Enabling Efficient Web Data-Record Interaction for People with Visual Impairments via Proxy Interfaces

JAVEDUL FERDOUS, Old Dominion University, USA
HAE-NA LEE, Stony Brook University, USA
SAMPATH JAYARATHNA, Old Dominion University, USA
VIKAS ASHOK, Old Dominion University, USA

Web data records are usually accompanied by auxiliary webpage segments, such as filters, sort options, search form, and multi-page links, to enhance interaction efficiency and convenience for end users. However, blind and visually impaired (BVI) persons are presently unable to fully exploit the auxiliary segments like their sighted peers, since these segments are scattered all across the screen, and as such assistive technologies used by BVI users, i.e., screen reader and screen magnifier, are not geared for efficient interaction with such scattered content. Specifically, for blind screen reader users, content navigation is predominantly one-dimensional despite the support for skipping content, and therefore navigating to-and-fro between different parts of the webpage is tedious and frustrating. Similarly, low vision screen magnifier users have to continuously pan back-and-forth between different portions of a webpage, given that only a portion of the screen is viewable at any instant due to content enlargement. The extant techniques to overcome inefficient web interaction for BVI users have mostly focused on general web-browsing activities, and as such they provide little to no support for data record-specific interaction activities such as filtering and sorting – activities that are equally important for facilitating quick and easy access to *desired* data records. To fill this void, we present InSupport, a browser extension that: (i) employs custom machine learning-based algorithms to automatically extract auxiliary segments on any webpage containing data records; and (ii) provides an instantly accessible proxy *one-stop* interface for easily navigating the extracted auxiliary segments using either basic keyboard shortcuts or mouse actions. Evaluation studies with 14 blind participants and 16 low vision participants showed significant improvement in web usability with InSupport, driven by increased reduction in interaction time and user effort, compared to the state-of-the-art solutions.

CCS Concepts: • **Human-centered computing → Accessibility technologies**; *Empirical studies in accessibility*.

Additional Key Words and Phrases: Web accessibility, Blind, Low Vision, Visual impairment, Screen reader, Screen Magnifier, Data records

1 INTRODUCTION

Interaction with web data records (e.g., products, flights, posts, job listings, emails) is an integral part of everyday web activities such as shopping, seeking information, communication, and social networking. To facilitate convenient and efficient interaction with web data records, modern web applications provide an assortment of *auxiliary segments* such as filters, sort options, search form, and multi-page links as shown in Figure 1. While sighted users can easily and almost instantly access these auxiliary segments using a pointing device such as a mouse, blind and visually impaired (BVI) users on the other hand struggle to do the same using their go-to assistive technologies such as a screen reader (e.g., JAWS, NVDA, VoiceOver) or a screen magnifier (e.g., ZoomText,

Authors' addresses: Javedul Ferdous, Old Dominion University, Norfolk, VA, 23529, USA, mferd002@odu.edu; Hae-Na Lee, Stony Brook University, Stony Brook, NY, 11794, USA, haenalee@cs.stonybrook.edu; Sampath Jayarathna, Old Dominion University, Norfolk, VA, 23529, USA, sampath@cs.odu.edu; Vikas Ashok, Old Dominion University, Norfolk, VA, 23529, USA, vganjigu@odu.edu.

2160-6455/2023/1-ART
https://doi.org/10.1145/3579364

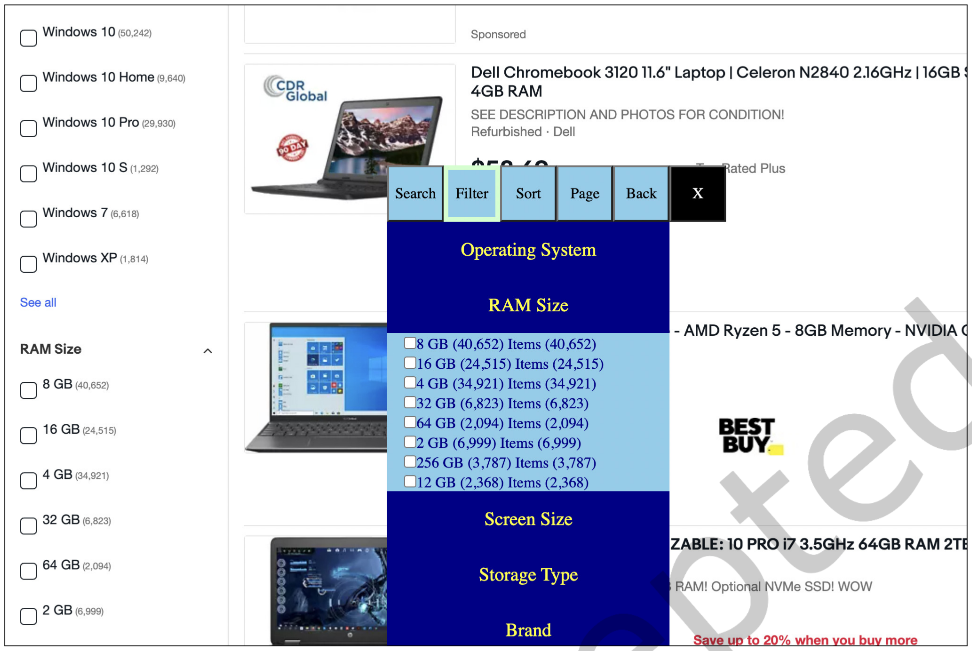

Windows 10 (50,242)
Windows 10 Home (9,640)
Windows 10 Pro (29,930)
Windows 10 S (1,292)
Windows 7 (6,618)
Windows XP (1,814)
See all
RAM Size
8 GB (40,652)
16 GB (24,515)
4 GB (34,921)
32 GB (6,823)
64 GB (2,094)
2 GB (6,999)
Sponsored
Dell Chromebook 3120 11.6" Laptop | Celeron N2840 2.16GHz | 16GB
4GB RAM
SEE DESCRIPTION AND PHOTOS FOR CONDITION!
Refurbished · Dell
Rated Plus
Search
Filter
Sort
Page
Back
X
Operating System
RAM Size
8 GB (40,652) Items (40,652)
16 GB (24,515) Items (24,515)
4 GB (34,921) Items (34,921)
32 GB (6,823) Items (6,823)
64 GB (2,094) Items (2,094)
2 GB (6,999) Items (6,999)
256 GB (3,787) Items (3,787)
12 GB (2,368) Items (2,368)
Screen Size
Storage Type
Brand
- AMD Ryzen 5 - 8GB Memory - NVIDIA
BEST BUY
ZABLE: 10 PRO i7 3.5GHz 64GB RAM 2TE
RAM! Optional NVMe SSD! WOW
Save up to 20% when you buy more

Fig. 1. InSupport for convenient BVI interaction with web data records. The auxiliary segments such as filters and sort options are automatically extracted and instantly 'delivered' to a BVI user via a proxy interface that is easily navigable using either basic shortcuts such as arrow and TAB keys or simple mouse actions. Without the proxy interface, the BVI user has to perform multiple key presses and/or panning-and-scrolling to manually navigate between data records and auxiliary segments – a process that includes searching and traversing realms of irrelevant content.

Windows Magnifier). This is because these assistive technologies are not tailored for interaction scenarios where content is distributed all over the screen as in case of webpages containing web data records and auxiliary segments (see Figure 1).

For instance, a screen reader primarily supports one-dimensional navigation of content (e.g., 'H' key for navigating to and hearing the contents of the next heading, 'A' for the next link), and therefore blind users have to endure an arduous and tedious process involving a multitude of key presses for navigating back-and-forth between data records and auxiliary segments (see Figure 1) [8, 19]. While this interaction burden is reduced to a certain extent in some screen readers which support alternative modalities (e.g., *rotor* in VoiceOver allows users to choose and selectively navigate certain types of HTML elements on the webpage), the onus of locating the auxiliary segments nonetheless still remains on the users. Moreover, to exploit this additional screen reader support, the users should have apriori knowledge of the HTML element types (heading, link, button, etc.) of auxiliary segments and their components; remembering such webpage-specific details can induce cognitive overhead for the blind users [16]. Similarly, low vision screen magnifier users too need to expend considerable 'panning' and scrolling effort for not only navigating to-and-fro between data records and auxiliary segments but also finding desired search filters, since only a small portion of the screen is visible to the users at any instant due to content enlargement.

Existing research solutions to address this problem are mostly based on speech interaction [8, 27], which are not only susceptible to noise but also require a tighter integration of a third party speech service with the user's screen reader, thereby limiting their practicality to a few open-source screen readers. The use of speech in public settings has also shown to cause privacy issues for BVI users [2, 63]. The few extant non-speech solutions on the other hand have all primarily focused on general webpage navigation [8, 15, 17]. Specifically, these existing methods strived to improve interaction efficiency by allowing BVI users to skip irrelevant content during navigation, i.e., skip directly to the beginning of the records from anywhere in the page [8], skip to the next/previous record irrespective of underlying HTML markup [9], navigate the webpage at a *semantic* level of logically meaningful segments [15, 17], etc. However, these solutions did little to enable convenient and efficient execution of data record specific activities such as filtering, sorting, and searching, which are also essential for quickly locating the desired records, especially when the number of data records is large and distributed across multiple pages.

To overcome these limitations of current approaches, in this paper we present InSupport, a browser extension that automatically extracts auxiliary segments from a webpage containing data records, and then *pushes* these segments to BVI users instantly on demand via a proxy interface (see Figure 1) that is easily navigable with either basic keyboard shortcuts or mouse actions. To extract the auxiliary segments in the webpage, InSupport uses custom devised machine learning-based detection algorithms that can identify subtrees in the DOM of the webpage that correspond to these segments. With this instant access support from InSupport, the BVI users will no longer have to spend significant time and effort manually exploring and navigating between different webpage segments while perusing web data records. This is especially beneficial for novice blind screen reader users who typically remember only a handful of basic keyboard shortcuts [7]; these users currently spend significantly more time than expert blind users in performing web tasks [7], as they are unaware of advanced shortcuts that enable faster navigation of web content.

A user study with 14 blind participants showed that with InSupport, the average time and number of input actions required by the participants for locating their desired data records were significantly lower than those with a state-of-the-art solution [9] as well as those with just their preferred screen readers. Similar observations were made in another user study with 16 low vision participants, where the participants' task completion times with InSupport were significantly and consistently lower than those with an extant solution [17] as well as their preferred screen magnifier. Note that this work is a direct extension of our prior work that was published in IUI 2022 [26]. In sum, our contribution in this manuscript (including those in our prior work) are:

- Machine learning-based algorithms to automatically identify auxiliary segments such as filters, sort options, search form, and multi-page links in webpages containing data records.
- A novel browser extension InSupport that enables BVI users to conveniently and instantly access auxiliary segments (e.g., filters, sort options) while interacting with web data records.
- Findings of two user studies with 14 blind and 16 low vision participants respectively where the participants evaluated InSupport against both state-of-the-art solutions and their status-quo assistive technologies.

2 RELATED WORK

Our work closely relates to existing research on the following topics: (i) web usability for blind screen reader users; and (ii) web usability for low vision screen magnifier users.

