Optimizing User Interaction for Web-based Mobile Tasks

Dong Zhou, Ajay Chander
DOCOMO USA Labs
3240 Hillview Avenue
Palo Alto, CA 94304, USA
{zhou, chander}@docomolabs-usa.com

Hiroshi Inamura
Research Laboratories
NTT DOCOMO Inc.
inamurah@nttdocomo.co.jp

ABSTRACT

This paper describes the design and prototype implementation of MIntOS (Mobile Interaction Optimization System), a system for improving mobile interaction in web-based activities. MIntOS monitors users' interactions both for gathering interaction history and for the runtime construction of *interaction context*. A simple approach based on interaction burstiness is used to break interaction sequences into *Trails*, which approximates user tasks. Such Trails are used to generate rules for online, context-sensitive prediction of future interaction sequences. Predicted user interaction sequences are then optimized to reduce the amount of user input and user wait time using techniques such as interaction short-cuts, automatic text copying and form-filling, as well as page pre-fetching. Such optimized interaction sequences are, at real-time, recommended to the user through UI enhancements in a non-intrusive manner.

Categories and Subject Descriptors

H.4.3 [**Communications Applications**]: Information browsers.
H.5.4 [**Hypertext/Hypermedia**]: Navigation.

General Terms

Algorithms, Performance, Design, Experimentation

Keywords

Web-based mobile tasks, user interaction monitoring, interaction trail, trail optimization

1. INTRODUCTION

Small form-factor and weaker and less stable connection are the two inherent characteristics of cell phones that negatively affect their usability for web-based mobile tasks. Their small form-factor makes it much harder to interact with: the small screen makes it difficult to read what's displayed or to click/tap at the right places; the lack of full-sized keyboard and stable arm-support makes it difficult to input text; and the lack of multi-window and mechanisms for easy copy/paste makes it difficult to move data from one application to another. In addition, cellular network connections are still weaker and less stable than their fixed-line counterparts. The result of this is longer and less predictable network latency and user wait time, and consequently worse user experience, especially for web sessions involving sequences of user actions and Internet requests/responses.

As a result, an otherwise simple task for PC users, such as "read an article at site *A* then search for phrase *X* contained in the article at site *B*", becomes considerably more demanding on mobile devices. Copying several text phrases from one page to another is hard, if not impossible to accomplish. Even simple navigation can become intolerable because of constant waiting for page loading, if the user is at a place with less than ideal network coverage.

While there are existing research work and commercial applications that help improve the usability of such web-based mobile tasks, they are still limited in one or more of the following aspects to varying extent:

- Interaction Sequence: many of these existing approaches provide optimization for individual steps, rather than for a sequence of interaction steps. However, an approach that is good for an individual step may not be good for a series of interactions.
- Minimal User Involvement and Context-Sensitivity: existing approaches typically require explicit user involvement, in creating dictionaries for form-filling, in building customized tasks, or in selecting and applying such customized tasks.
- Adaptability: optimizations supported by existing approaches are typically closely tied to the given context or task that the optimization is created for. Minor changes in context or task may render such optimizations inapplicable.
- Comprehensive Optimization Target: most of such existing approaches only target either reducing the amount of user input or the amount of user wait time, but not both.

The motivating vision for our research is a mobile user interaction optimization system that requires minimal explicit user involvement, is sensitive to users' interaction contexts, looks beyond users' immediate next step, targets costs in both user input and user wait time, and can to some extent adapt to changing user tasks. In this paper we propose the design and describe the prototype implementation of MIntOS, an initial framework towards the above goal.

MIntOS monitors user's web interaction to both gather history data and construct runtime interaction context, and uses an approach based on interaction burstiness and copy/paste dependency to break monitored interaction sequences into Trails, which are lightweight approximations to user tasks. Such Trails are then used by to generate rules for predicting future user interaction sequences. Predicted future interaction sequences are optimized to reduce user input and wait time. Such optimizations

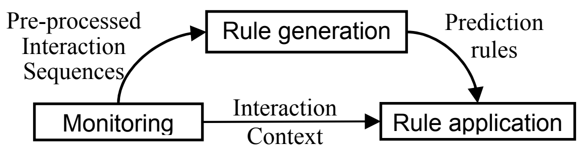

Figure 1. Components of the MIntOS framework

are then recommended to the user through UI widgets in a non-intrusive manner.

We have implemented a prototype of MIntOS and used several sample examples to demonstrate its improvements to mobile web interaction.

Our contributions are an interaction monitoring, simple reasoning and rule matching based framework for context-sensitive prediction of future interaction sequences; the optimization of such future interaction sequences and the UI mechanism for conveying such optimization to the user; and the implementation and demonstration of such framework.

