Breaking the Accessibility Barrier in Non-Visual Interaction with PDF Forms

UTKU UCKUN, Stony Brook University, USA
ALI SELMAN AYDIN, Stony Brook University, USA
VIKAS ASHOK, Old Dominion University, USA
IV RAMAKRISHNAN, Stony Brook University, USA

PDF forms are ubiquitous. Businesses big and small, government agencies, health and educational institutions and many others have all embraced PDF forms. People use PDF forms for providing information to these entities. But people who are blind frequently find it very difficult to fill out PDF forms with screen readers, the standard assistive software that they use for interacting with computer applications. Firstly, many of the them are not even accessible as they are non-interactive and hence not editable on a computer. Secondly, even if they are interactive, it is not always easy to associate the correct labels with the form fields, either because the labels are not meaningful or the sequential reading order of the screen reader misses the visual cues that associate the correct labels with the fields. In this paper we present a solution to the accessibility problem of PDF forms. We leverage the fact that many people with visual impairments are familiar with web browsing and are proficient at filling out web forms. Thus, we create a web form layer over the PDF form via a high fidelity transformation process that attempts to preserve all the spatial relationships of the PDF elements including forms, their labels and the textual content. Blind people only interact with the web forms, and the filled out web form fields are transparently transferred to the corresponding fields in the PDF form. An optimization algorithm automatically adjusts the length and width of the PDF fields to accommodate arbitrary size field data. This ensures that the filled out PDF document does not have any truncated form-field values, and additionally, it is readable. A user study with fourteen users with visual impairments revealed that they were able to populate more form fields than the status quo and the self-reported user experience with the proposed interface was superior compared to the status quo.

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

Additional Key Words and Phrases: PDF form accessibility, screen-reader users, accessible interfaces

ACM Reference Format:
Utku Uckun, Ali Selman Aydin, Vikas Ashok, and IV Ramakrishnan. 2020. Breaking the Accessibility Barrier in Non-Visual Interaction with PDF Forms. *Proc. ACM Hum.-Comput. Interact.* 4, EICS, Article 80 (June 2020), 16 pages. https://doi.org/10.1145/3397868

1 INTRODUCTION

PDF (Portable Document Format) documents are ubiquitous in the computing world. Almost anyone who interacts with computers will invariably come across PDF documents. While most of these

Authors' addresses: Utku Uckun, uuckun@cs.stonybrook.edu, Stony Brook University, Stony Brook, NY, USA; Ali Selman Aydin, aaydin@cs.stonybrook.edu, Stony Brook University, Stony Brook, NY, USA; Vikas Ashok, vganjigu@cs.odu.edu, Old Dominion University, Norfolk, VA, USA; IV Ramakrishnan, ram@cs.stonybrook.edu, Stony Brook University, Stony Brook, NY, USA.

2573-0142/2020/6-ART80 $15.00
https://doi.org/10.1145/3397868

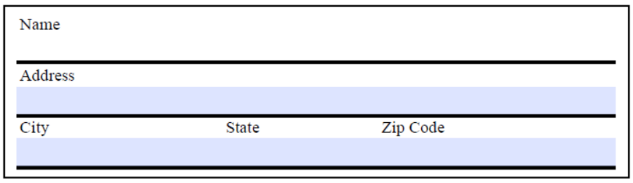


Fig. 1. The Name form field is not interactive; all others are interactive. (Modified from [12])

documents are created primarily for reading purposes, many kinds of fillable forms are also created in PDF. In fact, businesses big and small, government agencies, health and educational institutions and many others have all embraced PDF forms. People use PDF forms for providing information to these entities. However, these PDF forms are meant to be used by sighted people. Naturally, people who are blind find it challenging to fill out PDF forms.

People who are blind use screen readers (SRs) to interact with computers. SRs serially read out the text on the screen while ignoring all the graphics. The reasons blind persons (BPs) have difficulties with PDF form filling using SRs are manifold.

First, some of these PDF forms are images. In a review of over 856 PDF forms that we downloaded from the Web using Google search, we found out that nearly 1% were PDF images. Second, a majority of these forms are non-interactive, in the sense that the SR reads the textual elements in the forms but BPs are unable to key in any data in the form fields or select any of the options provided by check boxes/radio buttons in these forms (see Figure 1 - the Name field is not interactive while all the fields with "bluish" tinge are interactive). In our downloaded data set, 440 forms (i.e., 51.4%) were completely non-interactive, i.e., no form field was fillable. Third, even if the forms are interactive, the implicit labels of the form fields, which are read out when the SR's cursor is focused on these fields, are very often either non-existent or not meaningful such as the string "text". In Figure 2 the implicit labels for both the form fields are "text". So when the SR's cursor is focused on them it will readout "text", which is not helpful to the BP who is trying to get a sense of the kind of data required by the form fields. In a manual inspection of 103 of the remaining 407 interactive forms only around 44.6% had correct and meaningful implicit labels assigned to the form fields. Fourth, the top-to-bottom, left-to-right natural reading order of documents is often not adhered to by SRs. They tend to follow a reading order that is not consistent with the natural reading order and instead may read out the elements of the PDF document in seemingly arbitrary order. This is due to fact that SRs follow the order of words in the PDF's metadata which does not necessarily follow the correct reading order. This makes it difficult for BPs to associate the explicit labels that appear in the immediate locality of the form fields, notably, either in the front or on the top, with the form fields. In Figure 2 "Print name" and "Signature" are the explicit labels associated with the top and bottom form fields. But observe that in the SR's reading order of the form elements (shown as 1, 2, 3, 4) the SR first reads out "text" followed by "Print Name". Because the SR reads