2.1 Web Usability for Blind Users

There exist plenty of prior research works that have investigated the interaction issues faced by blind users while browsing the web [8, 12, 19, 29, 39, 46, 50, 51]. However, bulk of the existing works have predominantly focused on the accessibility of web content for blind users [4, 5, 10, 13, 30–32, 45, 50, 53, 59, 62], whereas the usability of non-visual web interaction has received comparatively lesser attention from researchers [8, 9, 12, 19, 22, 39]. Among these existing works, some approaches have pursued the idea of automatically annotating webpages by injecting JavaScript into the DOM of webpages in order to improve their usability [9, 22]. For instance, Brown et al. [22] presented a JavaScript based method that captured and classified dynamic changes in webpages, and subsequently provided this information to screen reader users via an injected ARIA live region. Similarly, a fairly recent work [9] employed visual saliency-capturing deep neural networks to identify the important parts or 'hot-spots' of a webpage, and then automatically injected ARIA landmark roles into the corresponding DOM subtrees of the identified hot-spots, so as to enable a screen reader user to quickly navigate to these hot-spots with special screen reader shortcuts (e.g., 'R' in the JAWS screen reader). While some screen readers also provide similar functionalities (e.g., the rotor feature in VoiceOver) for skipping irrelevant content, navigation is nonetheless one-dimensional and dependent on the DOM layout of the webpage; semantics and relative importance of the individual auxiliary segments are not considered by either these existing annotation methods or in-built screen reader functionalities.

Apart from automatic annotations, researchers have also investigated other approaches such as web automation [14, 52], speech assistants [8, 27], and alternative input modalities [15]. Web automation techniques [14, 18, 43, 48] facilitate automatic execution of certain repetitive web tasks (e.g., ordering a preferred pizza), thereby significantly reducing the user's manual effort and time for doing these tasks. A common aspect of most automation techniques is the use of task scripts or macros that contain the sequence of actions to execute the corresponding tasks. These macros can either be created through handcrafting [18, 48], or via user demonstration [6, 14, 44, 52]. While these techniques indeed improve interaction experience for blind screen reader users, they are limited to a small set of repetitive tasks, and therefore they may not be able to handle web tasks such as searching for a desired web data record, which involve considerable ad-hoc web browsing. Furthermore, end users face an extra burden of not only creating a script for each web task, but also maintaining and updating the script over time to accommodate changes to either the webpage or their preferences.

Accessibility assistants enable blind users to use speech input to interact with webpage content [7, 8, 27]. For instance, Gadde et al. [27] proposed a simple speech-based interface that enabled blind users to use simple voice commands to aurally obtain a quick overview of any webpage and also directly navigate (i.e., shift screen reader focus) to a few key segments. Ashok et al. [8] supported a more elaborate set of speech commands, including those to navigate and query web data records. Although speech interfaces facilitate faster access to content, they have several limitations, notably speech recognition accuracy (in noisy environments) [1], blind users' social concerns (e.g., drawing undesired attention from others), and privacy [3]. Also, many of these assistants (e.g., [8, 27]) require tighter integration with third-party screen reader framework, so their scope is limited to open-source screen readers.

Prior works have also explored novel input modalities to facilitate convenient webpage navigation by overcoming some of the core limitations of the keyboard based screen reader interaction. For example, Billah et al. [15] proposed using an off-the-shelf Dial input device as a surrogate for mouse to hierarchically navigate the semantically-meaningful segments (e.g., menu, forms, data records) on the page using simple rotate and press gestures. Similarly, Soviak et al. [56] presented a new tactile input device that enabled blind users to 'feel' the layout of any webpage via tactile sensations provided at boundaries of the webpage segments. Blind users could also employ this tactile device to navigate webpage content in a 2D space and directly select one of the segments on the page, akin to touch exploration on mobile devices. While these interfaces are effective in improving non-visual interaction with web content, they are limited to general navigation of the webpage semantic structure, and as such do not directly assist in accomplishing specific data record-related activities such as filtering, sorting, and searching. Moreover, these approaches are less adoptable as they require additional hardware, which can be potentially expensive to many blind users.

2.2 Web Usability for Low Vision Users

Usability of web interaction for people with low vision remains an understudied research topic [35, 42, 49, 57]. An early work by Jacko et al. [35] focused specifically on low-vision users who had age-related macular degeneration, and observed the mouse behavior of the users to understand their interaction strategies. Their findings indicated that user performance was significantly dependent on the size of icons – larger the icon size, better the performance. Szpiro et al. [57] conducted a more elaborate investigation to understand the low vision user behavior, interaction strategies and challenges when interacting with different computing devices such as desktops, smartphones, and tablets. They noticed that the low vision users sometimes needed to simultaneously rely on more than one assistive technology to overcome interaction challenges associated with computing applications. They also noticed that the low vision users frequently needed to make several adjustments to comfortably view the application content. Both of these works strived to study the general characteristics and pain points of low vision interaction with computing applications; they did not expend much effort to address low vision usability in specific web scenarios such as interacting with web data records and auxiliary segments.

Few research works have focused on improving low vision usability of computing applications including web browsing [11, 17, 28, 38]. A seminal work by Kline et al. [38] proposed a collection of tools that enabled low vision users to selectively enlarge portions of the screen and additionally keep track of mouse cursor. The ideas underlying these tools have now been incorporated in most modern screen magnifiers. Gajos et al. [28] on the other hand suggested transforming the application GUI itself via their proposed interface-generation technique tailored exclusively for people with low vision and motor impairments. By providing a custom interface specification, their technique could automatically generate a personalized GUI catering to the individual needs of low vision users. Similarly, Bigham [11] proposed the idea of altering web application GUI to improve usability, but he focused on space compaction; his magnification system automatically determined the extent to which webpage content could be enlarged without causing adverse side effects such as increased horizontal scrolling. Billah et al. [17] also proposed a space compaction based context preserving screen magnification system that strived to reduce panning by placing related webpage elements close to each other within the magnified viewport. Their magnification system also enabled users to sequentially navigate the important segments in a webpage using an external Dial input device. While space compaction can improve usability by preserving local relationships between content elements, its effectiveness is limited in scenarios where related content elements are distributed all across the screen such as in case of web data records and auxiliary segments.

Prior works directly addressing the low vision usability of web data records [40, 41] have only focused on the content of the records themselves and as such they do not support efficient access to the auxiliary segments such as filters and sort options, which are also important to ensure convenient low vision interaction with data records. Specifically, these existing works explored the idea of automatically extracting information in data records and then presenting it as a compactly arranged table which was more amenable for screen magnifier interaction. In their evaluation studies, many low vision participants explicitly expressed that they needed an assistive mechanism that facilitated efficient access to auxiliary segments while interacting with data records.

In sum, while the aforementioned approaches were effective in improving general low vision usability of web interaction, their impact on data record-specific activities involving auxiliary segments was limited, since they were not tailored for convenient low-vision access of spatially distributed web content. Therefore, in this paper we present InSupport, a scalable approach to improve the usability of interaction with auxiliary segments associated with web data records for BVI users.

3 APPROACH

Figure 2 presents an architectural schematic illustrating the workflow of InSupport [26]. As shown in the figure, InSupport was implemented as a browser extension that has two core components: (i) Segment Extractor and (ii) Proxy Interface. The Segment Extractor analyzes the webpage DOM and extracts the auxiliary segments (i.e., filters, sort options, search form, and multi-page links) using custom machine learning-based identification algorithms. The content of identified auxiliary segments is then replicated and presented to a user via the

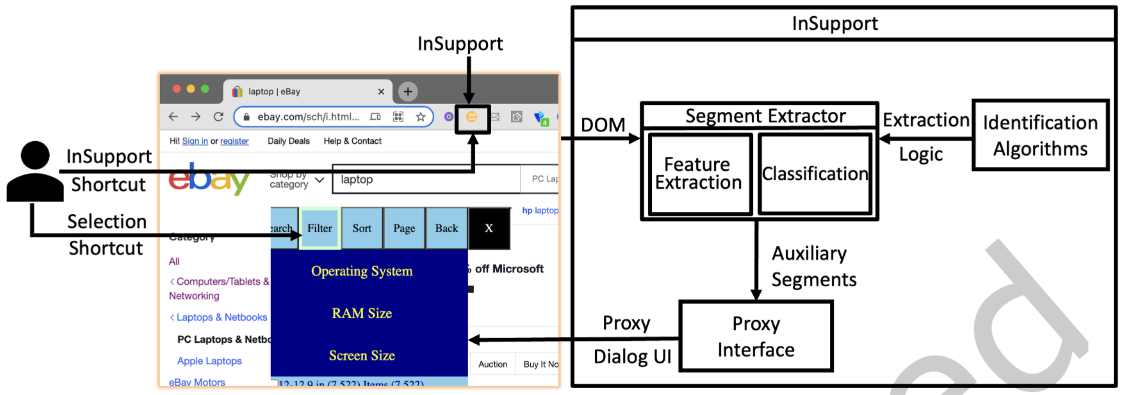


Fig. 2. InSupport architectural workflow.

Proxy Interface. The user can instantly access the InSupport Proxy Interface using a special 'CTRL+SHIFT+Z' shortcut, and then navigate the content using TAB and arrow keys, or alternatively using mouse clicks and scrolls. All user selections in the Proxy Interface (e.g., "filter by price", "sort by most recent", "next page") are automatically translated by InSupport into equivalent actions on the actual auxiliary segments on the webpage, thereby achieving the same intended effect.

3.1 Extracting Auxiliary Segments

The Segment Extractor leverages custom identification algorithms to extract the auxiliary segments from webpages containing data records. Specifically, InSupport extracts the following four types of auxiliary segments: (i) filter options; (ii) sort options; (iii) search form; and (iv) multi-page Links (see Table 1).

3.1.1 Identification Algorithms. The four algorithms for extracting the four types of auxiliary segments all have a similar workflow as shown in Algorithm 1. The algorithms start by first extracting all the candidate DOM nodes by referring to a predefined tag list as presented in Table 1. This predefined list was compiled after manual analysis of 100 webpages from top-visited websites belonging to different domains such as shopping, travel, job search, and classifieds. Then the algorithms extract a set of handcrafted features (see Table 5 in the Appendix) from the subtrees of all candidates. The extracted features representing each candidate are then fed to the custom trained machine learning classifiers (Table 1) that determine whether the candidate is the intended auxiliary segment.

Features. The detailed list of features along with their descriptions, for each auxiliary segment type is presented in Table 5 in the Appendix. We handcrafted a unique set of features for each segment type. As seen in Table 5, most features were binary (0 or 1) whereas the remaining features were numerical (integer). We designed several keyword based features (e.g., presence of *search* keyword) because we noticed in our earlier manual analysis that the HTML metadata typically contained several of such keywords within the attribute values of container nodes (e.g., div), even if these words were not shown on the webpage.

Algorithm 1: Auxiliary segment extraction algorithm

```
Input  : HTML DOM of the loaded web page, binary auxiliary segment classifier with classes 'AUX' and
         'NON-AUX'
Output : Auxiliary Segment or Null
$document \leftarrow$ Root node of DOM
// Get all the candidate nodes
$candidates \leftarrow getAllCandidates(document)$
foreach $candidate \in candidates$ do
    // Extract corresponding Table 5 features
    $features \leftarrow extractFeatures(candidate)$
    // Classify the candidate
    $class \leftarrow AUX\_classifier(features)$
    if $class ==$ 'AUX' then
        $result.append(candidate)$
    end
end
// Arbitration: Select the candidate with the highest probability score
$AuxSegment = argmaxScore(result)$
if $AuxSegment == Null$ then
    // No auxiliary segment detected
    return Null
end
return $AuxSegment$
```

Auxiliary Segment	Candidate Tags	# Features	Classification Model
Filter options	div, li, section, article, dt, desktop-facet, ul, form, fieldset, button, dl	5	MLP classifier
Sort options	div, ul, select	4	MLP classifier
Search form	form	5	Logistic regression
Multi-page links	div, nav, li, ul, span, section, button, tr, footer, aPage, pagination, bPage	6	Logistic regression

Table 1. Extraction algorithm details for the auxiliary segments.

