Speed-Dial: A Surrogate Mouse for Non-Visual Web Browsing

Syed Masum Billah[1] Vikas Ashok[1] Donald E. Porter[2] IV Ramakrishnan[1]
[1]Department of Computer Science, Stony Brook University, NY, USA
{sbillah, vganjiguntea, ram}@cs.stonybrook.edu
[2]Department of Computer Science, University of North Carolina at Chapel Hill, NC, USA
porter@cs.unc.edu

ABSTRACT
Sighted people can browse the Web almost exclusively using a mouse. This is because web browsing mostly entails pointing and clicking on some element in the web page, and these two operations can be done almost instantaneously with a computer mouse. Unfortunately, people with vision impairments cannot use a mouse as it only provides visual feedback through a cursor. Instead, they are forced to go through a slow and tedious process of building a mental map of the web page, relying primarily on a screen reader's keyboard shortcuts and its serial audio readout of the textual content of the page, including metadata. This can often cause content and cognitive overload.

This paper describes our Speed-Dial system which uses an off-the-shelf physical Dial as a surrogate for the mouse for non-visual web browsing. Speed-Dial interfaces the physical Dial with the semantic model of a web page, and provides an intuitive and rapid access to the entities and their content in the model, thereby bringing blind people's browsing experience closer to how sighted people perceive and interact with the Web. A user study with blind participants suggests that with Speed-Dial they can quickly move around the web page to select content of interest, akin to pointing and clicking with a mouse.

CCS Concepts
• **Human-centered computing ~ Haptic devices**
• **Human-centered computing ~ Accessibility technologies.**

Keywords
Semantic web browsing; tactile interaction; Microsoft Surface Dial; visual impairment; screen reader; tactile exploration.

1. INTRODUCTION
For web browsing, people with vision impairments employ screen readers (e.g., JAWS, VoiceOver, NVDA, and many more listed in [2]), which reads aloud the textual content of the Web in serial order, generally ignoring layout and graphics.

Blind users predominantly rely on keyboard shortcuts to accelerate content navigation. Without shortcuts, blind users cannot skip past or skim content; skimming is essential functionality, as websites often have complex visual layouts, inline advertising, and aggregate content from multiple sources. Forcing a listener to hear all content would significantly and disproportionately harm users with vision impairments. Thus, modern screen readers feature numerous shortcuts to accelerate web navigation.

Even with these shortcuts, however, screen readers have significant usability problems for web browsing [11, 27]. First, blind users have to train on these shortcuts and learn a number of browsing strategies before they can efficiently browse the Web [11]. Second, because one cannot judge the importance of content before listening to it, blind users must listen to reams of irrelevant content before they find what they need. Third, screen readers alone are of little help in understanding the semantics of a page's contents, as screen readers are not aware of the fact that web content is organized into several logical blocks or semantic entities. For example, consider a list of products with their prices, features, and short descriptions in a tabulated HTML list. Most screen readers do not capture the relationships between these elements, and therefore treat these list elements as separate items. Consequently, the user cannot easily know that elements are related to each other, or when she has navigated past the end of the list. In short, there is no overview, and the user must assemble the semantic structure in her mind as the content is being read. The end result of this is that the current state of web browsing places significant and needless cognitive load on people with vision impairments.

These problems are particularly acute for common tasks in content-rich web sites with several semantic entities. For example, sighted users can purchase something online or make a reservation in just a few minutes with a few mouse clicks, whereas screen-reader users often need 10 minutes or more to complete the same tasks [34].

The reason for this difference is that sighted users can perceive the semantic structure of a page at a glance, via visual cues such as white space, fonts, and colors. For example, a sighted user can easily differentiate search results from advertisements on a side panel. Further, with the mouse, the user can quickly place the cursor on any element of interest in the page, select it, and operate on it. In contrast, blind users must explore the screen by brute force, read aloud by a screen reader, in order to construct their own semantic model of the page. The visual feedback of a mouse is not useful for blind users.

To address the aforementioned problems in non-visual web browsing, speech and haptic interfaces have been explored as alternatives to keyboards (e.g. [7, 39]). Although these interfaces can alleviate the extensive use of keyboard shortcuts to a certain extent, they still fall short of adequately addressing the cognitive and content overload problem in non-visual web browsing (details in sections 2 and 5).

This paper describes our research contribution towards reducing the gap in the web-browsing experience between sighted people and those with vision impairments. Specifically, this paper presents the Speed-Dial Non-Visual Browsing System for desktop platforms—a novel interaction paradigm that provides an intuitive and quick

ASSETS '17, October 29-November 1, 2017, Baltimore, MD, USA © 2017 Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-4926-0/17/10…$15.00
https://doi.org/10.1145/3132525.3132531

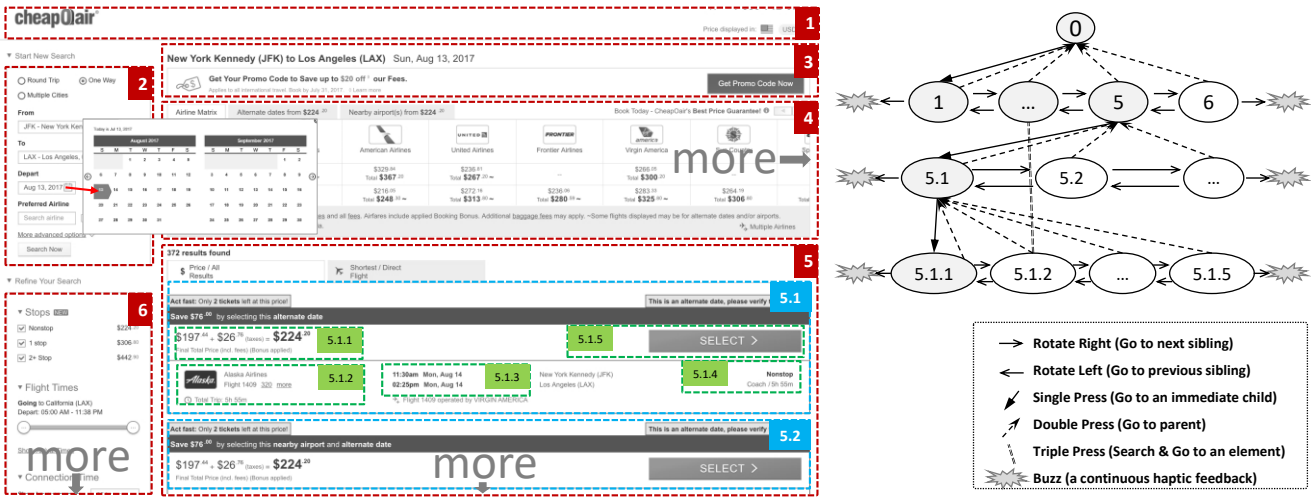

Figure 1: An illustration of Speed-Dial. On the left is a flight-reservation web site, which is divided into six logical blocks or semantic entities. Entity 5 is further divided into (sub)-entities. On the right is the corresponding semantic tree, and a list of valid gestures offered by the Dial to navigate on the tree.

access to the various entities and their content on the web page. Speed-Dial automatically generates a model, based on [7], explicitly capturing the semantics of the page that sighted users implicitly construct visually. This semantic model is interfaced with the interaction mechanics of an off-the-shelf physical Dial for web navigation.

