Joint Master in Computer Science
Department of Informatics
University of Fribourg

Multimodal Interfaces
Spring Semester 2013

SmartBrowse

Written by
Schaller Georg[1]
Kläy Pat[2]

May - 2013

[1]georg.schaller@unifr.ch
[2]pat.klaey@unifr.ch

Contents

Abstract

SmartBrowse is an approach to use several modalities for a remote control of a web browser. The basic idea is to replace abstract tools like a keyboard or a mouse completely and use for human beings intuitive methods to command the browser instead, like speech or gesture recognition. The concept as well as the realisation and the evaluation of SmartBrowse are discussed briefly.

1 Introduction

There was a time when only a couple of people were able to use the internet due to some technical difficulties. The intention of the browser as a client made it possible for a bigger range of users to connect to the internet. Even though these browsers were very primitive in terms of functionality and user interaction at that time, this lack has not been closed yet. So we wanted to make a first step onto the direction of a new browse experience, using only the basic modalities of human beings as communication medium only, rather than the abstracted mouse and keyboard modalities. What does that actually mean, human being? We use this term to consider each way of direct and physical interaction of a human being to another one. And in this case, the other one is not another person but a virtual device interpreting our output.

Studying the different ways of human to human interaction one can probably claim that the most intuitive and easiest communication techniques are verbal- as well as non verbal speech and gesture. Thus the most intuitive modalities that can be used for interaction between a human user and a systems might be for example speech, gestures, touches, pressure and of course also all possible combinations of these outputs. This would then be referred as multi modal, since we are dealing with multiple modalities as representation for an order or another input to our system. We consider these kind of user output and system input respectively as very intuitive for human beings since all these modalities are part of its daily life.

The idea is that some device recognizes the output of the user and sends it to the computer. In the computer a server application is waiting for such an input to transform it into known orders for the web browser.

As a concrete example we could want to have a system, which opens a website in a browser if I say "open " followed by the address of my website.

To navigate back to the previous page another human being based interaction modality could be used, for example a movement / gesture of a hand holding a device to the left side. Consecutively the navigation forth would then be represented by a movement of the hand to the right side. Since these movements are much more comfortable if the device is wireless connected to the computer instead of a wired connection, both - the computer and the device - have to be enabled for a bluetooth connection.
A complete list of all modalities and their meanings used in SmartBrowse is shown in chapter 3.
Having explained what we refer to when being talking about human intuitive outputs we can now couch the goal of this system in terms. SmartBrowse should be able to allow browsing in the internet without any help of a mouse or a keyboard but only with human outputs. We consider these outputs as much more intuitive for the user than using an abstract model as a mouse for example.

2 Architecture

The main questions about the technical issues were how to choose a device which is able to measure the user outputs and which software should be used or developed.

2.1 Hardware

For economical as well as for practical reasons we decided to develop SmartBrowse on a smart phone. That makes sense since the number of different sensors in a modern mobile phone is beyond the scope of the needs for SmartBrowse. On the other side a smart phone is not as expensive as some specific sensor devices might be and most people which are browsing in the net nowadays have access to a mobile phone.
Along with the sensors a SmartBrowse compatible smart phone needs bluetooth for connection to a computer and for reasons of performance also WIFI. So the final device we have chosen for SmartBrowse is a phone running android operating system.

2.2 Software

The server part of SmartBrowse is a Java application and thus operating system independent. Another main reason to use Java was the fact, that the client application on the smart phone is running with Java anyway. The whole software of SmartBrowse can be divided into three main parts. As mentioned above a client and a server part and in addition a driver part which is running a browser (in this case Mozilla Firefox[3]).

1. The client application which is running on the mobile phone is responsible to recognize and record the users output and to send it to the server application. This client is based on the Android SDK for developers of android software. The SDK provides a lot of functionalities to access the sensors of the mobile phone.

2. The server application which is running on a computer with a browser. The server is constantly waiting for some commands coming from the mobile phone. As soon as a command is being received, the server application is responsible for translating this command into a valid order for a browser driver. The server needs a bluetooth connection to the client running on the mobile phone. So we used the bluecove[4] library for Java for the connection between the phone and the computer. Each command from the phone is first translated into a JSON[5] object and then sent as JSON-String to the server. The server then has to interpret the JSON-String in order to understand the command.

3. The browser driver finally executes the command and displays the result in the chosen browser window. The driver is a selenium[6] application that allows the control of a browser using Java commands.