Model evaluation. To evaluate the classification models, we constructed four separate datasets for each of the four types of auxiliary segments. The source for these datasets was a custom built collection of manually-annotated 209 webpages (webpage collection/datasets and details available on the GitHub repository[1]). These webpages were chosen from a diverse set of websites belonging to over 15 different domains including shopping, travel, finance, and sports (e.g., *Best Buy*, *Airbnb*, *Shutterstock*, *NBA*). Note that these datasets did not overlap with

[1]https://github.com/javedulferdous/InSupport

the earlier dataset of 100 websites that was used for manual analysis to determine candidate tags. We annotated each webpage in the collection by manually inserting custom data attributes, one for each type of auxiliary segment; these data attributes were then exploited while constructing the corresponding datasets. Therefore, each webpage in the collection produced 1 positive data point for each type of auxiliary segment, thereby totalling 209 positive data points per auxiliary segment type. As the number of negative data points from each webpage can be more than 1, we randomly picked one negative data point per auxiliary type from each webpage in order to have balanced datasets. In sum, for each type of auxiliary segment, we had a total of 418 data points in the corresponding dataset, with equal number of positive/negative points.

In each of these datasets, we randomly picked and set aside 320 data points (160 positive and 160 negative) for training and the remaining 98 for testing. As for the machine learning model, in this paper, we explored logistic regression and multi-layer perceptron classifiers, given their widespread popularity across a variety of domains for binary classification tasks [24, 36, 37, 58]. For each of these two algorithms, we performed a 5-fold cross validation on the training dataset for optimization. The best models (based on F-score) for each classifier were then evaluated on each corresponding test dataset. The performances of these classifiers for each auxiliary segment type are presented in Table 6 (see Appendix). As noticeable in Table 6, with the handcrafted features, both machine learning algorithms performed very well in discriminating between auxiliary segments and arbitrary webpage segments. The few erroneous classifications were mostly due to unconventional HTML realizations of auxiliary segments in some test webpages. For example, in one such instance, the filter options were implemented as a collection of drop down menus instead of the conventional group of checkboxes or links. Similarly, in another instance, the sort options were implemented as independent buttons instead of the traditional drop-down list. In such scenarios, which were pretty rare, the models were not able to correctly recognize the auxiliary segments, thereby causing a slight drop in the recall performance of these algorithms. The best performing model (based on F-score) for each segment type was then selected for the corresponding identification algorithm.

Algorithm evaluation. To evaluate the algorithms as a whole and *in the wild*, we built another test dataset comprising 100 manually-annotated webpages (webpage collection/datasets and details available on the same GitHub link as that for model datasets described earlier). These webpages were randomly sampled from a diverse set of websites belonging to over 5 different domains including shopping, lifestyle, finance, and sports (e.g., *American Eagle*, *Flipkart*, *99acres*, *Under Armour*). We ensured that this dataset did not overlap with the previous dataset of 209 webpages which was used for training and evaluating models. We annotated each of the 100 webpages in the dataset by manually inserting data attributes in their DOMs – one attribute for each type of auxiliary segment. Out of these 100 webpages, 26 did not have a search form and 44 did not have sort options. All webpages had filter options and multi-page links. The algorithms were then executed on the test dataset and their outputs were compared with the manual annotations to determine their performance.

Auxiliary Segment	**Precision**	**Recall**	**F-score**
Filter options	0.85	0.95	0.90
Sort options	0.86	0.98	0.92
Search form	0.98	0.83	0.90
Multi-page links	0.95	0.89	0.92

Table 2. Performance of extraction algorithms.

Table 2 shows the performance values for all the algorithms. Overall, all four algorithms showed high performance on the test dataset. The errors were mainly due to one of the following reasons: (i) the candidate

HTML tag list (Table 1) did not contain the HTML tag of an auxiliary segment, so it completely missed it during extraction (i.e., false negative); (ii) the machine learning model inaccurately assigned a higher probability to another candidate which was not the intended auxiliary segment (i.e., false positive); and (iii) the machine learning model did not correctly classify a candidate as an auxiliary segment (i.e., false negative). For example, the Aurate website[2] had an image button on top of the webpage to bring up the search overlay and then a <div> tag in the overlay was the root of the search form, therefore the algorithm missed this search form since the list of candidate tags (see Table 1) only included the <form> tag. Another example of false negative was in the ASOS website[3], where the Multi-page links auxiliary segment was presented as a single "LOAD MORE" link instead of the conventional list of numbered links (e.g., "1, 2, 3, Next"); therefore, our algorithm was unable to identify this segment. As for examples of false positives, in the American Eagle website[4], the Sort options segment was incorrectly identified as Filter options segment, presumably because the two segments were clubbed together in the webpage. Similarly, in the Points2 website[5], the implementation of filters resembled that of a search form, so the Filter options segment was incorrectly recognized as the Search form segment. All (100%) of the false negative errors due to incompleteness of the candidate tags list (Table 1) occurred in case of Search form segment. The false negatives for all other auxiliary segments were due to incorrect classification. As shown in Table 2, false negatives mostly occurred in case of the Search Form and the Multi-page links segments. All false positive errors were due to incorrect classifications by the underlying machine learning models. As observable in Table 2, these errors mostly affected the Filter Options and the Sort Options segments; the examples provided earlier illuminate the causes underlying the errors.

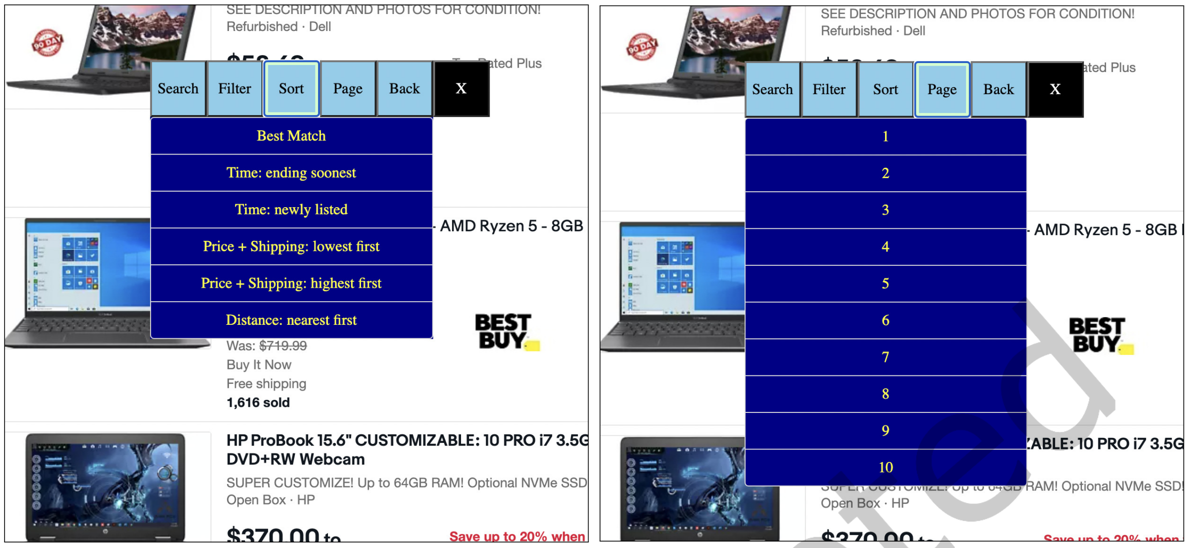
3.2 InSupport Proxy Interface

The proxy interface of InSupport was designed to be navigable with simple TAB/arrow keys or mouse click/scroll actions. As seen in Figure 3, it is a "one-stop" interface where a user can access all four auxiliary segments (if available). To access the interface, the user needs to press the 'CTRL+SHIFT+Z' shortcut. Initially the focus is set to the first auxiliary segment, namely the *Search* form if available, otherwise it is set to Filter Options. To navigate to other segments, the user can either use the mouse or press TAB/SHIFT+TAB or LEFT/RIGHT arrow keys. To select one of the segments, the user has to simply click on it or press the ENTER key. Within each segment, the user can navigate the various options using mouse scroll or TAB/SHIFT+TAB or UP/DOWN arrow keys. To select an option, the user can click on it or press ENTER key. For a screen reader user, to directly move focus back to the list of segments, the user can press the ESCAPE key.

When the user selects an option (e.g., "filter by price less than $50"), InSupport automatically translates this action into an equivalent action on the actual auxiliary segment on the webpage, thereby producing the same effect. Furthermore, InSupport refocuses the screen-reader cursor to the beginning of the data records each time they are updated or refreshed, thereby letting the blind user avoid the burden of pressing numerous screen-reader shortcuts to navigate back to the data records from the top of the webpage (by default, the screen reader starts narrating from the top of a page whenever the page is loaded or refreshed). To detect the beginning of the data records, we relied on the STEM algorithm [25] which is a robust and efficient state-of-the-art technique for identifying web data records. Also, the use of InSupport to access the auxiliary segments is purely optional; the user can always rely on the standard screen-reader shortcuts or mouse magnifier panning to interact directly with the webpage including auxiliary segments.

Note that InSupport does not show the tab option for a given auxiliary segment in its proxy interface if the corresponding extraction algorithm is unable to: (i) detect the segment in the webpage (i.e., false negative), and (ii)

[2]https://auratenewyork.com/collections/all
[3]https://www.asos.com/us/search/?q=shoes
[4]https://www.ae.com/us/en/c/men/shoes/cat4840024?pagetype=plp
[5]https://www.point2homes.com/US/Apartments-For-Rent/VA/Norfolk.html



(a) (b)

Fig. 3. InSupport proxy interface: (a) Sort options, (b) Multi-page links.

detect labels from the extracted segment possibly due to improper webpage design or due to inaccurate segment identification (i.e., false positive). In case InSupport is able to detect labels from a non-auxiliary segment that is a false positive, InSupport unfortunately does end up showing the extracted content in its interface, as it presently lacks the ability to handle such scenarios. However, we observed that such scenarios were a minority in our test dataset (described earlier), specifically, InSupport was able to detect labels in only 6.8% of false positives. Also, as users typically have certain apriori expectations regarding the content of an auxiliary segment based on their prior experiences on e-commerce websites such as shopping websites, they are likely to easily detect these false positives, and therefore avoid interacting with the corresponding non-auxiliary segments.

4 EVALUATION WITH BLIND SCREEN READER USERS

To assess the effectiveness of InSupport's proxy interface, we conducted an IRB approved user study with blind participants [26]. The details of the study and the findings are described next.

4.1 Participants

We recruited 14 blind participants via email lists and snowball sampling. Table 3 presents the participants' demographic details. The average age of the participants was 43.57 (Median = 44, Min = 26, Max = 64), and the gender representation was equal (7 female, 7 male). Our inclusion criteria required the participants to be proficient with JAWS screen reader and Chrome web browser. No participant had any motor impairments that affected their ability to do the study tasks. All participants stated that they regularly accessed a wide range of e-commerce websites for doing activities such as shopping, searching for jobs, and browsing classifieds.

