AutoDesc: Facilitating Convenient Perusal of Web Data Items for Blind Users

Yash Prakash
Old Dominion University
Norfolk, USA
yprak001@odu.edu

Mohan Sunkara
Old Dominion University
Norfolk, USA
msunk001@odu.edu

Hae-Na Lee
Stony Brook University
New York, USA
haenalee@cs.stonybrook.edu

Sampath Jayarathna
Old Dominion University
Norfolk, USA
sampath@cs.odu.edu

Vikas Ashok
Old Dominion University
Norfolk, USA
vganjigu@odu.edu

ABSTRACT

Web data items such as shopping products, classifieds, and job listings are indispensable components of most e-commerce websites. The information on the data items are typically distributed over two or more webpages, e.g., a *'Query-Results'* page showing the summaries of the items, and *'Details'* pages containing full information about the items. While this organization of data mitigates information overload and visual cluttering for sighted users, it however increases the interaction overhead and effort for blind users, as back-and-forth navigation between webpages using screen reader assistive technology is tedious and cumbersome. Existing usability-enhancing solutions are unable to provide adequate support in this regard as they predominantly focus on enabling efficient content access within a single webpage, and as such are not tailored for content distributed across multiple webpages. As an initial step towards addressing this issue, we developed *AutoDesc*, a browser extension that leverages a custom extraction model to automatically detect and pull out additional item descriptions from the 'details' pages, and then proactively inject the extracted information into the 'Query-Results' page, thereby reducing the amount of back-and-forth screen reader navigation between the two webpages. In a study with 16 blind users, we observed that within the same time duration, the participants were able to peruse significantly more data items on average with AutoDesc, compared to that with their preferred screen readers as well as with a state-of-the-art solution.

CCS CONCEPTS

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

KEYWORDS

Web accessibility, Blind, Visual impairment, Screen reader

IUI '23, March 27–31, 2023, Sydney, NSW, Australia

ACM ISBN 979-8-4007-0106-1/23/03.
https://doi.org/10.1145/3581641.3584049

ACM Reference Format:
Yash Prakash, Mohan Sunkara, Hae-Na Lee, Sampath Jayarathna, and Vikas Ashok. 2023. AutoDesc: Facilitating Convenient Perusal of Web Data Items for Blind Users. In *28th International Conference on Intelligent User Interfaces (IUI '23), March 27–31, 2023, Sydney, NSW, Australia.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3581641.3584049

1 INTRODUCTION

Web browsing inevitably entails interaction with data items such as shopping products, classified ads, job postings, hotel or car rentals, and available flights. To facilitate convenient interaction with the web data items, modern websites typically provide ancillary tools such as filters and sort options, and furthermore organize the content of the items into multiple webpages, e.g., a *'Query-Results'* page showing the summary of data items and the *'Details'* pages presenting the full information of the corresponding items (see Figure 1). While sighted users immensely benefit from such organization of content, thanks largely to the abundant visual cues that enable them to quickly scan and obtain desired information about any item, blind users on the other hand have to expend additional time and effort to obtain the same information, due to their dependence on screen reader assistive technology that predominantly supports one-dimensional access of web content [22].

A screen reader, as the name suggests, reads out the content on the screen, and additionally supports special keyboard shortcuts or gestures to navigate the content. In the context of web browsing, a typical screen reader (e.g., JAWS [23], NVDA [46], VoiceOver [4]) enables blind users to traverse a webpage in multiple ways (e.g., navigating by headings, paragraphs, or links). However, navigation is still pretty much one-dimensional – the blind users have to typically listen to reams of text before reaching the webpage segment containing the desired information [14, 36, 51], unless the users are aware of the exact keywords to search for in the webpage using the browser's in-built search functionality. As a consequence, blind screen-reader users typically need significantly more time and effort (i.e., input actions) to do everyday web browsing tasks compared to their sighted peers [11, 14].

This usability divide is likely to be higher for interaction concerning web data items, since the relevant content is usually split across multiple webpages. This is because in such scenarios (e.g., see Figure 1), blind users have to constantly navigate back-and-forth between the different pages to obtain additional relevant information about each of the items. The presence of other HTML content (headers,

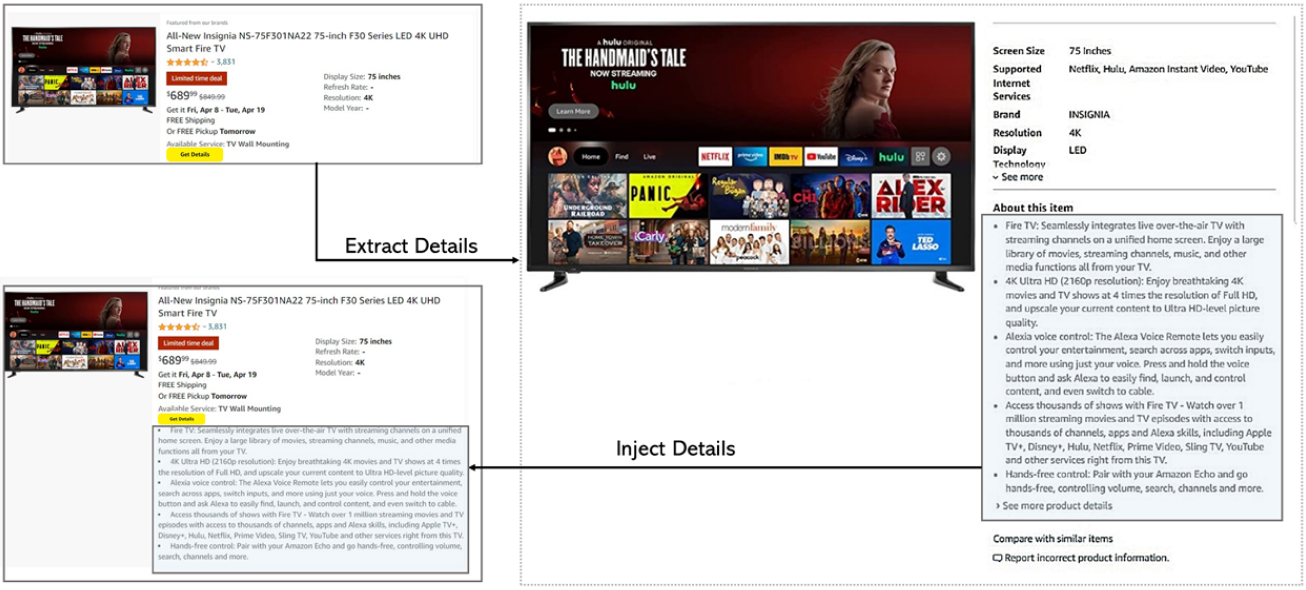


Figure 1: AutoDesc illustration. Additional item descriptions are automatically extracted from the 'Details' pages and injected into the corresponding item summaries in the 'Query-Results' page, so that users can quickly peek at these item details *in-place* without having to navigate to another page.

footers, menu, side panels, etc.) in the *'Details'* pages further exacerbates the interaction overhead, since the blind users have to spend additional time listening to and skipping such content to reach the desired information on the 'Details' page; sighted users on the other hand can rely on visual cues to almost instantly locate the same desired information. Therefore, it is imperative to bridge this usability gap between the sighted and blind users, considering the increased human reliance on the web in recent years to perform day-to-day activities [32], and that approximately 70% of blind users regularly use the web [61].

Existing solutions in this regard have primarily focused on efficient and convenient non-visual content access within a single webpage [13, 38] and as such are not geared towards content that is distributed across multiple webpages, as in case of web data items. Therefore, in this paper, we present AutoDesc, a browser extension that lets a blind user "peek" into some of the additional description about any item in the *'Query-Results'* page itself, without having to actually navigate to the corresponding *'Details'* page (see Figure 1), thereby saving the user's time and effort by reducing the number of back-and-forth traversals between the *'Query-Results'* and *'Details'* pages. To automatically identify and extract the relevant information about an item from its corresponding *'Details'* webpage, AutoDesc leverages a deep learning-based extraction model (specifically, a Mask R-CNN model) that was built using a manually-annotated dataset comprising 1050 ground-truth examples.

