

UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER
SCIENCE
Institute of Computer Science
Computer Science Curriculum

Gleb Stsenov

**Measuring mobile search tasks on Android
platform**

Bachelor's thesis (6 EAP)

Supervisors: Ulrich Norbistrath, Ph.D.

Author:

“.....“ June 2011

Supervisors:

“.....“ June 2011

Approved

Professor

“.....“ June 2011

Table of contents

Chapter 1 Introduction.....	4
1.1 Search tasks and their measurement.....	4
1.2 Thesis content overview	5
Chapter 2 Overview of related work	6
2.1 Common strategies and research on search task measurement and user behavior.....	6
2.2 Related work at the University of Tartu	9
Chapter 3 Search Experiment research.....	15
3.1 Research description.....	15
3.2 Search Experiment client-side application description	17
3.3 Server front-end and data store description.....	22
Chapter 4 Overview of the Search Experiment project.....	27
4.1 Implementation details	27
4.2 Possible improvements for the Search Experiment application and data storage	28
4.3 Analyzing results	31
Summary.....	33
Resümee	34
References	36
Appendices	39
Appendix A. The Search Experiment user guide	39
Appendix B. The source code and installation package of the Search Experiment (client and server)	45

Chapter 1

Introduction

1.1 Search tasks and their measurement

In July 2010, over 8.8 billion searches were conducted in the United States using Internet search engines [1]. This number is constantly growing all around the world. People use search engines more and more, and their queries become more and more complicated [26, 27]. People prefer to ask more complicated things of the search engines, and expect to get results quicker.

When the first search engines appeared, they were used only to crawl web pages, index their content and identify the pages that matched the user's query [2]. Nowadays users want to ask a question in human speech form, without thinking of how to formulate a query to get best results. And while queries such as "mass of Earth" are simple to handle, a query "What is the cheapest hotel in the center of Bangkok?" requires not only searching, but also analyzing.

To help search engine developers to improve their products, a lot of statistics must be collected. One of crucial factors is the user's behavior during the search session: how many pages they visited, how many links user opened one-by-one, taking him away from the search engine website. Measuring of search queries is a process of collecting such statistics during user activity when searching.

While our world becomes more and more mobile, the number of searches made from mobile devices, such as smartphones and tablet computers, is also growing. This thesis focuses on research that measures search queries on the Android mobile platform, which enjoys constantly growing popularity and is installed on a large variety

of devices [3]. Thesis also presents the design and development process of a mobile search logging tool.

1.2 Thesis content overview

Chapter two gives an overview of research of search tasks measurement and user behavior logging techniques, and also describes related work from the University of Tartu. A special emphasis is given on the mobile domain and the Search Logger project, which is used as the basis for the Search Experiment project presented herein.

Chapter three explains the platform and framework choice, describes the process of the research, usage of the Search Experiment application, the questionnaires used, and details of the logging process. As it is a client-server architecture project, server-side software and data storage are also described.

Chapter four gives an overview on results of the implementation of the application, provides some information about experiments undertaken, and describes the possible further work to develop the Search Experiment and improve the research.

The summary provides an overview of the thesis.

Appendix A contains a user guide for the Search Experiment application.

The source code of the Search Experiment is linked in the Appendix B of this thesis.

Chapter 2

Overview of related work

2.1 Common strategies and research on search task measurement and user behavior

Search engine quality measurement strongly connects search engine performance measurement and user search experience research. The first of these is more a server-side area, while the second one is related to the client side [4]. Research in this area has been conducted since the mid-nineties, and usually consisted of analyzing search engine logs [5] from one side and work with users from the other side. The analysis of logs usually consists of removing duplicates, finding correlations and judging their strength, for sets of two, three and more words. Research on user behavior usually requires dividing people into different groups, conducting interviews, investigating the users' knowledge of different domains and their skills in Web search, and, crucially, the client-side logging of user activities [6].

The division of people into groups is required because of different levels of Web experience and domain knowledge. According to Navarro-Prieto, Web searchers with high and low experience use different strategies of search [7], and as the research of Hölscher and Strube shows, the lack of relevant domain knowledge becomes a large obstacle even for Web search experts [6]. To divide people, an interview is usually conducted, in which the user's experience with the Internet and the use of search engines is investigated. People are asked about their Internet usage frequency and duration; what they are usually doing in the Web; what they are searching for; which search engines they use; and whether they are satisfied with the engines they commonly use for their searches. As the results show,

people who use Web search engines often need less time to find the necessary information than people who are new to the Internet and information search. Research also shows a strong correlation between performance of search and presence of domain knowledge [6].

When a search task is specific to some domain, such as economics or medicine, and a factual query is given, the behavior of groups differs. About 20% of the people with domain knowledge access certain websites directly, while others (up to 55%) prefer to use search engines. People with significant experience in Internet search will use the same terms that were provided by the study's authors while using search engine. 40%-50% of people with domain knowledge (regardless of Web experience) reformulated the factual queries, for example including synonyms or abbreviations, which helped them to find the answer more quickly [6].

Logging user events (page opening, search query input, viewing search results) can be implemented simply on several levels, which can be defined as the human level, the browser or network level and the system level.

Human-level logging is perhaps the most complicated form, as it involves recording human logic and thoughts. For example, in Hölscher and Strube's research, participants were forced to make every step of the interaction process verbally explicit, and were asked to think aloud about their activity [6]. All speech was taped and later transcribed. Analyzing it together with Web page requests and search queries provided the entire sequence of users' actions.

At the browser level, user events such as opening and closing a website can be logged. Requests and search engine interaction are also logged, specifically events such as going back and forward through search results, the Uniform Resource Locators (URL) and HTTP result codes. This can be easily implemented via a proxy server on the local machine or on some gateway between the user's computer and the Internet [6, 7].

System-level logging helps to log events of all kinds. It is usually a desktop application that logs system information, such as running processes, window and clipboard changes, and keys pressed. However, the concept of system-level logging excludes the concept of search tasks; it only provides a picture of system changes. It is more of a secondary logging tool. The example of such a program is Wrapper, written by Bernard J. Jansen in 2006 [25]. Slightly different from system state level logging is the concept of browser extension. This can log system events related to the browser, such as tab switching, browser activation and deactivation, clipboard changes, bookmark management events, and some others. It can also interact with browsed content in order to log browser-level events. An example of such a tool is the Internet Explorer add-on developed by S. Fox in 2005 [8].

As our world becomes more and more mobile, and the speed of broadband wireless connections increases, more and more mobile traffic is generated. There are two kinds of such traffic: the first one is traffic from regular portable computers such as laptops, and the second one is traffic from handheld devices such as personal digital assistants (PDA), communicators, and cellular phones. It is clear that user behavior on a personal computer (laptop) is usually the same or very similar to behavior on desktop computers, as the same input and output methods (full keyboards and large displays) exist both on laptop and desktop computers, and the speed of connection in modern 3G networks is even higher than the speed of wired connections a few years ago. The difference when using a handheld device is that the screen is usually smaller, the keyboard is often on-screen, or is a phone number pad with a number and few letters on one key, and the amount of memory on the device is limited, so the user cannot open 10-20 tabs. Often there is not even an ability to use tabs, with only one browser window, like in earlier versions of Microsoft Internet Explorer. Search techniques are also different. As the usage of Internet on

handheld devices increases, mobile search has to be measured in order to improve the quality of search results and to provide quick access to information.