4.2 Design

In the study, the participants were asked to perform the following two tasks related to typical data record interaction:

ID	Age	Gender	Age of Vision Loss	Preferred Screen Reader	Hours Per Day	Computer Type
P1	39	M	Since birth	JAWS	5-6	Laptop
P2	26	M	Age 3	JAWS	5-6	Laptop
P3	52	F	Age 5	JAWS	2-3	Laptop
P4	45	M	Age 6	JAWS	3-4	Desktop
P5	34	F	Age 11	JAWS	5-6	Laptop
P6	48	F	Cannot remember	JAWS	4-5	Laptop
P7	34	M	Cannot remember	JAWS	3-4	Desktop
P8	54	F	Cannot remember	JAWS	1-2	Laptop
P9	57	F	Since birth	JAWS	2-3	Desktop
P10	28	M	Age 2	JAWS	5-6	Desktop
P11	64	F	Since birth	JAWS	1-2	Desktop
P12	51	F	Since birth	JAWS	3-4	Laptop
P13	35	M	Since birth	JAWS	5-6	Desktop
P14	43	M	Since birth	JAWS	3-4	Laptop

Table 3. Participant demographics for the user study with blind participants. All information was self-reported by the participants. Hours per day indicates the average time a participant spent per day on web browsing.

- T1 – Locate a data record on a travel website that matches a predefined criteria (e.g., morning flight, Delta carrier, price less than $300).
- T2 – Locate a desired data record on a shopping website based on the participant's own personal preferences.

In a within-subjects experimental setup, the participants were asked to perform the above two tasks under three study conditions:

- Screen reader – The participants could rely only on their preferred screen readers to complete the tasks. This condition represented the status quo for all the participants.
- SaIL [9] – The participants could rely only on their screen readers in this condition too, except that the webpage was preprocessed by a state-of-the-art transcoding technique SaIL [9]. Specifically, SaIL uses visual saliency to detect important regions of a webpage, and then injects ARIA landmarks to the webpage DOM so that screen reader users can access the salient regions via special shortcut (e.g., 'R' in JAWS screen reader).
- InSupport – The participants could not only interact with webpages directly via screen reader shortcuts, but also instantly access the four auxiliary segments (i.e., filter options, sort options, search form, and multi-page links) via the InSupport interface.

To mitigate learning effect and avoid confounds, we used three different travel websites (Kayak, Travelocity, and Orbitz) for T1, and also three different shopping websites (Amazon, eBay, and Target) for T2. Furthermore, for T2, we also chose three different but similar query items ('laptop', 'desktop', and 'tablet'). For T1, the target data record was located in the second data records webpage (between fifth and eighth positions) on all three chosen websites. Also, in all the websites, the SaIL model [9] was able to identify all four auxiliary segments as being salient, and therefore these segments had corresponding aria landmarks injected into them for the SaIL study condition. We also ensured that InSupport too was able to accurately extract the auxiliary segments so as to prevent the confounding impact of extraction algorithm accuracy on InSupport user interface evaluation. The assignment of websites to conditions, items to websites, and the ordering of both tasks and conditions were counterbalanced using the Latin square method [20].

4.3 Apparatus

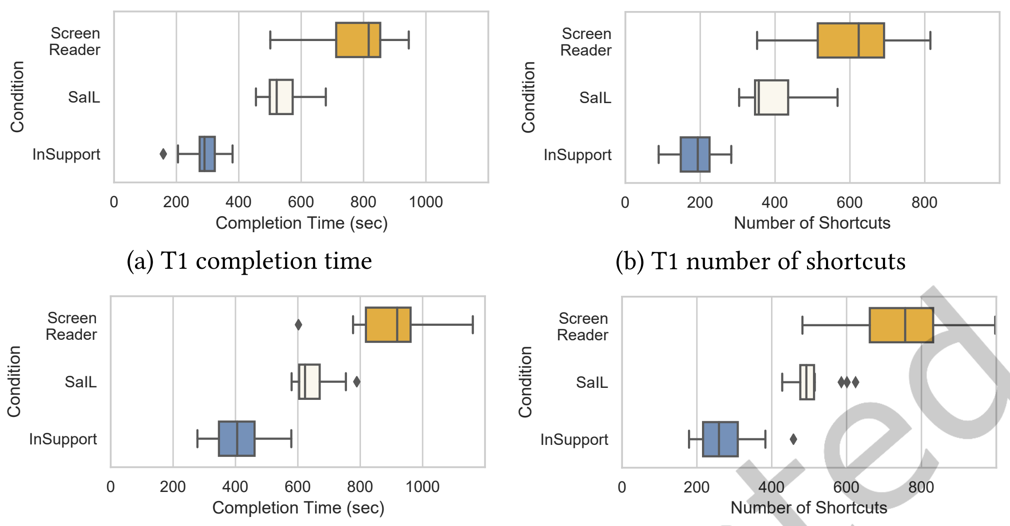
The experiment was conducted remotely, and the participants used their own computers – either laptop or desktop ('Computer Type' column in Table 3). All participants had JAWS screen reader and Google Chrome web browser installed on their computers. The InSupport extension was sent to the participants via email (as a Google Drive link) just prior to the study, and the experimenter also assisted the participants (via Zoom or Skype conferencing software) in installing the extension onto their Chrome browser. Four participants (P3, P8, P11, and P14) needed assistance from their cohabiting family members or friends to install InSupport. Note that for convenience, both the SaIL and InSupport systems were included in the single extension that was emailed to the participants. A special shortcut was provided to turn 'ON/OFF' each condition, and only one condition could be turned on at any given time. Specifically, if the SaIL condition was turned 'ON', the InSupport condition was automatically turned 'OFF', and vice versa.

4.4 Procedure

The experimenter first assisted the participant in downloading and installing the InSupport extension. Next, the experimenter gave the participant enough time to practice (∼ 20 minutes) so as to make them familiar and comfortable with the study conditions. The participant was then asked to complete all the tasks under different study conditions in the predetermined counterbalanced order. For each task, the experimenter allowed a maximum of 20 minutes for the participant to complete the task. The study lasted for a maximum of 3 hours, and all conversations were in English. After completing the tasks, the participant was asked to respond to subjective questionnaires (System Usability Scale (SUS) [21] for measuring usability and NASA Task Load Index (NASA-TLX) [33] to measure perceived user workload), and also participate in an exit interview to collect suggestions and other qualitative feedback. Throughout the study, the screen-sharing and recording features were turned on so as to capture (with the participant's permission) all user interaction activities for subsequent data analysis.
Measurements. During the study, the experimenter recorded task completion times and the number of user actions for each task performed by the participant. The experimenter also recorded the responses to the System Usability Scale (SUS) [21] and the NASA Task Load Index (NASA-TLX) [33] questionnaires. Qualitative feedback and peculiar interaction behavior during the study were also noted by the experimenter. We used an open coding technique [54] for analyzing the transcribed qualitative feedback from the participants. We iteratively went over the user responses and identified key recurring themes or insights in the data.

4.5 Results

4.5.1 Task T1 - Travel. **Task completion time.** Figure 4a presents the results for task completion times of the participants under all three study conditions for Task T1. Overall, in the screen reader condition, the participants spent an average of 780.64 seconds (Median = 816.5, Min = 501, Max = 945), whereas they spent an average of 540.85 seconds (Median = 521.5, Min = 455, Max = 679) with SaIL, and 288.64 seconds (Median = 290, Min = 158, Max = 380) with InSupport. A Kruskal-Wallis test showed that the difference in task completion times between the three study conditions was statistically significant (see Table 7 in Appendix).



(a) T1 completion time

(b) T1 number of shortcuts

(c) T2 completion time

(d) T2 number of shortcuts

Fig. 4. User performance statistics for the two study tasks T1 and T2.

Number of user actions. Figure 4b shows the statistics regarding the number of input actions performed by the participants under the three study conditions. In the screen reader condition, the participants needed an average of 611.85 input actions (Median = 623.5, Min = 352, Max = 815) to complete the task, whereas in the SaIL condition they needed an average of 395.92 input actions (Median = 356.5, Min = 304, Max = 567) to finish the task. However, in the InSupport condition, the participants performed significantly fewer input actions – an average of 187.21 input actions (Median = 193.5, Min = 89, Max = 283) to complete the task. As in case of task completion times, we observed a statistically significant difference between the number of user actions for the three study conditions (Table 7 in Appendix).

An analysis of the study data revealed the underlying reasons for the observed difference in task completion times and the number of actions between conditions for Task T1. In the screen reader condition, most users (12) exhibited the following two types of interaction behavior: (i) navigate the data records one-by-one while accessing only the multi-page links auxiliary segment (7 participants); and (ii) navigate back-and-forth between data records and the filters segment, specifically, repeat the process of navigating the first few records one-by-one and then go back to selecting filters, until the target data record is found (5 participants). These two types of behavior significantly contributed to the increase in completion time and number of actions as the participants traversed a vast amount of DOM content in the process. Only the two remaining participants decided to set all the desired filters first and then serially navigate the data records to locate the target record. Even in the SaIL condition, despite practice, slightly over one-third of the participants (5) did not use the special shortcut for quickly navigating between different landmarks injected by SaIL that covered both the data records and the filters segment. Instead, they chose to navigate over the data records one-by-one as in case of the screen reader condition, which considerably increased the time and input actions overhead. Even in case of the remaining 9 participants who used special SaIL shortcuts, they still had to navigate over quite a few irrelevant segments that were present between the filter options and the data records, since these irrelevant webpage segments too were landmarked by the SaIL saliency model. This problem did not manifest in the InSupport condition as the participants could directly access the auxiliary segments including filter options.

4.5.2 Task T2 - Shopping. **Task completion time.** Figure 4c presents the results for task completion times under all three study conditions for Task T2. Overall, in the screen reader condition, the participants spent an average of 894.35 seconds (Median = 918.5, Min = 602, Max = 1161), whereas they took an average of 646.14 seconds (Median = 623, Min = 580, Max = 789) with SaIL, and 413.64 seconds (Median = 405.5, Min = 278, Max = 579) with InSupport. Similar to Task T1, a Kruskal-Wallis test showed that the difference in task completion times between the three study conditions was statistically significant (see Table 7 in Appendix). All participants were able to successfully complete the task in all conditions.

Number of user actions. Figure 4d presents the statistics regarding the number of input actions performed by the participants in T2 under the three study conditions. In the screen reader condition, the participants needed an average of 751.64 input actions (Median = 756.5, Min = 482, Max = 997) to complete the task, whereas in the SaIL condition, they needed an average of 506.85 input actions (Median = 492.5, Min = 428, Max = 624) to finish the task. However, in the InSupport condition, the participants only performed an average of 276.64 input actions (Median = 259, Min = 179, Max = 458) to finish the task. This difference in the number of user actions between the three study conditions was statistically significant (see Table 7 in Appendix).