In a user study with 16 blind participants, we observed that on average, AutoDesc significantly reduced the number of user accesses to the *'Details'* pages of items, compared to that with a state-of-the-art solution (SaIL [8]) as well as their preferred screen reader. Consequently, we also noticed that the average task times and the number of key presses of users while doing the representative study tasks were significantly lower with AutoDesc, compared to those with SaIL and default screen reader. A majority of the participants also explicitly stated that with AutoDesc, they felt that they could peruse more items in the *'Query-Results'* page by quickly filtering out undesirable items, than with their screen readers before interaction fatigue, and therefore enjoy a higher chance of selecting "better deals". In sum, our contributions are as follows:

- Design and development of AutoDesc, a web browser extension that facilitates easy and quick access to additional description of web data items, in order to significantly reduce the interaction overhead for screen-reader users.
- An algorithm to automatically identify and extract additional item description from the *'Details'* webpages of data items.
- Findings of a user study with 16 blind screen-reader users evaluating AutoDesc.

2 RELATED WORK

Our work closely relates to the literature on the following topics: (i) Usability of web data items; and (ii) Web data extraction algorithms.

2.1 Usability of Web Data Items for People with Visual Impairments

Plethora of prior research works have addressed various interaction issues faced by visually impaired users while browsing the web [7, 14, 40, 43, 56, 67]. While a majority of prior research has focused primarily on the accessibility of web content for screen reader users [5, 10, 12, 26, 42, 50, 59, 63], previous works that have explored and addressed the usability of web content interaction for screen reader users have been relatively few [5–9, 13, 18, 22, 37, 39, 60].

A study by Bose et al. [15] aimed to provide recommendations to the WCAG 2.0 accessibility guidelines for improving screen reader users' experience on e-commerce platforms. The study found that on e-commerce platforms, sighted users could select a data item and obtain addition information in the 'Details' page with ease, and then quickly come back to the 'Query-Results' page to resume navigation of the items list. However, the screen reader users experienced multiple usability issues while doing the same tasks: (1) The 'Query-Results' page set was large; (2) Traversing between multiple pages was tedious and cumbersome; and (3) No shortcuts were available to access detailed information on the product directly without content navigation. Similarly, Prati et al. [49] also found that addressing accessibility issues in e-commerce websites for visually impaired users was not sufficient, and hence proposed certain usability guidelines to improve content navigation. Specifically, the guidelines suggested organizing content into hierarchies, placing important content on appropriate pages, and making use of structured navigation flows to avoid coonfusion and disorientation within the websites.

To improve web navigation via screen readers, many extant research works have focused on the use of annotations. For instance, Yesilada et al. [68] developed the "DANTE" system, which transcoded webpages by reducing significant portions of webpages into smaller fragments and then annotating semantic information on each fragment using WAfA ontology[28], thereby significantly improving orientation and spatial awareness of blind users on webpages, and also ensuring ease of navigation whilst using screen readers. A more recent work by Aydin et al. [8] leveraged deep neural networks trained on gaze data to identify 'hot-spots' of the webpage and then inject ARIA landmark roles into the corresponding salient sections in the webpage document object model (DOM). These ARIA landmarks could then be leveraged by a screen reader user to quickly navigate to the corresponding segments using special screen reader shortcuts (e.g., 'D' in NVDA).

Other than general non-visual web usability enhancement techniques, there also have been a few prior works directly addressing the usability of web data items for people with visual disabilities. For instance, Ferdous et al. [22] focused on providing instantaneous access support for the various auxiliary webpage segments such as filters and sort options that are scattered all over the 'Query-Results' page, so that blind screen readers could quickly apply filters and narrow down the search space before perusing data items. In another work by Lee et al. [39], alternative presentation techniques for data items were explored to facilitate convenient and efficient interaction for low vision screen magnifier users. Specifically, they proposed rendering the list of data items as a single table where the columns corresponded to the various attributes (e.g., price) of items and the rows corresponded to the different items in the list. An evaluation with low vision users showed that their system significantly reduced panning effort for low-vision users. While all these aforementioned approaches assist blind users in conveniently interacting with web data items, their scope is limited to a single webpage; they are not geared for scenarios where data is split across multiple webpages, e.g., the item summaries in the 'Query-Results' page and descriptions, reviews of items in separate 'Details' pages. Therefore, in this paper, we present AutoDesc, a scalable approach that improves the usability of non-visual interaction with web data items, by enabling quick and easy access to item information distributed over multiple webpages.

2.2 Web Data Extraction

Websites publish large amounts of information on a daily basis to suit the needs of their users. Therefore, extracting content from the websites plays a vital role for gathering relevant information [54]. Plenty of extraction techniques exist in this area to extract multitude of data from web pages [2, 7, 19, 48], including data items or records [3, 21, 69, 71]. For instance, Alvarez et al. [3] observed that all the items present in the DOM had consecutive sibling subtrees, and all the attributes for each item had the same path from the root node. Based on this observation, they grouped similar subtrees in the DOM using a clustering algorithm to detect data items and then further extracted the item attributes using a multiple string alignment-based method. Their approach as well as other similar methodologies work well in practice since the organization of web data items in webpages usually follows a certain well-defined structure. For our approach, we leveraged one such state-of-the-art STEM algorithm [21] to extract the web data items, due to its robustness and tolerance towards noise in the DOM.

Other than extracting web data items, techniques have also been proposed to extract other web data such as widgets [43], news articles [35], and auxiliary segments [22]. For instance, a recent work by Ferdous et al. [22] presented machine learning-based algorithms to identify and extract support segments such as filter options, sort options, search form, and multi-page links in webpages. Similarly, Melnyk et al. [43] built machine learning models based on handcrafted features to extract an assortment of web widgets such as calendars, popup menus, and chat boxes. However, to the best of our knowledge, none of the extant web data extraction algorithms have focused on item descriptions in 'Details' pages of websites. We therefore, built our own Mask R-CNN [30] based extraction model in AutoDesc for extracting item descriptions.

3 AUTODESC DESIGN

Figure 2 presents an architectural schematic illustrating the workflow of AutoDesc. When the 'Query-Results' webpage is loaded in the browser, AutoDesc leverages the STEM algorithm [21] to identify and extract all web data items and their respective attributes (e.g., title, price, rating). The AutoDesc extension then injects a 'Get details' button into each data item summary as shown in Figure 2. When a user presses this button for an item, AutoDesc first leverages a custom-built Mask R-CNN [30] model to automatically identify the region in the 'Details' page that corresponds to the description of the item. From this region, AutoDesc extracts the text using the Tesseract OCR engine [55] and then post-corrects the text using pre-trained BERT model [20]. Finally, AutoDesc displays the text right below the activated 'Get Details' button on the 'Query-Results' page as shown in Figure 2. Given this quick access to the item descriptions, blind users can potentially save a significant amount of time and effort by avoiding unnecessary back-and-forth navigation between the 'Query-Results' page and the 'Details' pages.

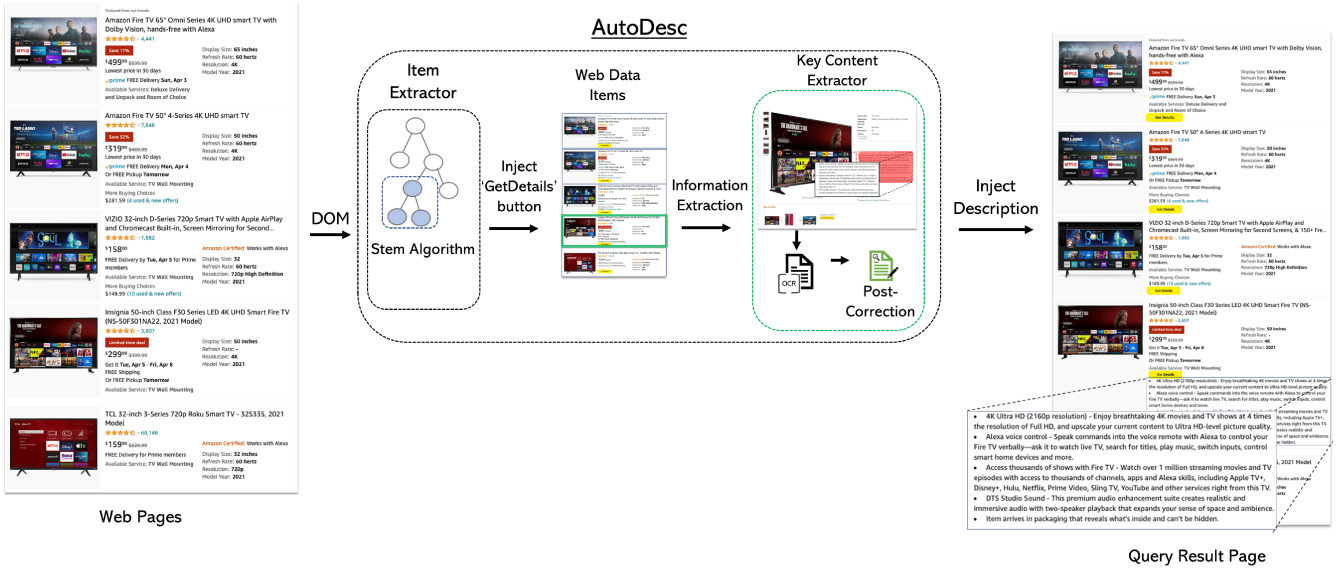


Figure 2: AutoDesc architectural workflow.