Experiments and research in measuring user behavior on mobile platforms are being conducted constantly. For example in 2007, when Nokia designed its browser for the S60 platform, the study called *Smartphone 360* was conducted in the UK, France and Germany. The study was 3 months long and involved 547 users of Nokia S60 smartphones. They downloaded a special logging tool onto their phones, which logged the activity of users on the Web. It worked at the system and browser levels, and participants were also interviewed, so the researchers also collected some information about users and their experience [9].

2.2 Related work at the University of Tartu

In March 2011, the Search-Logger experimentation environment was presented by H. Kikkas, U. Norbistrath, G. Singer and E. Vainikko from the University of Tartu in conjunction with D. Lewandowski from Hamburg University of Applied Science [4]. It was a product of research in the field of exploratory search tasks and their measurement.

Search Logger consists of an add-on for the Firefox browser and a server part, a PHP front-end and database. The add-on works mostly at the browser level, with a small workaround at the system level in order to log clipboard changes.

After Search Logger is installed on the browser, it asks the user to fill in the so-called demographics form, which is in fact an interview with questions about the participant's age, sex and other parameters, as well as his experience in using the Internet and search engines. After the form is filled out, the data is sent to the server through the Web front-end and stored in the database, or logged locally in a file. Then the add-on asks the user to complete one of the available search tasks, which differ by their complexity, starting from the simplest questions and ending in large complex tasks that require the user to collect and

analyze a lot of data. When the user selects a task, the experiment starts. A small questionnaire is shown, consisting of questions about the user's expectations of the given task. This is also submitted and logged. All user events, such as opening a Web page, opening a new tab or going to a search engine page, are logged. Data is logged in a file or sent directly to the Web front-end. When the user indicates that the task is finished, a questionnaire about the task is presented. This is also logged. After the user finishes all tasks, survey is conducted about user's experience with the experiment as whole. The experiment can be paused at any time by pressing a Pause button. Pausing is logged in the data storage facility and then no logging activity is performed until the experiment is resumed. The extension also has an option to log the data in a file on the local computer.

Type of event	Explanation
Search task started	Log entry for start of search task
Search task stopped	Log entry for end of search task
Search experiment started	Start of search experiment
Search experiment stopped	End of search experiment
Web page visited	User visited a website
Query entered	User entered a query
Link clicked	User followed / clicked on a link
Tab opened	User opened a tab
Tab closed	User closed a tab
Bookmark added	User created a bookmark
Bookmark deleted	User deleted a bookmark
Text copied	Copy/Paste event
Search Logger paused	Pausing the Search Logger extension
Introducing a new user	Extension started for the first time
Demographics displayed	Demographics form displayed
Demographics submitted	Demographics form submitted
Pre-search task form submitted	Pre-search task feedback form submitted
Post-search task form submitted	Post-search task feedback form submitted

Table 1. User events logged by Search-Logger [4]

The server front-end and data storage facility consist of a set of PHP scripts and a MySQL database. There are three main scripts and some additional ones. Additional scripts contain some functions such as generating and checking user identifications,

database connection handling, and some others. The main scripts are used for adding a user to the database, adding a log event, and displaying the log.

There are two data tables: one for users' list and one for logged events.

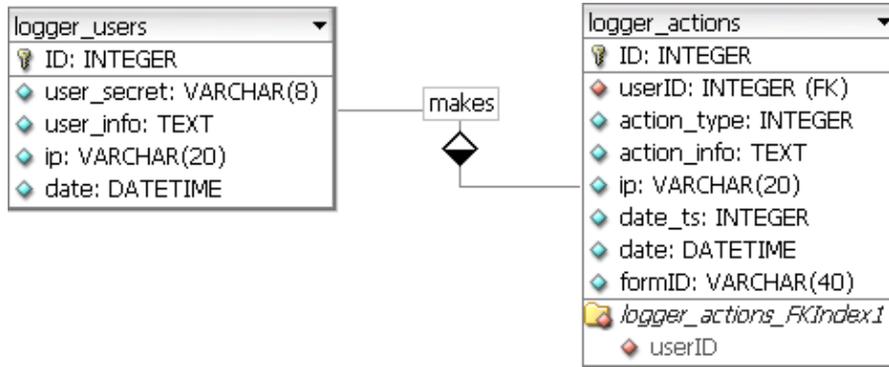


Figure 1. Search Logger data model

The Search Logger server part has a very simple interface for displaying collected data. Information is presented as a table of actions with timestamps, user identifications, addresses, and a dump of the data sent. The data table contains event types, so the sequence of user actions can be reconstructed and some statistics can be obtained by running specific SQL query against the database.

The main disadvantage of the current data storage system of Search Logger is that demographic data and that from other forms is sent encoded into one string, as can be seen in Table 2, on row marked (a). The string "(b=(5))(c=(1))(d=(1))" means that in the form field named "b", the answer was number 5, and question "c" was answered with first option. This means that when compiling statistics from this table, an SQL query has to use full-text search statements to find strings that contain a specific answer. For example, to get the number of users who answered "Totally disagree" to question "c" for the 5th search case post-survey, generic SQL query has to be following:

```

SELECT COUNT(id) FROM logger_actions WHERE formID = '5_FREE_5'
AND action_info LIKE '%(c=(1))%';
    
```

Or, with more MySQL-specific syntax

```
SELECT COUNT(id) FROM logger_actions WHERE formID = '5_FREE_5'  
AND MATCH(action_info) AGAINST '(c=(1))';
```

This is not the best query in terms of performance, because it uses a full-text search operation to match a string. Of course adding indexes to the table can improve performance, but in any case text search is slower than matching the row by a short definite value of one of the fields. However, is not critical in context of small research, and may slightly affect the performance, if research involves really large number of participants.

Firefox is a cross-platform browser, supporting the Windows, Linux and Mac OS X operating systems, which covers the majority of desktop users world-wide, and Search Logger is a Firefox extension. For mobile platforms, the Mozilla Foundation has a version of Firefox called Fennec, for Android and Maemo platforms. It shares a plug-in model with the desktop version, and the development of mobile add-ons mostly differs in user interface model. If extension has no desktop-specific features, the process of porting does not take long [12].

Maemo platform has only two actual devices, the Nokia N810, released in 2007, and the Nokia N900, released in 2009. They were not very popular among casual users, and in the year 2010 it was announced that Maemo will be merged into the MeeGo platform [13] and developed jointly by the Intel and the Nokia corporations. In May 2011, the Mozilla Foundation presented version 6.0a1 of Fennec at the MeeGo Conference in San-Francisco, but there is still a lot of work required before MeeGo can be released [24].

As for the Android platform with its growing popularity, the most known Android development problem is hardware. There are lots of vendors, producing very different devices, from low-price to hi-end, from mobile phones to netbooks. They have different processors, and the Android version of Fennec supports only ARMv7 (ARM version 7)

architecture processors. Its system requirements mention that the best performance is achieved on devices with at least 512MB of RAM [14]. These are typically middle and high price class devices, and cheaper middle-class devices and low-class devices are not covered. The Mozilla Foundation has tried to develop nightly versions of Fennec for ARMv6 architecture processors, but it has failed. ARMv7 architecture supports a newer instruction set, and it is easier for Mozilla developers to develop only an ARMv7-specific version of Fennec. So, porting of existing Search Logger version to Fennec can be done, but it will cover only a limited set of Android devices used worldwide. This means that there are many opportunities to conduct additional search measurement research and experiments in the mobile domain.

