A Voice-Controlled Web Browser to Navigate Hierarchical Hidden Menus of Web Pages in a Smart-TV Environment

Sungjae Han*, Geunseong Jung†, Minsoo Ryu†, Byung-Uk Choi† and Jaehyuk Cha†

*Department of Electronics Computer Engineering
Hanyang University
Seoul, South Korea
sjhans@hanyang.ac.kr

†Department of Computer Science and Engineering
Hanyang University
Seoul, South Korea
{aninteger, msryu, buchoi, chajh}@hanyang.ac.kr

ABSTRACT
This paper proposes a new voice web browser that can be operated in smart TV environments. Previous voice web browsers had the limitation of being run under limited conditions; for example, a list of the specific contents of a page was outputted by voice, or the user entered a search term by voice. In our method proposed in this paper, all the hierarchical menu areas on a web page are recognized and controlled with voice keywords so that page navigation according to a menu can be conveniently done in a voice supported web browser. Although many studies have been conducted on web page menu recognition, most of them provide insufficient information to recognize the hierarchical menu structure. In other words, most web pages in recent browsers showed submenus only as a result of a specific user interaction, since these previous studies had no way of recognizing or controlling the submenus. Therefore, in the web browser proposed in this study, a hierarchical menu structure, which is inserted dynamically via user interaction, is recognized and selected by voice, thus making it possible to maneuver on the web page. Furthermore, the core code of the browser is implemented in JavaScript, so it can be flexibly used not only for a web browser on Smart TVs, but also as functional extensions of existing web browsers in a PC environment.

Categories and Subject Descriptors
H.4.3 [**Communications Applications**]: Information Browsers

General Terms
Design, Experimentation, Human Factors

Keywords
Browser, HTML, Voice, Web

1. INTRODUCTION
Most web navigation on modern smart TVs (e.g., Google TV, Apple TV, etc.) is controlled through a combination of handheld remote and voice input. While standard voice commands support search term entry and hyperlink navigation, they currently offer no means of navigating the hierarchical menus found in many modern web pages. Given the importance of these menus to common website navigation, this deficiency in voice command is especially acute [1].

Some progress has been made in analyzing the rendered portions of a webpage for reference by user commands. Notably, the VIPS (Vision-based Page Segmentation) algorithm can collect content nodes according to visual associations, DOM (Document Object Model) structure, HTML (HyperText Markup Language) tag attributes, etc. [2]. Unfortunately, website sub-menus are not always fully rendered. In many cases, when a user selects a top-level menu item, the sub-menu for that item is rendered dynamically. Furthermore, since nodes in the upper and sub-menus are not always adjacent or placed in a containment relation within the DOM, they may not respond to VIPS detection. Clearly, a more sophisticated means of detection is needed.

WWW'14 Companion, April 7–11, 2014, Seoul, Korea.
ACM 978-1-4503-2745-9/14/04.

2. RELATED WORKS

2.1 WSR
WSR (Windows Speech Recognition) is available for web page navigation in many versions of Microsoft Windows. Among the WSR voice commands useful for web browsing are "Show Number" and "Mousegrid." "Show Number" displays all hyperlinks currently on screen and in menus. By speaking the number (i.e., index) of one of these links, the user moves the control pointer to the location of that link. "Mousegrid" divides the screen into 3x3 grid areas. By speaking the number of one of these areas, the user shifts focus to that area, and can further refine focus recursively. Note that using this command to achieve a particular goal on a webpage can be cumbersome [3].

2.2 VIPS
A number of methods have been proposed for splitting up the contents of a webpage according to topic or purpose. Many of these methods analyze HTML/DOM tree structure to partition information hierarchically. For example, VIPS (a Vision-based Page Segmentation Algorithm) forms information blocks by collecting nodes that are determined to have similar meaning visually, based on DOM structure and HTML tag attributes [2]. These blocks are then reconstructed by a separator that splits the page further into several blocks through repetitive node navigation and share significantly the gap between each block visually. In this way, VIPS bridges the gap between DOM structure and semantic structure on the VIPS page. Unfortunately,

the algorithm has no way to detect nodes that are inserted dynamically in response to user interaction, since these changes are not reflected in the static document source.

3. ARCHITECTURE

The structure and behavior of our voice-controlled web browser for smart TVs are shown in Figure 1.