The Speed-Dial prototype uses the recently announced Microsoft Surface Dial[1]; in principle, Speed-Dial should work with other Dial devices as well. The Surface Dial is a small, cylindrical puck, 1 ½" tall and 2 ½" in diameter. The Dial accepts three very simple gestures as input—press it like a button and rotate it left and right. The Dial can provide haptic feedback through vibration as output. Pressing and holding the Dial can activate a radial menu, which the user can navigate by rotating the Dial left or right. The Dial's behavior can be customized per-application.

The semantic model on which Speed-Dial operates is tree-structured, to provide blind users with an understanding of the hierarchical relationship of the elements in the page. Individual elements, such as buttons or text, are represented as nodes, whereas aggregate objects with clear visual boundaries, such as frames or lists, are represented as sub-trees. The semantic model used in Speed-Dial is created using techniques published in the extensive research literature on understanding the semantics of web pages (detailed in Section 2). The challenge is presenting the semantic model in a format amenable to quick access by a user.

Figure 1 illustrates how Speed-Dial can be used to navigate the search results page corresponding to a flight search, and the corresponding semantic tree. The page has six semantic entities, numbered 1 through 6, reflected by the six nodes at the 1st level in the tree. Each node at this level can be expanded further to reveal its (sub)-entities. The tree is traversed breadth-first. The press gesture at a node navigates down the subtree rooted at that node. The rotate right/left gestures traverse the nodes in a level bi-directionally. A special haptic feedback, denoted by the "buzz" indicates no more nodes remain to be traversed. When a node is visited, the screen reader reads out all the information associated with the entity, such as the price in node 5.1.1.

This illustration shows the distinctive features of Speed-Dial. Unlike shortcuts, there are a few, simple gestures. Gestural motions such as rotations and presses are almost instantaneous. The top-level semantic entities give a quick, high-level overview of the page, helping the user find an entity of interest quickly (i.e., with few gestures). The haptic feedback helps orient to the user within the page, such as indicating they have reached the end of a list. Speed-Dial can work with any screen reader, and provides uniform experience across different screen readers—an attribute that is highly desired by blind screen reader users [10]. Lastly, Speed-Dial exemplifies the well-known "information visualization mantra", namely, *overview, zoom, details on demand* [38] but in the context of non-visual web browsing.

A user study with 12 blind participants indicates that Speed-Dial is a promising approach to solve the content and cognitive overload problem with non-visual web browsing—the participants took 77% less time with Speed-Dial compared to that with a screen reader, and 49% less time compared to that with a screen reader augmented with speech interface to complete their tasks. In summary, Speed-Dial enables users to rapidly access any web content, thereby serving as a surrogate for the mouse.

2. RELATED WORK

Augmenting screen readers with different kinds of interaction modalities for web browsing continues to attract serious research attention, with audio-haptic and speech being the most popular.

Audio-haptic Interfaces for Web Browsing

In addition to the standard Braille displays [1], several research works [23, 24, 31, 39, 42, 43] have demonstrated the usability and applicability of audio-haptic interfaces for web browsing. These interfaces not only provide vibrations and audio cues, but also communicate the presence of HTML elements and various shapes on a page by using force-feedback [24, 28, 43], and tactile pin representations [33, 36, 39]. Prior work proposed guidelines for

[1] https://www.microsoft.com/en-us/surface/accessories/surface-dial

designing appropriate haptic sensations (e.g. tactons) for people with vision impairments [23].

Although audio-haptic devices are very helpful in obtaining an overview of a webpage, the unrestrained freedom of movement within the page is a hindrance for precise navigation and information finding; users may navigate past visual delimiters, such as white space. The user may have a sense of direction from the overview, but may still spend a lot of time searching for specific content of interest, such as the fare of a flight.

Extant audio-tactile interfaces incorporate a very limited notion of content semantics; in particular, they only represent the boundaries of various web entities on the page. The Dial-based system described in this paper utilizes the semantic meaning of the entities to organize navigation.

Closely related to haptic interfaces are tactons that provide vibrotactile feedback through the cutaneous senses to convey non-visual information [9, 13, 26, 32, 37, 40]. Sensory pulses of different durations that encode different kinds of taps such as single, double, etc. and amplitude of sensory stimulation that encode different values are some examples of tactons [12]. Speed-Dial system incorporates the idea of tactons for navigating the semantic model.

Speech Interfaces for Web Browsing

Natural-language interfaces for web browsing have been explored since inception of the Web, and have the potential to alleviate the extensive use of keyboard shortcuts by using spoken commands. House et al. [20] proposed a modified version of the NCSA Mosaic system [5] that was capable of translating user utterances to simple browsing actions. However, their system supported only a few basic actions on browser controls such as opening URLs or new windows. Dragon NaturallySpeaking [30] supports a slightly larger set of user commands to control websites by speech—besides browser controls, it has commands to operate on a few syntactic elements like links, buttons, forms, and form fields—all of which are readily identifiable by their HTML tags [21]. However, Dragon [30] relies on visual cues and a graphical user interface, thereby reducing accessibility. JSay [19] improves the accessibility of Dragon by interoperating with the popular JAWS screen reader. But JSay is also limited to a handful of basic functions, such as accessing application controls, scrolling, and refresh. A far richer set of more flexible commands is supported by Capti-Speak [6] that, like JSay, integrates a speech-interface with a screen-reader. All of the previously-mentioned, speech-driven interfaces, including Capti-Speak, operate only at the syntactical level, and do not incorporate the semantics of page content.

We also mention that generic voice-based Assistants such as Siri, Cortana, and Echo are being used to manage one's daily activities, but their focus is not web browsing.

Perhaps the closest related research to Speed-Dial is the spoken-dialog Assistant system proposed by Ashok et al. [7]. With their Assistant, users can not only navigate the page using spoken dialog, but also query the page content for certain information, such as the price of a product or duration of a flight. Further, the Assistant supports interaction at a semantic level, i.e., the users can communicate directly with the web entities present in the page without having to understand HTML markup. However, the Assistant, like all other speech interfaces, is heavily reliant on the availability and accuracy of automatic speech recognition (ASR).

Speech interfaces have several open challenges to widespread use for non-visual browsing. ASRs require high-fidelity microphones to accurately capture speech and eliminate spurious background noise; the built-in microphones in most desktops or laptops often corrupt the speech and cause substantial recognition errors. Moreover, the latency of ASR-based browsers is significant. Speech cannot be used in all scenarios, such as in public places where privacy and security may be compromised [3]. Speech alone is also unsuitable and tiring for ad-hoc navigation. For instance, skimming through a list of several products on a shopping website requires speaking "go to next product" repeatedly. We believe that rotating a dial is faster and less cumbersome for users. Under the best of circumstances, speech only addresses the element selection problem, not the content navigation problem.

Web Page Semantics

A principal component driving the Dial interface is the semantic model or semantic representation of the web page content. Semantic understanding of web pages is a classic research topic dating back to the inception of the Web (e.g., [16-18]). The use of web page semantics in accessibility is a long-standing research topic (e.g., [22, 35]). The starting point for semantic understanding of web pages is a web ontology. Broadly speaking, an ontology explicitly encodes a shared and common understanding of a particular domain, in this case the Web. The encoding captures commonly-used concepts in the domain, namely descriptions of web entities in the Web domain such as forms, text-boxes, menus, buttons, and widgets; attributes associate these concepts and relationships among them. With a web ontology in hand, one can create a semantic model of a web page. Web entity data extraction for constructing this semantic model is based on well-established web segmentation techniques [4, 8, 15, 44, 45]. Broadly speaking, these techniques partition a web page into semantically-related units, based on the observation that related items in a Web page often exhibit consistency in presentation style and spatial locality. We note that Apple's VoiceOver screen-reader also employs a basic notion of semantics using Web spots—landmarks that separate different segments on the webpage, some of which may be semantic entities. However, the caveat is that the user must manually associate segments with semantic entities.