3.1 Extracting Data Items

As mentioned earlier, we leveraged the STEM (suffix tree-based extraction method) algorithm [21] given its state-of-the-art performance as well as its application in prior usability solutions [37, 39]. The core idea of STEM is to extract all the HTML tag paths (i.e., <body>, <ul>, <li>) from all nodes in the DOM of the given web page and then assign a unique integer value called HTML tag path identifier (HTPI) to the respective HTML tag paths also known as web page sequence (e.g., 1,2,3,4,2,3). Then a suffix tree [58] is constructed on this web page based on the web page sequence of the webpage. Once a robust tree is constructed, four custom refining filters (i.e., frequency filter, tree filter, gap filter, repetition filter) are used to identify the most optimal node sequence from repeating node sequences in the tree structure. To validate a data item, we computed the Rate Validation Rate (RVR), i.e., the proportion rate of a sub-sequence and a candidate sequence. A greater RVR value for a candidate sequence indicates a higher level of validation for the data item. Corresponding HTML paths are then extracted for all the attributes of each data item, which then is used by the AutoDesc extension to inject the 'Get Details' button and also the extracted description into the 'Query-Results' page.

3.2 Extracting Item Descriptions

As mentioned earlier, to automatically extract item descriptions, we built a custom Mask R-CNN model [65] that can identify the region of the screen belonging to item descriptions in the 'Details' pages of websites. From these regions, we extracted the text using Tesseract OCR engine [55] and then post-processed the text using pre-trained BERT model [44].

Dataset. To train our Mask R-CNN model, we constructed a custom dataset (Github[1]), as no prior datasets regarding the same are publicly available. Webpages for creating the dataset were randomly sampled from a diverse set of websites spanning different domains such as Business and Finance, Home and Garden, Shopping, Style and Fashion, Technology and Computing[2]. Overall, the dataset consisted of 750 images for training and 300 images for validation. Once the dataset was curated, ground truth for the dataset was created by masking relevant information (i.e., item description) on the webpage image. To create respective masks, we made use of the publicly available GIMP [57] software. The original images and masked images were scaled to a standard size of 640x360. Following this, mask definitions were provided to co-relate images with their respective masks. Data information definitions were also created to describe basic features of the dataset (name, URL, creator, etc.) along with licenses associated with respective websites used to build the dataset. Finally, images, mask annotations, and definitions were used to create data instances. (Note: All the images were manually extracted; no synthetic images were generated via data augmentation techniques).

Training. The hardware configuration used was Nvidia V100 GPU with 128GB memory per node. The model was trained for 10 epochs with 500 steps per epoch and validation steps were set to 5. The entire training process was accomplished on 1 GPU with 1 image per GPU. We trained the model in 2 phases. In the first phase, we trained only the head layers of Mask R-CNN model while freezing all the backbone layers. In this phase, the learning rate was set to 0.001. In the second phase, we fine-tuned the entire model by training all the layers with the learning rate set to 0.0001. These parameter values were chosen after optimizing for accuracy using the validation dataset.

We trained Mask R-CNN [1] with ResNet-101 [31] and Feature Pyramid Network (FPN) [41] as the backbone network. FPN uses

[1]https://github.com/accessodu/dataset_repository

[2]The full list of web pages is also available in the above GitHub link

a top-down architecture along with lateral connections, whereas ResNet-FPN backbone improves the overall speed and accuracy of the model. The backbone network extracts feature maps, from which the Region Proposal Network (RPN) [70] generates appropriate webpage object proposals. Following this, Region of Interest Align (RoIAlign) [30] extracts accurate and precise feature maps. Finally, masked branch fully convolution layers are used to create appropriate masks along with existing parallel convolution layers for classification and bounding box regression. The final step is to overlap the identified bounded object with the generated masks as seen in Figure 2 (the red rectangular shaded region within the Key Content Extractor). We monitored the total loss which is the sum of all the losses (loss mask, loss classifier, and loss box regression). Overall, our model on the dataset reported a total loss of 0.5805.

Evaluation. For testing purposes, 20 unseen websites were picked at random, and Average Precision (AP) was calculated for every value of Intersection over Union (IoU). IoU signifies the amount of overlap between the predicted and ground truth bounding box. Precision and recall values were calculated for every IoU following which the precision-recall curve was plotted. The area under the precision-recall curve is defined as AP. As for our results, for an IoU of 50%, the AP was 94.45%, and for a higher IoU of 75%, the AP was as 89.69%.

Post processing. After identifying the regions corresponding to the item descriptions using the custom Mask R-CNN model, we leveraged the Tesseract OCR service [55] to extract all the text in these regions. Note that before using Tesseract, we also preprocessed the images using Otsu's adaptive threshold method [45] to improve image quality. It is well-known that even though OCR has an accuracy of 98 percent at character level, it incorrectly predicts words 10-20 percent of the time [33]. Therefore, we used post-processing mechanisms in two phases [44]. In the first phase, we split the input into tokens [64] and then generated GloVe word embeddings [47] for these tokens. We then fed these embeddings to the pre-trained BERT language model [20] to perform token-level classification for identifying valid and invalid tokens. In the second phase, we used neural machine translation at character level [34], to rectify invalid tokens.

3.3 User Interface

The AutoDesc's user interface design was largely influenced by the findings of Prati et al. [49], who observed in their study that there existed a lack of awareness and shared design guidelines for the visually impaired in the e-commerce community. They then proposed three design principles: i) Uniformity and coherence; ii) Navigation clarity; and iii) Logical provisions after performing expert analysis on current user experience and guidelines. The first principle stated that consistency was very important across web pages and that users should always interact with the same schema on the webpage, for example, the arrangement of web data items and all their corresponding details should be such that the user would not have to search for required information every time. The second and third principles addressed systematic design and navigation flow to avoid the user from getting disoriented within web pages. Following their suggestions, we designed our interface such that the 'Get Details' button and the extracted item descriptions were injected right below each web data item summary in the 'Query-Results' page itself. When the user presses the 'Get Details' button for a particular data item using the screen reader shortcut key 'ENTER', the description associated with the data item is presented right below the button. After the user navigated through the description, the user could press the 'X' shortcut key to close the description, and the screen reader would automatically refocus the cursor to the beginning of the next data item in the list. Thus, with only a few basic shortcut keys, a blind user can quickly access item descriptions using AutoDesc.

3.4 Implementation Details

We implemented AutoDesc as a Chrome browser extension based on the development guidelines publicly available on the Google website[3]. When the extension was enabled, AutoDesc leveraged the browser's in-built JavaScript (JS) functions to extract the entire HTML DOM tree of the webpage containing data items, and then forwarded the DOM content as a POST request to the Item Extractor module running in the back-end server as shown in Figure 2. To extract the data items, we implemented the STEM algorithm [21] ourselves in Python, as there was no publicly available code for the same. Once the data items were identified, AutoDesc extension again used the browser's in-built JS functions to inject two child nodes into each data item's DOM subtree – a visible 'Get Details' button and an invisible ' ' container for future on-demand display of additional item description extracted from the 'Details' page of the item.

To build the Mask R-CNN model for extracting the item description from the 'Details' page, we leveraged the publicly available Matterport project on GitHub[4]. For generating the input to the Mask R-CNN model, we used the Selenium driver [25] that supported the necessary feature to take a snapshot of the entire 'Details' page. As explained earlier in this section, post-processing of the Mask R-CNN output was done using the Tesseract OCR engine [55] and the pre-trained BERT model [20]. After the item description was extracted from the 'Details' page, AutoDesc used built-in browser functions to inject it into the ' ' container of the corresponding item. For establishing a communication channel between the extension JavaScript modules and the back-end server modules, we used the Flask rest API [52]. All the code associated with AutoDesc is available on GitHub[5].

4 EVALUATION

We conducted an IRB-approved user study with blind screen-reader users to assess the efficacy of AutoDesc and compare it with the status-quo screen readers as well as a state-of-the-art web usability-enhancing solution.