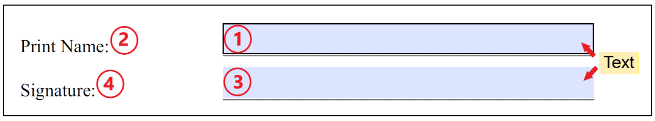
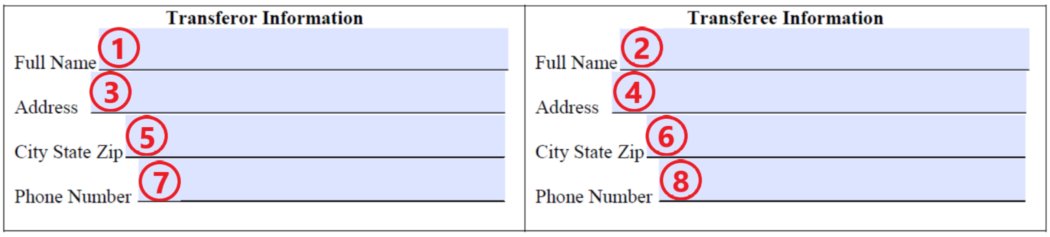


Fig. 2. Implicit labels of the two form fields are both "text". SR's reading order of the form elements (shown by numbers 1, 2, 3, 4) makes it difficult for BPs to associate the explicit labels "Print Name" and "Signature" to their corresponding form fields. (Modified from [11])



Transferor Information	Transferee Information
Full Name ①	Full Name ②
Address ③	Address ④
City State Zip ⑤	City State Zip ⑥
Phone Number ⑦	Phone Number ⑧

Fig. 3. Mixing the readout of the form elements of two groups - Transferor and Transferee. The numbers denote the SR's reading order of the form elements. (Modified from [32])

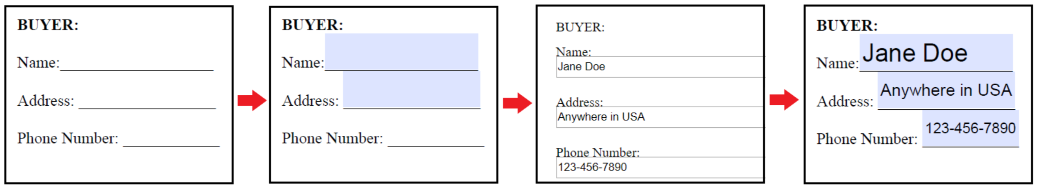
out in reverse order, it is not easy for a BP to associate "Print Name" as the label of the top form field. Fifth, even if the screen reading order is consistent with the natural reading order the readout often intersperses the text elements of one group of form elements with those of another group, which can confuse and disorient the BP. In Figure 3 it is natural and intuitive to readout all the form elements of the Transferor group before commencing the readout of the Transferee group. SRs cannot distinguish such groups and intersperse the elements of the two groups in its readout. In this figure the numbers denote the reading order. Last, because of length constraints some form fields cannot accommodate the complete text the BP wishes to key in without either truncating the text or shrinking the font size to such an extent as to make the text unreadable. Overall, all these problems make PDF form filling inaccessible for BPs. In fact, many BPs in our focus group mentioned that they shy away from PDF forms altogether and instead prefer web forms.

This paper proposes a novel approach to make PDF form filling accessible for BPs by addressing all of the aforementioned problems. Our solution leverages the fact that BPs are familiar with web browsing and are proficient at filling out web forms as noted in [9] as well in our own user study (see Section 4.2). Thus, we create a web form layer over the PDF form via a high fidelity transformation process that attempts to preserve all the spatial relationships of the PDF elements including forms, their labels and the textual content. The transformation also preserves the grouping of form fields in such a way that SRs complete the narration of all the text elements of a group before moving onto other groups. The transformation process also pairs up a form field's explicit and implicit labels in the transformed web form, thereby making the semantic meaning of a form field apparent to the BP, who now will know what data to fill in the form field.

BPs only interact with the web forms, and the filled out web form fields are transparently transferred to the corresponding fields in the PDF form. An optimization algorithm automatically adjusts the length and width of the PDF fields to accommodate arbitrary size field data. This ensures that the filled out PDF document does not have any truncated form-field values, and additionally, it is readable. Figure 4 illustrates transformation process on a non-interactive PDF form (the 1^{st} panel on the left). It is first transformed into a (semi)interactive PDF form (the 2^{nd} panel) and then into HTML (the 3^{rd} panel). Observe how all the spatial relationships amongst the form elements are preserved. The BP interacts with the HTML form and fills out all the form fields. The last step of the transformation populates the PDF form with the data filled in the web form (4^{th} panel).

Below are the main contributions of this paper:

- We present an analysis of PDF forms downloaded from the internet to demonstrate that a large portion of PDF forms are not accessible.
- We introduce and explain a transformation pipeline which converts an inaccessible PDF form to its Web form version to be filled by SR users, and fills the entered information to the PDF form.


BUYER:
Name:
Address:
Phone Number:

BUYER:
Name:
Address:
Phone Number:

BUYER:
Name:
Jane Doe
Address:
Anywhere in USA
Phone Number:
123-456-7890

BUYER:
Name: Jane Doe
Address: Anywhere in USA
Phone Number: 123-456-7890

Fig. 4. Illustration of the transformation steps: from non-interactive PDF form to a fully filled out PDF form. (Modified from [24])

- We demonstrate the effectiveness of our pipeline through both an offline assessment and a user study.

2 RELATED WORK

2.1 Lessons from Web Accessibility

There have been studies about PDF accessibility in the past, both in the context of web accessibility and as a topic on its own. One of these studies were carried out by Lazar et al[25]. Their survey investigates the sources of frustration for people who are blind while using Web. While the results in this paper represent the state of the Web in 2007, we detected common issues that impact the accessibility of PDF forms today. As this study indicates, top causes of frustration on the internet are complex layouts that are confusing the screen-reader's reading order and poorly designed unlabeled forms[25]. Another important finding is inaccessible PDF format being a cause of frustration. Even though there has been many attempts towards more accessible Web since these works have been published, we believe that these changes were not widely adopted in PDF forms.

2.2 Screen Readers

There are various software and hardware solutions that try to make computers more accessible to people with visual impairments. Braille display [38] [21] [37] is an example of a hardware solution while screen-reader software such as JAWS [20], NVDA [30] and VoiceOver[6] are examples of software solutions [9]. By nature software solutions are more commonly used and less expensive than hardware solutions therefore, screen reader is the preferred tool for most people with blindness. Among other tasks, screen readers have been the go-to tools for interacting with PDF forms for people with visual impairments, which is why our method focuses on screen-reader users.