Type	Action	Information	IP	Date	FormID
15	User viewed a related link	Location: http://www.piletilevi.ee/est/piletid/teater/muu/?show=19628... Title: ABBA. Kontsert - tantsuetendus - Piletilevi.ee	90.86.28.12	2011-05-04 19:31:07	5_FREE_5
15	User viewed a related link	Location: http://www.piletilevi.ee/widget.php?type=sportshows209&l=est... Title:	90.86.28.12	2011-05-04 19:31:07	5_FREE_5
15	User opened a tab	User opened a tab	90.86.28.12	2011-05-04 19:31:55	5_FREE_5
15	User viewed a related link	Location: http://www.facebook.com/plugins/likebox.php?href=http%3A%2F%2Fwww.facebook.com%2Fvanemuine&width=190.. Title: Facebook	90.86.28.12	2011-05-04 19:32:02	5_FREE_5
15	User viewed a related link	Location: http://www.vanemuine.ee/... Title: Vanemuine	90.86.28.12	2011-05-04 19:32:05	5_FREE_5
15	User closed a tab	User closed a tab	90.86.28.12	2011-05-04 19:32:33	5_FREE_5
15	User viewed a related link	Location: file:///C:/Users/kiku/AppData/Roaming/Mozilla/Firefox/Profiles/o6y4hxgj.de/fault/extensions/%7B8176c4... Title: Search case description template	90.86.28.12	2011-05-04 19:35:16	5_FREE_5
15	User opened a tab	User opened a tab	90.86.28.12	2011-05-04 19:35:16	5_FREE_5
16 (a)	User submitted post-SC form	Form data: (b=(5))(c=(1))(d=(1))(formID=(search_case_5_2)) Form ID:	90.86.28.12	2011-05-04 19:35:33	5_FREE_5
15	User viewed a related link	Location: http://www.search-logger.com/tiki-index.php?page=Case+Ended... Title: Case Ended : The Search_Logger Project	90.86.28.12	2011-05-04 19:35:37	5_FREE_5

Table 2. Part of the Search Logger log

Chapter 3

Search Experiment research

3.1 Research description

Android was chosen as the platform on which to conduct a search measurement experiment in the mobile domain, as it is fast-growing, popular, and mostly open-source. It is easier to start developing for the Android platform than for Apple iOS, for example because Apple's Xcode IDE (*integrated development environment*) works under Mac OS X or iOS only [29], while the main Android development tool ((Eclipse) is cross-platform. Also, at the beginning of the development process, a Java-based framework used to create Android applications is quicker and easier to understand than Objective-C which was born in 1986 and is used for creating Apple iOS applications [28].

The idea was to create a program that allows users to conduct an experiment in measuring their search habits and techniques, and provide researchers with feedback in the form of collected data and user opinions. The collected data must allow results to be divided by groups according to the participants' search experience and social criteria, and provide statistics about user preferences in search. As the application is based on the Search Logger project, it was possible to take the tasks and surveys from Search Logger, with minimal changes required to adapt them to the mobile platform.

As the Search Logger project is used as a base, creating a browser extension is the most natural way. Android's WebKit-based browser supports the Netscape plug-in model called NPAPI (Netscape Plugin Application Programming Interface), but it involves native programming and diving into Android source code, which can be very dependent on hardware and software versions. The concept of Search Experiment requires that everyone must be able to take part in the research, so the method of creating an application had to be

more universal, working with no hardware limitations and a software limitation of Android versions above 2.0. Older versions 1.5 and 1.6 are used on about 5% of devices as of the beginning of May 2011, according to Google's statistics [17].

A brief scan of browser customizations in the Android Market showed that a lot of so-called browser extensions are in fact individual applications with the WebView browser component as the main part of their user interface. This is the best model for experiment, as it does not interfere with the user's normal Web activity, and removes the need for pause / resume plug-in functionality. Interaction with the browser component allows logging events, and as the component is integrated with the device's user account, existing bookmarks and cookies can be used.

The approach with a proxy server, used for example in Hölscher and Strube's research, was not used as it takes more time to deliver data to the user's device. Also, using of on-device application allows logging more events (like bookmarking and deactivating / activating of application) easier. The Search Logger measuring tool, taken as a basis, also does not use a proxy server.

The logic of the experiment is very similar to the Search Logger browser extension. When used for the first time, it presents a short questionnaire, covering the user's social background and experiences. After this is completed, a search task can be selected for the experiment. A short survey about user expectations is conducted, after which the user starts the experiment, either by using the search engine or by opening a website according to the task's theme. While the user is searching, events are logged. When the user decides to finish the experiment, a survey about search results is conducted, and this data is also logged. When the user finishes all search tasks, a questionnaire is presented about his impression of the experiment at whole, helping the researchers to improve the application and create new tasks.

Type of event	Value	Explanation
FORM_DATA	0	Initial value or form submission data
ACTIVITY_STARTED	1	Search Experiment started
ACTIVITY_FINISHED	2	Search Experiment stopped
ACTIVITY_PAUSED	3	Activity paused (is not foreground activity)
ACTIVITY_RESUMED	4	Activity resumed
WEB_PAGE_VISITED	5	User visited a web page
SEARCH_ENGINE_PAGE_VISITED	6	User visited search page
INTRODUCED_NEW_USER	7	New user
DEMOGRAPHICS_DISPLAYED	9	Demographics form displayed
DEMOGRAPHICS_SUBMITTED	10	Demographics form submitted
FINAL_DISPLAYED	11	Final form displayed
FINAL_SUBMITTED	12	Final form submitted
PRESEARCH_DISPLAYED	13	Pre-search task feedback form displayed
PRESEARCH_SUBMITTED	14	Pre-search task feedback form submitted
POSTSEARCH_DISPLAYED	15	Post-search task feedback form displayed
POSTSEARCH_SUBMITTED	16	Post-search task feedback form submitted
BOOKMARK_ADDED	21	User created a bookmark

Table 3. User events, logged by the Search Experiment

3.2 Search Experiment client-side application description

As was described before and dictated by research logic, the application is in fact a simple browser that logs different events during the experiment. It consists of one main window (an Activity in Android terminology). This has an address bar at the top of the window, with a button to open a provided URL. The remaining part of the screen is used by the WebView component, which renders the opened page. The browser does not support using tabs for having several different pages opened at the same time, which allows the sequence of user events (such as moving back and forth through the search query results) to be logged correctly. The application utilizes a standard Android activity menu, called by pressing the menu button (which is available on all Android devices). The menu contains items that Start/Stop the search experiment, add a bookmark of the current page to device-wide bookmarks (available for viewing later in the native browser), access the program preferences, and an Exit command to close the program.



Figure 2. Search Experiment main activity with menu

The Settings menu item calls a typical Android preferences screen. It has a structure similar to the Search Logger preferences menu, and is divided into two parts. The first one is Settings which lets the user choose the method of logging (File and/or Web logging), change the address of the server that collects logged data, and change the automatically generated participant identification number. By default, the participant ID is randomly generated using a current UNIX timestamp (time from January 1, 1970) in milliseconds combined with a random integer number between 0 and 2^{32} . This is different from approach which is used in the Search Logger. Search Logger uses user's IP address and random integer number between 0 and 23, so theoretically two users from one local network segment can get same client identifications with a high probability, representing a particular case of well-known "birthday problem" [32].

