SaIL: Saliency-Driven Injection of ARIA Landmarks

Ali Selman Aydin*
Stony Brook University
aaydin@cs.stonybrook.edu

Shirin Feiz*
Stony Brook University
sfeizdisfani@cs.stonybrook.edu

Vikas Ashok
Old Dominion University
vganjigu@odu.edu

IV Ramakrishnan
Stony Brook University
ram@cs.stonybrook.edu

ABSTRACT

Navigating webpages with screen readers is a challenge even with recent improvements in screen reader technologies and the increased adoption of web standards for accessibility, namely ARIA. ARIA landmarks, an important aspect of ARIA, lets screen reader users access different sections of the webpage quickly, by enabling them to skip over blocks of irrelevant or redundant content. However, these landmarks are sporadically and inconsistently used by web developers, and in many cases, even absent in numerous web pages. Therefore, we propose SaIL, a scalable approach that automatically detects the important sections of a web page, and then injects ARIA landmarks into the corresponding HTML markup to facilitate quick access to these sections. The central concept underlying SaIL is visual saliency, which is determined using a state-of-the-art deep learning model that was trained on gaze-tracking data collected from sighted users in the context of web browsing. We present the findings of a pilot study that demonstrated the potential of SaIL in reducing both the time and effort spent in navigating webpages with screen readers.

CCS CONCEPTS

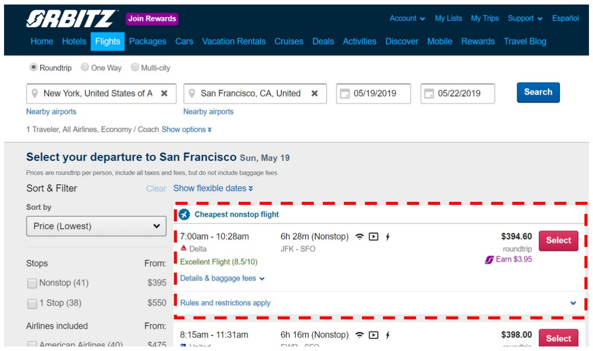
• **Human-centered computing → Human computer interaction (HCI); Accessibility systems and tools**.

KEYWORDS

WAI-ARIA, screen reader, landmarks, web accessibility.

ACM Reference Format:
Ali Selman Aydin, Shirin Feiz, Vikas Ashok, and IV Ramakrishnan. 2020. SaIL: Saliency-Driven Injection of ARIA Landmarks. In *25th International Conference on Intelligent User Interfaces (IUI '20), March 17–20, 2020, Cagliari, Italy*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3377325.3377540

*Both authors contributed equally to this research.



Roundtrip One Way Multi-city
New York, United States of A
San Francisco, CA, United
05/19/2019
05/22/2019
1 Traveler, All Airlines, Economy / Coach
Select your departure to San Francisco Sun, May 19
Prices are roundtrip per person, include all taxes and fees, but do not include baggage fees.
Sort & Filter
Sort by
Price (Lowest)
Stops From:
Nonstop (41) $395
1 Stop (38) $550
Airlines included From:
Cheapest nonstop flight
7:00am - 10:28am 6h 28m (Nonstop) $394.60
Delta JFK - SFO roundtrip
Excellent Flight (8.5/10) Earn $3.95
Details & baggage fees
Rules and restrictions apply
8:15am - 11:31am 6h 16m (Nonstop) $398.00

Figure 1: A typical flight reservation website from: www.orbitz.com. Reaching the first flight result (highlighted in red) requires around 70 'p' or 9 'h' shortcuts in JAWS.

1 INTRODUCTION

People who are blind rely on screen readers (e.g., JAWS [9], NVDA [13], VoiceOver [4]) for web browsing. A screen reader enables users to navigate a webpage in a sequential "press-and-listen" fashion, and additionally use an assortment of shortcut keys to speed up the navigation process, by skipping blocks of irrelevant content.