2. RELATED WORK

One way to reduce user wait time in web browsing is to break larger web pages into smaller pages. [1][2][3]. While web page segmentation techniques generally reduce user wait time for each page-load request as pages now become smaller, they may not reduce the total amount of user wait time for a given task, as reducing the sizes of pages likely also increases the number of navigation steps a user must take to complete a task. An approach to improving users' task performance is to optimize the hierarchy of the information that resides on the server.

WebVCR [7] and CoScripter [4] are systems that can record and automatically replay user interaction steps for web browser based tasks. A user can explicitly request to record his interactions with the browser, and save the recorded information for later playback. Both systems depend on explicit user instruction for recording, and choosing and playing back scripts. This puts burden on the user to identify frequent tasks and to remember scripts corresponding to frequent tasks. In addition, although such systems reduce the amount of user input for recorded tasks, it doesn't make it a priority to reduce user wait time for such tasks.

Highlight builds a re-authoring environment that allows mobile users to manually customize web-based applications that are originally designed for desktop users.[5][6] It uses a programming-by-demonstration based approach similar to that of CoScripter, and allows users to directly specify content to be included on mobile pages. Unlike CoScripter, Highlight explicitly targets user wait time for optimization in web-based mobile interaction. But similar as CoScripter, Highlight requires significant amount of explicit user involvement.

3. FRAMEWORK

The framework of MIntOS is composed of three conceptual modules: Monitoring, Rule Generation, and Rule Application (Figure 1). Note that these three modules can be decoupled in terms of both time and space, i.e., they can occur in different places at different times.

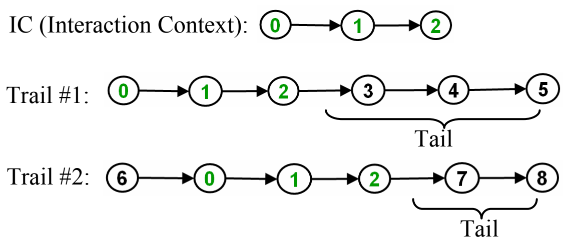

Figure 2. Example for target Trail selection

3.1 Mobile Interaction Monitoring

The Monitoring module observes users' interaction with their web browsers, and pre-processes interaction sequences for prediction rule generation. MIntOS relies on interaction history, i.e., users' past and current interactions with their devices or PCs, to predict what interaction sequences a mobile user is likely to perform next and to make such predicted likely interaction sequences easier to perform for the user. Within the scope of MIntOS, interaction history is defined by a set of *interaction state* transition sequences, where the interaction state is defined by the following state variables:

- The page that is currently open in the browser,
- The content of the URL bar,
- The highlighted text on the current page,
- Content of the clipboard, and
- Form inputs made to the current page

An *interaction event* is an event that changes the value of one of the above interaction state variables. In MIntOS, types of interaction events include: 1) Page-Load event, 2) URL-Input event, 3) Text-Highlight event, 4) Text-Copy event, 5) Text-Paste event, and 6) Form-Input event, which can be caused by a Text-Paste event or simply by user typing text in an input field.

The monitoring module breaks the captured interaction sequences into Trails. A *Trail* is a segment of user interaction sequence. It reflects a burst of user interaction activities. A long sequence of user interaction with a device can be divided into multiple Trails at abnormally long time gaps along the sequence. Such long time gap is usually resulted from the user reading a page in detail, or from the user shifting attention away from the browser. For example, assuming a user goes to "gadgettell.com", navigates to an article describing a gadget called "ZVBox", spends several minutes reading the article, and then goes to "amazon.com" and types in the search box to search for "ZVBox". This sequence of actions can be broken into two Trails at the point that the user reads the "ZVBox" article. Note that a Trail does not strictly correspond to a user level task, but it is a lightweight approximation of such task.

3.2 Prediction Rule Generation

The Rule Generation module creates interaction prediction rules from databases of collected trails. It runs periodically on each recently updated Trail database. A prediction rule created by the module takes the following form:

<Interaction_Context> :- <Optimized_Trail_Tail>

where <Interaction_Context> is defined as one of the segments of one of the Trails in the database (i.e., a subsequence of monitored interaction steps), and <Optimized_Trail_Tail> is the tail of a candidate Trail after optimization.

For each unique prefix of any Trail in the database, the Rule Generation module goes through two steps to try to create prediction rule(s): 1) find candidate Trails that "match" the prefix, and 2) optimize the tails of the candidate Trails.

Candidate Trail selection is through a recursive algorithm which takes an Interaction Context (IC), and either goes to the next level of recursion or returns a pair (IC, MS), where MS is a set of Trails matching TS (the size of MS could be zero). The purpose of the algorithm is to find likely future interaction sequences (i.e., MS) given interaction context IC.