4.1 Participants

We recruited 16 blind participants through email lists and word-of-mouth (See Table 1). The inclusion criteria was: (i) familiarity

[3]https://developer.chrome.com/docs/extensions/mv3/architecture-overview/
[4]https://github.com/matterport/Mask_RCNN
[5]https://github.com/accessodu/code_repository

ID	Age/ Gender	Age of Vision Loss	Screen Reader	Proficiency	Computer Type	Typical Online Activities
P1	34/M	Age 5	JAWS	Expert	Desktop	Shopping, Email, Social networking, News, Banking
P2	46/F	Since birth	VoiceOver	Intermediate	Desktop	Shopping, Email, Social networking, Classifieds
P3	55/F	Don't know	JAWS	Beginner	Laptop	Shopping, Social networking, News
P4	28/F	Age 18	NVDA	Intermediate	Laptop	Shopping, Email, News
P5	25/M	Since birth	VoiceOver	Expert	Laptop	Shopping, Email, Social networking, News, Document editing
P6	39/M	Since birth	JAWS	Expert	Laptop	Shopping, Email, News, Video streaming
P7	40/F	Age 6	JAWS	Intermediate	Laptop	Shopping, Email, News
P8	56/F	Don't know	JAWS	Beginner	Desktop	Shopping, Email, Social networking
P9	64/M	Don't know	JAWS	Beginner	Laptop	Shopping, Social networking
P10	66/M	Age 43	JAWS	Beginner	Desktop	Shopping, Email, News
P11	51/M	Since birth	NVDA	Intermediate	Desktop	Shopping, Email
P12	27/M	Age 16	NVDA	Expert	Laptop	Shopping, Email, Social networking, Stock trading, Document editing, Video streaming
P13	21/F	Age 3	VoiceOver	Expert	Laptop	Shopping, Email, Social networking, News
P14	33/M	Don't know	NVDA	Intermediate	Laptop	Shopping, Email, Social networking
P15	36/M	Don't know	VoiceOver	Expert	Desktop	Shopping, Email, Social networking
P16	44/F	Age 8	VoiceOver	Intermediate	Laptop	Shopping, Email, Social networking, News

Table 1: Participant demographics. All information is self-reported by the participants. 'Proficiency' indicates the participants' self-assessed proficiency in using screen readers, and 'Computer Type' represents the computer types used in the user study.

with web browsing using a screen reader on desktop/laptop environment; (ii) familiarity with the Chrome web browser; and (iii) ability to communicate in English. The gender representation was approximately equal (7 female, 9 male), and the average age of the participants was 41.56 (Median = 39.5, Min = 21, Max = 66). All participants stated that they relied exclusively on screen readers for interaction with websites, and that they browsed the web on a daily basis. Typical web activities as stated by the participants included shopping, social media, video streaming, classifieds, news reading, and booking. All participants mentioned that they owned a computer – either a desktop or a laptop. Regarding screen reader proficiency, 6 participants considered themselves to be experts, 6 participants self-assessed to have intermediate proficiency, and 4 participants felt like they were still beginners. No participant reported having other difficulties (e.g., hearing, motor control) that could possibly affect their ability to perform the study tasks. Table 1 presents the participants' demographic information.

4.2 Design

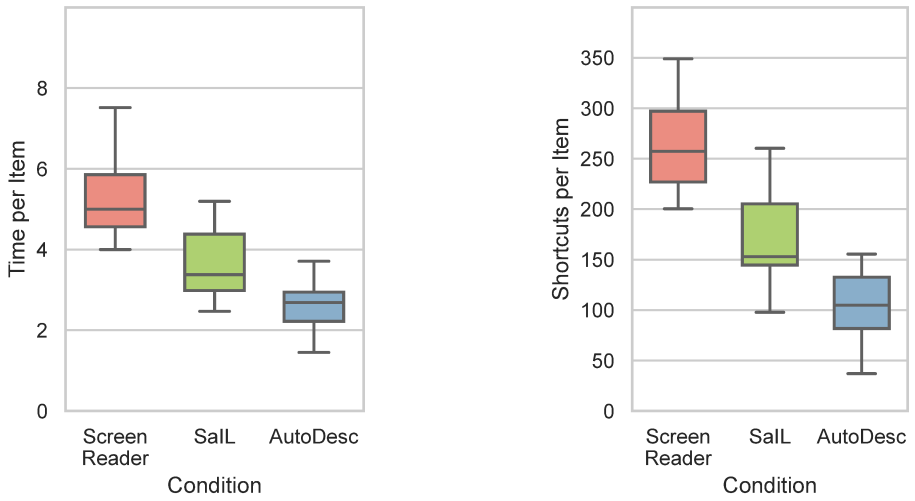
In a within-subject experimental setup, we asked the participants to perform representative tasks under 3 conditions:

- Screen Reader – The participants do the tasks on webpages that have not been externally enhanced for usability.
- SaIL – The participants do the tasks on webpages that have special ARIA landmarks injected by a state-of-the-art method, namely SaIL [8]. SaIL uses a visual saliency model to determine the important segments of a webpage and then injects landmarks at the beginning of these segments, so that the users can quickly navigate to these segments using special screen reader shortcut (e.g., 'R' with the JAWS).
- AutoDesc – The participants do the tasks on webpages that have additional pre-fetched item descriptions injected into them by our AutoDesc browser extension.

In each of these conditions, the participants performed the task of perusing a list of products on a shopping website and selecting a product that best matches their personal preferences. We selected the task to mimic realistic scenarios where people usually go over a list of items, compare their attributes, and finally select one of the items that they like the most in the list. To mitigate learning effect, we did not use the same website more than once while doing the task in different conditions. Specifically, we selected the following three different shopping websites: GAP, Best Buy, and Macy's. Also, the SaIL and AutoDesc algorithms did not exhibit any errors on these selected websites, thereby avoiding any confounding effect of algorithm accuracy on final performance values. Furthermore, we used three similar queries for the task - 'television', 'shirts', and 'furniture'. An illustration of one of the study tasks has been provided in Appendix A. The assignment of websites to conditions and the ordering of tasks and conditions were counterbalanced using the Latin square method [16].

4.3 Apparatus

The study was conducted remotely via Zoom or Skype (depending on the participants' preferences), and the participants used their own computers and preferred screen readers to do the study tasks. All the webpages corresponding to the tasks were pre-processed, cached, and hosted on a secure web server. At pre-processing, the landmarks or the AutoDesc descriptions were apriori determined and injected into the task webpages before caching them onto the web server. This way, the apparatus allowed us to evaluate



(a) Avg. time spent per item

(b) Avg. shortcut presses per item

Figure 3: Time and shortcut statistics for all the study conditions. The box plots shows the average time spent per item and the average number of shortcut presses per item across all study participants.

AutoDesc against SaIL and default screen reader without having to actually share and install the corresponding browser extensions on the participants' computers. Specifically, we just emailed the URLs of the webpages to the participants who then opened these pages in their browsers to do the corresponding tasks.

4.4 Procedure

The experimenter started the study by obtaining the participant's formal consent and then briefly explaining to the participant the study purpose. The experimenter then introduced the SaIL landmarks and the AutoDesc descriptions to the participant and conducted a practice session to let the participant become familiar and comfortable with SaIL and AutoDesc. After the practice session, the experimenter shared the URLs of all the cached task webpages to the participant, and asked the participant to complete all the tasks one-by-one according to the predetermined counterbalanced order. The experimenter allowed the participant to spend a maximum of 20 minutes to complete each task; note however that the experimenter did not explicitly convey this time limit to the participants in advance, in order to not affect their natural interaction behavior with the data items.

After the participant finished doing all the study tasks, the experimenter administered the System Usability Scale (SUS) [17] and NASA Task Load Index (NASA-TLX) [29] subjective questionnaires for measuring perceived usability and interaction effort respectively for each of the study conditions. In the exit interview, the experimenter engaged in an open-ended discussion with the participant to obtain qualitative feedback including feature requests and suggestions for improvement. With the participant's permission, screen-sharing and recording features were active throughout the study to capture all interaction activities for post study analyses. All conversations were in English, and the participants were compensated with an Amazon gift card.

