Contextual Adaptive User Interface For Android Devices

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

Abstract— In this paper we propose a framework to adapt the user interface (UI) of mobile computing devices like smartphones or tablets, based on the context or scenario in which user is present, and incorporating learning from past user actions. This will allow the user to perform actions in minimal steps and also reduce the clutter. The user interface in question can include application icons, menus, buttons window positioning or layout, color scheme and so on. The framework profiles the user device usage pattern and uses machine learning algorithms to predict the best possible screen configuration with respect to the user context. The prediction will improve with time and will provide best user experience possible to the user. To predict the utility of our model, we measure average response times for a number of users to access certain applications randomly on a smartphone, and on that basis predict time saved by adapting the UI in this way.

Keywords—adaptive user interface; mobile phones; application usage

I. INTRODUCTION

The average user has a number of apps installed on their smartphone [1]. Finding the application of interest currently is cumbersome and takes time as the applications are either categorized alphabetically or based on the type of the application. According to research by Weidenbeck [2], some users prefer to find their desired apps by looking at icons rather than search the labels by name. Therefore, the search by name option currently available on some mobile devices is not much use. In an ideal scenario, the user should be able to see only the specific apps on their screen which they actually want to access. Currently, such a system is not available.

In this paper, we propose a framework to improve the user experience by providing context specific modifications to the user interface (UI). The framework uses machine learning to learn the usage patterns depending on the context, when accessing different applications. This is then used to make the predicted applications or elements of the user interface more prominent and accessible, thus saving time and being more intuitive.

Figure 1 illustrates a system where the UI is modified to display context specific application icons while using a call application.

II. CONTEXTUAL FACTORS TO ADAPT THE USER INTERFACE

As mentioned, we are proposing to modify the user interface of a smartphone in response to certain contextual factors. In this section we look at some of these factors.

- Device sensor readings: Output of other sensors on the device including the ambient light sensor (to infer whether the user is indoors or outdoors), accelerometer and gyroscope (to say if the user is stationary or moving) can also be used to derive
additional contextual information in order to better predict the user’s chosen application and modify the UI appropriately.

- **User location**: One of the factors to adapt the UI is the location of the user. This is based on the premise that 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. The location is determined by means of the GPS sensor on the mobile device.

- **Duration of contact**: Based on how long a call takes, the apps accessed by the user may vary. A long call or chat conversation might require extensive use of the browser application, while a call of short duration might only require the appointments. The mobile device can access the call, text or chat logs to find out the average duration of the contact and map it with the applications accessed.

- **Category of the person being contacted**: In a mobile user’s contact list, the contacts may be of different categories, including work colleagues, family, friends and casual acquaintances. In case of a call coming from a colleague at work, one may need to access certain kind of apps during the call e.g. productivity apps. In case of calls from friends, one may need to access other apps like maps or calendar. The category can be specified by the user, or can itself be inferred by the device from other factors including user location, time of contact, duration of call and so on.

- **Day and time**: The type of applications accessed on weekdays might be different from the applications accessed on a weekend or on holidays. Similarly, in the morning the user may access different apps than the ones they do at night. A logging service running in the device would have to log the types of apps accessed at specific times of day or day of the week, and use it to make the appropriate UI modifications.

- **Application usage logs**: Logs of the past application usage, the frequency at which the particular app was accessed and the user actions and interactions while using the app can act as another source of contextual information.

This paper focuses on predicting the applications or actions the user is expected to perform and modifying the UI accordingly to make it easier for the user to perform the suggested tasks. Moreover, at any time the user is free to disregard the suggestions and choose their own tasks, which would be incorporated the next time a similar context arose. We do not invoke the suggested applications or actions automatically, therefore this approach is less invasive. More invasive approaches, such as pre-emptively invoking the predicted applications, might cause annoyance and would be counter-productive to our goal of enhancing the user experience.

### III. RELATED WORK

A number of researchers have proposed UI adaptations in the prior art [2]. A number of related ideas have also been patented [3-6].

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 UI.

Xu et al [8] developed a model to predict the application usage in smartphones, 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.


In the above studies, the focus has been on predicting and generalizing the application usage patterns by combining data from multiple users, and using this to optimize the performance of such apps. In this paper, we focus on improving the user interface by context specific modifications in real time, thus reducing the time taken by the user to launch applications. Also, our model works on individual data, and the UI modifications are also customized for individual users. We do not generalize from data collected from a number of users. Our learning algorithm runs inside the mobile device, although a cloud server might be needed to combine data from different devices belonging to the same user.