In contrast to Task T1 where the participants were focused on locating a pre-specified target record, in Task T2 they exhibited a more exploratory navigational behavior. In the screen reader condition, all participants on at least one occasion navigated to-and-fro between the data records and the filters segment. However, each back-and-forth added significant time and effort overhead given the large of number of DOM elements they had to traverse while navigating between these segments. A majority (9) of the participants exhibited a similar behavior in the SaIL condition, however, the time and effort were considerably reduced due to the advantage of special landmark shortcuts. However, these participants still had to navigate over extraneous segments to go back-and-forth between data records and filters segment. The remaining 5 participants did not use the landmark shortcut and instead performed the task just like how they did in the screen reader condition. In the exit interviews, these participants stated that they forgot the special landmark shortcut (despite practice) and that they were hesitant to try out their guesses for the fear of losing context in the task webpage. In the InSupport condition, all participants were more liberal in their use of filters via the InSupport proxy interface – all participants accessed the filters at least three times while doing the tasks. As the InSupport interface was instantly accessible, the participants did not expend any time or effort accessing the filters. Instead, most of their time and effort (i.e., input actions) were dedicated towards navigating the data records and also the linear list of filters in the proxy interface.

4.5.3 Usability. As mentioned earlier, we administered the standard SUS questionnaire to measure usability [21]. The SUS questionnaire requires the participants to rate alternating positive and negative statements about each study condition on a Likert scale from 1 - strongly disagree to 5 - strongly agree, with 3 - neutral. These responses are then assimilated into a score between 0 to 100, with higher values indicating better usability ratings. Overall, the SUS ratings for the InSupport condition were much higher (μ = 86.07, σ = 6.38) compared to those for the screen reader (μ = 53.92, σ = 13.61) and SaIL (μ = 68.92, σ = 14.66) conditions. A one-way ANOVA test revealed that this difference in SUS scores was statistically significant ($F = 22.865$, $p < 0.0001$). In the exit interviews, most (12) participants attributed their high ratings to the instant interface access feature of InSupport that saved multitude of key presses which were otherwise necessary to navigate between webpage segments. 8 participants also mentioned that the short learning curve of InSupport motivated them to provide favorable usability ratings for InSupport.

4.5.4 Perceived Workload. For workload estimation, we administered the NASA-TLX questionnaire to the participants [33]. The TLX questionnaire measures perceived task workload as a value between 0 and 100, with lower values indicating lower workloads, and hence better results. Overall, the TLX scores were significantly better for InSupport (μ = 25.61, σ = 6.71) than those for screen reader (μ = 74.38, σ = 4.93) and SaIL (μ = 45.57, σ = 6.76). As in case of SUS score, the difference in TLX scores between the three study conditions was statistically significant (one-way ANOVA, $F = 203.401$, $p < 0.0001$). The individual subscales of TLX that contributed the most towards the high total workloads in the screen reader condition were temporal demand, effort, and frustration, i.e., the ratings for these subscales were significantly higher than those for the other subscales (mental demand, physical demand, and overall performance). Effort and frustration subscales were also the major contributors to the workloads in the SaIL condition. For the InSupport condition, however, the ratings were much lower and uniform across all subscales with no obvious patterns.

4.5.5 Qualitative Feedback. In addition to the responses to the questionnaires, the participants also provided subjective feedback in their exit interviews that also included suggestions for improvement and feature requests. Some of the notable recurring themes identified from the exit interview data are mentioned next.

Separate interface for auxiliary segments is important. Almost all (12) participants stated that having a separate proxy interface for accessing the auxiliary segments was important because it helped them "separate their concerns", i.e., use screen reader shortcuts only for navigating within the data records and not worry about how to navigate to the auxiliary segments. As quoted by P4, "I only have to remember the layout of content in each item of the results, and not the entire webpage." In fact, a few (3) participants even suggested providing a different input method such as speech to access the InSupport proxy interface, so as to completely disentangle InSupport from the screen reader keyboard shortcuts.

Mitigating webpage exploration reduces frustration and leads to better record selection. All participants stated much of the frustration and fatigue during web browsing stems from the tedious serial exploration of webpage content using keyboard shortcuts, and therefore they typically cannot explore many data records before their selection. A majority (11) of the participants further stated that due to limited exploration caused by fatigue, they often miss out on the "best deals". These participants expressed that as fatigue and frustration are significantly lower with InSupport, they can explore more data records and therefore take advantage of better deals. This was best quoted by P7, "More coverage means more options, and more options means more likelihood of finding a better product".

Remembering and reusing past filters can increase efficiency of data record interaction. More than one third (5) of participants expressed a desire for remembering the past selection of filters and then automatically applying in future interactions involving the same or similar data records. For example, P7 suggested that InSupport should be able to remember his flight preferences based on prior interaction data and then automatically apply these filters every time he searches for flights, not only on the same website but only on other travel websites. These participants indicated that such a feature would significantly reduce their interaction burden as the list of filters itself can sometimes be very long.

All data records on one single page is preferable. 7 participants mentioned that they would like to have all data records on single webpage, so that they did not have to rely on multi-page links to go over multiple webpages. The main reason given by these participants was that every new page load in the browser tends to refocus the screen reader cursor to the top of the page, and sometimes it is tedious and cumbersome to navigate to the data records from the top of the page. These participants desired the InSupport to be capable of prefetching all the data records and appending them to the list on the first webpage.

5 EVALUATION WITH LOW VISION SCREEN MAGNIFIER USERS

We also conducted a user study with screen magnifier users to evaluate the potential of InSupport in improving low vision usability of interaction with web data records.

ID	Age/ Gender	Diagnosis (C - Congenital, A - Adventitious)	Visual Acuity		Max Zoom	Daily Web Browsing
			Left Eye	Right Eye		
P1	25/F	Cataract (A)	20/200	20/500	5×	4 hours
P2	55/F	Macular degeneration (A)	20/100	20/200	3×	2 hours
P3	43/M	Retinitis pigmentosa (A)	20/400	20/400	6×	3 hours
P4	53/F	Glaucoma (A)	NA	NA	4×	6 hours
P5	49/F	Chorioretinal scarring (C)	20/400	20/200	6×	2 hours
P6	31/M	Glaucoma (A)	0	20/200	6×	4 hours
P7	49/F	Glaucoma (C)	20/200	20/400	5×	5 hours
P8	39/M	Stevens-Johnson syndrome (A)	20/200	20/200	4×	2 hours
P9	53/F	Leber congenital amaurosis (C)	NA	NA	6×	2 hours
P10	36/M	Optic atrophy (A)	20/200	20/100	4×	6 hours
P11	39/M	Astigmatism (C)	20/400	20/200	5×	3.5 hours
P12	31/F	Retinitis pigmentosa (A)	20/700	20/200	8×	2 hours
P13	57/M	Diabetes (A)	NA	NA	6×	2.5 hours
P14	22/M	Congenital retinal scar (C)	20/400	20/400	6×	5 hours
P15	36/F	Glaucoma (A)	20/200	20/100	3×	3 hours
P16	52/M	Macular degeneration (A)	20/200	20/200	4×	2 hours

Table 4. Participant demographics for the user study with low vision participants. The participants self-reported all the data.

5.1 Participants

We recruited 16 low vision participants who had a variety of low vision conditions as shown in Table 4. The average age of the participants was 41.8 (Range = 22-57), and the gender representation was equal (8 female, 8 male). For this study, we only considered BVI users who relied on screen magnifiers to interact with applications; BVI users with extremely poor visual acuity who could not use screen magnifiers and instead need to rely on screen readers, were excluded from the study. All participants stated that they used one of the following screen magnifiers: Windows Magnifier [47], ZoomText [55], and Apple Zoom [34]. As noticeable in Table 4, the participants spent at least 2 hours (average) a day browsing the web. Also, the visual acuity of the participants ranged between 20/100 (good eye) and 20/700 (bad eye).

5.2 Design

In a within-subject experiment, the participants were asked to perform representative tasks under the following study conditions:

- *Screen Magnifier* (Magnifier): The participants only used their preferred screen magnifier (e.g., ZoomText) to do the assigned task.

- *Screen Magnifier + Space Compaction* (Compaction): The participants used their preferred screen magnifier to do the task, but a state-of-the-art space-compaction method [17] (see Section 2.2) was also applied to the magnified webpage to reduce panning and preserve local relationships between elements.
- *Screen Magnifier + InSupport* (InSupport): The participants used their preferred screen magnifier to do the assigned task, however they could also leverage the InSupport popup interface while interacting with the data records.

The tasks selected for the study were the same as those used in the earlier user study with the blind participants:

- T1 – Locate a data record on a travel website that matches a predefined criteria (e.g., morning flight, Delta carrier, price less than $300).
- T2 – Locate a desired data record on a shopping website based on the participant's own personal preferences.

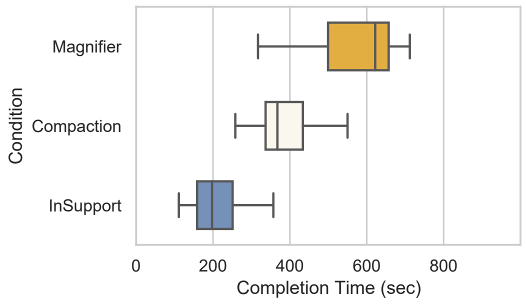
The rest of the study design was identical to that in the earlier study with blind users. To tackle learning effect and confounds, we used three different travel websites for T1 (Kayak, Travelocity, and Orbitz) and three different shopping websites for T2 (Amazon, eBay, and Target). Moreover, we chose three different but similar query items ('laptop', 'desktop', and 'tablet') for T2. For T1, the target data record was located in the second data records webpage (between fifth and eighth positions) on all three chosen websites. We also ensured that InSupport was able to accurately extract the auxiliary segments so as to prevent the confounding impact of extraction algorithm accuracy on InSupport user interface evaluation. The assignment of websites to conditions, items to websites, and the ordering of both tasks and conditions were counterbalanced using the Latin square method [20].

5.3 Apparatus

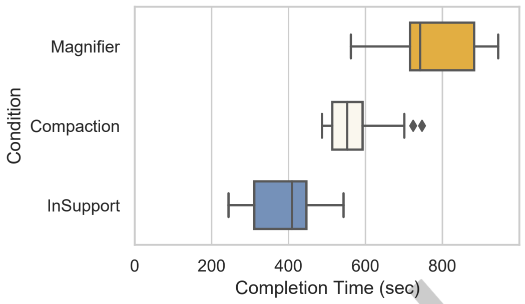
The study was remotely conducted, where the participants used their own computers to perform the study tasks. We used the Zoom or Skype conferencing software for communication and screen-sharing, and we recorded the whole study session after obtaining the participants' consent. The participants P4, P5, P8, P12, and P14 used the Apple's built-in Zoom magnifier on their Macbook laptops, and all others used either the ZoomText or Windows screen magnifier on their desktop computers/laptops. All participants had Google Chrome web browser installed on their computers. The InSupport extension was emailed as a shared Google Drive to the participants, and the experimenter helped the participants setup the browser extension via the conferencing software. For convenience, space compaction method was also included in the InSupport extension, and it could be turned ON/OFF using a special shortcut. When this option was turned ON, InSupport features were disabled to simulate the Compaction study condition, and when it was OFF, InSupport features were enabled to simulate the InSupport study condition. No participant required any additional assistance from their family members to setup the extension.