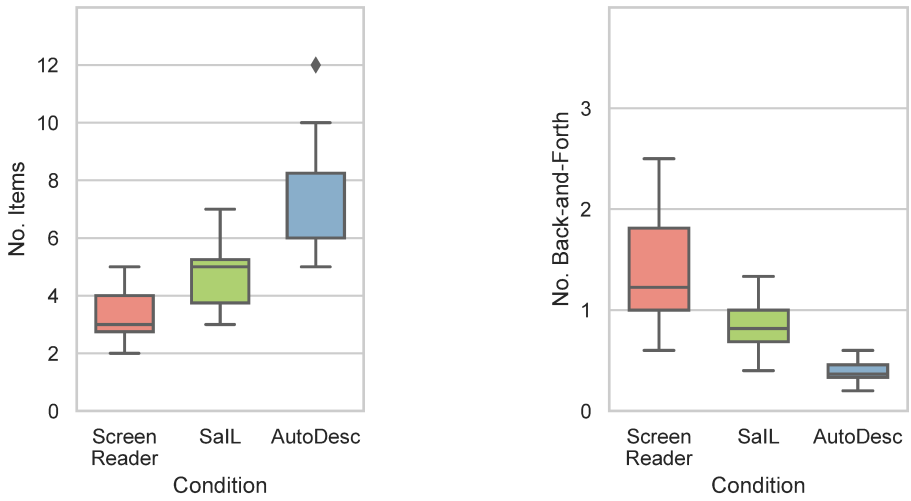
4.5 Measures

From the study data, we computed the following metrics for each participant: (i) Average time spent per item; (ii) Number of items covered while doing a task; (iii) Average number of shortcuts pressed per item (including navigating the *details* page) while doing a task; (iv) Number of back-and-forth navigation between the *results* and the *details* pages in each task; (v) SUS usability scores; and (vi) NASA-TLX workload scores. We then used these metrics to compare the different study conditions, and determine if there was a significant positive impact of AutoDesc on the overall user experience of the participants with web data items.

4.6 Results

4.6.1 Average time spent per item. The average time spent on an item by a participant during a task was computed by dividing the total time the participant spent doing the task (regardless of completion) by the number of *distinct* items visited during the task. Figure 3a shows the box plots for the average time spent per item for all three study conditions. Overall, participants spent an average of 5.29 minutes (Median = 5.00, Min = 4.00, Max = 7.51) with screen readers, 3.68 minutes (Median = 3.37, Min = 2.47, Max = 5.19) with SaIL, and 2.57 minutes (Median = 2.68, Min = 1.45, Max = 3.71) with AutoDesc. A Kruskal-Wallis test revealed that the difference in time spent per item between the three study conditions was statistically significant ($H = 29.57$, $p < 0.001$). A post-hoc Dunn's test showed that AutoDesc significantly outperformed both the screen reader ($Z = 5.43$, $p < 0.01$) and the SaIL ($Z = 2.69$, $p = 0.007$) conditions.

A deeper analysis of the study data showed that in the Screen Reader condition, the following two interaction activities significantly contributed to the average time per item: (i) Searching for item description and other relevant information (e.g., reviews) in the 'Details' page via one-dimensional shortcut-based content navigation; and (ii) Moving screen reader focus back to the item summary



(a) Avg. distinct items covered in a task

(b) Avg. back-and-forth navigation between pages

Figure 4: Box plots for the average number of distinct items visited and the average number of back-and-forth navigation between 'Query-Results' and 'Details' pages in all three study conditions.

in the 'Query-Results' page after navigating back from the 'Details' page of the item. While the participants also performed these two activities in the SaIL condition, their contribution to the time spent per item was considerably lower, presumably due to the presence of SaIL-injected ARIA landmarks that enabled the participants to skip through large portions of irrelevant content during navigation. Nonetheless, the participants still had to go through quite a few webpage segments that were also deemed as salient by the SaIL algorithm, before reaching the item descriptions and other relevant segments such as reviews in the 'Details' page. Similarly, after going back to the 'Query-Results' page from a 'Details' page, the participants had to first navigate through other salient segments such as search form, filters, and sort options, before reaching the list of items. Moreover, 4 participants forgot their screen reader shortcut for navigating ARIA landmarks in the middle of the task, so they could not fully exploit the benefits of SaIL. In such a scenario, these 4 participants reverted to their status-quo one-dimensional screen reader interaction, which increased the time spent per item. The effect of the above two activities on the time spent per item was much lower in the AutoDesc condition, as the participants increasingly leveraged the *Get Details* button in the item summaries to obtain desired information "inline" in the 'Query-Results' page itself, instead of navigating to the corresponding 'Details' pages. However, for a very few items (mostly *television* and *shirts*), the participants did navigate to the corresponding 'Details' pages even if the description was accurately extracted and made available on the the 'Query-Results' page. The reason for this behavior was that the participants wanted to access other desired information such as *customer reviews* and *product comparisons* in the 'Details' pages.

4.6.2 Average number of shortcut presses per item. Like the *avg. time spent per item* metric, the average number of shortcuts per item pressed by a participant in a task was computed by dividing the total number of shortcuts the participant pressed during the task (regardless of completion) by the number of *distinct* items visited during the task. Figure 3b shows the box plots for the average number of shortcuts per item for all three study conditions. Overall, participants pressed an average of 264.63 shortcuts (Median = 257.37, Min = 200.4, Max = 349.0) with screen readers, 172.49 shortcuts (Median = 153.0, Min = 97.8, Max = 260.33) with SaIL, and 105.82 shortcuts (Median = 104.8, Min = 37, Max = 155.5) with AutoDesc. A Kruskal-Wallis test revealed that the difference in shortcuts pressed per item between the three study conditions was statistically significant ($H = 32.04$, $p < 0.001$). A post-hoc Dunn's test further showed that AutoDesc significantly outperformed both the screen reader ($Z = 5.65$, $p < 0.001$) and the SaIL ($Z = 2.63$, $p = 0.008$) study conditions. As in case of the average time per item metric, the average number of shortcuts per item was influenced by the amount of navigational effort expended by the participants in: (i) locating the item descriptions in 'Details' page, and (ii) navigating back to the corresponding item summary in the 'Query-Results' page after returning from the 'Details' page. This navigational effort was the highest in the Screen Reader study condition and lowest in the AutoDesc condition.

4.6.3 Number of items covered during task. The metric captured the number of distinct items that a participant explored while doing a task. All revisits to previously explored items were not included in the final count. Figure 4a presents the results for this metric across all three study conditions. Overall, the participants visited an average of 3.37 items (Median = 3, Min = 2, Max = 5) with screen reader, 4.62 items (Median = 5, Min = 3, Max = 7) with SaIL, and 7.06 items (Median = 6, Min = 5, Max = 12) with AutoDesc. The differences in the number of items covered across the three study conditions was statistically significant based on a Kruskal-Wallis test ($H = 27.5$, $p < 0.001$). As in case of previous metrics, a post-hoc Dunn's test further showed that AutoDesc significantly outperformed both the screen reader ($Z = 5.21$, $p < 0.001$) and the SaIL

($Z = 3.08$, $p = 0.002$) study conditions. As the participants spent significantly more time navigating irrelevant content in the Screen Reader and SaIL conditions, they couldn't cover many distinct data items in the 'Query-Results' page before either interaction fatigue set in or allotted time for the task expired.

4.6.4 Number of back-and-forth navigation between webpages. This metric was computed by dividing the total number of visits to 'Details' pages (recall that the participant starts a task in the corresponding 'Query-Results' page) by the total number of distinct items visited while doing the task. Figure 4b presents the results for this metric across all three study conditions. Overall, the participants navigated back-and-forth between 'Query-Results' page and 'Details' pages an average of 1.38 times (Median = 1.22, Min = 0.6, Max = 2.5) per item under the screen reader condition, 0.86 times (Median = 0.81, Min = 0.4, Max = 1.33) under the SaIL condition, and only 0.38 times (Median = 0.36, Min = 0.2, Max = 0.6) under the AutoDesc condition. The difference in the metric values between the three study conditions was found to statistically significant (Kruskal-Wallis test, $H = 31.96$, $p < 0.001$). A post-hoc Dunn's test further showed that AutoDesc significantly outperformed both the screen reader ($Z = 5.58$, $p < 0.001$) and the SaIL ($Z = 3.56$, $p < 0.001$) study conditions.

This result shows that providing instant access to just the descriptions of items can significantly reduce the number of visits to the 'Details' pages, and therefore decrease the time and shortcuts needed per item. As observable in Figure 4, for a majority of items in the AutoDesc condition, the participants did not access the corresponding 'Details' pages after listening to the extracted descriptions in the 'Query-Results' page itself, which indicates that these descriptions enabled the participants to quickly filter out items that did not match their preferences. Only for a few select items that matched their preferences, the participants visited the 'Details' page to access other relevant information such as customer reviews. In the Screen Reader and SaIL conditions, however, such instantaneous access to item descriptions was not available, so the participants had to spend additional time and effort navigating the 'Details' pages themselves to find out more information about the corresponding items and decide if these items matched their personal preferences.