When a page is loaded into the browser, menus inside the page are detected using VIPS. Any menu element with a CSS property of "visibility:hidden" or "display:none" is classified as a sub-menu candidate group. This candidate group corresponds to a hyperlink object anchored by text values or an image resource. The characters displayed for each menu object are extracted using optical character recognition (OCR). If users enter unique words that correspond to menu objects, the system will dynamically trigger a "mouseover" JavaScript event for that menu object, as would occur when a mouse pointer moved over the object. The CSS properties of the candidate group for previously classified sub-menus are used to detect visual object changes, such as when a hidden object is made visible, and cues a return to the voice input step.

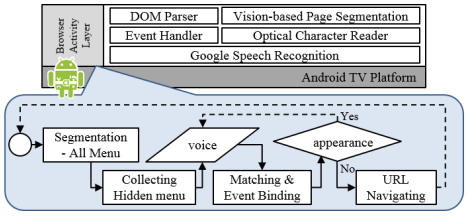

Figure 1 System architecture of the proposed browser

4. IMPLEMENTATION

4.1 Environment

The voice browser proposed in this paper is primarily for smart TVs. In order to check whether the proposed method can be used on other platforms such as PCs, an expanded module for general web browsers was additionally implemented.

First, a browser app for smart TVs was developed by utilizing the 'webView' component in Android. The basic functions of a web browser were implemented the same as in the other one, whereas a menu area detection and control module were implemented such that they could be inserted dynamically. For voice input, the basic API of Google Android was used, and a menu area detection and control module were implemented using only JavaScript. In order to check the operation of the implemented application, we performed an experiment using the Android 3.2 version for Google TV 2.0 [4].

Next, implementation of a voice browser in a PC environment was completed using the Google Chrome extension [5]. The extension of Google Chrome is a small plug-in program that is installed for functional expansion of a browser. We added a voice support function to the existing Chrome browser as an extension. All the codes for menu detection and control were inserted onto a web page dynamically via control of Content-Scripts in Chrome. Therefore, menus on a web page displayed in Google Chrome could be controlled by the user's voice.

The functions of menu detection and event binding were implemented using JavaScript, and the functions of voice recognition, such as TTS, were implemented using an API that was produced externally. In the Android-based environment, Google TTS and Google Voice for Android were used, whereas an API provided by Google Chrome was used for a browser running in a general PC environment [6][7].

4.2 User Interface

Figure 2 shows an example of a user interface screen as proposed in this paper. In the figure, 2-1) shows a basic browser address bar and page-move buttons. 2-2) indicates a basic main menu area, and provides a function with which submenus, which are activated by the user's voice, can be verified visually, using a box symbol in the object. In addition, 2-3) shows a button for performing a detailed environment setup with regard to the browser and voice control functions. Finally, 2-4) shows a console area, where a developer can check the contents of the user's voice that was inputted, or some other processing procedure.

Table 1 Scenario example

Step	Description
1	Open Favorites: The user opens the Favorites by voice. In this scenario, the page moves to the e-Bay site, as shown the first screenshot in Figure 2.
2	Grid menu activation: The main menu area is detected from the page, and is indicated as a green box.

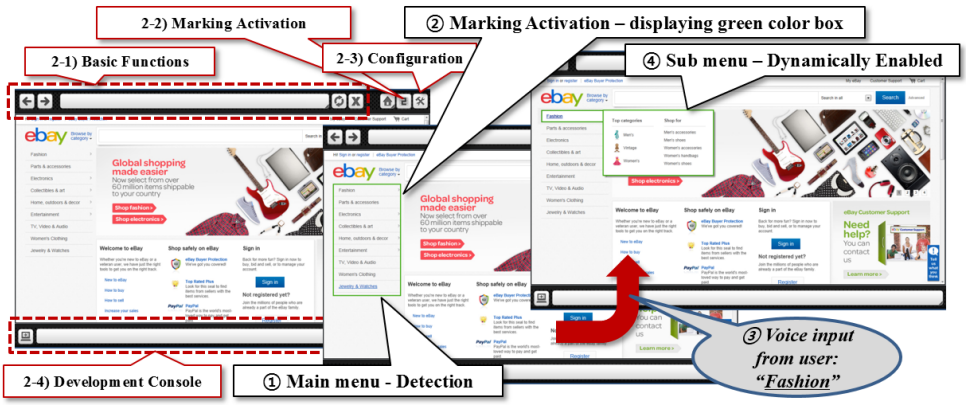

Figure 2 Example screenshots of the user scenario

3	Visual check 1: When a mouse cursor is placed over the e-Bay main page, the submenus are displayed within the main menu. However, as in a PC environment, without using a mouse, the user cannot see the submenus.
4	Voice input for main menu selection: When the user attempts menu navigation via voice input, the submenus are displayed to the user via control of a page event if submenus are detected from the corresponding main menu.
5	Activation of submenus: If the submenus on a page are detected dynamically, the browser waits for the user's command without performing further page navigation.
6	Voice input for selection from submenus: The submenu links are activated via a user voice command.
7	If no more submenus are found on the final selected menu, one moves to a corresponding hyperlink page.