5.4 Procedure

The experimenter procedure was identical to that in the earlier study. First, the experimenter helped the participants in setting up the InSupport extension in their Chrome browser. Next, the experimenter allowed the participants to practice enough ($\sim$ 20 minutes) to get comfortable using both the *Compaction* and the *InSupport* study conditions. After practice, the experimenter asked the participants to do the tasks in the predetermined counterbalanced order. The participants were given a maximum of 20 minutes for each task. The duration of the study did not exceed 3 hours for any participant and the conversations were in English. After completing the tasks, the participants were administered subjective questionnaires such as System Usability Scale (SUS) [21] and NASA Task Load Index (NASA-TLX) [33]. The participants were finally debriefed in an exit interview where subjective opinions about the study conditions and other qualitative feedback were collected. During the study, the screen-sharing and recording features were turned on to capture (with the participants' permission) all interaction activities for subsequent data analysis.



(a) T1 completion time



(b) T2 completion time

Fig. 5. Completion time statistics for the two study tasks T1 and T2.

Measurements. Completion times for the study tasks were recorded during the study. Subjective data included responses to the System Usability Scale (SUS) [21] and NASA Task Load Index (NASA-TLX) [33] questionnaires, as well as the qualitative feedback and suggestions from the exit interviews. The completion times, SUS scores, and TLX scores were analyzed using inferential statistics; qualitative data was analyzed using an open coding technique [54], where we iteratively went over the user responses and identified key recurring themes or insights.

5.5 Results

5.5.1 Task Completion Time. The completion time statistics for the two study tasks in each of the three conditions are shown in Figure 5. For both tasks, the performance of the participants was the highest under the proposed InSupport condition, followed by the Compaction condition and lastly, the default Magnifier condition. Specifically, for the task T1, the average task completion time under the InSupport condition was 207.87 (Median = 197.5, Min = 111, Max = 357), whereas it was 386.06 (Median = 367.5, Min = 258, Max = 550) and 570.43 (Median = 622, Min = 317, Max = 712) respectively in the Compaction and Magnifier study conditions. For the task T2, the average task completion time under the InSupport condition was 393.18 (Median = 409, Min = 244, Max = 543), which was much lower than that in the Compaction condition (Average = 574.06, Median = 552.5, Min = 487, Max = 747) as well as that in the Magnifier condition (Average = 774.87, Median = 742, Min = 562, Max = 945). The difference in task completion times between the three study conditions was also found to statistically significant in both tasks by a Kruskal-Wallis test (see Table 8 in Appendix). Note that all participants successfully completed all the tasks in all conditions.

Analysis of the study data revealed that in the Magnifier condition, almost all (14) participants spent significant amount of time panning to-and-fro between the filter options and the web data records. This was because they did not select all the desired filter options in one sitting (presumably because only a few options were visible within the magnifier viewport at any instant). Instead, they repeated the process of selecting one or two filters and then perusing the refreshed list of data records before going back to the filter options to change/select other filters. Only two participants spent time panning and scrolling over the entire list of filter options only once and selecting all the desired filters before navigating back to the data records. Some participants also spent a significant amount of time scrolling through the enlarged list of data records to access the multi-page links.

We noticed that participants' behavior was similar in the Compaction condition, but the amount of panning and scrolling was significantly reduced due to decrease in the overall area of enlarged content. However, as noticeable in Figure 5, this decrease in user effort (and thereby time) was still incomparable to the drastic reduction in user effort under the InSupport condition. In the InSupport condition, all participants mostly relied on the instantly accessible popup interface to select the desired filters, sort options, and multi-page links, and therefore they did not expend much panning-and-scrolling effort in accessing the actual auxiliary segments on the webpage. Only on a few handful (6) occasions, the participants accessed the auxiliary segments on the webpage instead of the InSupport popup interface, presumably due to habit.

As in case of the previous study with blind users, all low vision participants were more explorative in the task T2 than in the task T1. The participants accessed the popup interface significantly more times during T2 (μ = 3.56, σ = 0.99) than in T1 (μ = 1.68, σ = 0.68). They especially did this to try out an assortment of filter options and explore as many data records as possible before making a decision to choose one of the records for completing the task T2.

5.5.2 Usability. As specified earlier, the usability of each condition was measured via the well-known SUS questionnaire [21]. In the SUS questionnaire, the participants had to rate alternating positive and negative statements about each study condition on a Likert scale from 1 - strongly disagree to 5 - strongly agree with 3 - neutral. The individual responses to the statements were then combined into a single score between 0 to 100, where higher scores indicate better usability. Overall, the average SUS score for the InSupport condition was much higher (μ = 84.21, σ = 5.42) than those for both Compaction (μ = 63.28, σ = 14.65) and Magnifier (μ = 48.75, σ = 13.54) conditions. This difference in SUS scores between study conditions was also found to be statistically significant (one-way ANOVA test, $F = 33.45$, $p < 0.001$). All participants mentioned in the exit interviews that their high usability scores for InSupport were primarily due to its simplicity and also due to its 'push-based' instant-access support for auxiliary segments. Five participants also mentioned that it was easy and quick to get comfortable using the InSupport popup interface.

5.5.3 Perceived Workload. We measured interaction workload using the NASA-TLX questionnaire [33]. TLX scores too have a range of 0 to 100, however lower scores indicate reduced workloads and hence better performance. Overall, InSupport had the least average TLX score (μ = 31.83, σ = 5.37), followed by the Compaction (μ = 55.93, σ = 6.84) and Magnifier (μ = 68.06, σ = 5.28) study conditions. The difference in TLX scores between the study conditions was also found to be statistically significant (one-way ANOVA, $F = 147.7$, $p < 0.001$). A closer inspection of participants' responses revealed that poor ratings for the *Effort* and *Frustration* TLX subscales were the main drivers of higher TLX scores for the Compaction and Magnifier study conditions, compared to the InSupport condition where the ratings for all six TLX subscales were lower and more uniform.

5.5.4 Qualitative Feedback. Analysis of the participants' exit interview data revealed the following notable observations.

It is important to be able to quickly go over the full list of available filters. Nearly two-thirds (10) of the participants stated that it was important for them to know all the filters that were available for selection, as this could help avoid spending time and effort navigating over data records that were irrelevant to them. 7 of these participants stated that it is presently difficult for them to get a quick glance of all available filters, and therefore they just try to leverage the limited number of filters they come across in the first few seconds of panning. They further mentioned that they only look for other filters if their present selection of filters is unable to provide them with relevant data records that satisfy their custom needs. This explains why the participants spent significant amount of time panning to-and-fro between records and filters in both the Magnifier and Compaction study conditions.

Additional sort options based on record attributes will be helpful. Half (8) of the participants expressed that the sort feature in most webpages was inadequate in terms of the available sort options. These participants stated that additional options corresponding to the various attributes of current data records on the page should also be included in the sort feature. This explains why only a few participants accessed the sort options during the study.

It should be possible to manually set the number of records per page. A majority (12) of the participants mentioned that it should be possible to manually set the number of data records per page on any arbitrary website. They mentioned that very few websites support this feature, but they would like this feature to be available on all webpages containing data records. 5 of these 12 participants stated that fewer records per page helped reduce panning and scrolling, and also helped them make better comparisons between the records on the page. The remaining 7 participants had a contrary opinion; they instead wanted a large number of data records on the same webpage so that they didn't have to move between different pages to make comparisons.

Automatic selection of filters for repetitive tasks will improve user experience. One fourth (4) of the participants mentioned that for some of the repetitive web browsing activities like shopping and travel, they would like to avoid selecting the same filters every time they engage in these activities. They wished InSupport could remember their selections and automatically apply them in subsequent interactions with data records. Recall that in the earlier study, blind users too expressed a desire for such an automatic filter selection feature, thereby indicating that this need is common across all users with visual disabilities.

6 DISCUSSION

The user studies demonstrated how InSupport was effective in significantly improving the overall interaction efficiency and usability for BVI users with web data records. There are however a few limitations of InSupport which need to be addressed in the future for further enhancing user experience with data records.

Limitations. A limitation of our work is that the InSupport's proxy interface was evaluated on webpages where the extraction algorithms could accurately identify all the auxiliary segments, and as such we did not consider webpages where one or more of the extraction algorithms had errors. Further elaborate validation of the algorithms and InSupport interface is required to determine the extent to which our findings generalize across arbitrary webpages having different kinds of content layouts and designs. Moreover, the extraction algorithms were designed exclusively for webpages in English, and therefore they need to be extended with additional language-agnostic features to be able to accurately identify auxiliary segments in webpages written in other languages.

Another limitation of our work is that in our study with blind users, our focus was limited to only JAWS screen reader users. Apart from JAWS, a recent survey also found that a larger proportion (59.9%) of blind users rely on other screen readers including NVDA and VoiceOver [60]. While our observations are very likely to generalize across different screen readers due to similarities in how they support web browsing, a formal study to compare the interaction experiences between user groups relying on different screen readers can shed light on the effectiveness of InSupport both within and between groups. Also, the current InSupport is only supported for the Chrome web browser. While Chrome browser is currently the most widely used browser within the BVI user community, there are still significant proportions of BVI users relying on other browsers such as Firefox [60]. In future, we will explore options to extend InSupport support for other browsers, which is mostly an engineering effort.

For training classifiers in identification algorithms, we only considered logistic regression and multi-layer perceptron models due to their widespread popularity in classification tasks. Although these models exhibited high performance values on our test datasets, it is still unclear if these models perform better than other alternatives such as support vector machine, decision tree, and random forest. Exploring these alternatives is the scope of future work.

Also, InSupport presently is unable to handle a false positive scenario where it can extract label information from a non-auxiliary segment incorrectly classified as an auxiliary segment. Although, we observed that such scenarios were rare in our dataset, further analysis and improvements to the InSupport algorithms are necessary to understand and eliminate the possible negative consequences of these classification errors.

The current InSupport prototype supports only desktop/laptop web browsing, and does not yet support mobile web browsing. As smartphones are becoming ubiquitous and users are increasingly browsing the web on smartphones, support for convenient interaction with data records is especially important for BVI users, given that smartphone screen readers offer very few input gestures for blind users, and the screen size is extremely limited for low vision screen magnifier users. Recognizing this emerging need, we will explore options in the future to port InSupport for smartphone browsers.

Lastly, in our study with low vision users, we only analyzed the general trends common across multiple low vision participants. However, low vision community is inherently a heterogeneous group comprising diverse set of eye conditions, where each subgroup (e.g., tunnel vision, peripheral vision) itself may have specific interaction needs and challenges. Therefore, much larger user studies having sufficient representation from the different low vision subgroups need to be conducted in the future, to excavate and address the needs of individual subgroups.

Automatic filter selection and reordering. As mentioned by the participants in both studies, sometimes the list of filters can be very long, and therefore navigating this list can be tedious and cumbersome even with InSupport. Therefore, mechanisms are needed that can automatically determine the set of filters that the user will mostly likely apply, given the user's past interaction history. By leveraging these mechanisms, InSupport will be able to proactively either apply the desired filters on user's behalf or suggest them to the user. This will especially be beneficial in case of repetitive online browsing tasks. InSupport will also be able to dynamically reorder the filters in the proxy interface so that the filters with high likelihood of being selected are placed near the front of the list. Exploring such solutions is the scope of future research.

Setting the number of data records per page. In both studies, some participants expressed a desire to have most of the data records on one single page to reduce navigation between different webpages for making record comparisons. While a few websites do provide an option to choose the number of records per page, most websites do not offer this feature. Prefetching a large number of data records from possibly a multitude of webpages can be challenging due to the significant time overhead that may potentially render the prefetching method impractical for real-time interaction. Exploring feasible alternative approaches to address this issue under such challenging constraints will also be part of our future research.

