigate to the correct screen as well as access the desired application.

To reduce bias, the users were given a phone they were not previously familiar with, having a standard Android interface with the latest Android OS, some common applications pre-installed and the application icons arranged in alphabetical order. Since the users would be more familiar with the icons after they knew the positioning the first time, we used this method to reduce skewing of the time data.

The mobile device was also equipped with a voice and text interface for searching applications, but we prohibited the user from using those in order to study the invoking times for application icons in the default neutral case.

TABLE I. TIME (IN SECONDS) TO ACCESS APPLICATION ICONS BY MULTIPLE USERS

App name	User 1	User 2	User 3	User 4	Average
Temple Run	3	6.7	4.49	2.64	4.2
Play music	5	3.8	10.33	5.48	6.15
Maps	10	1.5	12.21	7.28	7.75
Whatsapp	4.47	2.36	7.76	2.59	4.29
Chrome	2.5	3.09	1.67	2.15	2.35

The applications the users were asked to find and invoke are Temple run, Play music, maps, Whatsapp, and Chrome, in random order. We measured the time taken to access each of the applications with a stopwatch.

The results are shown in Table 1. While there is a wide variation of the times taken to access and invoke the correct application icons, the average times (average of all applications 4.9 seconds) are substantially higher than the minimum time it took to invoke a single application (1.5 seconds). The average human response time is around 200 milliseconds [15-16] which is an order of magnitude lower than the times measured in our experiment, and can thus be ignored while measuring the times taken to access the applications. This shows there is good scope to enable the users to save substantial time if the icons are positioned or shown prominently as per the predicted usage for that user.

VII. FUTURE WORK AND CONCLUSION

Our proposed system to develop an adaptive user interface is still at a developmental stage and we are trying to develop a prototype for the same. Once it is completed, in future we hope to implement this system on mobile devices and extend the system for other kinds of adaptations rather than limiting it to application icons only. We also hope to perform a more meaningful user evaluation, improve the prediction system to give more relevant predictions as per the current context of use.

Addition of the feature to customize the user interface or applications based on the context would reduce clutter on the screen, increase the user satisfaction and save time.

REFERENCES

[1] http://thenextweb.com/insider/2012/05/16/nielsen-us-smartphones-have-an-average-of-41-apps-installed-up-from-32-last-year

[2] S. Weidenbeck. The use of icons and labels in an end user application program: an empirical study of learning and retention. Behavior and Information technology, 1998, Vol 18 No 2

[3] http://en.wikipedia.org/wiki/Context-sensitive_user_interface

[4] Telephone answering machine with caller ID dependent message operation. European Patent EP0844773A2

[5] System and methods for selecting advertisements based on caller identifier. US Patent US20090043657

[6] Method and apparatus for content personalization over a telephone interface. US Patent US6807574

[7] Kamisaka et al, “Operation Prediction for Context-Aware User Interfaces of Mobile Phones”, Proc. Ninth Annual International Symposium on Applications and the Internet, 2009.

[8] Ye Xu et al, “Preference, Context and Communities: A Multi-faceted approach to Predicting Smartphone App Usage Patterns”, Proc. ISWC 2013, Zurich, Switzerland, September 2013

[9] Chang Tan, Qi Liu, Enhong Chen, Hui Xiong, “Prediction for Mobile Application Usage Patterns”, Nokia Mobile Data Challenge Workshop 2012.

[10] Matthias Böhmer, Antonio Krüger, “A Study on Icon Arrangement by Smartphone Users”, Proc. ACM SIGCHI Conference on Human Factors in Computing Systems. Paris, France, 2013

[11] Matthias Böhmer and Gernot Bauer, ”Exploiting the icon arrangement on mobile devices as information source for context-awareness”, Proc. MobileHCI’10, 2010

[12] Yusuke Fukazawa, Mirai Hara, Masashi Onogi and Hidetoshi Ueno, “Automatic mobile menu customization based on user operation history”, Proc. MobileHCI’09, 2009

[13] Ghmm : www.ghmm.org

[14] Svmtool : www.lsi.upc.edu/~nlp/SVMTool/

[15] http://en.wikipedia.org/wiki/Mental_chronometry

[16] http://www.humanbenchmark.com/tests/reactiontime/stats.php

[17] J. Caminero, J. Vanderdonckt, F. Paterno et al, "The SERENOA Project: Multidimensional Context-Aware Adaptation of Service Front-Ends", Proc. LREC'12, 23-25 May 2012, Istanbul.

[18] C. Stephanidis, A. Paramythis, M. Sfyrakis et al, "Adaptable and Adaptive User Interfaces for Disabled Users in the AVANTI Project", Proc. 5th International Conference on Intelligence and Services in Networks: Technology for Ubiquitous Telecom Services, 1998.

[19] K. Verpoorten, K. Luyten, and K. Coninx, "Task-based prediction of interaction patterns for ambient intelligence environments", Proc. HCI, 2007.

[20] L. Balme, A. Demeure, N. Barralon, J. Coutaz and G. Calvary, "CAMELEON-RT: A Software Architecture Reference Model for Distributed, Migratable, and Plastic User Interfaces", Proc. EUSAI 2004.

[21] T. Clerckx, C. Vandervelpen, K. Luyten and K. Coninx, "A TaskDriven User Interface Architecture for Ambient Intelligence Environments", Proc. IUI, 2006.

[22] D. A. Smith and H. Lieberman, "The Why UI: Using Goal Networks to Improve User Interfaces", Proc. IUI, 2010.

[23] P. Michalak and J. Allen, "Improving User Taught Task Models", Proc. UM 2007.

[24] M. Hartmann and D. Schreiber, "Prediction Algorithms for User Actions", Proc. ABIS, 2007.

[25] X. Zou, W. Zhang, S. Li and G. Pan, "Prophet: What App You Wish to Use Next" Proc. Ubicomp 2013, Zurich, Switzerland.