Type	User input	Command	Examples
Speech	"Open ..."	Opens the specified website	"Open Wikipedia.org" "Open Google Maps"
Speech	"Search ..."	Making a google web search for specified topic	"Search Niagara falls"
Gesture	Accelerate to the left	Navigate backward	
Gesture	Accelerate to the right	Navigate forth	
Gesture	Two finger disperse (figure 2)	Zoom in	
Gesture	Two finger approaching (figure 2)	Zoom out	
Touch and Gesture	Touch and tilt in a Direction (figure 3)	Scroll to the chosen direction	
Speech and Touch	"Show links" and click	Open a link	
Speech and Touch	"Show buttons" and click	Press a button	

Table 1: Listed SmartBrowse commands.

3 Modalities

3.1 Voice

Voice is one of the most common instrument for human interaction. Thus we decided to use voice / speech as an input to our system. We enabled SmartBrowse to understand simple spoken commands as for example "open wikipedia.org", "search Niagara falls", "show links" and "show buttons". A detailed explanation of these commands can be found in table 1. To have some good skills in voice recognition we use the built-in speech recognition

[3] `http://www.mozilla.org/de/firefox`
[4] `http://bluecove.org`
[5] `http://www.json.org`
[6] `http://docs.seleniumhq.org`

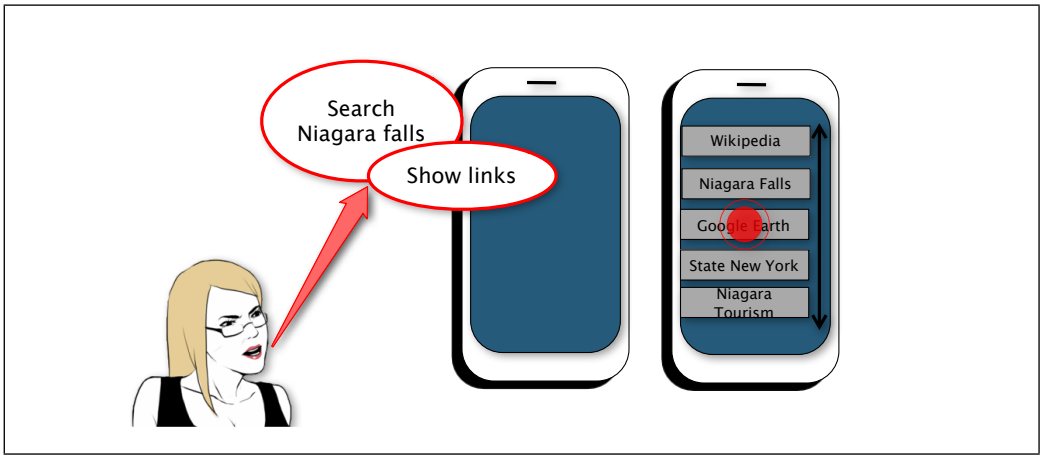

Figure 1: A user is searching for a google earth map of the Niagara falls. The red dot shows a touch contact.

software from google. The only disadvantage of this approach is, that the smart phone needs a internet connection to be able to run speech recognition. So a WIFI connection increases the performance of the recognition in terms of time. The speech command mode has to be activated by a touch before a command can be spoken.

3.2 Touch

Touch is used primary for navigation between different activities within the client application. So touches allow to switch to speak mode or to scroll mode respectively. In addition touch is used in combination with voice (for example to show and open links and buttons) or in combination with gesture to scroll on a web side. So if a user wanted to browse a website using the links the speech command "show links" enables a view on the smart phone where each link of the website is shown as button which can be clicked using touch (see figure 1).

3.3 Gesture

Gestures are used for zooming in or out on a website (see table 1). This is useful especially for web applications as google maps. You can simply use two fingers with dispersing or approaching movements to zoom in or out re-

spectively (see figure 2).

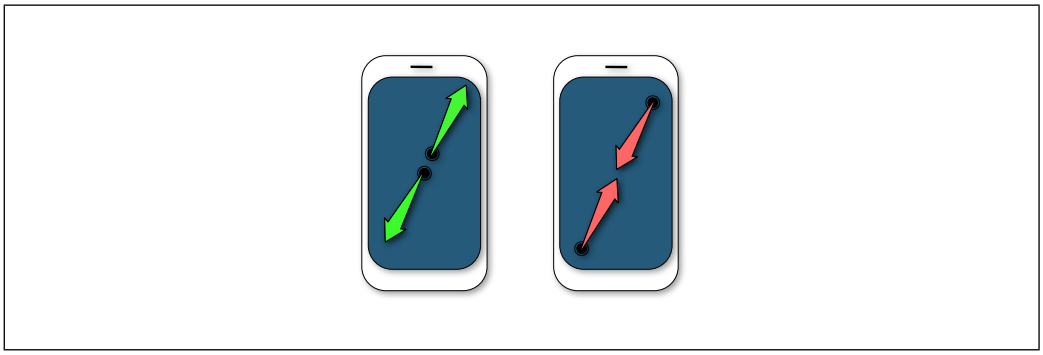

Figure 2: Zoom in (left) and zoom out (right) in SmartBrowse with two fingers

Also scrolling is possible in google maps as well as on every other website. It is enough to touch the the surface of the phone and to tilt it into a direction to scroll into the same direction within the website (see figure 3). As soon as the touch is released, the scrolling stops immediately.