Additional sort options based on context. In the study with low vision users, some of the participants expressed a need for additional sort options corresponding to the various attributes of data records currently on the webpage. Providing such a functionality will require InSupport to first identify these attributes. Fortunately, there are plenty of extant works on automatic data record extraction [23, 25, 61] that we can exploit for this purpose. Exploring these algorithms and devising mechanisms to support additional sort options also fall under the scope of future work.

Needs of blind users vs. needs of low vision users. The two studies also illuminated many similarities and some differences between the needs and experiences of blind screen reader users and those of low vision screen magnifier users. For instance, as mentioned earlier, some of the participants in both studies desired an option of viewing all the data records in a single webpage. Similarly, both blind and low vision participants stated that automatically applying search filters for repetitive tasks based on prior filter selections would help significantly reduce interaction burden and improve usability of interaction with web data records. Regarding sort options however, low vision users wanted additional options corresponding to attributes of data records, whereas blind users did not specify any additional requirements. Also, the low vision users stated that they preferred to explore a webpage to figure out all available filter options. In contrast, blind users mentioned that they did not like exploring a webpage as it significantly increased the amount of listening and shortcut presses. These similarities and differences suggest that interface-customization features should be included in InSupport so that users with different visual conditions can alter the interface according to their preferences.

Societal impact. Usability of webpages is important to ensure the equality of access to digital content for people with disability, including those with visual impairments. As most websites are primarily designed for convenient sighted interaction, BVI users have to expend significantly more time and effort to do even basic web tasks which their sighted peers can accomplish in a matter of few seconds [8], thereby creating a usability gap in the interaction experience. This paper seeks to narrow this gap for one of the important everyday web tasks – interaction with web data records. By facilitating more convenient interaction with web data records, BVI users too will be able to find 'better deals', complete transactions faster on e-commerce websites, read more posts from their friends with less effort, and so on.

7 CONCLUSION

Interaction with web data records is an important and ubiquitous activity in web browsing. The present interface design for data records however is primarily tailored for sighted interaction and therefore BVI users struggle to locate desired data records with the same ease and efficiency as their sighted peers. To reduce this usability gap between sighted and BVI users, this paper presented InSupport, a browser extension that automatically extracts the important auxiliary segments such as filters and multi-page links from webpages containing data records, and subsequently makes them instantly accessible to BVI users via an easy-to-use proxy popup interface. Evaluation of InSupport in two separate user studies with blind and low vision participants respectively showed that InSupport significantly reduced the interaction effort and also improved the usability and task workload, compared to both state-of-the-art contemporary techniques as well as the participants' own assistive technologies. The studies also illuminated potential avenues for further research on this topic, which included automatic data-driven selection of record filters that a user will most likely apply for a given set of data records.

ACKNOWLEDGMENTS

We thank Yash Prakash and Mohan Sunkara for their assistance in evaluating the algorithms. This research work was funded by NSF award 1805076 and NIH award R01EY030085.

REFERENCES

[1] Ali Abdolrahmani, Ravi Kuber, and Stacy M Branham. 2018. " Siri Talks at You" An Empirical Investigation of Voice-Activated Personal Assistant (VAPA) Usage by Individuals Who Are Blind. In *Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility*. 249–258.

[2] Tousif Ahmed, Roberto Hoyle, Kay Connelly, David Crandall, and Apu Kapadia. 2015. Privacy Concerns and Behaviors of People with Visual Impairments. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 3523–3532. https://doi.org/10.1145/2702123.2702334

[3] Tousif Ahmed, Patrick Shaffer, Kay Connelly, David Crandall, and Apu Kapadia. 2016. Addressing physical safety, security, and privacy for people with visual impairments. In *Twelfth Symposium on Usable Privacy and Security ({SOUPS} 2016)*. 341–354.

[4] Chieko Asakawa and Hironobu Takagi. 2000. Annotation-based transcoding for nonvisual web access. In *Proceedings of the fourth international ACM conference on Assistive technologies*. 172–179.

[5] Chieko Asakawa, Hironobu Takagi, Shuichi Ino, and Tohru Ifukube. 2002. Auditory and tactile interfaces for representing the visual effects on the web. In *Proceedings of the fifth international ACM conference on Assistive technologies*. 65–72.

[6] Vikas Ashok, Syed Masum Billah, Yevgen Borodin, and IV Ramakrishnan. 2019. Auto-Suggesting Browsing Actions for Personalized Web Screen Reading. In *Proceedings of the 27th ACM Conference on User Modeling, Adaptation and Personalization* (Larnaca, Cyprus) *(UMAP '19)*. Association for Computing Machinery, New York, NY, USA, 252–260. https://doi.org/10.1145/3320435.3320460

[7] Vikas Ashok, Yevgen Borodin, Yury Puzis, and IV Ramakrishnan. 2015. Capti-speak: a speech-enabled web screen reader. In *Proceedings of the 12th International Web for All Conference*. 1–10.

[8] Vikas Ashok, Yury Puzis, Yevgen Borodin, and IV Ramakrishnan. 2017. Web screen reading automation assistance using semantic abstraction. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*. 407–418.

[9] Ali Selman Aydin, Shirin Feiz, Vikas Ashok, and IV Ramakrishnan. 2020. SaIL: saliency-driven injection of ARIA landmarks. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*. 111–115.

[10] Sean Bechhofer, Simon Harper, and Darren Lunn. 2006. Sadie: Semantic annotation for accessibility. In *International Semantic Web Conference*. Springer, 101–115.

[11] Jeffrey P. Bigham. 2014. Making the Web Easier to See with Opportunistic Accessibility Improvement. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (Honolulu, Hawaii, USA) *(UIST '14)*. ACM, New York, NY, USA, 117–122. https://doi.org/10.1145/2642918.2647357

[12] Jeffrey P Bigham, Jeremy T Brudvik, and Bernie Zhang. 2010. Accessibility by demonstration: enabling end users to guide developers to web accessibility solutions. In *Proceedings of the 12th international ACM SIGACCESS conference on Computers and accessibility*. 35–42.

[13] Jeffrey P Bigham and Richard E Ladner. 2007. Accessmonkey: a collaborative scripting framework for web users and developers. In *Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A)*. 25–34.

[14] Jeffrey P. Bigham, Tessa Lau, and Jeffrey Nichols. 2009. Trailblazer: Enabling Blind Users to Blaze Trails through the Web. In *Proceedings of the 14th International Conference on Intelligent User Interfaces* (Sanibel Island, Florida, USA) *(IUI '09)*. Association for Computing Machinery, New York, NY, USA, 177–186. https://doi.org/10.1145/1502650.1502677

[15] Syed Masum Billah, Vikas Ashok, Donald E. Porter, and IV Ramakrishnan. 2017. Speed-Dial: A Surrogate Mouse for Non-Visual Web Browsing. In *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility* (Baltimore, Maryland, USA) *(ASSETS '17)*. Association for Computing Machinery, New York, NY, USA, 110–119. https://doi.org/10.1145/3132525.3132531

[16] Syed Masum Billah, Vikas Ashok, Donald E. Porter, and I.V. Ramakrishnan. 2017. Ubiquitous Accessibility for People with Visual Impairments: Are We There Yet?. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 5862–5868. https://doi.org/10.1145/3025453.3025731

[17] Syed Masum Billah, Vikas Ashok, Donald E Porter, and IV Ramakrishnan. 2018. SteeringWheel: a locality-preserving magnification interface for low vision web browsing. In *Proceedings of the 2018 CHI conference on human factors in computing systems*. 1–13.

[18] Michael Bolin, Matthew Webber, Philip Rha, Tom Wilson, and Robert C Miller. 2005. Automation and customization of rendered web pages. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*. 163–172.

[19] Yevgen Borodin, Jeffrey P. Bigham, Glenn Dausch, and I. V. Ramakrishnan. 2010. More Than Meets the Eye: A Survey of Screen-reader Browsing Strategies. In *Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (W4A)* (Raleigh, North Carolina) *(W4A '10)*. ACM, New York, NY, USA, Article 13, 10 pages. https://doi.org/10.1145/1805986.1806005

[20] James V. Bradley. 1958. Complete Counterbalancing of Immediate Sequential Effects in a Latin Square Design. *J. Amer. Statist. Assoc.* 53, 282 (1958), 525–528. https://doi.org/10.1080/01621459.1958.10501456 arXiv:https://amstat.tandfonline.com/doi/pdf/10.1080/01621459.1958.10501456

[21] John Brooke et al. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.

[22] Andy Brown and Simon Harper. 2013. Dynamic injection of WAI-ARIA into web content. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*. 1–4.

[23] Zehuan Cai, Jin Liu, Lamei Xu, Chunyong Yin, and Jin Wang. 2017. A Vision Recognition Based Method for Web Data Extraction. *Advanced Science and Technology Letters* 143 (2017), 193–198.

[24] Isabelle Colombet, Alan Ruelland, Gilles Chatellier, François Gueyffier, Patrice Degoulet, and Marie-Christine Jaulent. 2000. Models to predict cardiovascular risk: comparison of CART, multilayer perceptron and logistic regression.. In *Proceedings of the AMIA Symposium*. American Medical Informatics Association, 156.

[25] Yixiang Fang, Xiaoqin Xie, Xiaofeng Zhang, Reynold Cheng, and Zhiqiang Zhang. 2018. STEM: a suffix tree-based method for web data records extraction. *Knowledge and Information Systems* 55, 2 (2018), 305–331.

[26] Javedul Ferdous, Hae-Na Lee, Sampath Jayarathna, and Vikas Ashok. 2022. InSupport: Proxy Interface for Enabling Efficient Non-Visual Interaction with Web Data Records. In *27th International Conference on Intelligent User Interfaces*. 49–62.

[27] Prathik Gadde and Davide Bolchini. 2014. From screen reading to aural glancing: towards instant access to key page sections. In *Proceedings of the 16th international ACM SIGACCESS conference on Computers & accessibility*. 67–74.

[28] Krzysztof Z. Gajos, Jacob O. Wobbrock, and Daniel S. Weld. 2007. Automatically Generating User Interfaces Adapted to Users' Motor and Vision Capabilities. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology* (Newport, Rhode Island, USA) *(UIST '07)*. ACM, New York, NY, USA, 231–240. https://doi.org/10.1145/1294211.1294253

[29] Steven Gardiner, Anthony Tomasic, and John Zimmerman. 2015. EnTable: Rewriting Web Data Sets as Accessible Tables. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility* (Lisbon, Portugal) *(ASSETS '15)*. Association for Computing Machinery, New York, NY, USA, 443–444. https://doi.org/10.1145/2700648.2811344

[30] Cole Gleason, Amy Pavel, Himalini Gururaj, Kris Kitani, and Jeffrey P Bigham. 2020. Making GIFs Accessible.. In *ASSETS*. 24–1.

[31] Cole Gleason, Amy Pavel, Emma McCamey, Christina Low, Patrick Carrington, Kris M. Kitani, and Jeffrey P. Bigham. 2020. Twitter A11y: A Browser Extension to Make Twitter Images Accessible. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*

(Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376728

[32] Darren Guinness, Edward Cutrell, and Meredith Ringel Morris. 2018. Caption crawler: Enabling reusable alternative text descriptions using reverse image search. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–11.