An example of a scenario in which a page is navigated by a voice command using the proposed browser, is shown in Table 1 and Figure 2.

5. EVALUATION

Existing web browsers and voice input can be used to select static main menu output on screen, but not dynamic sub-menu output. However, in case of WSR to be supported by Microsoft's Windows, sub-menu choice is possible in other browsers as well as Explorer. Furthermore, WSR has the advantage of universal availability. Unfortunately, when executing WSR's "Show Number" command, invisible content becomes problematic, since the number of the link object is meant to be displayed as an overlay of something rendered on screen. Furthermore, when trying to access sub-menus with "Mousegrid," the procedures are very complicated and there is a drawback to ask users for more than voice commands. The alternative is to select top-level menu items by entering the corresponding text through voice input, and then selecting the appropriate submenu item. Even if the text of the menu is rendered as an image rather than character output, keywords could be detected through the OCR module. Table 2 shows qualitative comparisons of proposed system and existing browsers.

Table 2 Qualitative comparison of proposed system and existing browsers

browser	web search	voice controlled navigation			
		main menu	visible sub-menu	hidden sub-menu	image menu
Siri[8]	O	X	X	X	X
Safari[9]	X	X	X	X	X
Opera[10]	△	X	X	X	X
IE&WSR [3][11]	△	△	△	△	△
Proposed browser[12]	△	O	O	O	O

6. CONCLUSION

Hierarchical menus are the most common means of navigation within a website, and should therefore be supported by voice navigation. In this paper, we proposed a new kind of web browsing system in which webpages are first analyzed using the VIPS algorithm, and then used as a basis for detecting changes to submenus and other dynamic content, as a result of user input. This browsing system offers the following advantages:

1. The ability to control the hierarchical menu structures of websites with relatively simple voice input.

2. Access and navigation for the sub-menu generated dynamically depending on the user's input are given.

3. Even if a menu object is rendered using images, it can be recognized through OCR and made available as keywords.

Note that the system requires a DOM structure for parsing, and thus will not function with compiled Flash or Silverlight based content. In the future, we would like to enhance the system to function with these alternate file types. We would also like to conduct further quantitative evaluation of voice-based browsing.

7. ACKNOWLEDGMENTS

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea and LG Electronics., under IT/SW Creative research program supervised by the NIPA (National IT Industry Promotion Agency). (NIPA-2013-(H0504-13-1003))

8. REFERENCES

[1] Sireesha K., Supriya, A., Haritha, D., Swetha, K. S., and Sastry, J. 2011. Voice Recognition browser for reduced vision and vision loss Learners. *International Journal of Scientific & Engineering Research.* 2, 12 (Dec. 2011).

[2] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma. VIPS: a vision-based page segmentation algorithm. *Microsoft Technical Report*, MSR-TR-2003-79, 2003.

[3] R. Brown. Exploring new speech recognition and synthesis APIs in Windows Vista. *Talking Windows, MSDN Magazine.* http://msdn.microsoft.com/hi-in/magazine/cc163663.aspx.

[4] GoogleTV Development. https://developers.google.com/tv.

[5] Google Chrome Extensions – Content Scripts. http://developer.chrome.com/extensions/content_scripts.html

[6] Google Voice Search. http://www.google.com/mobile/voice-search.

[7] Google Chrome Extensions – SpeechInput. https://developer.chrome.com/extensions/experimental.speechInput.html.

[8] Siri. http://www.apple.com/ios/siri.

[9] Apple Inc., Safari 6. http://www.apple.com/safari/.

[10] Opera Software ASA, Opera 11. http://www.opera.com/.

[11] Microsoft Corporation, Internet Explorer 10. http://windows.microsoft.com/en-us/windows-8/get-started-ie-10#1TC=t1.

[12] The demo screencast. http://db.hanyang.ac.kr/voiceweb.