A semantic model, in essence, is a tree that is similar in structure to the HTML DOM of the web page. Conceptually, the semantic layer is an abstract layer over the DOM tree. The semantic model in this paper is similar to the one developed by Ashok et al. [7], in that a collection of hand-coded web entity descriptions in the web ontology provides the necessary blueprints for identifying and representing various web entities, their characteristics, and their relationships as a hierarchy in the semantic model.

3. SPEED-DIAL INTERFACE DESIGN

We now explain the various components of Speed-Dial in detail.

Figure 2 depicts the workflow and the architectural schematic of the Speed-Dial system. A user can interact with a webpage using either the standard screen reader shortcuts or the Dial gestures. A special gesture (press & hold) brings up a radial menu (Figure 3) listing out several choices, such as navigate by heading or web-entity overview; each option determines the meaning of the other dial gestures, such as *rotating* or *tapping*.

We note that the radial-menu choices, as well as the interpretation of the gestures, also depend on the semantics of the page content. For example, if the screen-reader focus is presently on a text-box, the Dial radial menu contains characters (Figure 4); if the focus is on a date form field, the radial menu contains options to change the day, month, and year (Figure 7); if the focus is on a flight-result item (Figure 8), a single tap with subsequent *rotate* gestures will

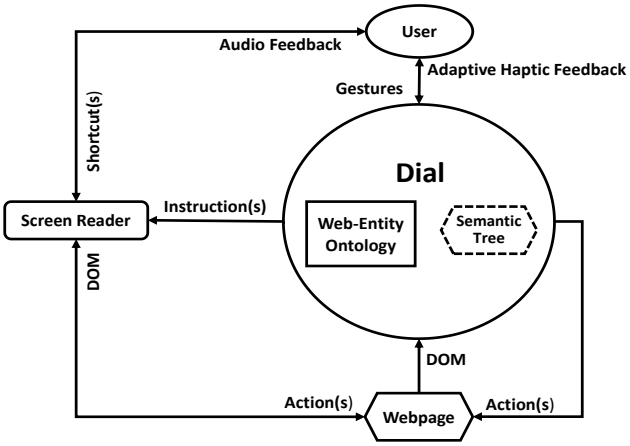

Figure 2. Architectural schematic and workflow of Speed-Dial system.

enable the user to quickly navigate through the important details, such as price or duration.

To obtain the semantic structure and composition of the webpage from the DOM, the Dial leverages a custom-defined Web-Entity Ontology [7] that provides the necessary blueprints or "class definitions" to identify and represent various web entities, such as forms, filters, menu, search-results, and calendars, as well as their characteristics and relationships. The Dial models this abstracted information as a Semantic Tree – a "cleaned-up" semantics-based non-visual presentation of the HTML DOM. The Semantic Tree then forms the basis for gestural navigation as illustrated in Figure 1. Additionally, a blueprint in the Web-Entity Ontology may also specify custom radial menu choices and gesture interpretations, which will be applied to the Dial interface whenever the corresponding entity is in focus.

Note that it is also possible to navigate the raw HTML DOM with Dial. But this reduces to navigating the HTML DOM element-by-element at the syntactic level, and hence inherits all the "content and cognitive" overload problems associated with navigating using keyboard shortcuts.

Finally, depending on the radial-menu option selected and gesture performed by the user, the Dial either performs the intended action (e.g., fill a form text field) directly on the webpage content, or instructs the screen reader to perform certain actions (e.g., navigate to the next heading). For users' convenience, the Dial also provides audio-haptic feedback, such as a tick with a small vibration when navigating to the next element.

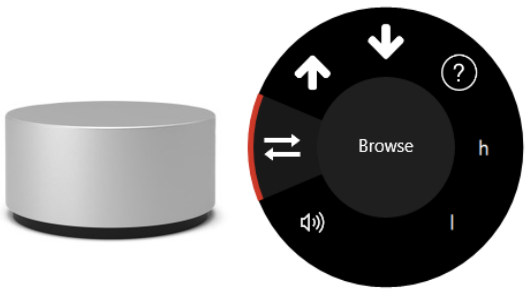

Figure 3: Surface Dial (left) and custom Dial menu (right)

It is instructive to mention that it is certainly possible to navigate the semantic model with keyboard alone without using the Dial device, although this will require either new shortcuts that do not overlap with standard shortcuts or overriding existing shortcuts. Consequently, the new shortcuts will likely be non-uniform across screen-readers. In contrast, Speed-Dial interface is uniform across all screen-readers, and requires users to remember less to efficiently navigate webpages.

3.1 The Speed-Dial

This subsection describes the important components of the Speed-Dial interface.

3.1.1 Hardware and Gestures

We did not build our own hardware; instead we used an *off-the-shelf* input device—the Surface Dial (see Figure 3) from Microsoft. It is a wireless rotational peripheral device (dimensions: 59 mm x 30 mm, weight: 145 g) with adaptive haptic feedback and 3,600 points of precision. The Surface Dial is designed to be placed on the screen of the Microsoft Surface tablet, but the Surface Dial also works off-screen with any PC, laptop or tablet running the Windows 10 Anniversary Update platform version.

Gestures. By default, the Surface Dial supports four basic types of gestures; a user can (i) rotate left, (ii) rotate right, (iii) quick press or just *press*, and (iv) press and hold. The press and hold or *long press* gesture brings up a radial Dial menu (Figure 3). Additionally, adaptive haptic feedback lets the user feel every action. Using the Surface Dial SDK, we customized the interface and the controls to suit our specific needs, including the addition of two custom gestures: double presses and triple presses (Figure 1, bottom-right).

Haptic Feedback. Speed-Dial also uses two tactons (haptic feedback) [32] to inform users about either the movements of the virtual navigational cursor, or the boundaries of the entities on the page. For navigational movements, we used a short, tick-like feedback on every rotation; and for entity-boundary notifications, we used a continuous buzz-like vibratory feedback (Figure 1).

3.1.2 Web-Entity Ontology

Web ontology has been extensively studied in the context of information extraction from web-pages. It has also been applied to improve accessibility [25]. The Web-Entity Ontology is based on our prior work on Web-Entity-Description Library described in [7], which followed well-studied methodologies and algorithms [4, 7, 29, 44, 45]. The Ontology contains hand-crafted descriptions or blueprints of various entities, such as forms, tables, filters, menu, search results, and calendars. The descriptions of web entities can be thought of as Java classes (from a programmer's perspective), whereas the entities in the Semantic Tree generated from the webpage can be thought of as corresponding instances or objects. Each description contains the following information:

(i) The characteristics and properties (i.e., data and methods) of the corresponding entity type. For example, a search-results entity contains fields to store the list of result items, a pointer to the root node (in the DOM), and a list of strings containing keywords such as "flight" or "tickets;
(ii) Algorithms to identify or extract the entities belonging to that entity type on the page; and
(iii) Custom Dial-interface specifications (more details in the next subsection).