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

### IV. COMPONENTS OF THE SOLUTION

The aim of the contextual adaptive user interface is to store the context of various user actions, predict the next user actions based on the context and on that basis to modify the user interface.

Figure 2 illustrates the architecture of the system and the relation between the different modules/ components of the proposed framework. The components include the following:

![Module diagram for the system with Adaptive User Interface](image)
• **Data extraction service:** This service runs in the background and extracts and logs various user data including location data, app usage data, call data and time information.

• **Database:** The database stores the contextual user data collected by the data extraction service. The database can reside wholly on the mobile device, or partly in the cloud.

• **Machine learning module:** The learning module uses data stored in the database and appropriate machine learning algorithms to derive rules associating the context with user actions regarding interaction with the device. The rules themselves are then stored in the database. The module also uses the stored rules to make real time predictions for future user actions based on the present context.

• **User interface adaptation module:** Based on the predictions of the machine learning module, this component adapts the user interface in real time to make it easier for the user to perform the predicted actions. This adaptation can take a number of forms, such as displaying the icons of the predicted actions prominently, changing the menus to list the predicted items at the top, changing the relative sizes individual UI elements such as buttons to prominently display the element which the user is expected to click. Some of the possible adaptations are discussed in more detail in a later section.

In the following section we look at some of the adaptations.

V. RULES DERIVED BY THE MACHINE LEARNING MODULE

As mentioned in the previous section, the machine learning module takes the data stored by the data extraction service and derives rules associating the specific context with user actions. The rules are then each given a weight, which itself changes with time. The weighted average of the rules is then used to predict the UI adaptation to be made.

The rules are in the format of context parameter associated with a certain user action or UI adaptation. Some example rules are given below:

<Currently running application> ➔ <Menu item predicted to be chosen>
<Currently running application> ➔ <User interface element predicted to be clicked>
<Caller or contact person id> ➔ <Application icon predicted to be invoked during the call or chat conversation>
<Caller or contact person id> ➔ <Application predicted to be invoked just after the call or chat within a threshold of time T>
<Id of person being called> ➔ <Application predicted to be invoked during the call>
<Time of day, Caller Id> ➔ <Application invoked>
<Location of user> ➔ <Application invoked>

Along with the database, the learning can also take place either locally on the device, or else on a remote cloud server. Each of these has its advantages. Having the database and algorithm running locally would lead to faster response times, especially if the network connection is slow. However, this might slow down the device at the same time, so a local storage and processing of learning rules is more appropriate for high end devices. Storing and processing the algorithms on a remote cloud server, on the other hand, has the advantage of incorporating data from various devices owned by the user, as well as freedom from constraints of memory and processing speed.

The learning algorithm can be chosen from any of the standard supervised learning algorithms including Support Vector Machines (SVMs), Hidden Markov Models (HMMs), Bayesian algorithms or Associative Memory Neural Networks. Existing open source tools or libraries, such as ghmm for HMMs [11] or SVMTool for SVMs [12] can 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.

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

VI. USER INTERFACE ADAPTATIONS

The user interface adaptations possible in response to the present context and applying the learned rules are many. In this section we look at some of the adaptations.

A. Changing ordering of the menu items

The menu item that is predicted to be chosen are ordered as per the probability of the prediction. The items most likely to be chosen are displayed at the top of the menu and the least likely ones at the bottom.

B. Prominently displaying predicted UI elements

The user interface elements (such as buttons) that are expected to be next chosen as per the user’s current context are displayed prominently or towards the center of the application, and the ones less likely to be chosen are temporarily hidden or displayed towards the end.

C. 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.

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.
D. Changing positioning 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, along with other icons on that screen. The increased prominence would make it easier for the user to invoke the applications of choice. 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.

The changing icon positioning to give increased prominence to predicted application icons can be accomplished by a number of ways:

- 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 to invoke the predicted applications for the particular screen

E. 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. For example, during a phone call, the icons to be invoked are displayed along with the call screen. Figure 2 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.

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.

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.

VII. 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 people and 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 and 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.

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). 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.

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

VIII. 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.

Addition of this feature of customizing the user interface or applications based on the user profile would reduce clutter on the screen and increase the user satisfaction with using the device and save the user time while accessing a specific application, leading to higher productivity.

REFERENCES


[12] Svmtool : wwwlsiupcedu/~nlp/SVMTool/