The algorithm is initially called with a Trail prefix as input. At each recursion point, a new input for next level of recursion is generated by removing the first state of the Interaction Context that represents the current input. In the example shown in Figure 2, the new IC for the next level of recursion will be a Trail segment consisted of states 1 and 2.

A recursion level returns the (IC, MS) pair, and consequently terminates the whole algorithm, when one of the following conditions is reached:

1. Rule(s) with <Interaction_Context > identical to IC exists;
2. A non-empty set of Trails matching IC is found;
3. IC contains only one state (and thus no need for further recursion).

The tail of each matching Trail in the MS of the (IC, MS) pair is then optimized to reduce user input, as well as to reduce user wait time for page loading. Specifically, for each event in the tail,

- If the event is a Form-Input event, then we determine if such input is resulted from copy/paste operations by tracking back along the Trail. If it is, we associate the value(s) of the text string(s) copy/pasted with the Form-Input event.
- If the event is a Page-Load event, we check whether the URL or other request parameters of the page to be loaded is associated with one or more of preceding Form-Input events. This is accomplished by comparing form action name and input field name of a Form-Input event against the action name and the parameter names of a page load request. We then link the Page-Load event with its associated Form-Input event(s).
- If the event is a Page-Load event and there is no Form-Input event associated with it, we create a new Page-Prefetch event for it and add the new event to the start of the tail.

We then discard remaining URL-Input, Text-Highlight, Text-Copy, and Text-Paste events. The resulting tail is finally combined with the Interaction Context to form a prediction rule.

3.3 Matching and Applying Rules

The Rule Application module matches a user's current interaction context (provided by the Monitoring module) against generated rules and, if there are any matching rules, communicates the optimized future interaction sequences to the user non-intrusively.

The optimized Trail tail of each matching prediction rule is applied for interaction optimization including:

- Page prefetching, for reducing user wait time in page loading. Each Page-Prefetch event in the optimized Trail tail indicates an opportunity of pre-loading a page before the user requests for it. Whether such pre-fetching actually takes place depends on factors including current network condition. Page prefetching, if takes place, is transparent to the user.
- Navigation short-cuts, for allowing a user to skip ahead and avoid downloading un-used intermediate pages. Each Page-Load event in the optimized Trail tail is presented as a UI widget that, when activated, loads the page indicated by the event.
- Form pre-filling, for reducing the need for user to type in or copy/pasting text. Each Form-Input event in the optimized Trail tail is presented as a text-input widget whose value is initialized with the text value of the event. The user can change the value of the widget just like editing the text input field of a HTML form. Changes made to such widgets are automatically propagated to Page-Load event(s) that are associated with the Form-Input event. Note that with such input widget, the user can input text in form fields without first loading the page that contains the form.

3.4 State, Trail and Rule Generalization

To make it more likely to find matching Trails in candidate Trail selection, and more likely to find prediction rules matching the user's current interaction context, it is necessary to generalize interaction states and Trails. *Interaction state generalization* is the process of clustering states previously considered different into an equivalence state, so that established prediction rules can be applied to user tasks that are slightly different, but are still similar to some extent.

While there are other possibilities for interaction state generalization, in MIntOS we focused on two fundamental approaches: content-based and structure-based interaction state generalization. Here by *content* we refer to the textual information on the page visible to the user. This includes the text nodes of the document tree, text labels of hyperlinks (or "inner text") and buttons, as well as text input elements. By *structure,* we mean the hierarchy of the tags in the document tree, as well as the names (but not values) of the attributes of these tags.

In content-based interaction state generalization, we consider pages with the same content as the same for state comparison purposes, and set the state variable for current page to this equivalence page. And similarly in structure-based interaction state generalization, we consider pages with the same structure as the same for state comparison purposes, and set the state variable for current page to this equivalence page.

4. IMPLEMENTATION & DEMONSTATION

We have implemented prototypes of MIntOS for Firefox and the Android platform. In the former case, MIntOS is implemented as a Mozilla extension which listens to DOM events to monitor user interaction with the browser. UI widgets mentioned in section 3.3 are displayed on the status bar of the browser (Figure 3). In the latter case, we modified the Browser application included in the

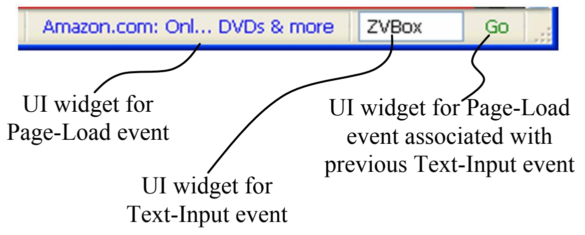
Amazon.com: Onl... DVDs & more
ZVBox
Go
UI widget for Page-Load event
UI widget for Text-Input event
UI widget for Page-Load event associated with previous Text-Input event

Figure 3. Widgets for optimized rule body displayed on browser status bar

Android "Donut" branch at strategic program locations to intercept user interaction events. UI widgets are displayed as horizontal bar at the bottom of the page (Figure 4(a)). Note that when a widget acquires focus, a popup window showing the snapshot image of the page corresponding to the widgets appears.