Although shortcuts help navigating the content faster, there are several problems associated with the use of shortcuts. First, the user needs to train and remember a wide range of shortcuts, and devise strategies to use them effectively. Second, even with shortcuts, the users frequently need to sequentially hop through a long list of web elements to reach the desired element. An example that illustrates these challenges is provided in Figure 1. In order to reach the first flight result in this typical flight reservation website, the user needs to press the paragraph shortcut ('p' in JAWS) 70 times. Even with prior knowledge about the website and custom navigation strategies, reaching the desired location is likely to take considerable number of key-presses, as the webpages are usually cluttered with content, and the user needs to listen and parse the audio feedback from the screen reader after each key press to determine the relevance of this content.

Web browsing efficiency can be improved by specifying ARIA (Accessible Rich Internet Applications) landmarks [22] on different sections or regions of interest in a webpage, where each region (e.g., search-result item) is potentially made up of several semantically related web elements. These landmarks serve as reference points that could be conveniently and quickly accessed using keyboard shortcuts [21], thereby enabling screen reader users to skip blocks of web elements while navigating to these marked sections or regions of interest. The extant screen readers already provide shortcuts to navigate such landmarks (e.g., R key in JAWS, D key in NVDA, VO + Alt + Arrow keys in VoiceOver). Therefore, if ARIA landmarks are strategically placed in a way that accurately reflects the visual/functional components of a webpage, it is possible to improve navigation efficiency for web screen reading.

Although ARIA landmarks can be implemented without significant effort by web developers, the current adoption of ARIA landmarks is not widespread. For instance, we found out that 54% of top-50 website homepages (Alexa ranking [1]) and 50.4% of Moz-500 website homepages [12] contained no ARIA landmarks, while the average number of landmarks for Alexa top-50 website homepages is 2.0 (median=0, max=2 min=0) and the average number of landmarks for Moz-500 website homepages is 1.868(median=0, max=29, min=0). This paucity of landmarks in web pages is mainly due to the fact that they are created at the discretion of the web designer, and many of them simply choose not to create them, and even those who do, create very few of them.

Although ARIA landmarks are not adequately utilized across the web, their creation does not necessarily depend on developers. It is possible to add ARIA landmarks to webpage regions by creating "role attributes", assigning values to them, and then designating these regions as ARIA landmarks. It is possible to dynamically inject such landmarks at the client side, once the webpage has been rendered in the browser. But doing this manually is impractical. This is because both the positions and the roles of ARIA landmarks need to be determined by inspecting the visual and functional properties of the webpage regions. Considering the ever-changing nature of the webpages, this task is too laborious to be manually performed. Specifically, manual annotation or crowd-sourcing approaches are not largely scalable, and this is likely to render these approaches inadequate in addressing the rate of change on the Web.

Considering inadequate adoption of ARIA landmarks despite their low-effort implementation, and scalability challenges associated with manual annotation of ARIA landmarks, we propose SaIL (Saliency-Driven Injection of Landmarks), a system that can automatically analyze the visual and structural properties of a webpage, and then dynamically inject ARIA landmarks into the webpage markup accordingly. SaIL is rooted in the idea of leveraging visual saliency in the context of webpages to locate important regions based on their visual and functional properties. SaIL uses a state-of-the-art deep learning model trained on gaze data from sighted users, to identify salient or important regions in the page, and inject ARIA landmarks that users can exploit to quickly navigate the page. Since it is entirely automated, SaIL system is highly scalable, allowing for injecting ARIA landmarks on any webpage without manual effort.

We summarize our contributions as follows:

- We present SAIL, an automated and scalable system for injecting ARIA landmarks, rooted in the idea of identifying salient regions in a web page, using a deep learning model trained on gaze data.
- We report the findings of a preliminary user study with 6 screen reader users demonstrating the potential of SaIL in improving web screen reading efficiency and user experience for blind screen reader users.

2 RELATED WORK

The purpose of ARIA markups is to create a representation of the structure of the webpages for screen readers so that the user can navigate the webpages effectively [22]. The scope and extent of web accessibility that is provided by the ARIA markups has been a topic of research [8]. Injection of such markups to the static regions in the web page are typically implemented by developers. However, injection of ARIA markups to the live regions of web pages have proven to be challenging. For example, in the case of live regions, the Document Object Model (DOM) gets dynamically updated without having the proper ARIA markups in these regions.