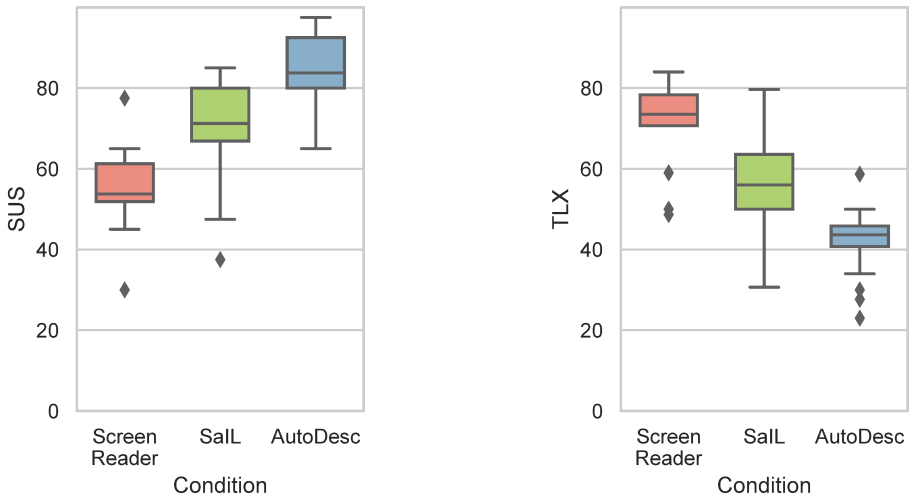
4.6.5 System Usability Scale (SUS). We used the standard SUS questionnaire to assess usability [17]. The SUS questionnaire asks participants to rate alternating positive and negative Likert items on a scale of 1 to 5, where 1 being strongly disagree, 3 being neutral, and 5 being strongly agree. The responses are then combined into a single score between 0 and 100, with higher scores indicating better usability evaluations. Figure 5 presents the SUS statistics for the three study conditions. Overall, the SUS ratings for the AutoDesc condition (Average = 84.53, Median = 83.75, Min = 65, Max = 97.5) were higher than those for both the screen reader (Average = 55.31, Median = 53.75, Min = 30, Max = 77.5) and SaIL (Average = 70.15, Median = 71.25, Min = 37.5, Max = 85) conditions. This difference in SUS scores was statistically significant, according to a Kruskal-Wallis ANOVA test ($H = 27.79$, $p < 0.001$). Pairwise comparisons between conditions using post-hoc Dunn's test showed that AutoDesc had significantly better SUS scores than both the screen reader ($Z = 5.27$, $p < 0.001$) and the SaIL ($Z = 2.56$, $p = 0.01$) study conditions. In the exit interviews, almost all (15) participants attributed their high usability ratings to the *instant access of item description* feature of AutoDesc that saved them a lot of time and key presses that would have been otherwise necessary to navigate between and within 'Query-Results' and 'Details' webpages. Nearly two-thirds (10) of the participants also mentioned that they did not have to learn anything new to leverage AutoDesc, which prompted them to give positive usability feedback for AutoDesc.

4.6.6 Perceived interaction workload. We administered the standard NASA-TLX questionnaire [29] to gauge the workload experienced by the participants while they did the assigned tasks in different study conditions. Like SUS, NASA-TLX also generates a score between 0 and 100 to estimate perceived task workload, however unlike SUS, lower TLX values indicate lesser workloads and hence better performance. Overall, we observed that the study conditions had a significant impact on NASA-TLX scores (Kruskal-Wallis ANOVA, $H = 28.67$, $p < 0.001$). The TLX scores for the AutoDesc (Average = 41.93, Median = 43.66, Min = 23, Max = 58.66) condition were significantly lower than those for the SaIL (Average = 56.4, Median = 56, Min = 30.66, Max = 79.66) and screen reader (Average = 71.60, Median = 73.5, Min = 48.66, Max = 84) conditions (post-hoc Dunn's test, $Z = 5.35$, $p < 0.001$ for Screen Reader vs. AutoDesc, and $Z = 2.73$, $p = 0.006$ for SaIL vs AutoDesc). A closer inspection of the responses to individual subscales of the TLX questionnaire revealed that the *Mental Demand*, *Effort*, and *Frustration* subscales contributed more towards the higher workload scores in the Screen Reader and SaIL conditions, i.e., the ratings for these subscales were significantly higher than those for the other subscales (*Temporal Demand*, *Physical Demand*, and *Overall Performance*). For the SaIL condition, *Mental Demand* and *Effort* were the discriminating subscales that received significantly higher ratings than the other subscales. The ratings for AutoDesc however were much lower and uniform across all six subscales.

4.6.7 Qualitative feedback. We also collected subjective feedback from the participants in their exit interviews. Some of the notable observations that were common across multiple participants are briefly described next.

Tiring to explore data items with a screen reader. Almost all (14) participants stated that they found it tiring and frustrating to interact with web data records. The reasons provided by these participants to explain this fatigue included "too much listening", "don't know where to go for information", and "need to remember a lot". To Quote P4: " There is simply too much stuff on these websites and they are scattered all over the place. It is frustrating to go here and there looking for stuff." Five participants also expressed that they would like to obtain "all information in one place".

Selection typically made after exploring only the first few items. A majority (11) of the participants stated that they typically peruse only a few initial items in the available items list before deciding to purchase one of the items. The underlying reasons specified by these participants were mainly "lack of time" and "fatigue". Four of these 11 participants further mentioned that they often missed out on "good bargains", since some of the preferred items were buried deep in the list, and they simply did not have the time or patience to explore the items list up to that depth. To



(a) SUS (b) NASA-TLX

Figure 5: Perceived usability (SUS) and task workload (NASA-TLX) for all three study conditions.

counter this issue, these 4 participants also stated they often relied on filters in the 'Query-Results' page to *bring* desirable items to the top of the list, however they mentioned that even with this strategy, they often missed out on great deals.

Multiple revisits to item summaries and 'Details' pages due to content overload. Three-fourths (12) of the participants mentioned that they often forgot some key information about items they had previously visited, so they had to revisit these item summaries and sometimes even their 'Details' pages to refresh their memories. These participants attributed this forgetfulness to the heavy content load on e-commerce websites where they had to listen to reams of content while perusing data items, and also remember a lot of details about multiple items to make comparisons. Half of these 12 participants expressed that AutoDesc would help them counter this issue, by enabling them to quickly access the desired details without having to revisit the 'Details' pages.

AutoDesc helps quickly filter out undesired items. Over half (9) of the participants mentioned that AutoDesc served as a " filter" that enabled them to quickly weed out items that didn't match their preferences from final consideration. These participants further explained that the instantly available descriptions helped them in deciding whether it was worth exploring the item more (e.g., read the customer reviews), and therefore spend the extra time and effort only if necessary.

Instant access to other item information is also desirable. A few (4) participants expressed a desire for AutoDesc to include support for instantly accessing other item-related information such as user reviews, multiple item comparisons, shipping details, similar item recommendations, etc. However, 2 of these participants further mentioned that only summaries of other item-related information would be beneficial given the risk of content overload (e.g., customer reviews can be lengthy with thousands of reviews).

5 DISCUSSION

The user study findings clearly demonstrate the effectiveness of AutoDesc in substantially improving the perusal efficiency and usability for blind screen reader users while interacting with web data items. However, the study also illuminated some of the shortcomings of our approach and therefore exposed some of the avenues for future research. A few important ones are discussed next.

Limitations. One of the limitations of our approach is that in our evaluation study, we selected websites where both SaIL and AutoDesc algorithms were able to accurately identify the item description in the 'Details' webpages. Although this experimental setup helped us avoid confounding effect of algorithm accuracy on performance values, it prevented us from understanding how algorithmic errors would have impacted the interaction experience of blind participants and how the participants would have coped with such inaccuracies. Understanding how AutoDesc's extraction algorithm accuracy affects the usability of data item interaction is the scope of our future work. Another limitation is that AutoDesc only pre-fetches item descriptions. As mentioned by some participants in the exit interviews, users also preferred instant access to other item-related information such as reviews, similar item suggestions, multiple item comparisons, etc., if such information was present in the website. We therefore plan to expand the repertoire of extraction algorithms to include support for these additional item-related details.