The second part of the preferences screen concerns the completed forms. It allows the user to re-submit the initial demographics form and the final form, which is usually displayed when all tasks have been completed. The user can also retry the search tasks by removing their number from "Used Cases" field. Preferences management is done

according Android's recommended practices, which takes a minimum of effort by the developer.

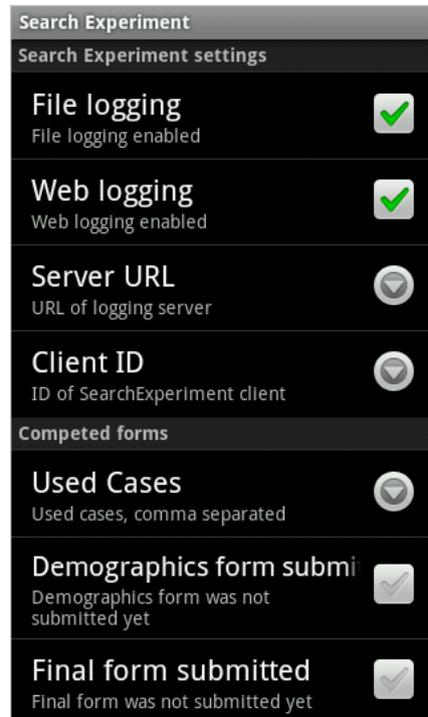


Figure 3. Search Experiment Preferences activity

The original idea was to use existing HTML pages with surveys from the Search Logger project, but some problems were discovered. First of all, due to the Firefox plug-in nature of Search Logger, the form is not actually submitted; rather, a Mozilla JavaScript API is called, which forms the request URL using the plug-in configuration and sends the data to the server. This was the only way to add user identification to the request. Second, the HTML code of pages was semantically invalid, despite the HTML Transitional Doctype used. This could cause problems later when rendering the page [31]. Small changes were required to transform JavaScript-interacting Search Logger Web pages to the form that best worked with the WebView component.

There were 4 search cases taken from the standard version of Search Logger [18]:

1. Find the date when Mozart was born;
2. Find value of 20000 EEK in US dollars;
3. Find the date the invention of penicillin and the name of the inventor;
4. Make a plan of a budget wedding, writing down a list of the 20 most important points to consider

These four tasks are arranged in an ascending order of complexity. As they were taken from an older version of Search Logger, there is only one task for real exploratory search that requires analyzing a lot of information – the wedding budget task. In fact, this kind of search is not usually done from compact handheld devices, but with revolutionary advent of tablet devices it is now easier for users to perform an exploratory search tasks from mobile devices. In the current implementation, the tasks are packed into the application install package and are the only available tasks for the experiment.

When the application is first started, a message dialog appears, asking the user to fill out a demographics form. It contains fields for name, age, sex and contact data (such as e-mail), and a short survey about the user's experience in using the Internet and search engine. It also contains questions about the user's comfort level with carrying out search tasks on the Internet.

After choosing the Start command in the application menu, the user can select one of the search cases from list.



Figure 4. Selection of the search case

	disagree	1	2	3	4	5	agree
I expect the search task to be easy	<input type="radio"/>						
I think I can find the desired information with search engines only	<input type="radio"/>						
I expect that I will need other sites (like Wikipedia) to fulfill this task	<input type="radio"/>						
I think I will need less than 5 queries to find the desired information	<input type="radio"/>						
I think it will take me less than 5 minutes to get a sufficient result	<input type="radio"/>						

Figure 5. Pre-search task feedback form

When a case is selected, a form is displayed, asking the user about his expectations of the task. These help to classify the user's experience levels with Web search, and later analyze the ways of finding information depending on user skills. After the form is submitted, the user is redirected to the search engine page, which is Google Mobile Search.

The first search task given in the application is very simple. For example, when using the Google search engine, it is enough to enter the word "mozart" in the query text box. When the search result page is opened, the first item in the results is a Wikipedia article, and the beginning of this article is displayed. The user can see immediately that Mozart was born on January 27th, 1756.



Figure 6. Search results page

Now the user can select the Stop command from the menu. The post-task form is then displayed. It contains questions about how the experiment has gone, in comparison with the expectations, expressed in the previous survey. When the form is submitted, the experiment on given task is successfully finished. When starting another task, the user cannot pick the question about Mozart from the list. The list can be reset by removing the task number in Settings. Parallel making of several tasks is not supported at the moment.

When the user searches for information, user events are logged. When Web logging is enabled, data is immediately sent to the server logging application, helping researchers to reconstruct to the course of the experiment. File logging allows results to be submitted later.

3.3 Server front-end and data store description

The server part of the application consists of a database and a single PHP script that stores the collected data into two database tables.

The first one is called "sl_log", and contains user events logged from the application. It provides information on the actions made by user during the search.

From this table, researchers can reconstruct the experiment. For example, table 5 contains the application log after first launch.

1. Row 35 and 36 – Search Experiment application was launched and a blank page loaded;
2. Row 36 – the demographics form was displayed;
3. Row 37 has ACTION_TYPE=0, which means a form was submitted (the demographics form, as seen from the FORM_ID column); the REQUEST_DATA array has 22 elements, these are the answers;
4. Row 38 - the demographics form was successfully submitted;
5. Row 39 – the browser was redirected to Google search;
6. Row 40 – the pre-search form was displayed, which indicates that the search task was selected and started. The URL indicates that this was the 1st task;
7. Row 41 – received the search_case_1 form with 7 answers;
8. Row 42 – the pre-search form was successfully submitted;
9. Row 43 – the browser was redirected to Google search;
10. Row 44 – the search query "mozart" was entered; result page loaded;
11. Row 45 – the post-search form was displayed, which indicates that the search task was stopped. The URL indicates that this was 1st task;
12. Row 46 – received the search_case_1_2 form with 7 answers;
13. Row 47 – the post-search form was successfully submitted;

According to the log, it took approximately 4 minutes to fill out the demographics form, and then about 15 seconds for the pre-search form. Then the browser was redirected to the search engine page, and afterwards the query "mozart" was entered into the search engine

and the results page was displayed. After this, the search case post-form was opened, because the user selected the Stop button. After 21 seconds of filling out, the form was sent to the server, and the experiment was finished. The search task took 1 minute to complete.

Answers to the surveys are stored in the separate table "sl_answers". The ID of form submitting event is taken from "sl_log" table, and user answers can be found by this ID. For example, in table 5 the ID of the pre-form reception record for the Mozart task is 40. Looking for records with a LOG_ID equal to 40 in "sl_answers" table produces the following information:

ID	LOG_ID	QUESTION	ANSWER
80	40	b	5
81	40	c	5
82	40	d	1
83	40	e	5
84	40	f	5

Table 4. Answers to the search case 1 pre-form in the Search Experiment

- For questions 1, 2, 4 and 5, the user chose the 5th option, which was "Totally agree";
- For the 3rd questions the user gave the answer "Totally disagree".

As questions 1, 2, 4 and 5 were in fact statements such as "I think it will be simple", and the 3rd one was "I think I'll have to use other sites except the search engine", it is clearly understandable that the task looked simple to the users. Similarly, after analyzing the post-form answers for the same task researchers will find that it was really simple.