A body of work has focused on making such dynamically changing web regions accessible. For example, the evaluation by Thiessen et. al. [17] reveals the extent of accessibility that is provided by the ARIA 'live' region, which is a static tag indicating the parts of DOM that will be updated dynamically. These regions are identified and marked by the developer in the HTML source. The 'live' ARIA markup is utilized for the analysis of live widgets to provide accessibility solutions for screen reader users [7, 18, 19]. A recent work [2] on live regions uses machine learning techniques to identify the drop-down menu widgets on the screen to make them accessible by guiding the screen reader through these menus.

Another solution provided as part of ARIA markups is the 'landmark' feature. Landmarks are defined based on the already-existing 'role' attribute of HTML tags. According to [22], the purpose of this technique is to provide programmatic access to different regions of a web page. Using a landmark shortcut, the screen reader user can jump between such regions. In the work presented in [5], the web accessibility is improved by utilizing ARIA landmarks to change the order of the content visited by the screen reader.

Utilizing landmarks can potentially save the user from tabbing/navigating through web content especially in case of a familiar webpage. Nonetheless, there are several challenges associated with using the landmarks. Firstly, these landmarks are based on the definition of role attributes that is provided by the web developer. Since the role attribute is not primarily defined for accessibility purpose, developers may not follow accessibility guidelines when defining these roles. However, these landmarks are sporadically and inconsistently used by web developers, and in many cases, even absent in numerous web pages. For instance, the work in [20] revealed that among studied web accessibility features of U.S institutional websites, the ARIA landmarks are least likely to be present. This finding is in accordance with our own experiments on ARIA landmark statistics provided in the introduction section. VoiceOver screen reader offers 'auto web spots' feature [3]. Auto web spots are determined based on the structural and visual analysis of the webpage. However, our inspection of the top 100 websites, ranked by Moz-500 [7] showed that almost none of them had auto web spots feature available.

In sum, landmarks are sporadically and inconsistently used by web developers. Our SaIL system is a scalable approach that automatically detects the important sections of a web page, and then injects ARIA landmarks into the corresponding HTML markup. Utilizing visual saliency is one of the main components of the SAIL system. There has been voluminous work on visual saliency both in the context of natural images and webpages [6]. For natural images,

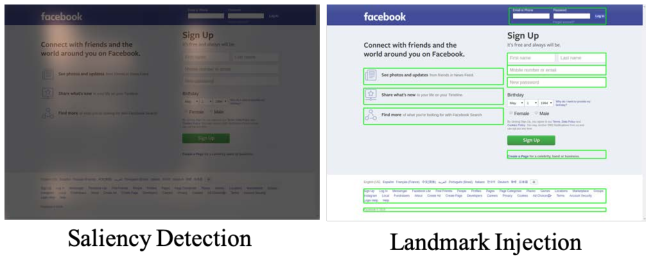


Figure 2: SaIL system overview. (LEFT): the visual saliency information in the form of a heatmap - bright regions correspond to salient regions; (RIGHT): injected landmarks on web segments corresponding to salient regions. Images: www.facebook.com.

some of the deep-learning based approaches include SALICON [11], SalNet [15] and SalGAN [14]. For predicting webpage saliency, authors of [16] propose a multiple kernel learning based saliency model. A more recent webpage saliency model that considers various frequently encountered tasks has been proposed in [23]. In our work, we used SalGAN saliency model as the basis and we adapted it using the dataset proposed in [16].

3 APPROACH

SaIL dynamically injects the "role" landmark attributes into the HTML source to provide easy access to different webpage sections. Towards that, SaIL (i) detects regions of interest (ROIs) using a combination of visual cues and HTML features, (ii) finds the visible web elements that correspond to detected ROIs, and finally, (iii) injects an ARIA landmark role to these web elements.