To create the Web-Entity Ontology, we built and analyzed a training dataset of 200 popular websites from various domains including shopping, social networking, music, sports, banking,

flight/hotel reservations, government, video-streaming, and news. We identified the common aspects of the entities belonging to the same type but on different websites (e.g., list of products on different shopping websites), and designed generic descriptions for these types. Overall, our Web Entity Ontology consists of over 100 descriptions. Note that new descriptions for new web-entity types can be easily added to the Ontology manually without modifying the other components of Speed-Dial, thereby making it scalable across multiple websites.

Accuracy. Depending on the type of semantic entity, our extraction accuracy varied from 75% to over 90%. For example, the accuracy was over 90% for identifying calendars, menus, sort-options, and filter-options; over 85% for forms and search-results; and 75% for individual news-articles. Almost all errors were false negatives (i.e., not able to identify the entity on the page) as opposed to false positives (i.e., identifying one entity incorrectly as some other entity). In case of false negatives, the users can still find the corresponding entities using keyboard shortcuts.

3.1.3 Custom Interface Specifications

In this subsection, we describe how we encode custom navigational behavior for different semantic entities onto the Dial interface. Because the Dial gestures, such as rotation, are strictly one dimensional, we convert any two-dimensional (2-D) information on the web page into multiple one-dimensional (1-D) lists.

Not all web entities require a custom Dial interface. For example, the Dial can directly operate (press) on buttons, radio-buttons, and check-boxes. However, some semantic elements require special adaptation, including the autocomplete text-box; date-picker or calendar; drop-down or combo-box; and generic search-results list. As shown in prior studies [29], screen-reader users have a hard time interacting with these complex UI entities due to their inconsistent behavior, appearance, and the inaccessibility of underlying JavaScript libraries. Therefore, whenever such elements are in focus, the Dial offers customized interfaces to improve their usability. Next, we present the details of custom interfaces for a few key entity types.

Basic/Leaf. We consider an entity that does not have any children or sub-entities to be a leaf entity. Examples of this type include buttons, radio-buttons, check-boxes, and links. For these entities, the Dial does not offer any custom interface; it simply brings focus to the HTML element underlying the entity. User action (or the press operation) is directly performed on the HTML element by calling the HTML element's corresponding in-built function.

Text-Box. In text-box entity, user can enter text using either the default keyboard, or an onscreen radial keyboard provided by the Dial. To bring up the radial keyboard (Figure 4), user needs to perform the press & hold gesture (*long press*). Once the radial keyboard is on, the user rotates the Dial left/right to select a character and then *press* once to add the character to the text-box. Although this radial keyboard is not suitable for lengthy textual input, it is handy for entering a few characters.

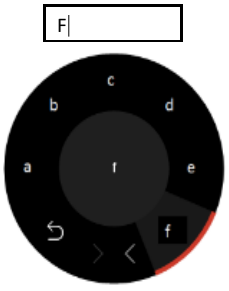

Figure 4: On-screen radial keyboard in Dial for entering text

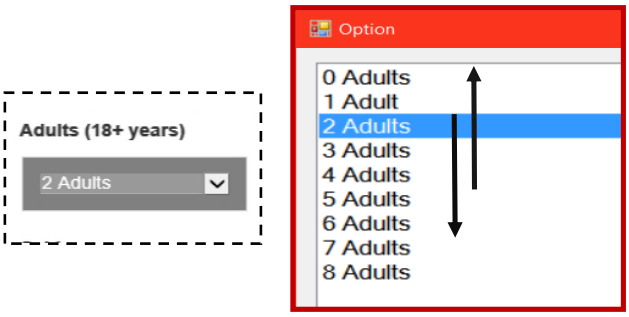

Figure 5: A sample HTML drop-down (left), and its equivalent custom drop-down widget for Dial (right).

Drop-Down/Combo-Box. The custom interface for a drop-down is a popup containing a list-box that populates data dynamically from the available options in the underlying HTML element (i.e. `<li>` elements under a `<ul>` HTML markup tag). The user rotates the Dial left/right to move down/up the list. A *press* gesture finalizes the selection and dismisses the custom-interface popup (Figure 5).

Autocomplete. Similarly, the custom interface for the autocomplete text-boxes includes a popup that contains a text-box for input, and a list-box for storing dynamically-generated auto-complete suggestions (Figure 6). The user can type the first few characters using either the keyboard, or the radial keyboard (Figure 4). Then the user rotates the Dial left/right to go through the suggested list items, and performs *press* to select a list item. The selected value is entered into the HTML autocomplete text-box and the custom-interface popup is dismissed automatically.

Calendar. Calendar is one of the most difficult widgets to interact with a screen reader [29]. Different websites may have different presentations of the calendar widget (e.g., showing only days of a month, or days of two months; or having small arrows and dropdowns to change the month and year). A few websites do not even permit users to manually enter the date, thereby forcing them to use a cumbersome and often inaccessible calendar widget. Even if the date-field is editable, sometimes there is no mention of the expected date format (e.g., 01/05/2017, or 01-05-2017, or 01/05/17, or May 1), and thus the users are unable to proceed due to form-validation errors, which screen readers often times fail to report.

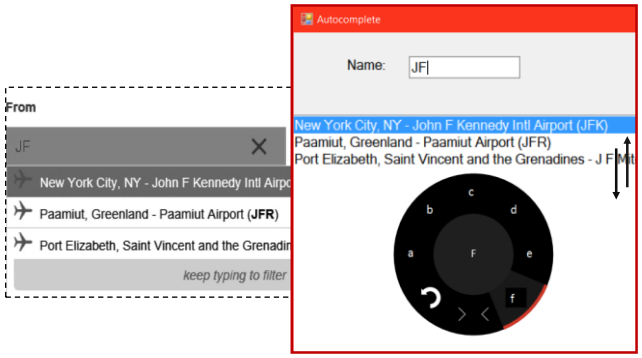

Figure 6: A sample HTML autocomplete box (left), and its equivalent custom autocomplete widget for Dial (right).

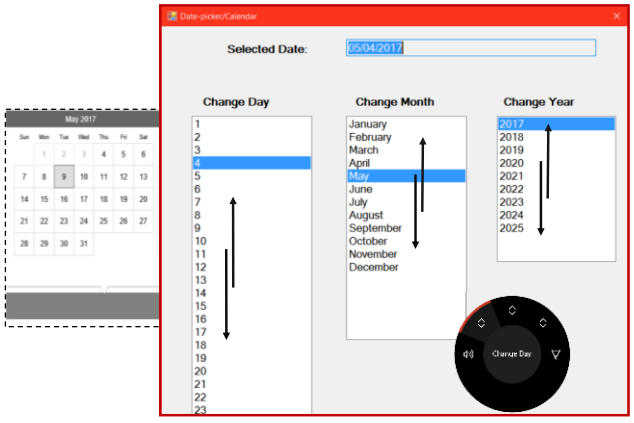

Figure 7: A sample HTML calendar (left), and the custom (generic) calendar widget for Dial (right)

To address these usability concerns, our custom calendar interface (popup) has three separate 1-D lists containing (i) days of a month, (ii) months of a year, and (iii) recent years (Figure 7). The access to these 3 lists is provided via 3 separate options on the radial menu, and the contents of each list can be easily accessed by simply rotating the dial after selecting that list with a press gesture. Once the user has settled on a date, a double press gesture populates the underlying HTML date-field with the chosen date while dismissing the custom popup.