The demographics form and post-experiment form are handled in the same way.

When a lot of users have completed the experiment, it will be very easy to compile statistics about different tasks. Already, researchers can use SQL queries to ask complex questions such as "How many users aged between 25 and 30 were totally agree that they will have to use other sites when searching for Mozart's birth date?" The query is:

```
/* find a number of users */
SELECT COUNT(DISTINCT sl2.client_id) FROM sl_log sl2
WHERE sl2.client_id IN (
```

```

SELECT DISTINCT s11.client_id FROM sl_log s11 WHERE s11.id IN
/* who are aged between 25 and 30 */
(SELECT sa1.log_id FROM sl_answers sa1 WHERE sa1.question = 'a1'
AND CAST(sa1.answer AS UNSIGNED INTEGER) BETWEEN 25 AND 30))
AND s12.form_id = 'search_case_1' AND s12.action_type = '0'
/* and answered 'Totally agree' ('5') on question about other sites
(marked 'd') */
AND (SELECT sa2.answer FROM sl_answers sa2 WHERE sa2.log_id = s12.id
AND sa2.question = 'd') = '5'

```

The last step is to select a data visualisation framework to present the data in the form of charts and tables.

The application also logs pausing and resuming events. These occur when the user switches to another window or an incoming call is received, and then the focus comes back to the Search Experiment application.

Search Experiment has a permission to add bookmarks to device-wide bookmarks storage. This event is also logged on the server. Usually when a bookmark is added, the page is revisited later. There is a high probability that this page contains a lot of data that requires reading and analysis, so users usually put the page into their bookmarks, trying to find a quick answer from another search result, and in case of failure come back to the bookmarked page. Logging of bookmark addition can help researchers to improve search engine functionality, for example by creating a new analysis system that scans a page with a large amount of data, and searches for the specific information that the user is looking for, using text data mining techniques. It could really improve the process of exploratory search.

ID	CLIENT_ID	CLIENT_IP	ACTION_TYPE	LOCATION_TITLE	LOCATION_ADDRESS	DATETIME	FORM_ID	REQUEST_DATA
35	BBB13006 2a0957	93.40.12.4	1			2011-05-24 22:38:21		array(5) { ... }
36	BBB13006 2a0957	93.40.12.4	9	Demographics form	file:///android_asset/htmls/demog_f orm.html	2011-05-24 22:38:27		array(5) { ... }
37	BBB13006 2a0957	93.40.12.4	0			2011-05-24 22:42:33	demog_for m	array(22) { ... }
38	BBB13006 2a0957	93.40.12.4	10		http://www.noom.ee/~gleb/sl/index .php?a=BBB+User&a...	2011-05-24 22:42:33		array(5) { ... }
39	BBB13006 2a0957	93.40.12.4	6	Google	http://www.google.com/m	2011-05-24 22:42:36		array(5) { ... }
40	BBB13006 2a0957	93.40.12.4	13	Search case description template	file:///android_asset/htmls/search_c ase_1.html	2011-05-24 22:45:19	search_case _1	array(7) { ... }
41	BBB13006 2a0957	93.40.12.4	0			2011-05-24 22:45:33		array(5) { ... }
42	BBB13006 2a0957	93.40.12.4	14		http://www.noom.ee/~gleb/sl/index .php?b=5&c=5&d=1&...	2011-05-24 22:45:34		array(5) { ... }
43	BBB13006 2a0957	93.40.12.4	6	Google	http://www.google.com/m	2011-05-24 22:45:36		array(5) { ... }
44	BBB13006 2a0957	93.40.12.4	6	mozart - Google	http://www.google.com/m/search?q =mozart&mshr=&mbsb...	2011-05-24 22:45:56		array(5) { ... }
45	BBB13006 2a0957	93.40.12.4	15	Search case description template	file:///android_asset/htmls/search_c ase_1_2.html	2011-05-24 22:45:39		array(5) { ... }
46	BBB13006 2a0957	93.40.12.4	0			2011-05-24 22:46:20	search_case _1_2	array(7) { ... }
47	BBB13006 2a0957	93.40.12.4	16		http://www.noom.ee/~gleb/sl/index .php?b=5&c=1&d=1&...	2011-05-24 22:46:20		array(5) { ... }

Table 5. Part of a Search Experiment log (some data is truncated for better appearance)

Chapter 4

Overview of the Search Experiment project

4.1 Implementation details

The Search Experiment client part was developed as an Android Java application and can be classified as a Web App, as it uses the WebView browser component and interacts with web content [20].

Eclipse IDE v3.6.1 with the Android Development Tools extension (ADT) v 10.0.0 was used in the process of making the application for mobile devices, and the Notepad++ text editor was used to write the front-end PHP scripts. Scripts do not use some version-specific aspects of PHP, so they can be easily installed on almost every Web server with PHP4 and MySQL support.

One issue was discovered during implementation and may occur when configuring the server. A standard HttpClient class used for logging in the Search Experiment application sends data by the POST method. It cannot handle redirects correctly, so if the server is configured to redirect from one address to another, data will not be stored. A direct address must be used instead of redirections.

Initially, an Android emulator was used for designing the activity layout and user interface. The emulator uses a QEMU processor emulator [21], and its performance was adequate while using an Intel Core2Duo T5400 processor under Windows 2003. Several sessions were done in a VirtualBox virtual machine with Android X86 [22], but it required some additional steps to get it to work with Eclipse and debugging tools. Emulators support typing on the PC hardware keyboard, which has its own influence on testing the application as an end-user. The final stages of development were done on an LG P500 (LG Optimus One) device with Android 2.2 Froyo, a middle-class device with an ARMv6

processor. The application was also tested on an HTC Desire (Android 2.2, high-class device, ARMv7 processor) and a Sony Ericsson X8 (custom Cyanogen Mod firmware based on Android 2.2, ARMv6 processor). One issue was encountered on the latter two devices. Both used the HTC keyboard application, and there was a problem with showing the on-screen keyboard when the application was launched for the first time. The problem disappeared after an application restart.

The latest version of the application was also tested on Android X86 running in VirtualBox. It worked fine, so theoretically will run nicely on any computer using Android (including Android X86, which can be installed as a fully functional operating system onto a PC's hard drive). Although Android X86 is fast, it is hard to use a touch-oriented operating system on a desktop computer without a touch screen because of the large number of mouse moves required.

The machine used as a Web server had an Intel Pentium D 2.8 GHz processor and ran Debian Linux 2.6.32-30 with PHP 5.3.3 and MySQL 5.1.49.

The application was not localized, containing only English texts and captions.

4.2 Possible improvements for the Search Experiment application and data storage

Web pages containing pre-search and post-search surveys were taken from the Search Logger application's source code [18]. Their HTML code was changed minimally. Further changes can be easily done automatically, using a script in one of the common scripting languages like *bash*, *awk* or PHP.

Another aspect is the page layout and content. The layout can be re-designed to better fit the screen without horizontal scrolling. Of course, any changes made must satisfy

an HTML Doctype, which best suits for mobile devices. The most common one, recommended by Google, is XHTML Basic [20].

As for content, Search Logger interview pages contain a small manual and recommendations for the experiment. These must be changed to reflect the mobile domain of the Search Experiment application.