3.1 Saliency Network

The deep-learning saliency-network model used in SaIL is based on SalGAN [14], which is a generative adversarial network (GAN) [10] used for identifying salient regions in images. Similar to other GANs in the literature, SalGAN consists of two sub-networks: a generator network for predicting saliency maps for given images, and a discriminator network designed to classify these saliency maps as either valid or invalid. Joint training of these two networks prompts the generator network to produce more realistic and accurate saliency maps, while the discriminator network is trained to discriminate between the salient and non-salient regions more accurately. Once the optimization of the joint objective of the two sub-networks is completed, the generator sub-network could then be used to predict saliency maps for any given image, which in our case is a snapshot of a webpage. Figure 2 provides an example of a saliency map generated by our model for the Facebook homepage.

3.2 Dataset and Training

In contrast to the original SalGAN network that was trained on a natural image dataset, our GAN model was trained on a webpage saliency dataset provided by Shen et al. [16]. This dataset contains screenshots of 149 webpages, and the corresponding saliency maps inferred from the gaze tracking data collected by observing 11

ID	Age	Gender	Experience	Screen Readers
P1	35	F	Intermediate	Jaws, Voiceover, NVDA
P2	46	F	Beginner	Jaws
P3	31	F	Intermediate	Jaws, Voice Over
P4	58	M	Intermediate	NVDA
P5	28	F	Expert	Jaws, Voice Over, NVDA
P6	44	M	Expert	Jaws

Table 1: Participant demographic information

sighted users as they browsed these web pages. To train our GAN model on this dataset, we adopted the procedure of transfer learning, in which instead of cold-starting the training process, a neural network was fine-tuned from previously learned information.

3.3 Saliency Prediction and Post-processing

The trained GAN model infers saliency maps of webpages using their corresponding screenshots of size 256x192. A saliency map produced by the generator network is a grayscale image that has the same size as the original screenshot. SaIL then post-processes the saliency map to determine the most salient regions, i.e., filter out the ROIs from raw saliency data in the map.

The post-processing procedure is as follows: the salient regions are combined with the properties of the HTML tags to come up with candidate tags. To this end, the DOM tree elements are shortlisted using the tag type (*e.g.,* `<div>`, `<form>`) and visibility properties as some tags are more likely to be candidates for ARIA landmark injection based on domain knowledge. The shortlisted tags are then ranked based on their maximum saliency scores, and top k tags are tagged for landmark injection, where k is the adjustable number of desired ARIA landmarks. Our prototype implementation can take from seconds to minutes to complete this procedure, depending on the DOM tree size of the page, while an efficient production implementation can work significantly faster.

4 EVALUATION

4.1 Pilot Study

We conducted an IRB-approved pilot study with 6 fully-blind participants to evaluate SaIL. All participants had sufficient prior knowledge about screen readers. For demographic information of the participants, see Table 1. In this pilot study we asked each participant to access information from four websites, two with injected landmarks and two without injected landmarks. Websites with and without landmarks for each participant were chosen randomly. We selected the four websites to be: 'expedia.com' and 'tripadvisor.com' for flight searching tasks and 'usajobs.gov' and 'indeed.com' for job searching tasks (see Figure 3).

We started each session by asking about prior experiences with screen readers. All participants mentioned that they did not know about the landmark/auto web spot features of screen readers. Except for P4 who used screen readers about once a week, all other participants mentioned that they use screen readers on a daily basis. All participants were given time to familiarize with the use of landmarks on a test webpage for 10-15 minutes. The duration of each session was an hour.