Also, the training and testing examples for our extraction algorithm only included webpages in English, and therefore it is unclear how the algorithm will fare on webpages in other languages. Another limitation of our work is that the current AutoDesc only functions within the Chrome web browser. Although Chrome is currently the most popular browser within the blind user community, there are still significant number of blind people using other browsers such as Firefox and Internet Explorer [62]. In future, we will extend AutoDesc support for other browsers as well. Lastly, our

AutoDesc system is presently available only for the desktop/laptop environment; it currently does not support efficient perusal of web data records on mobile devices such as smartphones. Given the ubiquity of smartphones and the increasing reliance on smartphone web browsing for conducting e-commerce transactions, support for efficient non-visual interaction with web data items is vital to ensure equality of access for people of all abilities. Recognizing this emerging need, we will explore porting AutoDesc to smartphone browsers as part of future work.

Instant access to all item-related information. Providing instant access to all item-related information such as reviews, product comparisons, and shipping details, as requested by some of the participants in the study, involves the following challenges. First, unlike descriptions, user reviews and product comparisons can be extremely long, and can therefore cause content overload in the 'Query-Results' page. Second, new extraction algorithms have to be developed for each of the item-related information (e.g., reviews, comparisons), as one algorithm is unlikely to work for different types of item-related information due to differences in their rendering and markup. To address the content overload issue, we will explore summarization techniques [24], so that only short snippets of information are provided to the users in the 'Query-Results' page. The user will then be able instantly access these snippets and decide whether it is worth accessing the full information. For the second challenge, we will explore both vision-based Mask R-CNN models and hybrid (vision + HTML markup) transformer models [66] to determine the best model for accurately identifying different types of item-related information.

Automatically filtering data items. Given that blind users typically explore only a few initial data items before making a decision, automatic approaches that can predict and filter out undesirable items in real-time as the users peruse the list of items, can potentially assist the users in finding the item that best match their preferences. For example, if a user is ignoring shirt items from a brand 'B' present at the top of the list, then removing all other shirts belonging to that brand from the list will push the shirt items from other brands (which may be more preferred by the user) higher up in the list. To build such a prediction and filtering model, we plan to explore contextual embedding [53] and similarity analysis methods [27], that will help us represent and compare items with each other.

Societal impact. In addition to web accessibility, usability of web interaction is also equally important for ensuring equal access to digital content for people with severe visual disabilities. However, most websites are primarily tailored for sighted users, so blind users typically need to put in significantly more time and effort to do the same web activities that sighted users can accomplish in a matter of few minutes [7]. It is this usability divide that prevents blind users for enjoying the benefits of web to the same extent as their sighted peers. In this paper, we seek to bridge this divide for an important web activity – interaction with web data items. By facilitating more efficient and usable interaction with the contents of web data items, we envision that blind users too will be able to explore more data items in short duration with lesser effort, and score better *deals* like their sighted peers.

6 CONCLUSION

Web data items are essential components of most modern websites and interaction with them is inevitable during web browsing. However, the present distributed organization and visualization of content in data items across multiple webpages mostly favor sighted users; blind users on the other hand have to endure a tedious and frustrating process of navigating back-and-forth between multiple webpages, and also listening to reams of irrelevant content while perusing web data items. As a first step towards bridging this usability divide, in this paper, we presented AutoDesc, a novel browser extension that automatically extracts additional item descriptions from the 'Details' pages of items, and then proactively injects the extracted information into the corresponding item summaries on the 'Query-Results' page, thereby reducing the need for the blind user to explicitly visit the 'Details' pages. Evaluation of AutoDesc in the user study with 16 blind participants showed that AutoDesc significantly reduced the average time and shortcuts spent on exploring an item in the *query-results* list, when compared with a state-of-the-art solution and also the participants' preferred screen readers. The study also revealed promising future research directions that included efficient and convenient access to user reviews and item comparisons.

ACKNOWLEDGMENTS

This work was supported by National Institutes of Health (NIH) Award R01EY030085.

REFERENCES

[1] Waleed Abdulla. 2017. Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. https://github.com/matterport/Mask_RCNN.
[2] Julian Alarte, David Insa, and Josep Silva. 2017. Webpage menu detection based on DOM. In *International Conference on Current Trends in Theory and Practice of Informatics*. Springer, 411–422.
[3] Manuel Álvarez, Alberto Pan, Juan Raposo, Fernando Bellas, and Fidel Cacheda. 2010. Finding and extracting data records from web pages. *Journal of Signal Processing Systems* 59, 1 (2010), 123–137.
[4] Apple Inc. 2022. Accessibility - Vision - Apple. https://www.apple.com/accessibility/vision/.
[5] 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.
[6] 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.
[7] 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.
[8] 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.
[9] Rakesh Babu. 2013. Understanding challenges in non-visual interaction with travel sites: An exploratory field study with blind users. *First Monday* (2013).
[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, Anna C Cavender, Jeremy T Brudvik, Jacob O Wobbrock, and Richard E Ladner. 2007. WebinSitu: a comparative analysis of blind and sighted browsing behavior. In *Proceedings of the 9th International ACM SIGACCESS Conference on Computers and Accessibility*. 51–58.
[12] 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.
[13] 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.

[14] 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)*. Association for Computing Machinery, New York, NY, USA, Article 13, 10 pages. https://doi.org/10.1145/1805986.1806005

[15] Roopa Bose and Helmut Jürgensen. 2014. Accessibility of E-Commerce Websites for Vision-Impaired Persons. In *ICCHP*.

[16] 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

[17] John Brooke. 1996. Sus: a "quick and dirty'usability. *Usability evaluation in industry* 189, 3 (1996).

[18] 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.

[19] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. 2003. Vips: a vision-based page segmentation algorithm. (2003).

[20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[21] 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.

[22] 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.

[23] Freedom Scientific. 2022. JAWS ® – Freedom Scientific. https://www.freedomscientific.com/products/software/jaws/.

[24] Mahak Gambhir and Vishal Gupta. 2017. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review* 47, 1 (2017), 1–66.

[25] Boni García, Mario Munoz-Organero, Carlos Alario-Hoyos, and Carlos Delgado Kloos. 2021. Automated driver management for selenium WebDriver. *Empirical Software Engineering* 26, 5 (2021), 1–51.

[26] 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*. 1–12.

[27] Wael H Gomaa, Aly A Fahmy, et al. 2013. A survey of text similarity approaches. *international journal of Computer Applications* 68, 13 (2013), 13–18.

[28] Simon Harper and Yeliz Yesilada. 2007. Web authoring for accessibility (WAfA). *Journal of Web Semantics* 5, 3 (2007), 175–179.

[29] Sandra G Hart and Lowell E Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in psychology*. Vol. 52. Elsevier, 139–183.

[30] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*. 2961–2969.

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[32] Megan L Hilts. 2008. *Internet dependency, motivations for internet use and their effect on work productivity: The 21 st century addiction*. Rochester Institute of Technology.

[33] Ravi Ilango. 2019. Using NLP (BERT) to improve OCR accuracy. https://medium.com/doma/using-nlp-bert-to-improve-ocr-accuracy-385c98ae174c

[34] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810* (2017).

[35] Eduardo Sany Laber, Críston Pereira de Souza, Iam Vita Jabour, Evelin Carvalho Freire de Amorim, Eduardo Teixeira Cardoso, Raúl Pierre Rentería, Lúcio Cunha Tinoco, and Caio Dias Valentim. 2009. A fast and simple method for extracting relevant content from news webpages. In *Proceedings of the 18th ACM conference on Information and knowledge management*. 1685–1688.

[36] 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.

[37] 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)* (2022).

[38] Hae-Na Lee, Sami Uddin, and Vikas Ashok. 2020. iTOC: Enabling Efficient Non-Visual Interaction with Long Web Documents. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 3799–3806.

[39] 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*. 1–12.

[40] Barbara Leporini and Fabio Paternò. 2004. Increasing usability when interacting through screen readers. *Universal access in the information society* 3, 1 (2004), 57–70.

[41] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. 2017. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2117–2125.

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

[43] 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.

[44] Thi Tuyet Hai Nguyen, Adam Jatowt, Nhu-Van Nguyen, Mickael Coustaty, and Antoine Doucet. 2020. Neural machine translation with BERT for post-OCR error detection and correction. In *Proceedings of the ACM/IEEE joint conference on digital libraries in 2020*. 333–336.