Due to a large number of different form-factors of Android devices, the information collected from the demographics form can also contain some general device parameters such as screen size, availability of a hardware keyboard, and the version of the operating system used. This data can be added to the form automatically. When collected, it can help the researchers to design improvements for search engines, increasing their usability. Of course, the user must be notified that this data is collected. An End-User License Agreement (EULA) can be added as a link in the demographics form or as a popup message called from the activity menu.

In the current version of the application, the users cannot see their bookmarks and open them. This can be changed by allowing the application to read the user's browsing history and bookmarks, which requires the additional permission `READ_HISTORY_BOOKMARKS` and a corresponding expansion of the user interface. Once a page is visited, it can be added to the bookmarks to be visited again later during the search. This is important when making large exploratory searches that involve information analysis. At the same time, an exploratory search is traditionally conducted on a desktop computer or a laptop. The victorious offensive of the tablet computer can change this trend. Also, there are a lot of applications that synchronize bookmarks between handheld and desktop devices.

Currently, the Search Experiment uses Google Search as the default search engine, as it is better integrated with the Android platform than others. This can be changed to the

way Search Logger works, selecting a random search engine from a small list or from the Search Logger homepage.

Similarly to the Search Logger extension, the Search Experiment shows a list of search cases with numbers only, giving no information about the task's content. This can be improved and there are several different ways to do it.

The first one is simpler: the application has to parse the search cases available and use a content of the HTML *<title>* tag as the task's name.

Another way is to improve the experiment by moving the database to the server side. This opens up significant possibilities for making the experiment longer and more accurate. Questions and statistics about the tasks completed by each participant are stored on the server. The user can choose a task that had been never done before or try something again using new knowledge or another search engine. Surveys can also be stored in a database. Of course this requires a front-end to show the forms, which will be generated from questions' metadata such as questions type (yes/no, set of checkboxes, or text box). Researchers can add new questions and get new data from participants. Support for users adding their own search tasks can also be implemented. Of course, the name of the task will be available from the database and user will see that task topic.

The second way is very promising. Such a database can even be used by both the Search Experiment and the Search Logger, collecting statistics for computers and handheld devices in one place, with an ability to compare data and calculate correlations.

The server side must also have an additional interface to upload data collected from file log and store it in the database, in case Web logging is unavailable. This requires a change in the file format. Current file logging is very simple, and importing that data into a database is not implemented yet.

One of the possible improvements is the possibility to make two and search tasks simultaneously. To add this feature, menu must be re-organized and task selection added for Stop command.

Of course, localization has to be done, and the user language has to be stored in order to get language-specific statistics. This is important when searching, because of different search terms and their length in different languages.

4.3 Analyzing results

Unfortunately it was not possible to involve many people in testing the Search Experiment application. Also, the fourth question (budget wedding) may be too complex to attempt on a handheld device, especially on a smartphone. Two volunteers who tried this application completed the first three questions, but all left the budget question untouched. Both used smartphones, and said it would require too much effort and too much time to look through long texts on a small screen. It was mentioned that one of the possible ways would be to conduct an initial search, adding pages that might contain proper information to the device's bookmarks for later synching with a desktop computer, where the task could be completed. Both users said that they would like to make some searches which can be categorized as "local services", to see how simple it is to find, for example, the closest drug store or administrative office, as it is a very common requirement when the user is "in the field" and has only a handheld device.

All it means that questions used in the Search Experiment need to be more natural for mobile devices. It is difficult to imagine that somebody would make a plan of a wedding when walking around the city. However, it is very likely, for example, that they would search for currency exchange data while travelling by car in a foreign country. Mobile research must have "mobile tasks", and this is the important conclusion from small research made.

Further studies can involve more volunteers, for example students using Android on their devices. As smartphones become a trend [30], they will be used not only by advanced users, but also by casual consumers, with very different levels of search engine and Web search experience.

Summary

The main goal of this work was to conduct an overview of strategies for measuring search tasks and user behavior logging worldwide and at the University of Tartu, with special emphasis on the quickly grown mobile domain, and to introduce a solution for measuring search tasks on the Android platform.

The work describes some known studies on search measuring, including the Search Logger extension for the Firefox browser. It sheds some light on the lack of experiments in the mobile domain and presents its own solution for the popular Android platform called Search Experiment.

The second chapter contains a description of the techniques for research and analyzed results of experiments. The Search Logger extension is described, being a related work from the University of Tartu, and some advantages and disadvantages were highlighted. The lack of existing work in the mobile domain is explained.

The third chapter introduces the Search Experiment research method for the Android platform, describing the strategies used, as well as design and usage of the Android application, server-side front-end and data storage.

The fourth chapter contains material on development details and suggests possible improvements to Search Experiment functionality. A small analysis of conducted experiments is also provided.

The main goal of this work was met. A review of related works is presented; the lack of studies in the mobile domain is explained. Research for mobile platform is described and the Search Experiment application is designed and implemented. The results of this work can be used for further research.

Mobiilsete otsinguülesannete mõõtmine Androidi platvormil

Bakalaurusetöö (6 EAP)

Gleb Štšenov

Resümee

Käesoleva bakalaureusetöö põhieesmärk on anda ülevaade otsinguülesannete mõõtmisest ja kasutaja käitumise logimise strateegiatest üle maailma ja Tartu Ülikoolis. Autor pöörab erilist tähelepanu kiiresti arenevale mobiilvaldkonnale ning esitab lahendusi otsinguülesannete mõõtmiseks Androidi platvormil.

Bakalaureusetöös on kirjeldatud tuntud uuringud otsingu mõõtmistulemuste leidmiseks, siia hulka kuulub ka Search Logger laiendus Firefox veebilehitseja jaoks. Töös on selgelt välja toodud mobiilvaldkonna uuringute vähesus ning esitatud autoripoolne lahendus populaarse Android mobiilplatvormil nimega Search Experiment.

Teine peatükk sisaldab uurimismetoodikate kirjeldust ja sooritatud eksperimentide analüüsitud tulemusi. Uuringu aluseks olev Search Logger laiendus on esitatud Tartu Ülikoolis tehtud uuringus, ning mõned selle eelised ja puudused on lahti seletatud. Antud peatükis on välja toodud ka mobiilvaldkonnas tehtud uurimustööde puudus.

Kolmas peatükk tutvustab Search Experiment uuringut Androidi platvormi jaoks, kirjeldades kasutatud strateegiad ning Androidi rakenduse ja serveripoolse liidese ning andmekogu disaini ja selle kasutamist.

Neljandas peatükis on välja toodud informatsioon Search Experiment mobiilirakenduse arendamise detailidest ja autori poolt pakutud võimalikud parendused Search Experiment funktsionaalsuses. Samuti toodud väike analüüs tehtud eksperimentidest.

Töö põhieesmärk on saavutatud. Seotud tööde ülevaade on tehtud ning mobiilvaldkonnas uurimustööde puudus välja toodud. Mobiilplatvormi uuring on kirjeldatud ja Search Experiment rakendus disainitud ja arendatud. Töö tulemusi saab kasutada edaspidistes uuringutes.