Search Result/Data-grid. A lot of content on the web is tabular in nature. However, different websites may format tabular content differently, e.g., as lists, tables, or nested `div` elements. To represent tabular content, we use a custom data-grid view (Figure 8, in the foreground), where each column contains semantically-identical elements. For example, consider the search results in Figure 8 (in the background), and its equivalent custom interface (in the foreground). Each search-result item is represented as a row, and columns represent different properties of an item such as fare, duration, itinerary, and airline. The user can navigate either along the direction of a row or along a column using rotational gestures, with the press gesture for toggling between these two directions. Thus, the users not only can get a quick overview of the important details of any single flight, but also rapidly navigate over the properties (e.g. prices) of all the flights in the results list.

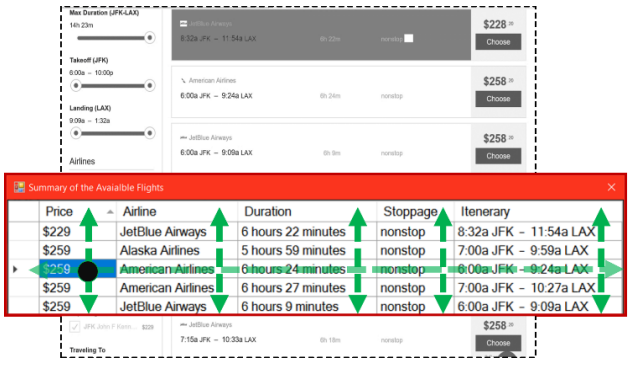

Figure 8: An HTML data-grid view (background), and its equivalent custom data-grid view for Dial (foreground). The green lines show the direction of Dial cursor.

3.2 Screen Reader Integration

We encoded a few essential screen-reader shortcuts (e.g., `h`: go to next heading, `l`: go to next link) in the default Dial radial menu (see Figure 3). Selecting such a menu option (e.g., `h`) enables users to navigate by headings using rotational gestures. This is particularly helpful for less-experienced screen-reader users with limited shortcut vocabulary. Additionally, it reduces cognitive overload by promoting uniformity, since different screen readers may have different key-bindings for the same function (e.g., `R` in JAWS vs. `D` in NVDA to go to the next landmark). In other words, given an action and a screen reader, the onus of determining the right shortcut for this purpose is shifted from the user to the Dial; the Dial examines what screen reader the user is currently using, and then simulates the appropriate, low-level keystroke. Thus, by playing the role of an intermediary between the user and a screen reader, the Dial ensures uniformity of user experience, and minimizes the disruption when a user switches from one screen reader to another—a feature highly desired by blind users [10].

3.3 System Implementation Notes

Speed-Dial was developed with the C# visual studio .NET framework version 4.6.2. It was a Windows Form application with an embedded browser (*WebBrowser* form control) object. Internally, this embedded browser shares the same rendering engine with Microsoft's Internet Explorer (IE) web browser. Therefore, from the perspective of screen readers, our application was equivalent to IE, and therefore the same screen-reader shortcuts can be used to browse the web.

In future work, we can integrate the Speed-Dial with any other browser through a custom plug-in or browser-extension. Our initial study selected IE because studies have shown that IE still remains the most popular browser among blind users [41].

For generating adaptive haptic feedback, we manipulated the value of the *RotationResolutionInDegrees* property exposed by the Dial API. For instance, setting this property value to 1.0 generates a continuous buzz-like feedback, whereas values greater than 10.0 generate a simple tick.

4. USER STUDY

To assess the effectiveness and usability of Speed-Dial, we conducted an IRB-approved user study aimed at validating the following primary and secondary hypotheses:

H: Speed-Dial significantly improves web usability compared to: (i) popular screen readers and (ii) screen readers augmented with speech interfaces. We test the following secondary hypotheses:

- **H1**: Higher efficiency in web tasks involving continuous navigation within the page.
- **H2**: Lesser effort to locate, understand and interact with desired web element.
- **H3**: Higher usability rating.
- **H4**: More intuitive interaction with webpage contents.

4.1 Participants

We conducted a user study with 12 visually-impaired participants who had no hearing or speech impairments. All participants were familiar with the IE web browser and the JAWS screen reader. They varied in age from 25 to 65 (mean=40, median=36, SD=12.5), and gender (7 males, 5 females). Regarding screen-reader expertise, 5 participants considered themselves to be "experts", whereas the remaining 7 participants were "beginners". On average, the participants browse internet for 1.5 hours per day.

4.2 Study Setup

We assigned tasks to the participants that were transactional in nature, requiring a sequence of steps spanning multiple web pages. Users with vision impairments typically find these tasks challenging. We assigned the following 3 types of tasks:

- **T1**: On a travel booking site, find the form for reserving a flight, booking a hotel, renting a car, or purchasing a vacation package. Fill-out the form with experimenter-provided data (such as *From city*, *To city*, *Departure date*, *Number of travelers*), and hit the *search* button.
- **T2:** On the search-results page of the same travel booking site, find the search-results list, and identify the top 3 search results satisfying predefined criteria (i.e., flight-related criteria were cheapest airfare, shortest travel time, or number of layovers).
- **T3:** On the search-result page, use different filters to narrow down the search-results list, and consider any special deals. This task was purely exploratory, and, as such, we did not record any objective measurements.

We choose the following 3 travel booking websites: priceline.com, expedia.com, and cheapoair.com, based on the observation that these websites differ considerably in appearance, accessibility support, richness of contents, and content placement. Additionally, all the participants indicated that they were not that familiar with these websites. For each website, a participant was supposed to perform the 3 aforementioned tasks under each of the following 3 conditions, thereby totaling 9 tasks per participant:

- **Screen-Reader Only (SR).** Participants could only use the standard keyboard shortcuts supported by JAWS. This was also considered as the baseline condition.
- **Screen-Reader with Speech Command (SR+Speech).** Participants could use the standard keyboard shortcuts of JAWS, and high-level speech commands described in [7], e.g., "select this item", "pick the next item", "go to search-result", or "choose 5th for departure date".
- **Screen-Reader with Speed-Dial (SR+Dial).** Participants could use the standard shortcuts of JAWS, as well as the gestures offered by the Speed-Dial interface.

To avoid confounds, we used the same underlying semantic model in both SR+Speech and SR+Dial conditions. To minimize the learning effect, we counterbalanced the ordering of the websites, the conditions, and the tasks T1 & T2.

Prior to the study, participants were given sufficient instructions and time (~30 minutes) to familiarize themselves with all three conditions. We chose the following 3 websites for practice runs: Craigslist (classifieds), Twitter (social), and CNN (news).

Each participant was allotted 30 minutes to complete tasks T1 & T2. If a participant completed both tasks within the allotted time window, only then he or she was requested to perform T3 for the remaining time. All conversations during the study were in English.

4.3 Data Collection and Analysis

We used the publicly-available software *globalmousekeyhook*[2] to record participants' keystrokes and time spent on each page. We also audio-recorded all sessions and transcribed them later for post-analysis. For tasks T1 and T2, we recorded the following measures: (i) completion time; (ii) number of shortcuts used under each condition; (iii) number of speech commands used; (iv) number of times each Dial gesture was used; (v) number of times a participant navigated back and forth between the boundaries of an entity; (vi) number of times the participant sought moderator's help; and (vii) reasons for failure, such as automatic speech recognition (ASR) errors, content inaccessibility, semantic error, or Speed-Dial error.