[33] Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In *Human Mental Workload*, Peter A. Hancock and Najmedin Meshkati (Eds.). Advances in Psychology, Vol. 52. North-Holland, 139 – 183. https://doi.org/10.1016/S0166-4115(08)62386-9

[34] Apple Inc. 2020. Change Accessibility Zoom preferences on Mac - Apple Support. https://support.apple.com/guide/mac-help/change-zoom-preferences-for-accessibility-mh40579/mac.

[35] Julie A. Jacko, Armando B. Barreto, Gottlieb J. Marmet, Josey Y. M. Chu, Holly S. Bautsch, Ingrid U. Scott, and Robert H. Rosa, Jr. 2000. Low Vision: The Role of Visual Acuity in the Efficiency of Cursor Movement. In *Proceedings of the Fourth International ACM Conference on Assistive Technologies* (Arlington, Virginia, USA) *(Assets '00)*. ACM, New York, NY, USA, 1–8. https://doi.org/10.1145/354324.354327

[36] Kaitlin Kirasich, Trace Smith, and Bivin Sadler. 2018. Random forest vs logistic regression: binary classification for heterogeneous datasets. *SMU Data Science Review* 1, 3 (2018), 9.

[37] Lyudmyla Kirichenko, Tamara Radivilova, and Vitalii Bulakh. 2019. Binary classification of fractal time series by machine learning methods. In *International Scientific Conference "Intellectual Systems of Decision Making and Problem of Computational Intelligence"*. Springer, 701–711.

[38] Richard L. Kline and Ephraim P. Glinert. 1995. Improving GUI Accessibility for People with Low Vision. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '95)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 114–121. https://doi.org/10.1145/223904.223919

[39] Jonathan Lazar, Aaron Allen, Jason Kleinman, and Chris Malarkey. 2007. What frustrates screen reader users on the web: A study of 100 blind users. *International Journal of human-computer interaction* 22, 3 (2007), 247–269.

[40] Hae-Na Lee and Vikas Ashok. 2022. Customizable Tabular Access to Web Data Records for Convenient Low-vision Screen Magnifier Interaction. *ACM Transactions on Accessible Computing (TACCESS)* 15, 2 (2022), 1–22.

[41] Hae-Na Lee, Sami Uddin, and Vikas Ashok. 2020. TableView: Enabling Efficient Access to Web Data Records for Screen-Magnifier Users. In *The 22nd International ACM SIGACCESS Conference on Computers and Accessibility* (Virtual Event, Greece) *(ASSETS '20)*. Association for Computing Machinery, New York, NY, USA, Article 23, 12 pages. https://doi.org/10.1145/3373625.3417030

[42] V Kathlene Leonard, Julie A Jacko, and JJ Pizzimenti. 2006. An investigation of handheld device use by older adults with age-related macular degeneration. *Behaviour & Information Technology* 25, 4 (2006), 313–332.

[43] Gilly Leshed, Eben M Haber, Tara Matthews, and Tessa Lau. 2008. CoScripter: automating & sharing how-to knowledge in the enterprise. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1719–1728.

[44] Ian Li, Jeffrey Nichols, Tessa Lau, Clemens Drews, and Allen Cypher. 2010. Here's what i did: sharing and reusing web activity with ActionShot. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 723–732.

[45] Letresa McLawhorn. 2001. Leveling the accessibility playing field: Section 508 of the Rehabilitation Act.

[46] Valentyn Melnyk, Vikas Ashok, Yury Puzis, Andrii Soviak, Yevgen Borodin, and IV Ramakrishnan. 2014. Widget classification with applications to web accessibility. In *International Conference on Web Engineering*. Springer, 341–358.

[47] Microsoft. 2020. Use Magnifier to make things on the screen easier to see. https://support.microsoft.com/en-us/help/11542/windows-use-magnifier-to-make-things-easier-to-see.

[48] Paula Montoto, Alberto Pan, Juan Raposo, Fernando Bellas, and Javier López. 2009. Automating navigation sequences in AJAX websites. In *International Conference on Web Engineering*. Springer, 166–180.

[49] Lourdes Moreno, Xabier Valencia, J. Eduardo Pérez, and Myriam Arrue. 2018. Exploring the Web Navigation Strategies of People with Low Vision. In *Proceedings of the XIX International Conference on Human Computer Interaction* (Palma, Spain) *(Interacción 2018)*. Association for Computing Machinery, New York, NY, USA, Article 13, 8 pages. https://doi.org/10.1145/3233824.3233845

[50] Christopher Power, André Freire, Helen Petrie, and David Swallow. 2012. Guidelines Are Only Half of the Story: Accessibility Problems Encountered by Blind Users on the Web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Austin, Texas, USA) *(CHI '12)*. ACM, New York, NY, USA, 433–442. https://doi.org/10.1145/2207676.2207736

[51] Alisha Pradhan, Kanika Mehta, and Leah Findlater. 2018. "Accessibility Came by Accident": Use of Voice-Controlled Intelligent Personal Assistants by People with Disabilities. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3174033

[52] Yury Puzis, Yevgen Borodin, Rami Puzis, and IV Ramakrishnan. 2013. Predictive web automation assistant for people with vision impairments. In *Proceedings of the 22nd international conference on World Wide Web*. 1031–1040.

[53] Yury Puzis, Yevgen Borodin, Andrii Soviak, Valentyn Melnyk, and IV Ramakrishnan. 2015. Affordable web accessibility: A case for cheaper ARIA. In *Proceedings of the 12th International Web for All Conference*. 1–4.

[54] Johnny Saldaña. 2021. *The coding manual for qualitative researchers.* sage.

[55] Freedom Scientific. 2020. ZoomText Screen Magnifier and Screen Reader - zoomtext.com. https://www.zoomtext.com/.

[56] Andrii Soviak. 2015. Haptic Gloves Prototype for Audio-Tactile Web Browsing. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility* (Lisbon, Portugal) *(ASSETS '15)*. Association for Computing Machinery, New York, NY, USA, 363–364. https://doi.org/10.1145/2700648.2811329

[57] Sarit Felicia Anais Szpiro, Shafeka Hashash, Yuhang Zhao, and Shiri Azenkot. 2016. How People with Low Vision Access Computing Devices: Understanding Challenges and Opportunities. In *Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility* (Reno, Nevada, USA) *(ASSETS '16)*. ACM, New York, NY, USA, 171–180. https://doi.org/10.1145/2982142.2982168

[58] Ryota Tomioka, Kazuyuki Aihara, and Klaus-Robert Müller. 2006. Logistic regression for single trial EEG classification. *Advances in neural information processing systems* 19 (2006).

[59] W3C. 2018. Web Content Accessibility Guidelines (WCAG) Overview. https://www.w3.org/WAI/standards-guidelines/wcag/.

[60] WebAIM. 2019. WebAIM: Screen Reader User Survey #8 Results. https://webaim.org/projects/screenreadersurvey8/

[61] Yan Wen, Qingtian Zeng, Hua Duan, Feng Zhang, and Xin Chen. 2018. An Automatic Web Data Extraction Approach based on Path Index Trees. *International Journal of Performability Engineering* 14, 10, Article 2449 (2018), 11 pages. https://doi.org/10.23940/ijpe.18.10.p21.24492460

[62] Shaomei Wu, Jeffrey Wieland, Omid Farivar, and Julie Schiller. 2017. Automatic alt-text: Computer-generated image descriptions for blind users on a social network service. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*. 1180–1192.

[63] Hanlu Ye, Meethu Malu, Uran Oh, and Leah Findlater. 2014. Current and Future Mobile and Wearable Device Use by People with Visual Impairments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 3123–3132. https://doi.org/10.1145/2556288.2557085

A FEATURES FOR AUXILIARY SEGMENT CLASSIFIERS

Search Form	
Feature	***Description***
Inner text present	Checks whether the candidate node has inner text in its subtree - binary
Search keyword	Checks for the keyword "search" in the inner text of candidate's subtree - binary
Number of search keyword	Number of "search" keyword matches with in the candidate's subtree - integer
Button present	Check whether the candidate has a button in its subtree - binary
Search attribute value	Checks if any attribute in any node within the candidate's subtree has "search" keyword in it - binary
Filter Options	
Feature	***Description***
Checkbox List	Checks if the candidate's subtree contains a list of check boxes - binary
Number of links	Counts the number of links in the candidate's subtree - integer
Number of inputs	Counts the number of input tags in the candidate's subtree - integer
URL valid	Check if all the links in the candidate's subtree contain valid URLs - binary
Button list	Checks if the candidate's subtree contains a list of buttons - binary
Sort Options	
Feature	***Description***
Keyword match	Checks inner text of nodes in candidate's subtree for keywords such as *price*, *recommended*, *ratings*, *distance*, *time*, and so on - binary
Keyword count	Counts the number of keyword matches using the same keywords as the previous option - integer
Sort keyword	Check if inner text values of nodes in candidate's subtrees have the keyword *sort* - binary
Option tag count	Counts the number of option tags (if any) in the candidate's subtree - integer
Multi-Page Links	
Feature	***Description***
Number of buttons	Counts the number of buttons in the candidate's subtree - integer
Number of links	Counts the number of links in the candidate's subtree - integer
Common URL	Counts the number of links that have the same domain and subdomain URL - integer
Number of values	Counts the number of nodes in the subtree that have only numeric text such as 1, 2, and 3 - Integer
Keyword present	Checks if inner text of subtree nodes contains keywords such as *page*, *show*, *next*, *previous*, and so on - binary
Keyword count	Counts the number of occurrences of select keywords in the candidate's subtree - integer

Table 5. Features for each auxiliary segment, along with their descriptions.

B CLASSIFICATION PERFORMANCE OF MACHINE LEARNING MODELS FOR AUXILIARY SEGMENTS

Segment Type	**Classifier**	**Precision (%)**		**Recall (%)**		**F-score (%)**	
		Negative	**Positive**	**Negative**	**Positive**	**Negative**	**Positive**
Filter options	Logistic regression	99	100	100	99	99.5	99.5
	Multi-layer perceptron	99	100	100	99	99.5	99.5
Sort options	Logistic regression	91.9	100	100	90.8	95.7	95.1
	Multi-layer perceptron	92.3	100	100	91.3	95.9	95.4
Search form	Logistic regression	90.4	94.5	94.7	90.0	92.5	92.2
	Multi-layer perceptron	90.7	93.2	93.3	90.4	93.5	91.7
Multi-page links	Logistic regression	83.4	100	100	79.9	90.9	88.8
	Multi-layer perceptron	83.4	100	100	79.9	90.9	88.8

Table 6. Classification performance of machine learning algorithms.

C SIGNIFICANCE TEST RESULTS FOR USER STUDIES

Task	**Completion Time**	**Number of User Actions**
Task T1	$H = 34.207$, $df = 2$, $p < 0.001$	$H = 34.465$, $df = 2$, $p < 0.001$
Task T2	$H = 34.369$, $df = 2$, $p < 0.001$	$H = 33.576$, $df = 2$, $p < 0.001$

Table 7. Kruskal-Wallis test for statistical significance between conditions in the study with blind users.

Task	**Completion Time**
Task T1	$H = 35.417$, $df = 2$, $p < 0.001$
Task T2	$H = 35.709$, $df = 2$, $p < 0.001$

Table 8. Kruskal-Wallis test for statistical significance between conditions in the study with low vision users.