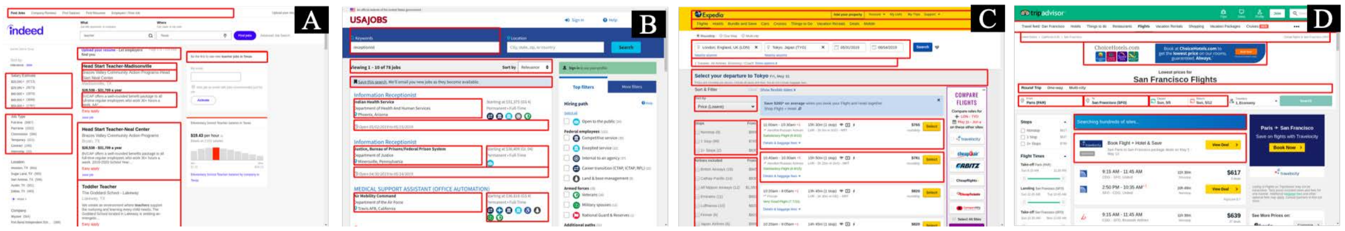


Figure 3: The results page in the four websites used in the user study. (A) indeed.com and (B) usajobs.gov are job search websites and (C) expedia.com, and (D) tripadvisor.com are flight reservation sites. The regions with landmarks injected by SaIL are enclosed in red boxes.

The order of injected/non-injected conditions were counterbalanced and in each case the participant viewed one flight search and one job search website that were randomly selected. Figure 3 shows regions of web pages containing information the user was asked to find. The requested information was selected to be in a similar position on all of the four web pages. The JAWS screen reader was used in this study. For each website, the participant was given 10 minutes to find the requested information. We observed that without the landmarks there were three cases (P2: tripadvisor.com, P2: indeed.com, P4: usajobs.com) where the user could not finish the task in the given time. In the case of SAIL injected landmarks, there was one case where the task could not be completed (P2: expedia.com). Another observation in this study was the time each user spent on the website to find the requested information. Table 2 shows the time spent in seconds on each website for each user. Although the limited number of participants and inadequate randomization does not allow statistical significance tests, we observed relatively shorter times when users utilized the landmarks. The observed times can be interpreted as follows: each landmark lets the user skip over unrelated information; therefore, the user can access the information in less time.

5 DISCUSSION

Our study observations indicate that utilizing landmarks can speed up the acquisition of information from web pages. We also noticed that users would go over the links or the headings sequentially or resort to tabbing through the entire page until they reach the requested information. However, in the case of the landmark injected websites, they would go over the landmarks first to get an overview of the page structure, then select the most relevant landmark and explore the vicinity of this landmark only. Users found the landmarks useful for the purpose of web overview and for not having to worry about the type of elements (e.g., heading or link). This could be helpful particularly in cases where the types of targets in a web page are unknown prior to browsing.

Our observations during the study revealed that users utilize the landmarks to escape from 'dead ends'. Depending on their experience, the users occasionally got stuck in a region different than the target region since they were not aware of the shortcuts that can move the focus to the target region. In some of these cases, they resorted to jumping to the beginning of the page, and making repeated attempts at reaching the target region. However, using landmarks the user can jump to the beginning of a region and does not have to go through all the previous elements.

Participants mentioned that they would like the landmark to be placed in the beginning of the semantic web structure, which would give them assurance on where to look after landing on a landmark.

Another important point made by participants is the requirement of landmarks to be comprehensive and consistent. For example, in case of three similar lists, the user would like each list to have a landmark or one landmark encompassing all three lists. However, if only the first list and the third list have landmarks, it is hard for the user to identify the second list. As a result, users believe that having assurance that the landmarks cover the entire page consistently will enable them to jump over regions and make better use of the landmarks. On the other hand, in case of incomplete landmarks, the user may need to spend extra effort to make sure that they did not miss the content between the landmarks.

ID	C1		C2	
P1	95(A)	250(D)	68(B)	38(C)
P2	🟥 (A)	🟥 (D)	112(B)	🟥 (C)
P3	556(A)	298(D)	268(B)	242(C)
P4	🟥 (B)	283 (C)	135 (A)	117(D)
P5	85(A)	229(D)	66(B)	74(C)
P6	154(B)	48(C)	118(A)	128(D)

Table 2: Task completion time matrix for participants (in seconds). Letters in parentheses indicate the visited website, as defined in Figure 3. Red cells denote cases where task was not completed within 10 minutes. C1: Without SaIL, C2: With SaIL.

6 CONCLUSION

In this paper we proposed SaIL, an automated system based on visual saliency for injecting ARIA landmarks. It was motivated by the limited adoption of ARIA landmarks across the Web. The results of our pilot study indicate that SaIL has the potential to decrease the time spent by screen-reader users in navigating webpage content, by enabling them to leverage the injected landmarks for quickly accessing the regions of interest.