References

- [1] The Nielsen Company's blog, Top U.S. Search Sites for July 2010, August 25, 2010
http://blog.nielsen.com/nielsenwire/online_mobile/top-us-search-sites-for-july-2010/ 18.04.2010
- [2] Wikipedia, Web search engine
http://en.wikipedia.org/wiki/Web_search_engine [18.04.2011]
- [3] Android Developers Youtube channel
<http://www.youtube.com/watch?v=fqFpq9WXbJo> [18.04.2011]
- [4] G. Singer, U. Norbistrath, E. Vainikko, D. Lewandowski, H. Kikkas: Search-Logger — Analyzing Exploratory Search Tasks. Submitted and accepted to 26th Symposium On Applied Computing (SAC'11), Web Technologies track, March 21-25, 2011, TaiChung, Taiwan
- [5] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz. Analysis of a very large web search engine query log. In ACM SIGIR Forum, volume 33, page 6-12, 1999
- [6] C. Hölscher, G.Strube. Web search behavior of Internet experts and newbies. In Computer Networks, volume 33, pages 337-346, 2000
- [7] R. Navarro-Prieto, M. Scaife and Y. Rogers, Cognitive strategies in web searching, in: Proc. 5th Conf. on Human Factors and the Web, June 1999 [WWW document],
<http://zing.ncsl.nist.gov/hfweb/proceedings/navarro-prieto/index.html> [21.04.2011]
- [8] S. Fox, K. Karnawat, M. Mydland, S. Dumais and T. White. Evaluating implicit measures to improve web search. ACM Transactions on Information Systems (TOIS), 23(2):147-168, 2005
- [9] Y. Cui, V. Roto, How people user Web on mobile devices. Proceeding of the 17th international conference on World Wide Web, 2008

- [10] Church, K., Smyth, B., Cotter, P., and Bradley, K. 2007. Mobile information access: A study of emerging search behaviour on the mobile Internet. ACM Trans. Web, 1, 1, Article 4 (May 2007), 38 pages.
- [11] M.Kamvar, S.Baluja. A Large Scale Study of Wireless Search Behavior: Google Mobile Search, CHI '06 Proceedings of the SIGCHI conference on Human Factors in computing systems, pages 701-709, 2006, ACM
- [12] Mozilla Wiki – Mobile/Fennec/Extensions
<https://wiki.mozilla.org/Mobile/Fennec/Extensions> [14.05.2011]
- [13] Maemo – <http://en.wikipedia.org/wiki/Maemo> [14.05.2011]
- [14] Mozilla Wiki - Mobile/Platforms/Android
https://wiki.mozilla.org/Mobile/Platforms/Android#System_Requirements
[14.05.2011]
- [15] ARM architecture http://en.wikipedia.org/wiki/ARM_architecture 14.05.2011
- [16] NPAPI - <http://en.wikipedia.org/wiki/NPAPI> [15.05.2011]
- [17] Android Developers: Platform versions
<http://developer.android.com/resources/dashboard/platform-versions.html>
[15.05.2011]
- [18] SVN Commit of Search Logger for Firefox plug-in
<svn://www.dougdevel.org/misc/publications/theses/Master/HannuKikkas/>
[25.05.2011]
- [19] Simple information interface for Search Logger
<http://server.search-logger.com/index.php> [18.05.2011]
- [20] Web Apps Overview
<http://developer.android.com/guide/webapps/index.html> [25.05.2011]

- [21] QEMU processor emulator
<http://en.wikipedia.org/wiki/QEMU> [26.05.2011]
- [22] Android X86 Project
<http://www.android-x86.org/> [26.05.2011]
- [23] Maemo – <http://en.wikipedia.org/wiki/Maemo> [29.05.2011]
- [24] MeeGo 1.2 Developer Edition for Nokia N900, MeeGoConf at San-Francisco, 23-25 of May, 2011
http://www.youtube.com/watch?v=Q4vN_Xn0jq0 [30.05.2011]
- [25] B. J. Jansen, R. Ramadoss, M. Zhang, and N. Zang. Wrapper: An application for evaluating exploratory searching outside of the lab. EESS 2006, page 14.
- [26] Neil Gandal, The dynamics of competition in the internet search engine market, International Journal of Industrial Organization, Volume 19, Issue 7, July 2001, Pages 1103-1117
- [27] A. Rangaswamy, C. L. Giles, Silvija Seres, A Strategic Perspective on Search Engines: Thought Candies for Practitioners and Researchers, Journal of Interactive Marketing, Volume 23, Issue 1, February 2009, Pages 49-60
- [28] Objective-C, <http://en.wikipedia.org/wiki/Objective-C> [13.06.2011]
- [29] Xcode <http://en.wikipedia.org/wiki/Xcode> [14.06.2011]
- [30] Increased penetration of smartphones will drive 3G traffic
<http://www.unwiredinsight.com/penetration-smartphones> [14.06.2011]
- [31] The Most Common HTML and CSS Mistakes to Avoid
<http://webdesignledger.com/tips/the-most-common-html-and-css-mistakes-to-avoid>
[14.06.2011]
- [32] Birthday problem
http://en.wikipedia.org/wiki/Birthday_problem [14.06.2011]

Appendices

Appendix A. The Search Experiment user guide

1. Installation notice

Because Search Experiment is not distributed through Android Market, which is an official place to download application from, an installation from unknown sources should be allowed in order to install Search Experiment application. This can be done by checking "Unknown sources" check box in Settings – Applications:

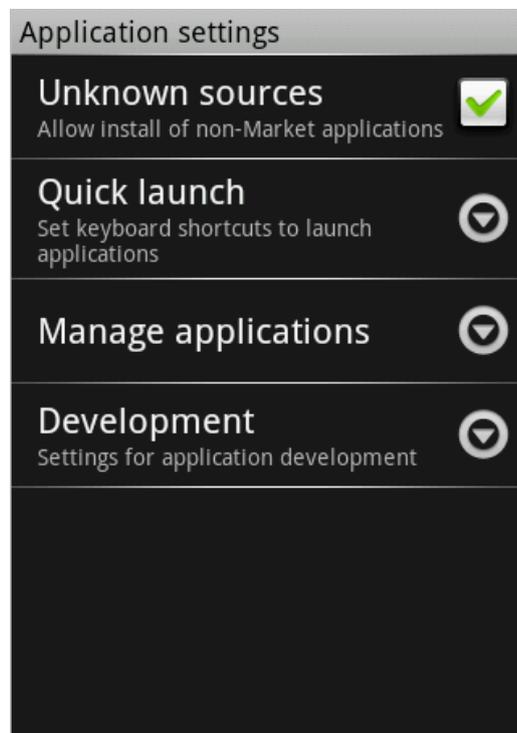


Figure 7. Enabling installation from the unknown sources in Android

User can disable this after installation to disallow to install programs from unknown sources later.

2. Using the Search Experiment

Application main window (or Activity in Android terminology) consists of address field, a button to open an address typed in address bar, and an area where web page is displayed.



Figure 8. The Search Experiment main window

The browser area supports zooming features as standard Android Web browser application, so the user can zoom in and out by multi-touch, by on-screen controls or quickly switch between zoom and fit to page modes by double-touching the page.

First launched, application asks user to fill out a demographics form.