2.3 PDF Accessibility

Certain characteristics of PDF format make it challenging to enable PDF accessibility for people with visual impairments. A major setback is the proprietary nature of the PDF format[35]. Open source documents are easier to work on while developing accessibility features. HTML is an example open source document type and there has been numerous attempts to making HTML documents more accessible to BPs[8] [7] [28]. Another challenge is due to frequent discrepancies between the presentation order of elements in a PDF document and the parsing order of the screen reader. Such cases are harder to cope with in PDF documents, since it's more difficult to alter the components' order in a PDF document compared to changing the component order in an HTML document.

Despite the prevalence and importance of PDF documents, the number of works investigating the non-visual accessibility of PDF documents has remained limited. One example is [35], which investigates not only non-visual accessibility of the PDF documents, but also other types of accessibility challenges with PDF documents. While this work is positive about the accessibility of the

PDF files overall, it does not focus on accessibility of forms. A properly labeled and annotated PDF form might be suitable for non-visual accessibility under certain conditions, but this is frequently not the case as we discuss in Section 1. Similarly, [29] examines articles from four journals and finds out that the majority of articles do not adhere to best-practices for PDF accessibility, such as providing alt text for figures used in the papers, or providing tags. Authors of [34] acknowledge the problem of accessibility of PDFs using screen-reader software, since many PDF files consist of scanned images. This is in line with our findings described in Section 1. Findings presented in [1] reveal that PDF accessibility is a problem in educational settings.

There has been previous attempts at making existing PDF files accessible or generating accessible PDF files. One such work focused on a specific application is [4], in which the authors introduce a framework to replace $\LaTeX$ formulae with an accessible replacement text for screen-readers. For generic PDF files, authors of [15] outline an architecture that consists of a client and a processing component for authoring accessible PDF documents. Another example is PAVE[16, 19], which is a web-based tool to detect and correct or provide feedback about PDF accessibility issues. In [17] the authors introduce two plugins for Microsoft Word and Microsoft PowerPoint based on the architecture described in [15] and in [14] they provide a comparison of these tools to the existing such tools. A comparison of tools to make PDF files accessible via post-processing, including PAVE[16, 19], is provided in [13]. These tools are compared on various criteria such as availability as a web-based tool, and availability without charge. The common theme of these works is they either aim to improve the PDF accessibility during the authoring process or make existing PDF files accessible, while our work differs from these works since it does not modify the accessibility properties of the PDF file other than adding new annotations and modifying implicit labels.

Another distinct group of related work regarding PDF accessibility is in the form of guidelines and best-practice manuals. One such example guideline is PDF Techniques for WCAG(Web content accessibility guidelines)[36]. Another example is [18], which briefly describes two processes for making PDF documents accessible. Although these works do not specifically target PDF forms, they provide useful tips for PDF accessibility. For example, PDF Techniques for WCAG requires PDF creators to provide labels for interactive form controls and to provide a correct reading order[36]. Our work does not attempt to make existing PDF accessible per se by modifying the file itself.

2.4 HTML Form Accessibility

Web accessibility has been one of the focal points of non-visual accessibility research in the previous two decades[10, 27, 33]. In addition, major screen-reader software provide features and shortcuts specifically targeted at web interaction. Consequently, thanks to efforts from both academia and the industry, interacting with simple web forms designed with accessibility concerns in mind is frequently an effortless task. We take advantage of this progress by choosing web forms as a replacement for PDF forms.

3 MAKING PDF FORMS ACCESSIBLE

Conceptually, a PDF form is composed of a collection of logical segments; each segment consists of a collection of basic PDF elements, namely text elements and form field elements. Informally, a logical segment represents a group of form elements that will be narrated by the SR as one contiguous piece without interspersing them with the elements of any other logical segment during the narration. In Figure 3 the form elements in the Transferor form and those in the Transferee form are two separate logical segments. The SR will be required to narrate all the elements in Transferor first before it proceeds to read out the elements in Transferee.

We will now describe the transformation pipeline, rooted in the idea of logical segments, to transform a non-accessible PDF into a PDF form that has its fields filled out by the BP.

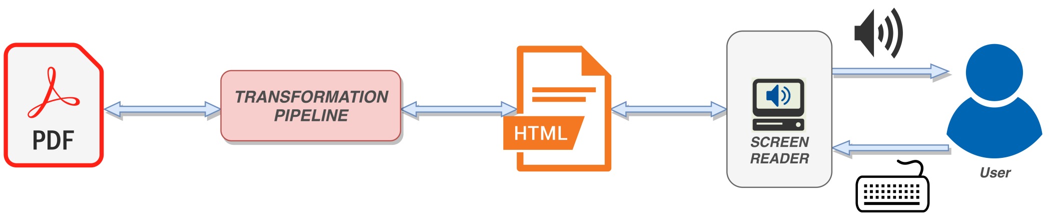


Fig. 5. End-to-end transformation process overview.

3.1 Transformation Pipeline

Figure 5 is a high-level overview of the end-to-end transformation process flow. The PDF document with embedded PDF forms is converted by the transformation pipeline into a HTML document with high fidelity, in the sense that the HTML document retains all the spatial relationships in the PDF documents including forms, their labels and the textual content. The embedded forms get converted into web forms in the HTML page. BPs only interact with the web forms using their SRs, and the filled out web form fields are transparently transferred to the corresponding fields in the PDF form. An optimization algorithm automatically adjusts the length and width of the PDF fields to accommodate arbitrary size field data, thereby ensuring that the filled out PDF document does not have any truncated form-field values, and in addition, is readable.

We now describe the "hows" of the transformation pipeline. Figure 6 shows the main operational steps of the pipeline, denoted by the oval boxes. Each of these steps is described next.

3.1.1 Extraction. In this step the basic elements of the PDF document, namely, text and form field elements are identified and extracted. Towards this, we employ Adobe Acrobat Pro [2] as well as Amazon's AWS Textract [5]. Both these tools identify the PDF elements and using them in combination provides a higher extraction yield, as opposed to using only one of them.