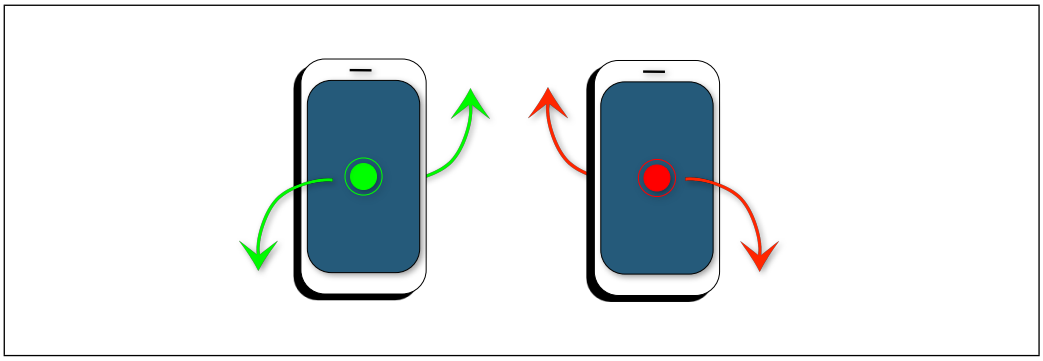

Figure 3: Scroll to the left side (left) and scroll to the ride side (right) in SmartBrowse while one finger pressed

3.4 CASE and CARE

Regarding CASE and CARE the following concepts can be found in our application:

CASE

On the machine side we can find:

Exclusive

Most of the tasks are executed one after the other without influencing each other.

Synergistic

To scroll: Using touch and gesture at the same time, touch activates the gesture recognition but touch needs to stay active.

Alternate

The activation of the voice modality is done using the touch modality. But they are not active at the same time.

CARE

On the human side we can find:

Complementarity

In parallel: To be able to scroll the user needs to touch the display and tilt the device at the same time. Only one of this modalities individually is not enough to fulfil the task.
Sequentially: When a user wants to click on a link he first has to say "show links" and finally he can click on the appearing buttons on the device which are representing the links.

Assignment

Switch between different views of SmartBrowse. Another example is to open a website. This task can only be done using the speach command "open".

Equivalence

Say "back" or use gesture to navigate back in the browser.

4 Problems

During the project we faced a lot of problems which needed to be solved. Most of them could be solved within a short brainstorming session, but some are still not solved.
One of the major problem was to correctly parse and analyse the content of a

given website. Even though we used external libraries (for example JSoup[7]) it still was very tricky to collect all links, buttons, textfields etc. Since every website is different and partially contains JavaScript, there was not a general rule to detect all elements we were looking for. There was always a link which was interpreted as button or vice versa. This also made it complicated for the user to interact with the system because he did not know whether what he sees is a button or a link. Especially images were hard to parse. If there is a href attribute specified in the image tag, should we treat the image as a link or as a button.
JavaScript was another uncomfortable thing to deal with. When some JavaScript action was triggered, the selenium webdriver did not always realize that the state of the webpage changed. Sometimes it was even not possible to execute the JavaScript functions (for example when a given area of an image should be clicked).
These problems actually created a new one. To solve them, there was a lot of logic needed. We spent hours trying to parse the google results and almost lost sight of the subject which clearly was Multimodal Interfaces and not Artificial Intelligence.

But we also faced problems regarding the modalities. When using the accelerometer sensors in Android, the developer can control the update rate only in a very limited way. In combination with the bluetooth connection this leads to the problem that either the update rate was too fast, so the bluetooth connection could not properly handle the traffic or the update rate was to slow so the user could already notice the delay.
Also password fields brought a lot of trouble. If you enter your username and password in a browser it is not a big deal. But in our application, most input is given by speech and no one is really willing to speak out loud his or her password. To solve this problem we thought about creating some sort of dictionary, which will map a given keyword (for example "password one") to the real password. So at the first time the user needs to enter this password, he can click some button, speak the keyword and then enter the password on the mobile phone using its keyboard. But there was no time to implement this functionality so the password problem still occurs in our application.