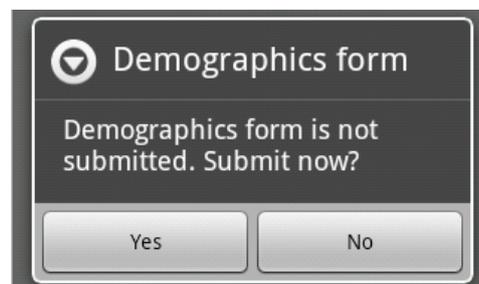
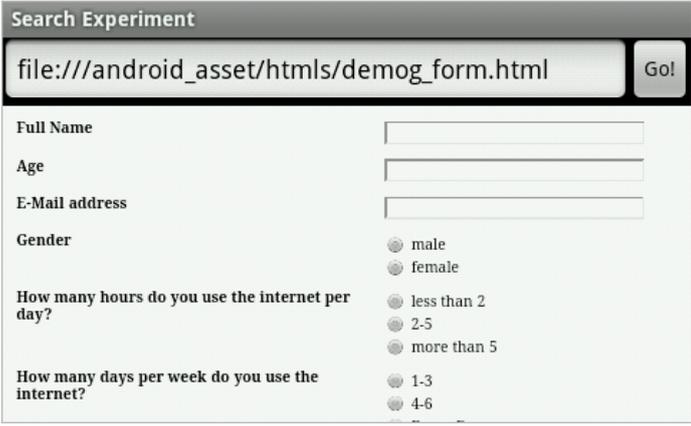


Figure 9. Confirmation dialog for submitting demographics form

It contains fields for name, age, sex and contact information, such as e-mail. Also a short survey is conducted about the user's experience in using Internet and search engines, and questions about the user's comfort level while carrying out search tasks on the Internet. The user has to fill out the form and press "Submit the demographics form" button in the end of the page.



The screenshot shows a mobile browser window titled "Search Experiment". The address bar contains the file path "file:///android_asset/htmls/demog_form.html" and a "Go!" button. The form includes the following fields and options:

- Full Name:
- Age:
- E-Mail address:
- Gender: male, female
- How many hours do you use the internet per day?: less than 2, 2-5, more than 5
- How many days per week do you use the internet?: 1-3, 4-6

Figure 10. Demographics survey

Then browser is redirected to Google Mobile search page.

To begin the experiment, user has to start some search case. It can be done by pressing Menu key of Android device and selecting Start command from application menu:



Figure 11. Application menu

Application will show a list of available search tasks:



Figure 12. List of search cases available to user

After a search task is selected, a pre-search survey similar to demographics form will be conducted. The task itself is described at the beginning of the survey. The user is asked about his expectations of the given task: supposed number of queries required, supposed amount of time to complete the task, and some other questions.

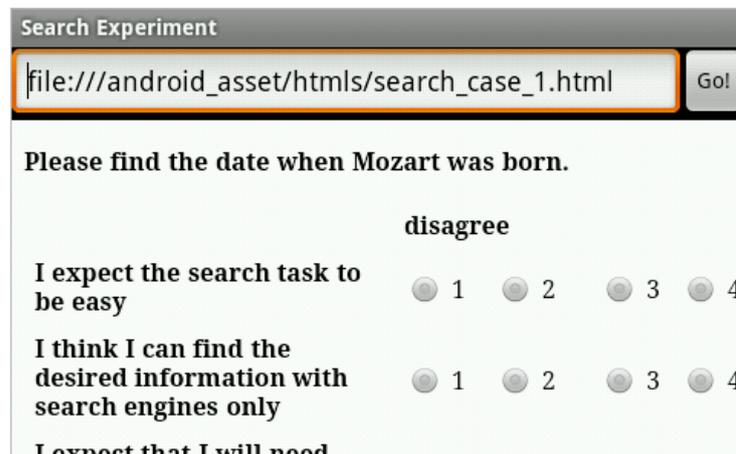


Figure 13. Search case pre-form

After pressing "Start the search case" button in the end of the form, user will again see Google Mobile search page. Now user can enter a query into the text box on displayed

page, or open some other page by typing its address to the address text box and pressing "Go" button to open it.

When user decides that search task is completed, the Stop command from application menu has to be selected:



Image 14. Application menu with active search case

When Stop is selected, a post-search survey is displayed. There are questions about the process of searching and the user satisfaction with the results.

After all tasks are completed, a confirmation dialog appears which allows user to fill so-called "final form". This is a survey about user experiences with the Search Experiment application. It asks users about things they liked in experiment and about possible improvements in program.

3. Adding bookmarks

During the search user can add bookmarks to system bookmarks storage. They can be revisited later using system browser. Bookmark adding is done by selecting Add bookmark command from application menu. A dialog window appears providing the possibility to set the caption of new bookmark before storing:

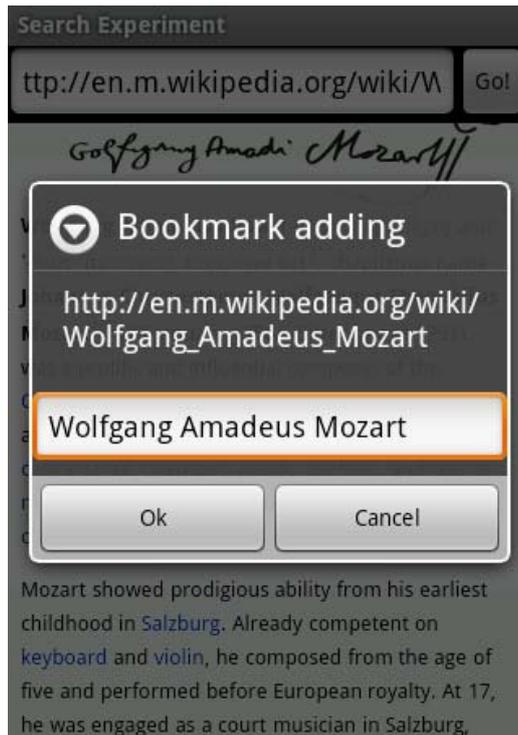


Figure15. Adding bookmark

Application can be closed by selecting Exit command from menu.

4. Preferences

Application has preferences page, which can be opened by selecting Settings from menu.

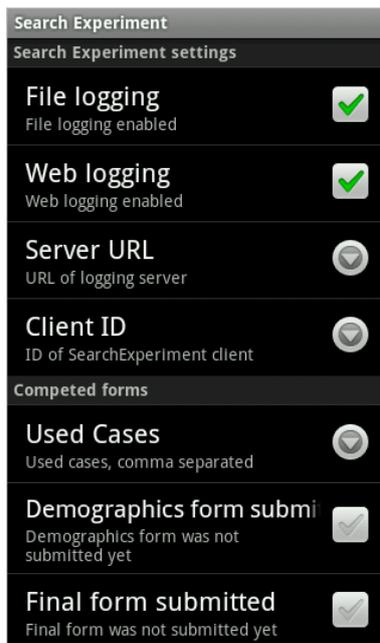


Figure 16. Preferences window

File logging and Web logging check boxes enable / disable appropriate ways of logging of user's activity.

Server URL is an address of logging server script.

Client ID is currently used identification of client. It can be changed if such a need arises. By default, ID is generated from timestamp combined with random integer number.

"Used cases" entry is a text value containing the numbers of search cases completed. If user wants to make some search task again, appropriate task's number can be removed from this field.

Demographics and final form submission check boxes allow user to fill out appropriate forms again.

Switching from Preferences view back to the main view is done by pressing device's Back button.

Appendix B. The source code and installation package of the Search Experiment (client and server)

The code is located on the compact disk attached to the thesis.