Each identified element is enclosed by a bounding rectangle and annotated with their 2-D coordinate locations. Both these tools also assign implicit labels to form elements, although they may not always be meaningful. We note that sometimes the original PDF form itself may have meaningful implicit labels. In such cases we do utilize these labels as well. The final operation in this step is associating meaningful labels with the form field. Rather than finding the explicit label which in general is difficult, we instead create a suggestion list of possible labels by grabbing the surrounding text elements in the immediate neighborhood of the form field, namely, text elements

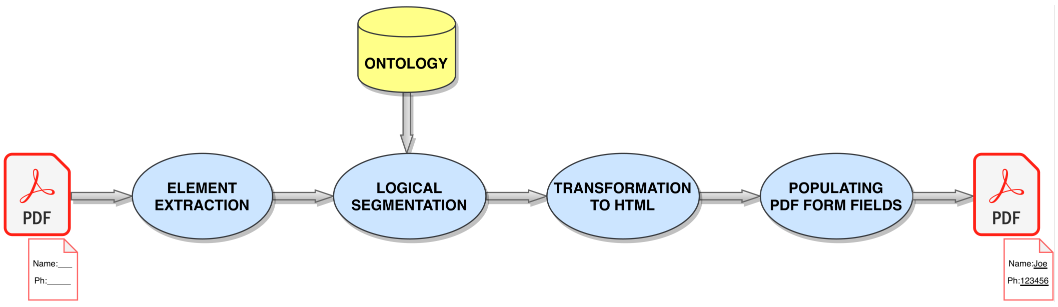


Fig. 6. Main operational steps of the transformation pipeline.

above, below, to the left and to the right of the form field. This list will also include the explicit label, provided it exists in the original PDF document. The list is then ranked according to the length of the text (shorter text is preferred) and distance of the text from the form field (closer is preferred). Our observations showed that combination of these metrics gave the best results when trying to find the explicit label. When the focus of the SR cursor shifts to a field, it first reads out the implicit label assigned to this field by either Acrobat Pro or AWS Textract. Next, it reads out the possible labels such as “text at the top is name”, “text to the right is city”, and so on. If the implicit label is not meaningful, the labels in the suggestion list provide contextual hints that assist the BP in making the correct label association in order to enter the right data value in the field.

3.1.2 Logical Segmentation. In this step we create logical segments from the PDF elements extracted in the previous step (see Figure 7a and Figure 7b). First we create logical lines which are text and form elements that align horizontally and constitute a line in the PDF document. Figure 7c shows an example. These logical lines are then merged into logical segments as shown in Figure 7d.

The process of creating logical segments is driven by an ontology that essentially is a set of machine processable rules. Next, we provide the overview of the rules of this ontology.

Recall that the extraction step results in a set of PDF elements annotated with their associated 2-D coordinates. To create the logical lines the algorithm will merge contiguous elements that satisfy the following constraints:

(1) The elements are horizontally aligned. Horizontal alignment means that their Y coordinates are “almost” identical, i.e., within some some small threshold δ_Y.
(2) The consecutive elements are not far apart, i.e., they must be within some threshold δ_D.

Both δ_Y and δ_D were experimentally determined from our data set of 856 PDF forms.

The logical lines are next merged into logical segments. As a prelude to the merging step, we identify explicit and implicit boxes enclosing a group of PDF elements, which can be regarded as self-contained logical segments. In the PDF document explicit boxes have distinct border lines (Figure 8a), whereas implicit boxes have a discernible background color enclosing the group of

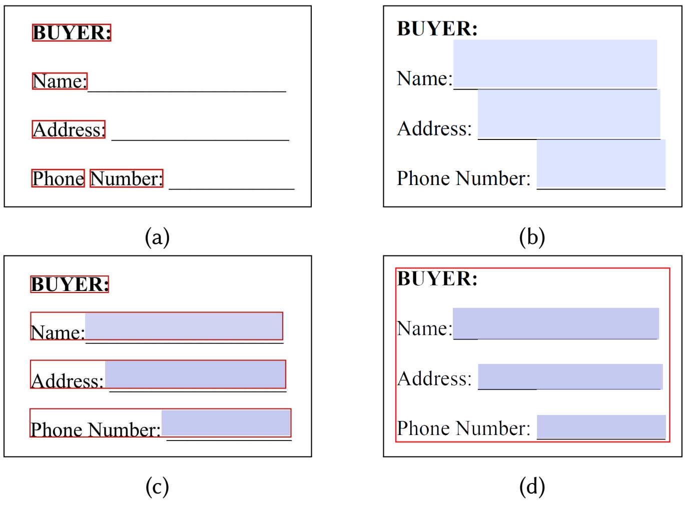


Fig. 7. Process of creating logical segments. (a): Example of text elements with bounding boxes. (b): Example of form field elements with bounding boxes. (c): Example of logical lines created from 7a and 7b. (d): Example of logical segment created from 7c. (Modified from [24])

elements (Figure 8b). Both explicit and implicit boundaries are detected using standard image processing methods (e.g., binarization, edge detection, erosion, dilation etc.). Logical lines within different boundaries cannot be merged into the same logical segment.

Staring with the set of all logical lines, the logical segmentation algorithm recursively merges them into larger logical segments.

The base step merges pairs of logical lines under these constraints:

(1) The pairs of logical lines are vertically aligned. Vertical alignment means that their X coordinates are "almost" identical, i.e., within some some small threshold δ_X.
(2) Logical lines have a high degree of overlap along the horizontal axis, i.e., above some threshold δ_O.

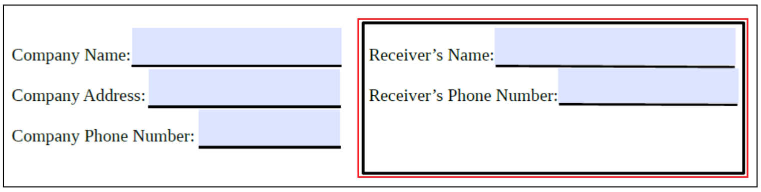
Both δ_X and δ_O were experimentally determined from our data set of 856 PDF forms.

Following this base step, we will get a set of elementary logical segments consisting of merged pairs of logical lines. We will use these elementary segments to bootstrap the merging process as follows: First, we identify elementary segments that are *unmergeable* (i.e., they cannot be merged). These are segments that have logical lines that are vertically aligned but do not have any horizontal overlap. Intuitively, such elementary segments will become independent logical segments. We create the set UM of pairs $< \alpha, \beta >$ where α and β are unmergeable segments.