[7]http://jsoup.org/

5 Evaluation

Having our prototype finished, we evaluated and compared it to the mobile phone built-in browsers. Therefore we asked the test users to fulfil the two following tasks and measured the time it took to finish them:

1. What is the population of the US-State in which the Niagara Falls are?
2. Locate the airport of Santa Monica (Los Angeles) on a map.

We choose these tasks because they represented for us the main activity in using the internet: Look up some information and locate something on a map. We would have liked to test some activities on a social-media platform but due to the problem we faced with the password (c.f. chapter 4) and the fact that we cannot ask/force our test users to write something on facebook or twitter (or they do not have an account), we dropped that idea.
We asked 12 people to do the mentioned tasks, six people used their own mobile-phone and the built-in browser (also for the second task we asked them to only use the browser) and six people used our application. In each group, three people started with task two, and the other three persons started with task one. The people using our application could test it 5 minutes before they had to do the tasks. And here are the results:

	Task 1		Task 2	
	Mobile phone	Application	Mobile phone	Application
User 1	48 sec	150 sec	49 sec	30 sec
User 2	250 sec	109 sec	58 sec	33 sec
User 3	221 sec	155 sec	118 sec	75 sec
User 4	67 sec	261 sec	127 sec	61 sec
User 5	70 sec	235 sec	37 sec	70 sec
User 6	66 sec	81 sec	71 sec	73 sec

Table 2: Time to achieve the given task

The people which were using our application, were asked some questions about the usability and the overall experience of the application. The feedback we got was very interesting. Mostly they mentioned that the application is working quite good, but they could not think of using it instead of the built-in browser of their mobile phone. They argued about the speech recognition

which is difficult to use in public places where it might be embarrassing to talk to your mobile phone. On the other hand, they really liked being able to control the browser on a big display with the mobile phone.

5.1 T-Test

With the raw data we performed a t-test to statistically test whether there is a difference between our application and the mobile phone built-in browsers. As null hypothesis we say that there is no difference and $\alpha = 0.05$. The result of the unpaired two-tailed t-test is 0.36 for Task 1 and 0.28 for Task 2. In both cases the result is clearly bigger than α which means that we cannot reject the null hypothesis. So statistically and in respect to the time it takes to complete the two tasks there is no difference between a mobile phones built-in browser and our application.

5.2 Interpretation of the result

Interpreting the observed and calculated results is important. And for this application it is even more. While watching the test users do the tasks we could observe a lot of interesting things which on one hand explain the result but on the other hand also show, that we should not rely too much on these number:

1. The first task (look up some information) was solved in different ways. The first step was equal for all users: take googles help. But from then on, every person had their own strategy how and what to search for. So the result more depended on what some was searching for and not primary whether it was with our application or the mobile phones browser.

2. The way how people use their phone is completely different. You could immediately say, whether a person is using his or her mobile device only for phone calls or whether the phone is also used to browse the internet. So obviously the time to complete the tasks fluctuates a lot (c.f. Table 2: User 2 and 3 for Task 1 with the mobile phone).

3. The language was a (big) handicap. While some persons did not have any problem to use the voice recognition (or the other way around, the voice recognition system had no problem to understand them),

there were other users which had to repeat the words or sentences they said over and over again until it was recognized correctly. People with mother tongue (swiss-)german pronounce english words different than native english speakers do.

6 Conclusion

As stated earlier, our goal was to create an application which could replace the common interaction methods (Keyboard and Mouse) to control a web browser. Of course this was a very ambitious goal and we soon realized that this was not possible. So we focused on mobile devices and said, that our application should replace the browsers on the mobile phones. Also this still seemed to be ambitious but at least we could go for it.
With the results of chapter 5.1 we can see, that the application could replace the mobile browser at least in respect to the efficiency of browsing the web. But considering the users feedback and also the still open problems (c.f. section 4) there is still a lot to do to make the application better. Now, after the project and having seen chapter 5.2, we know that it is not enough to just compare the time it takes to complete two task. We should have done more and also other test to be able to say whether or not our application can replace or at least fight the mobile phones built-in browser. But the results of section 5 also show, that our application could be used in an other field. It could be used for people with handicaps, it could help them to use the web in an efficient way. If we combine the voice recognition with a text-to-speech engine, blind people could "easily" browse the net. Or quadriplegic could also use the voice modality to access the web. Yet another group would be people with an amputated arm, they could use all the modalities and might be faster and more efficient than with the traditional mouse and keyboard.

List of Figures

List of Tables

References