We also recorded any unusual browsing behavior that delayed the completion of assigned tasks, such as repeatedly navigating over the same content, confusion due to form-validation errors, or failure to recognize the entities.

At the end of the experiment, the participants were asked to complete the standard System Usability Scale (SUS) questionnaire [14], and a set of open-ended questions eliciting comments and suggestions for all 3 study conditions.

5. RESULTS

In this section, we analyze the collected measurements and subjective feedback, as well as compare the participants' performance and web browsing experiences under the 3 different study conditions (i.e., SR, SR+Speech, and SR+Dial). We conducted one-way ANOVA test with Tukey's honestly significant difference (HSD) post hoc test to determine if any differences in measures between these 3 conditions were statistically significant.

Completion time. We found significant effect of study conditions on completion time for both tasks T1 ($F_{2,33} = 47.6$, $p < .0001$) and T2 ($F_{2,33} = 18.4$, $p < .0001$). The mean completion times for tasks T1 and T2 under each condition are shown in Figure 9. For task T1, when using SR+Dial, the mean completion time (60.4s) was reduced by 73.0% compared to that of the baseline SR (224.2s), and 5.7% compared to the SR+Speech condition (66.2s). For task T2, these reductions were 80.2% and 62.8% respectively (SR+Dial: *mean*=76.9s, SR: *mean*=389.6s, SR+Speech: *mean*=207.0s).

Number of shortcuts. We also found a significant effect of study conditions on number of shortcuts issued in tasks T1 ($F_{2,33} = 32.5$, $p < .0001$) and T2 ($F_{2,33} = 28.4$, $p < .0001$). Figure 10 shows the mean number of shortcuts. When using SR+Dial (*mean*=2.6), participants typed 97.1% and 90.6% fewer shortcuts than when using SR (*mean*=92.4) and SR+Speech (*mean*=27.91) in task T1. In task T2, these reductions were 95.5% and 89.5% respectively (SR+Dial: *mean*=8.1, SR: *mean*=181.1, SR+Speech: *mean*=77.6).

The post-hoc pairwise comparison with Tukey HSD test indicated that the differences in means in completion time and shortcut usage for tasks T1 and T2 were statistically significant ($p < .01$) between the pairs of SR & SR+Speech and SR & SR+Dial. However, between SR+Speech & SR+Dial the differences were not significant for task T1, but significant ($p < .05$) for task T2.

A closer inspection of the collected data reveals the reason behind this significant difference for task T2. For each spoken utterance, the Speech interface in SR+Speech condition incurred an (unavoidable) overhead of about 1.0 to 1.5 seconds for converting speech to text, interpreting this text, and translating the interpretation to actions. Additionally, the Speech interface occasionally encountered speech-recognition failures, errors, and natural-language ambiguities, such as vague commands matching two or more elements with identical names and tags. The situation was further exacerbated on the search-results page for task T2 that contained many semantic entities and related content. For navigation, participants had to speak the same commands (e.g., "go

[2] https://github.com/gmamaladze/globalmousekeyhook

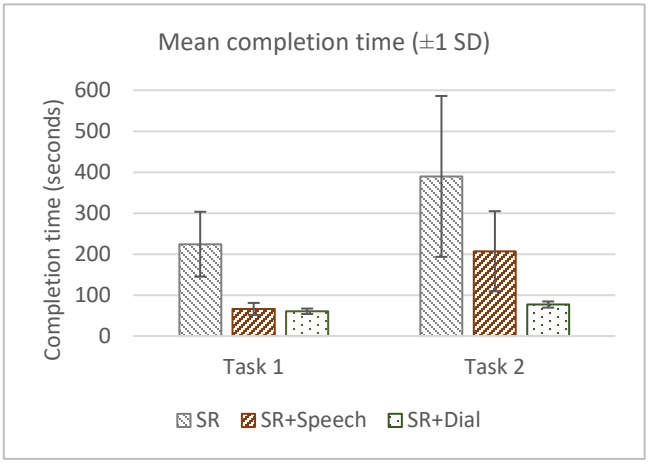

Figure 9: The mean time to complete task T1 (left) and task T2 (right) in 3 conditions. Error bars show ±1 SD.

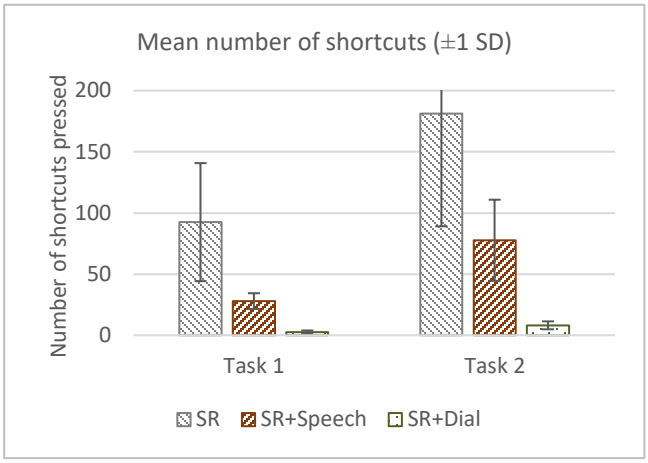

Figure 10: The mean number of shortcuts issued in task T1 (left) and task T2 in 3 conditions. Error bars show ±1 SD.

next", "how much is the price?", "what is the duration?") over-and-over again for each search-result item. The overhead accumulated to the point that participants reduced the usage of speech commands and reverted to using keyboard shortcuts.

In contrast, in SR+Dial condition, the participants did not encounter these issues because: (1) the Dial interface gave them feedback almost immediately—within 0.3 seconds on every rotation, while also providing haptic feedback whenever the focus reached the boundaries of entities. All participants liked this feature, and stated that it gave them confidence and a better understanding of their orientation in a webpage; and (2) the semantics-driven content-presentation style of Speed-Dial was straightforward to navigate, requiring only a few simple gestures. Therefore, even though the search-results webpage contained many entities, almost all participants systematically and *semantically* explored the page with Dial gestures rather than falling back to keyboard shortcuts that operated on the *syntactic* HTML content. The participants also explicitly stated that they pressed a few keyboard shortcuts in SR+Dial condition due to their habitual reflex or by accident, and not because they needed those shortcuts.

Note however that, for the form-filling task T1, there were no statistically significant differences in average measure values between SR+Speech vs SR+Dial. Because in SR+Speech condition, participants simply gave speech commands to choose the dates, thereby avoiding the painful screen-reader interaction with the calendar widget. Regardless, the above results and observations validate our hypotheses H1, H2, and H4.

5.1 Interaction Behavior and Subjective Feedback for Speed-Dial

We observed an interesting behavioral pattern with Speed-Dial—the participants kept rotating the Dial even after finding their content of interest until they reached the boundary. Then, they rotated the Dial to navigate back to the desired content, and perform the desired action. When asked, the participants explained that they were always afraid of missing important information as well as losing focus and orientation—something that they claimed as a frequent occurrence while browsing web with screen readers. For example, when they press a 'TAB' shortcut, the screen readers might sometimes inexplicably take them to a different, undesired section of the webpage. As a result, they subconsciously became cautious and less exploratory while navigating a webpage (with screen reader). But with Dial, the participants were more confident exploring the content given that they would never unknowingly navigate past the boundaries of entities—Speed-Dial always confines the navigation within these boundaries unless the user explicitly instructs it to move to other entities.