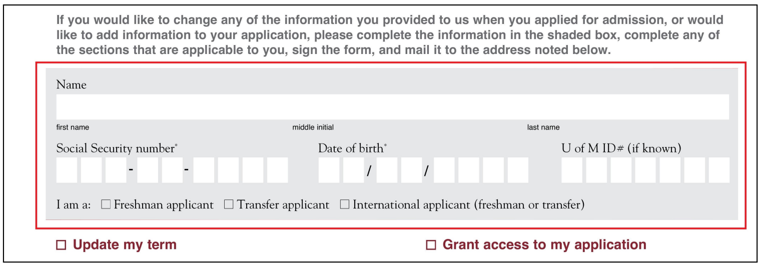
In the recursive step logical segment pairs $< \gamma, \omega >$ are merged into larger segments under the following constraints:

(1) γ and ω are vertically aligned.
(2) γ and ω overlap horizontally.
(3) There is no other logical segment β that cannot be merged with either γ or ω.

Following a recursive step, if an ummergeable segment becomes part of a bigger segment then the unmergeability information is propagated to the bigger segment and the unmergeable set UM gets updated with new pairs of unmergeable logical segments. In Figure 9, logical segments 2 and 3


Company Name:
Company Address:
Company Phone Number:
Receiver's Name:
Receiver's Phone Number:

(a) The red rectangle highlights how PDF forms utilize surrounding boxes for grouping PDF elements.


If you would like to change any of the information you provided to us when you applied for admission, or would like to add information to your application, please complete the information in the shaded box, complete any of the sections that are applicable to you, sign the form, and mail it to the address noted below.

Name

first name middle initial last name

Social Security number* Date of birth* U of M ID# (if known)

- - / /

I am a: ☐ Freshman applicant ☐ Transfer applicant ☐ International applicant (freshman or transfer)

☐ Update my term ☐ Grant access to my application

(b) Red rectangle highlights how a different background color can serve as a visual cue for a group. (Modified from [31])

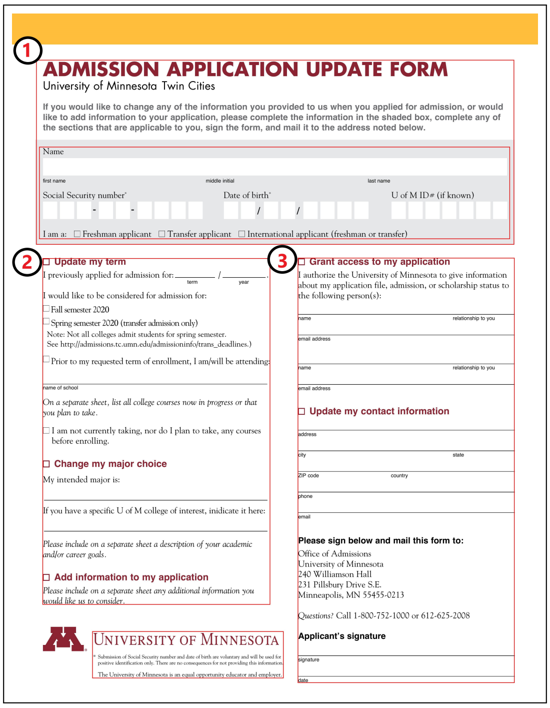
Fig. 8. Illustration of grouping of PDF elements with explicit and implicit boundaries.

are examples of unmergeable logical segments. Therefore, logical segment 1 cannot merge with either one of these two segments. Thus we preserve the logical structure of the two columns in this PDF form. The process stops when no logical segments are left to merge.

3.1.3 Transformation to HTML. The logical segments that are created as a result of the previous step are first topologically sorted based on their positions.The sorted segments are then converted into HTML. The BP interacts with the HTML document with a SR. The PDF form fields are converted to web form fields. These fields are annotated with their implicit labels. They are also augmented with the suggestion lists. Together they assist the BP in filling out the web form with the right data.

3.1.4 Populating PDF Form Fields. The last step pertains to populating the PDF form fields. The subtle problem here is in ensuring that the web form data can be accommodated in the PDF form fields without compromising readability. This is because form fields in PDF have length and width constraints. So text data that cannot be be accommodated within these constraints either gets truncated or the font size will get reduced drastically so as to make the text unreadable.

We address this problem in two steps: In the first step we assess if we can find the largest font size from a range of font sizes, with which the text can be inserted in the form field, splitting it across multiple lines of the form field, if needed. If this is not possible we expand the width and length of the form field by greedily grabbing the white space in the immediate neighborhood of the form field and repeating the previous step. If the text is split over multiple lines we ensure that the word is split at the right spot using the algorithm in [23]. Figure 10 shows an example where the length and width of the initial form field cannot accommodate the form field data without shrinking the font to an unreadable size. Our algorithm can detect this situation and accordingly


1

ADMISSION APPLICATION UPDATE FORM
University of Minnesota Twin Cities

If you would like to change any of the information you provided to us when you applied for admission, or would like to add information to your application, please complete the information in the shaded box, complete any of the sections that are applicable to you, sign the form, and mail it to the address noted below.

Name

first name middle initial last name

Social Security number* Date of birth* U of M ID# (if known)

- - / /

I am a: ☐ Freshman applicant ☐ Transfer applicant ☐ International applicant (freshman or transfer)

2

☐ Update my term

I previously applied for admission for: ________ / ________.
term year

I would like to be considered for admission for:

☐ Fall semester 2020

☐ Spring semester 2020 (transfer admission only)
Note: Not all colleges admit students for spring semester.
See http://admissions.tc.umn.edu/admissioninfo/trans_deadlines.)

☐ Prior to my requested term of enrollment, I am/will be attending:

name of school

On a separate sheet, list all college courses now in progress or that you plan to take.

☐ I am not currently taking, nor do I plan to take, any courses before enrolling.

☐ Change my major choice

My intended major is:

If you have a specific U of M college of interest, inidicate it here:

Please include on a separate sheet a description of your academic and/or career goals.