When there are multiple matching prediction rules, the user can get access to those matching rules with lower normalized interaction costs by pressing a button and revealing a popup window (Figure 4(b)). The popup window contains multiple horizontal lists of optimization UI widgets, with each list corresponding to a matching prediction rule.

4.1 Demo Description

We demonstrate the basics of MIntOS, including how it monitors user interaction, and how the user interacts with UI widgets that represent optimized interaction suggested MIntOS. We also demonstrate how interaction state generalization helps MIntOS deal with changes in Web pages and user tasks. Our demonstration employs three real world use scenarios:

- The "ZVBox" example, where a user goes to web site "gadgetell.com", navigates to an article describing a gadget called "ZVBox", and tries to search for "ZVBox" at Amazon.com. Later a different user opens the same gadgetell.com ZVBox article. We use this use case to demonstrate the basics of MIntOS.
- The Smart Copy/Paste example, where a user, at a paper summary page of an online library, copy/pastes citation information of the paper between the summary page and a citation database application. Later the user visits the summary page of a different paper. We use this example to demonstrate interaction state generalization.
- The mobile banking example, where the user each time needs to go through a number of time-consuming page loading steps to check his/her bank account balance. We use this use case to demonstrate short-cut support and interaction state generalization in MIntOS. (Figure 5)

5. CONCLUSION AND FUTURE WORK

In this paper we described MIntOS, a framework for enhancing user experience in web-based mobile tasks based on interaction monitoring, future interaction prediction and optimization, and adaptive UI for context-sensitive recommendation of such prediction and optimization. We have implemented prototypes of MIntOS and used real-world use cases to demonstrate the improvements to web user interactions and experiences.

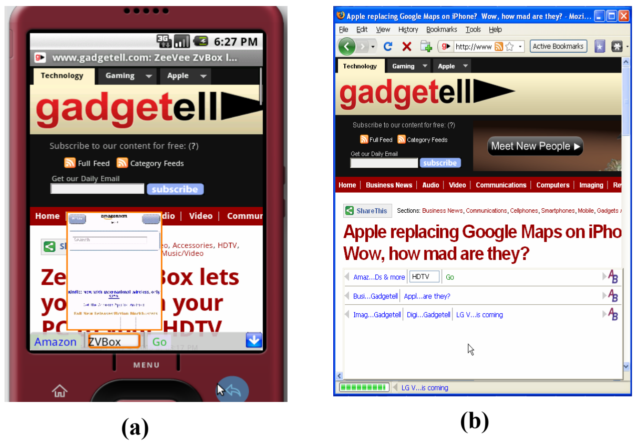

Figure 4. (a) User interface of the prototype on Android Platform. (b) A popup window showing multiple other predicted (but less likely) future interaction steps

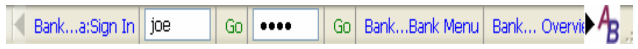

Figure 5. Status bar on Joe's browser when he visits the Mobile Banking application for the second time

Our future work will include a more extensive study of using MIntOS on mobile handsets with other usage scenarios, investigation on other generalization approaches, as well as customizing our system to, for example, different device characteristics and user expertise.

6. REFERENCES

[1] G. Hattori, K. Hoashi, K. Matsumoto, and F. Sugaya. Robust web page segmentation for mobile terminal using content-distances and page layout information. Proceedings of WWW 2006.

[2] S. Baluja. Browsing on small screens: recasting web-page segmentation into an efficient machine learning framework. Proceedings of WWW 2006.

[3] C. C. Yang and F. L. Wang. Fractal summarization for mobile devices to access large documents on the web. Proceedings of WWW 2006.

[4] G. Leshed, E. Haber, T. Matthews and Tessa Lau. CoScripter: Automating & Sharing How-To Knowledge in the Enterprise. Proceedings of CHI 2008.

[5] J. Nichols and T. Lau. Mobilization by Demonstration: Using Traces to Re-author Existing Web Sites. Proceedings of IUI'2008.

[6] J. Nichols, Z. Hua and J. Barton. Highlight: A System for Creating and Deploying Mobile Web Applications. Proceedings of UIST'2008.

[7] V. Anupam, J. Freire, B. Kumar, and D. Lieuwen. Automating Web navigation with the WebVCR. Proceedings of WWW 2000.