[45] Oliver Nina, Bryan Morse, and William Barrett. 2011. A recursive Otsu thresholding method for scanned document binarization. In *2011 IEEE Workshop on Applications of Computer Vision (WACV)*. IEEE, 307–314.

[46] NV Access. 2022. NV Access. https://www.nvaccess.org/.

[47] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[48] Jyotika Prasad and Andreas Paepcke. 2008. Coreex: content extraction from online news articles. In *Proceedings of the 17th ACM conference on Information and knowledge management*. 1391–1392.

[49] Elisa Prati, Simone Pozzi, Fabio Grandi, and Margherita Peruzzini. 2021. E-commerce Usability Guidelines for Visually Impaired Users. In *International Conference on Human-Computer Interaction*. Springer, 280–293.

[50] 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.

[51] IV Ramakrishnan, Vikas Ashok, and Syed Masum Billah. 2017. Non-visual web browsing: Beyond web accessibility. In *International Conference on Universal Access in Human-Computer Interaction*. Springer, 322–334.

[52] Kunal Relan. 2019. Building REST APIs with Flask. *Building REST APIs with Flask* (2019).

[53] Subendhu Rongali, Abhyuday Jagannatha, Bhanu Pratap Singh Rawat, and Hong Yu. 2020. Improved pretraining for domain-specific contextual embedding models. (2020).

[54] Brijendra Singh and Hemant Kumar Singh. 2010. Web data mining research: a survey. In *2010 IEEE International Conference on Computational Intelligence and Computing Research*. IEEE, 1–10.

[55] Ray Smith. 2007. An overview of the Tesseract OCR engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, Vol. 2. IEEE, 629–633.

[56] Tony Stockman and Oussama Metatla. 2008. The influence of screen-readers on web cognition. In *Proceeding of Accessible design in the digital world conference (ADDW 2008), York, UK*.

[57] The GIMP Development Team. 1998. GNU Image Manipulation Program. https://www.gimp.org

[58] Esko Ukkonen. 1995. On-line construction of suffix trees. *Algorithmica* 14, 3 (1995), 249–260.

[59] W3. 1997. WAI. W3C Web Accessibility Initiative. https://www.w3.org/WAI/.

[60] Ruolin Wang, Zixuan Chen, Mingrui Ray Zhang, Zhaoheng Li, Zhixiu Liu, Zihan Dang, Chun Yu, and Xiang'Anthony' Chen. 2021. Revamp: Enhancing Accessible Information Seeking Experience of Online Shopping for Blind or Low Vision Users. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–14.

[61] WebAIM. 2018. *Survey of Users with Low Vision #2 Results*. https://webaim.org/projects/lowvisionsurvey2/

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

[63] 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.

[64] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* (2016).

[65] Canhui Xu, Cao Shi, Hengyue Bi, Chuanqi Liu, Yongfeng Yuan, Haoyan Guo, and Yinong Chen. 2021. A Page Object Detection Method Based on Mask R-CNN. *IEEE Access* 9 (2021), 143448–143457.

[66] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020. Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1192–1200.

[67] Yeliz Yesilada, Simon Harper, Carole Goble, and Robert Stevens. 2004. Screen readers cannot see. In *International Conference on Web Engineering*. Springer, 445–458.
[68] Yeliz Yesilada, Robert Stevens, Simon Harper, and Carole Goble. 2007. Evaluating DANTE: Semantic transcoding for visually disabled users. *ACM Transactions on Computer-Human Interaction (TOCHI)* 14, 3 (2007), 14–es.
[69] Yanhong Zhai and Bing Liu. 2005. Web data extraction based on partial tree alignment. In *Proceedings of the 14th international conference on World Wide Web*. 76–85.
[70] Zhuoyao Zhong, Lei Sun, and Qiang Huo. 2019. An anchor-free region proposal network for Faster R-CNN-based text detection approaches. *International Journal on Document Analysis and Recognition (IJDAR)* 22, 3 (2019), 315–327.
[71] Jun Zhu, Zaiqing Nie, Ji-Rong Wen, Bo Zhang, and Wei-Ying Ma. 2006. Simultaneous record detection and attribute labeling in web data extraction. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 494–503.

A USER STUDY TASK EXAMPLE

Figure 6 illustrates a study task that required the blind participants to peruse a list of *television* data items on the Macy's website and then select one of the items best matching their preferences. Based on the predetermined counterbalanced order, the participants did this task in one of the three study conditions: Screen Reader, SaIL, and AutoDesc. Figure 6 specifically presents the AutoDesc condition, where the participants could leverage the additional "Get Details" buttons to quickly access descriptions of corresponding items. These buttons were not available in other study conditions, and the participants therefore had to access the descriptions by navigating to separate 'Details' pages.

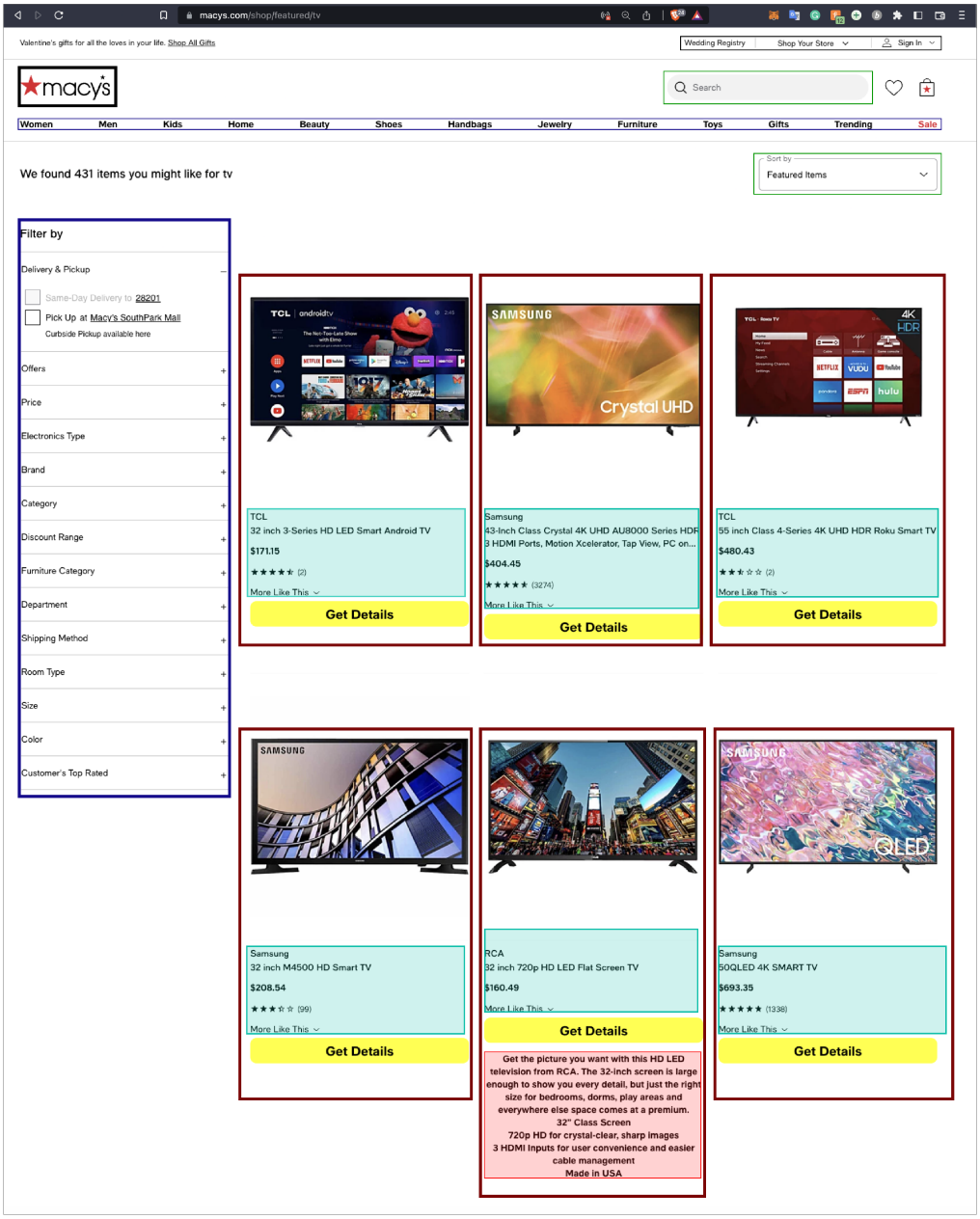


Figure 6: Illustration of one of the tasks blind participants did in the user study.