☐ Add information to my application

Please include on a separate sheet any additional information you would like us to consider.

University of Minnesota

* Submission of Social Security number and date of birth are voluntary and will be used for positive identification only. There are no consequences for not providing this information.

The University of Minnesota is an equal opportunity educator and employer.

3

☐ Grant access to my application

I authorize the University of Minnesota to give information about my application file, admission, or scholarship status to the following person(s):

name relationship to you

email address

name relationship to you

email address

☐ Update my contact information

address

city state

ZIP code country

phone

email

Please sign below and mail this form to:

Office of Admissions
University of Minnesota
240 Williamson Hall
231 Pillsbury Drive S.E.
Minneapolis, MN 55455-0213

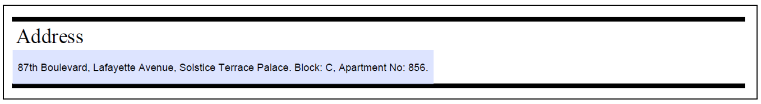
Questions? Call 1-800-752-1000 or 612-625-2008

Applicant's signature

signature

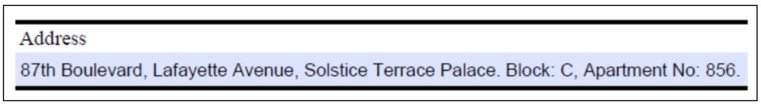
date

Fig. 9. Example of the final logical segments of a PDF form document. (Modified from [31])


Address

87th Boulevard, Lafayette Avenue, Solstice Terrace Palace. Block: C, Apartment No: 856.

(a) Address field, shown with bluish tinge, is too short for the address entered by the user; the font shrinks to accommodate the data and has become unreadable.


Address

87th Boulevard, Lafayette Avenue, Solstice Terrace Palace. Block: C, Apartment No: 856.

(b) Length adjusted PDF form field populated with user-filled data transferred from the web form.

Fig. 10. Populating PDF form field by automatic adjustment of its size. (Modified from [12])

expand the form field to the available space around it and thus preserve the readability of the form field data.

4 EVALUATION

4.1 Algorithm Accuracy

To evaluate our approach, we created a dataset consisting of 100 manually-annotated PDF forms. Specifically, we marked all the components (text, checkboxes and form fields) in each of these PDFs, and also the correct reading order in which they are supposed to be narrated linearly by a screen reader. The outcomes (i.e., field labels, reading order) of our algorithm on this dataset were then compared to the human "ground-truth" annotations to assess its accuracy and effectiveness.

To compare and match the extracted components with human annotations, we used the Levenshtein distance[26] between the component labels, as the extraction services sometimes had few extraneous characters, e.g., "Name:" instead of just "Name" for the same component.

We used two metrics to evaluate the accuracy of our algorithm: Kendall's Tau coefficient[22] and F1 score. To compare the reading order generated by the algorithm to the correct human-specified reading order, we used the Kendall's Tau coefficient, which is a rank correlation coefficient between two lists' ordering. Highest possible value of Kendall's Tau is 1 if the ordering is the same for both lists and the lowest possible value is -1 if a list is compared to its reverse. F1 score was used to measure the accuracy of field extraction and corresponding labels from the PDFs. Perfect extraction (i.e., no false positives and no false negatives) would result in an F1 score of 1, while the lowest possible F1 score is 0.

4.1.1 Results. Table 1 presents the results of our evaluations. We ran our processing algorithm described in Section 3 3 times, each time using the output of a different form-field detection service (i.e., Acrobat Pro, AWS Textract, and both). Results obtained only using the output of Acrobat Pro service are shown in first row while the results obtained only using the output of AWS Textract service are shown in second row. Lastly, the results obtained by combining outputs of both services are shown in the third row.

As seen in Table 1, using only Acrobat Pro extractions as input resulted in the best Tau score while using only AWS Textract extractions yielded the lowest Tau score. Our observations suggest that this is due to Acrobat Pro being more reliable in accurately determining the exact field locations in our dataset; since accurate field-location information leads to accurate form segmentation that improves the reading order. Reverse pattern can be seen in the F1 scores: AWS Textract has performed better

Sources	Tau Score	Precision	Recall	F1
Acrobat Pro	0.922	0.747	0.763	0.755
AWS Textract	0.907	0.817	0.806	0.812
Both Sources	0.921	0.793	0.831	0.811

Table 1. Offline evaluation statistics.

in detecting explicit labels in our dataset, resulting in more accurate element extraction. Using both services together produced the second highest Tau and F1 scores. However, both these scores were only marginally smaller than the best scores created by the other methods, which led us to incorporate output produced by both sources in our proposed method.

4.2 User Study

We conducted an IRB-approved user study with 14 screen-reader users (7 male, 7 female, with an average age of 47.3, median=46, min=32, max=62) (see Table 2). Recruitment of participants was done using contact email lists and word of mouth. The inclusion criteria for this study were (i) having a visual impairment severe enough to require screen-readers for interacting with computers, and (ii) at least basic proficiency with screen-reader software. All users were proficient in using JAWS [20] to varying degrees, which is the screen-reader software used in the study. Participants were given monetary compensation for their contributions.

4.2.1 Design. We designed a within-subject study with two conditions: (i) using Adobe Acrobat Reader[3] (baseline); and (ii) using the web page generated by our system (proposed). Testing of each condition involved three forms, totaling to $2 \times 3 = 6$ forms to be filled per user, excluding the practice forms which are used in both conditions. The 6 forms used are assigned to two conditions randomly to ensure uniform selection probability, and the order of conditions was counterbalanced. The order of forms assigned to each condition was also randomized.