Regarding the usage of custom gestures, we noticed that the participants frequently used the double press gesture to move "one level" up in the Semantic Tree so that they could reorient themselves. 9 out of 12 participants stated that traversing up in the Semantic Tree until reaching the top level with Dial was much better than pressing 'ESC' multiple times with JAWS to get to the beginning of a page. If they wanted to return to their previously-focused location, with JAWS, they had to use multiple shortcuts once again; whereas with the Dial, they could find the previous location with fewer steps (at most, the height of the Semantic Tree). Surprisingly, they did not use the triple press gesture often; only 2 out of 12 used that gesture, and said that it slowed them down.

While participants were using only the screen reader, we observed that the expert users spent quite a bit of time guessing the features supported by the entities. For example, they checked if a text-box, had the "autocomplete" feature by entering a few characters, and pressing the down-arrow key. With Speed-Dial, they said that they did not have to guess anymore due to the custom-Dial interface that is consistent across all websites.

5.1.1 Usability Rating

At the end of each session, each participant was asked the standard System Usability Scale (SUS) questionnaire where participants rated positive and negative statements about each study condition on a Likert scale from 1-strongly disagree to 5-strongly agree, and 3-being neutral. We found significant effect of study conditions on SUS score ($F_{2,33}$ = 15.5, p < .0001). The mean SUS score of SR+Dial (85.0) was 18.3% higher than that of SR (69.3) and 4.6% higher than that of SR+Speech (81.04). Post hoc comparisons using the Tukey HSD test indicated that the pairwise differences in means in SR vs. SR+Speech and in SR vs. SR+Dial were statistically significant (p < .01). However, it was not significant in case of SR+Speech vs. SR+Dial (p = .36). Still, participants stated that they preferred Speed-Dial to others, as SR was the most cumbersome to use, and the Speech interface was becoming cumbersome to use for task involving continuous navigation. Furthermore, with the Speech interface, speech commands often interfered with listening to the page content—a problem avoided by using Speed Dial.

5.2 Post-Evaluation

Finally, we administered a brief, Likert-type questionnaire (1-strongly unfavorable, 5-strongly favorable response) to solicit participants' opinions and suggestions regarding Speed-Dial. As shown in Table 1, all participants liked the "buzz" feedback indicating entity boundaries (Question 3). They also liked the uniformity of interaction experience (Question 10), and the ease of learning (Question 2). The participants were undecided about the potential of using Dial for text entry (Question 9) as well as the triple press gesture for the "search & go" feature. Finally, there was a strong consensus that Speed-Dial can indeed serve as a mouse surrogate (Question 6).

5.3 Feature Requests

The participants expressed a desire for more custom gestures, such as *press & turn* to immediately select a menu option. They also wanted a relatively stronger haptic feedback, and incorporation of rotational momentum to skip multiple entities in one go. Integration of Speed-Dial with smartphones was also highly coveted.

Table 1. Post-experiment questionnaire and responses.

Question: On a scale of 1 to 5	Avg	SD
1. How easy is it to rotate the Dial compared to pressing keyboard shortcuts?	4.50	0.65
2. How easy is it to learn Speed-Dial?	4.75	0.43
3. How important is it to know the entity boundaries?	5.00	0.00
4. How useful is vibration feedback?	4.17	0.90
5. How useful is the combination of audio and vibratory feedback?	3.17	1.40
6. Can Speed-Dial serve as a mouse?	4.50	0.65
7. Is "search & go" feature useful?	3.83	0.99
8. How easy is it to fill forms with Speed-Dial?	4.42	0.76
9. How useful is the radial keyboard?	3.33	1.11
10. Does Speed-Dial provide uniform interaction experience?	4.75	0.43

6. CONCLUSION

This paper explores the applicability of an off-the-shelf, physical Dial as a surrogate for a computer mouse in the context of non-visual web browsing. Our experimental results demonstrate that the combination of Dial & semantic model has a bestter constructive synergy over extant non-visual web browsing systems that combine speech interface and semantic model. A unique aspect of Speed-Dial is that it provides uniformity of interaction experience regardless of the underlying screen reader. Uniformity can eliminate the need to learn new screen readers, thereby reducing training effort and associated cost. In fact, this was affirmed by a participant who said: "If I have this, I shouldn't spend my time in computer class". Speed-Dial enables blind users to quickly move around the web page to select content of interest, akin to pointing and clicking with a mouse, i.e., it serves as a surrogate for the computer mouse.

7. ACKNOWLEDGMENTS

We thank Glenn Dausch, Yevgen Borodin, and the anonymous reviewers for their insightful feedback that helped shape this paper. Asmita Negi, Leela Bharath Kumar, and Rakesh Agarwal contributed to the implementation of Speed-Dial. This research was supported in part by NSF: IIS-1447549, CNS-1405641; National Eye Institute of NIH: R01EY026621; and NIDILRR: 90IF0117-01-00. The content is solely the responsibility of the authors and does not necessarily represent the official views of NIH nor represent the policy of NIDILRR.

8. REFERENCES

[1] AFB, Refreshable Braille Displays. Retrieved from http://www.afb.org/ProdBrowseCatResults.aspx?CatID=43.

[2] AFB, Screen Readers. Retrieved from http://www.afb.org/ProdBrowseCatResults.aspx?CatID=49.

[3] Ahmed, T., Hoyle, R., Connelly, K., Crandall, D., and Kapadia, A., 2015. Privacy Concerns and Behaviors of People with Visual Impairments. In *Proceedings of the Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 3523-3532.

[4] Álvarez, M., Pan, A., Raposo, J., Bellas, F., and Cacheda, F., 2010. Finding and extracting data records from web pages. *Journal of Signal Processing Systems 59*, 1, 123-137.

[5] Andreessen, M., 1993. NCSA Mosaic technical summary. *National Center for Supercomputing Applications 605*.

[6] Ashok, V., Borodin, Y., Puzis, Y., and Ramakrishnan, I. V., 2015. Capti-Speak: A Speech-Enabled Web Screen Reader. In *Proceedings of the Proceedings of the 12th Web for All Conference*. ACM, 327-328.

[7] Ashok, V., Puzis, Y., Borodin, Y., and Ramakrishnan, I., 2017. Web Screen Reading Automation Assistance Using Semantic Abstraction. In *Proceedings of the Proceedings of the 22nd International Conference on Intelligent User Interfaces*. ACM, 407-418.

[8] Baluja, S., 2006. Browsing on small screens: recasting web-page segmentation into an efficient machine learning framework. In *Proceedings of the Proceedings of the 15th international conference on World Wide Web*. ACM, 33-42.

[9] Bardot, S., Brock, A., Serrano, M., and Jouffrais, C., 2014. Quick-glance and in-depth exploration of a tabletop map for visually impaired people. In *Proceedings of the Proceedings of the 26th Conference on l'Interaction Homme-Machine*. ACM, 165-170.

[10] Billah, S. M., Ashok, V., Porter, D. E., and Ramakrishnan, I. V., 2017. Ubiquitous Accessibility for People with Visual Impairments: Are We There Yet? In *Proceedings of the Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 5862-5868.