ACKNOWLEDGMENTS

This work was supported by NSF Award: 1806076, NEI/NIH Award: R01EY026621, and NIDILRR Award: 90IF0117-01-00.

REFERENCES

[1] Amazon. [n.d.]. Alexa Ranking. https://www.alexa.com/topsites.

[2] Humberto Lidio Antonelli, Rodrigo Augusto Igawa, Renata Pontin De Mattos Fortes, Eduardo Henrique Rizo, and Willian Massami Watanabe. 2018. Dropdown menu widget identification using HTML structure changes classification. *ACM Transactions on Accessible Computing (TACCESS)* 11, 2 (2018), 10.

[3] Apple. [n.d.]. Chapter 6. Browsing the Internet. https://www.apple.com/voiceover/info/guide/_1134.html.

[4] Apple. [n.d.]. VoiceOver. https://www.apple.com/accessibility/mac/vision/.

[5] Mario Batusic and Daniela Steiner. 2010. Improving the accessibility of Fabasoft Folio by means of WAI-ARIA. In *International Conference on Computers for Handicapped Persons*. Springer, 476–483.

[6] Ali Borji. 2018. Saliency prediction in the deep learning era: An empirical investigation. *arXiv preprint arXiv:1810.03716* (2018).

[7] 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*. ACM, 14.

[8] Iyad Abu Doush, Faisal Alkhateeb, Eslam Al Maghayreh, and Mohammed Azmi Al-Betar. 2013. The design of RIA accessibility evaluation tool. *Advances in Engineering Software* 57 (2013), 1–7.

[9] FreedomScientific. [n.d.]. Jaws Screen Reader. https://www.freedomscientific.com/products/software/jaws.

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[11] Xun Huang, Chengyao Shen, Xavier Boix, and Qi Zhao. 2015. Salicon: Reducing the semantic gap in saliency prediction by adapting deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*. 262–270.

[12] Moz. [n.d.]. The Top 500 Most Popular Websites - Moz. https://moz.com/top500.

[13] NVAccess. [n.d.]. https://www.nvaccess.org/.

[14] Junting Pan, Cristian Canton Ferrer, Kevin McGuinness, Noel E O'Connor, Jordi Torres, Elisa Sayrol, and Xavier Giro-i Nieto. 2017. Salgan: Visual saliency prediction with generative adversarial networks. *arXiv preprint arXiv:1701.01081* (2017).

[15] Junting Pan, Elisa Sayrol, Xavier Giro-i Nieto, Kevin McGuinness, and Noel E O'Connor. 2016. Shallow and deep convolutional networks for saliency prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 598–606.

[16] Chengyao Shen and Qi Zhao. 2014. Webpage saliency. In *European conference on computer vision*. Springer, 33–46.

[17] Peter Thiessen. 2011. WAI-ARIA live regions and HTML5. In *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility*. ACM, 27.

[18] Peter Thiessen and Stephen Hockema. 2010. WAI-ARIA live regions: eBuddy IM as a case example. In *Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (W4A)*. ACM, 33.

[19] Peter Thiessen and Erin Russell. 2009. Wai-aria live regions and channels: Reefchat as a case example. *Disability and Rehabilitation: Assistive Technology* 4, 4 (2009), 276–287.

[20] Terrill Thompson, Dan Comden, Scott Ferguson, Sheryl Burgstahler, and Elizabeth J Moore. 2013. Seeking predictors of web accessibility in US higher education institutions. *Information Technology and Disabilities* 13, 1 (2013), 18.

[21] W3C. 2019. WAI-ARIA Authoring Practices 1.1. https://www.w3.org/TR/wai-aria-practices/#aria_landmark.

[22] WAI-ARIA. 2017. W3C Accessible Rich Internet Applications. https://www.w3.org/TR/wai-aria/.

[23] Quanlong Zheng, Jianbo Jiao, Ying Cao, and Rynson WH Lau. 2018. Task-driven webpage saliency. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 287–302.