The forms used in the study were chosen from the PDFs that lacked annotations, which constituted 51.4% of the PDF forms we downloaded from the web. Forms selected included a car accident

ID	Gender & Age	Condition	Desktop Screen Reader	Internet Usage Freq.
P1	Male/37	Blind	JAWS	Everyday
P2	Female/47	Blind	JAWS	Everyday
P3	Female/58	Blind	JAWS	Everyday
P4	Male/55	Blind	JAWS, NVDA	Everyday
P5	Female/58	Blind	JAWS	Every week
P6	Male/45	Peripheral vision	SystemAccess, JAWS	Everyday
P7	Male/39	Light perception only	JAWS, NVDA	Every week
P8	Male/60	Low-vision	JAWS, VoiceOver	Every week
P9	Male/45	Blind	JAWS	Everyday
P10	Female/62	Blind	JAWS	Every week
P11	Male/33	Blind	JAWS	Everyday
P12	Female/32	Blind	JAWS	Everyday
P13	Female/57	Blind	JAWS	Everyday
P14	Female/34	Light perception only	JAWS	Everyday

Table 2. Participant demographics.

report form, a background check request form, a sales form, and an entry form for a competition. These forms contained 24.5 form fields on average(min=10, max=48). Only one of the forms was originally a multi-page document, which was then truncated to crop the first page of the form. All the forms used in the user study were single page forms because of the limitations due to the user study duration and for ensuring similar length and level of difficulty for each form.

4.2.2 Procedure. The user study was conducted as follows: The participants were first introduced to each interface along with the useful modes of interaction (e.g., JAWS shortcuts), and asked to perform simple tasks such as locating and filling a particular text field inside a form using these modes of interaction. Once the users felt they achieved a certain level of proficiency with these interfaces, they were asked to perform the main study tasks. Some of the form fields contained questions that may not apply to the users or questions that users may not be willing to reveal, hence, during the practice phase, the participants were informed that they could freely make up dummy answers based on their liking. In addition, the users were instructed to type 'N/A' or 'NA' in the form fields when they were not sure about what to type in those fields.

In the baseline condition, the participants filled PDF forms using Acrobat Reader. These PDF forms were processed using Acrobat Pro to detect form fields that had not be annotated by default. In the proposed condition, the participants filled pre-computed HTML forms generated by our transformation pipeline. Information filled in by the users in both conditions were stored for future analysis.

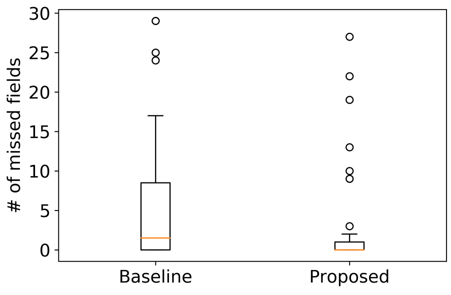
After completing each task, the participants were asked to answer a SEQ (Single Ease Question), on a scale of 1(easy) to 7(hard). We also tracked the time during the process, limiting the filling duration to 15 minutes for each form(Except a single form that took 16 minutes for the user to fill).

4.2.3 Results.

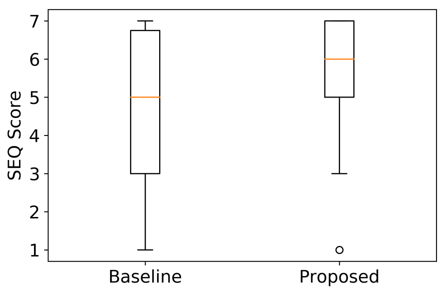
Unfilled Form Fields. The primary metric of performance for an accessibility aid for form-filling task is the number of form fields that are accessible to the users with screen readers. For this purpose, we recorded the number of text fields filled by each user in each and every form. We did not include checkboxes in our analysis since checkboxes might have been intentionally left blank by the users. Figure 11a shows the distribution of number of unfilled text fields by the users per document. On average, the number of text fields missed by the users was 2.81(11.5%) with our proposed method, compared to 5.79(23.6%) with the baseline method. This difference was found to be statistically significant (one-tailed Mann-Whitney U test, $U = 1180.5, p = 0.002$). Based on our observations, this difference can be mainly attributed to two factors: (i) higher form-field detection accuracy of the proposed method, and (ii) more accurate label generation leading to more fields being filled.

SEQ Results. Figure 11b shows the distribution of SEQ results for all forms and all participants. Average SEQ score for baseline was 5.54 while the average SEQ score for the proposed method was 4.52. As a one tailed Mann-Whitney U Test shows, the proposed method was rated significantly better than the baseline method($U = 668.5, p = 0.03$).

Subjective Feedback. We collected user feedback by administering a custom questionnaire. First, at the end of the study, the users were asked which interface they liked more. 9 out of 14 users have stated that they liked the proposed interface more than the baseline. Many of the participants that selected the proposed method over the baseline mentioned how suggestions of possible labels (referred to as 'Hints' in the web interface) were helpful for them to infer the label when the annotations are missing. For example, P1 mentioned that the hints can be helpful when one does not understand the question being asked in a field. P8 said that hints were helpful in describing what's coming next. Speaking of the proposed method, P7 said "More interactive, more descriptive, and more information... It was easier and faster".



(a) Boxplot for distribution of number of missed fields by the users.



(b) SEQ score distribution.

Fig. 11. Boxplots for comparing number of missed fields and SEQ scores for both conditions.

The effects of reading order on user experience became evident particularly in the baseline condition. Speaking of one of the forms filled during the baseline condition, P9 said the form was not laid out correct. P2 stated "i felt lost" for one of the forms filled during the baseline condition.

The users that preferred the baseline interface did so mainly due to their previous familiarity with Acrobat Reader. P13, who preferred baseline over the proposed interface, stated that while hints were interesting, it was a little challenging to get used to. P2 said hints may be confusing. This indicates there may be a learning curve associated with using label suggestions effectively. A longer practice period might result in better utilization of hints. P4 found the baseline more maneuverable, while P12 stated that she understood the forms better with the baseline interface.

5 DISCUSSION AND FUTURE WORK

The results clearly demonstrate the benefits of using web interfaces generated by our method as a substitute for the default PDF form-filling interfaces. However, some open questions and limitations remain to be addressed, which could spark future research.

Tabbing Behavior. During our user study, we observed that some users preferred to use "Tab" shortcut to jump between editable fields in the given form. However, this behavior jumps over text elements in the form, and therefore the screen reader does not read them aloud. During the user study, this caused some users to miss the contextual information required to understand and fill certain form fields, e.g., ‘From address’ vs. ‘To address’. As part of future work, we will explore different ways to compensate for this user behavior. For example, pairing form fields with relevant text in a way that results in screen reader reading this relevant text aloud could help users to stick with tabbing behavior without losing the context.