[11] Borodin, Y., Bigham, J. P., Dausch, G., and Ramakrishnan, I. V., 2010. More than meets the eye: a survey of screen-reader browsing strategies. In *Proceedings of the Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (W4A)*. ACM, 1-10.

[12] Brewster, S. and Brown, L. M., 2004. Tactons: Structured Tactile Messages for Non-Visual Information Display. In *Proceedings of the Fifth Australasian User Interface Conference (AUIC2004)*. ACS, 15-23.

[13] Brock, A., Kammoun, S., Macé, M., and Jouffrais, C., 2014. Using wrist vibrations to guide hand movement and whole body navigation. *i-com 13*, 3 (2014-12-10), pp. 19-28.

[14] Brooke, J., 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry 189*, 194.

[15] Cai, D., Yu, S., Wen, J.-R., and Ma, W.-Y., 2004. *VIPS: A vision based page segmentation algorithm.* Microsoft technical report.

[16] Chung, C. Y., Gertz, M., and Sundaresan, N., 2002. Reverse engineering for web data: From visual to semantic structures. In *Proceedings of the Proceedings 18th International Conference on Data Engineering*. IEEE, 53-63.

[17] Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R., Jhingran, A., Kanungo, T., Rajagopalan, S., Tomkins, A., and Tomlin, J. A., 2003. SemTag and Seeker: Bootstrapping the semantic web via automated semantic annotation. In *Proceedings of the Proceedings of the 12th international conference on World Wide Web*. ACM, 178-186.

[18] Fensel, D., Decker, S., Erdmann, M., and Studer, R., 1998. Ontobroker: Or how to enable intelligent access to the WWW. In *Proceedings of the Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*. CiteSeer.

[19] Hartgen Consultancy, 2017. J-Say. Talk to Your Computer, It Talks Back! Retrieved from http://www.ngtvoice.com/products/software/astec/j-say/.

[20] House, D., Novick, D., Fanty, M., and Walpole, J., 1995. Spoken-Language Access to Multimedia (SLAM).

[21] HTML5.2, 2017. Hyper-Text Markup Language v.5.2. Retrieved from http://w3c.github.io/html/.

[22] Huang, A. W. and Sundaresan, N., 2000. A semantic transcoding system to adapt Web services for users with disabilities. In *Proceedings of the Proceedings of the 4th International ACM Conference on Assistive Technologies*. ACM.

[23] Kuber, R., Yu, W., and McAllister, G., 2007. Towards Developing Assistive Haptic Feedback for Visually Impaired Internet Users. In *Proceedings of the Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, 1525-1534.

[24] Kuber, R., Zhu, S., Arber, Y., Norman, K., and Magnusson, C., 2014. Augmenting the non-visual web browsing process using the geomagic touch haptic device. *SIGACCESS Access. Comput.*, 109, 4-10.

[25] Labský, M., Svátek, V., and Nekvasil, M., 2008. Information Extraction Based on Extraction Ontologies: Design, Deployment and Evaluation. *Ontology-based Information Extraction Systems (OBIES)*, 9.

[26] Lawrence, M. M., Martinelli, N., and Nehmer, R., 2009. A Haptic Soundscape Map of the University of Oregon. *Journal of Maps 5*, 1, 19-29.

[27] Lazar, J., Allen, A., Kleinman, J., and Malarkey, C., 2007. What Frustrates Screen Reader Users on the Web: A Study of 100 Blind Users. *International Journal of human-computer interaction 22*, 3, 247-269.

[28] Magnusson, C., Tan, C. C. S., and Yu, W., 2006. Haptic access to 3D objects on the web. In *Proceedings of the Eurohaptics*.

[29] Melnyk, V., Ashok, V., Puzis, Y., Soviak, A., and Borodin, Y., 2014. Widget Classification with Applications to Web Accessibility. In *Proceedings of the International Conference on Web Engineering (ICWE)*.

[30] Nuance, 2017. Dragon Naturally Speaking Rich Internet Application. Retrieved from http://www.nuancesoftwarestore.com/dragon-naturallyspeaking-premium/.

[31] O'Modhrain, M. S. and Gillespie, B., 1997. The moose: A haptic user interface for blind persons. In *Proceedings of the Proc. Third WWW6 Conference*.

[32] Pielot, M., Poppinga, B., Heuten, W., and Boll, S., 2011. A tactile compass for eyes-free pedestrian navigation. In *Proceedings of the Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part II*. Springer-Verlag, 640-656.

[33] Pietrzak, T., Crossan, A., Brewster, S. A., Martin, B., and Pecci, I., 2009. Creating Usable Pin Array Tactons for Nonvisual Information. *IEEE Transactions on Haptics 2*, 2, 61-72.

[34] Puzis, Y., Borodin, Y., Puzis, R., and Ramakrishnan, I. V., 2013. Predictive Web Automation Assistant for People with Vision Impairments. In *Proceedings of the proceedings of the 22th international conference on world wide web*. ACM, 1031-1040.

[35] Ramakrishnan, I. V., Stent, A., and Yang, G. L., 2004. HearSay: enabling audio browsing on hypertext content. In *Proceedings of the International World Wide Web Conference (WWW)*.

[36] Rotard, M., Knödler, S., and Ertl, T., 2005. A tactile web browser for the visually disabled. In *Proceedings of the Proceedings of the sixteenth ACM conference on Hypertext and hypermedia*. ACM, 15-22.

[37] Rümelin, S., Rukzio, E., and Hardy, R., 2011. NaviRadar: a novel tactile information display for pedestrian navigation. In *Proceedings of the Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 293-302.

[38] Shneiderman, B., 1996. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the Visual Languages, 1996. Proceedings., IEEE Symposium on*. IEEE, 336-343.

[39] Soviak, A., Borodin, A., Ashok, V., Borodin, Y., Puzis, Y., and Ramakrishnan, I., 2016. Tactile Accessibility: Does Anyone Need a Haptic Glove? In *Proceedings of the Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility*. ACM, 101-109.

[40] Tsukada, K. and Yasumura, M., 2004. ActiveBelt: Belt-Type Wearable Tactile Display for Directional Navigation. In *UbiComp 2004: Ubiquitous Computing: 6th International Conference, Nottingham, UK, September 7-10, 2004. Proceedings*, N. DAVIES, E.D. MYNATT and I. SIIO Eds. Springer Berlin Heidelberg, Berlin, Heidelberg, 384-399.

[41] WebAIM, 2015. Survey of Preferences of Screen Readers Users. Retrieved from http://webaim.org/projects/screenreadersurvey6/.

[42] Wong, E. J., Yap, K. M., Alexander, J., and Karnik, A., 2015. HABOS: Towards a platform of haptic-audio based online shopping for the visually impaired. In *Proceedings of the Open Systems (ICOS), 2015 IEEE Confernece on*. IEEE, 62-67.

[43] Yu, W., Kuber, R., Murphy, E., Strain, P., and McAllister, G., 2006. A novel multimodal interface for improving visually impaired people's web accessibility. *Virtual Reality 9*, 2-3, 133-148.

[44] Zhai, Y. and Liu, B., 2005. Web data extraction based on partial tree alignment. In *Proceedings of the Proceedings of the 14th international conference on World Wide Web*. ACM, 76-85.

[45] Zhu, J., Nie, Z., Wen, J.-R., Zhang, B., and Ma, W.-Y., 2006. Simultaneous record detection and attribute labeling in web data extraction. In *Proceedings of the Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 494-503.