Form Label Suggestions. In our approach, we suggested all possible form-field labels that our program detected, based on our belief that more information would provide more context to the blind users who are trying to infer what the form field is asking of them. However, our user study showed that more information can likely cause more confusion. Creating a more delicate balance by selecting and presenting only useful labels to the blind users can improve the performance of our pipeline, and therefore is a topic of future research.

Making Form Label Suggestions Optional. Our observations suggest that the ability to estimate form layouts is directly related to experience the users have with regard to filling HTML or PDF forms with a screen reader. The two participants who self-rated their screen-reader experience as beginners both preferred the baseline method over the proposed method, while none of the users that preferred proposed method over baseline self-rated their screen-reader experience as beginners. This indicates that the hints need to be optional, so that the users can select this option when desired, and turn it off when they feel it is not beneficial for form filling.

Better Form-Field Recognition. Improving the systems that recognize and extract form fields is one of the main steps towards making PDF forms more accessible. It is also one of the primary bottlenecks that our system strives to mitigate. As our goal in this work was not to develop novel form-field recognition algorithms, we utilized state-of-the-art solutions in an effective manner. Form-field recognition has been commoditized, with improved versions being released constantly. While the current form-field recognition systems are frequently inaccurate in handling complicated forms where the form fields are embedded inside text or are not denoted with a line, they are highly accurate in forms where the explicit labels and corresponding form fields are visually discernible. Future developments by service providers and researchers will result in more accurate methods for form field recognition.

Accuracy of Logical Segments and Reading Order. We identify logical segments using the process described in Section 3. Accurate recognition of logical segments is vital for constructing a meaningful and correct reading order. However, as the results in Section 4.1 shows, reading order constructed by our algorithm is not always accurate. Improving reading order accuracy would allow for better preservation of context and hence would make the form filling process smoother.

6 CONCLUSION

In this work, we addressed the accessibility issues that screen-reader users face while filling PDF forms. With screen-readers, we noticed that many fields in majority of PDF forms are either inaccessible, or are narrated in an incorrect serial order, thereby making it extremely challenging for screen-reader users to fill them out with ease and also without errors. Therefore, we proposed a transformation pipeline that enables the users to substitute PDF form filling with the more accessible and convenient HTML form filling, with the pipeline transparently transferring the user input obtained from the HTML form back into the original PDF form. Furthermore, to ensure correct reading order, the pipeline analyzes the PDF form to identify chunks of semantically-related form elements, called logical segments, and narrates each segment as one contiguous piece without interspersing its members with other form elements not in the segment. Evaluation of our approach in a user study with 14 participants showed significant accessibility and usability improvements compared to the status quo, thereby validating its effectiveness in assisting screen-reader users fill out PDF forms.

ACKNOWLEDGMENTS

This work was supported by NSF Award: 1806076, NIH Awards: R01EY026621, R01EY030085, R01HD097188, NIDILRR Award: 90IF0117-01-00.

REFERENCES

[1] Patricia Acosta-Vargas, Sergio Luján-Mora, and Tania Acosta. 2017. Accessibility of Portable Document Format in Education Repositories. In *Proceedings of the 2017 9th International Conference on Education Technology and Computers*. 239–242.

[2] Adobe. [n.d.]. Adobe Acrobat Pro DC. https://acrobat.adobe.com/us/en/acrobat/acrobat-pro.html.

[3] Adobe. [n.d.]. Adobe Acrobat Reader DC. https://acrobat.adobe.com/us/en/acrobat/pdf-reader.html

[4] Dragan Ahmetovic, Tiziana Armano, Cristian Bernareggi, Michele Berra, Anna Capietto, Sandro Coriasco, Nadir Murru, Alice Ruighi, and Eugenia Taranto. 2018. Axessibility: a latex package for mathematical formulae accessibility in pdf documents. In *Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility*. 352–354.

[5] Amazon. [n.d.]. Amazon Textract | Extract Text & Data | AWS. https://aws.amazon.com/textract/.

[6] Apple. [n.d.]. Vision Accessibility - Mac - Apple. https://www.apple.com/accessibility/mac/vision/
[7] Vikas Ashok, Yevgen Borodin, Yury Puzis, and IV Ramakrishnan. 2015. Capti-speak: a speech-enabled web screen reader. In *Proceedings of the 12th Web for All Conference*. 1–10.
[8] Vikas Ashok, Yury Puzis, Yevgen Borodin, and IV Ramakrishnan. 2017. Web screen reading automation assistance using semantic abstraction. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*. 407–418.
[9] Yevgen Borodin, Jeffrey P Bigham, Glenn Dausch, and IV 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)*. ACM, 13.
[10] Peter Brophy and Jenny Craven. 2007. Web accessibility. *Library trends* 55, 4 (2007), 950–972.
[11] Attorney Grievance Committee. [n.d.]. Background Check Request Form. https://www.nycourts.gov/courts/ad1/Committees&Programs/DDC/Background%20Check%20Request%2001%2028%202019.pdf
[12] PENNSYLVANIA DEPARTMENT OF STATE BUREAU OF CORPORATIONS and CHARITABLE ORGANIZATIONS. [n.d.]. Articles of Incorporation-Nonprofit. https://www.dos.pa.gov/BusinessCharities/Business/RegistrationForms/Documents/5306-7102-%20Art%20of%20Inc-Dom%20Nonprofit.pdf
[13] Alireza Darvishy. 2018. PDF accessibility: Tools and challenges. In *International Conference on Computers Helping People with Special Needs*. Springer, 113–116.
[14] Alireza Darvishy and Hans-Peter Hutter. 2013. Comparison of the effectiveness of different accessibility plugins based on important accessibility criteria. In *International Conference on Universal Access in Human-Computer Interaction*. Springer, 305–310.
[15] Alireza Darvishy, Hans-Peter Hutter, Alexander Horvath, and Martin Dorigo. 2010. A flexible software architecture concept for the creation of accessible PDF documents. In *International Conference on Computers for Handicapped Persons*. Springer, 47–52.
[16] Alireza Darvishy, Hans-Peter Hutter, and Oliver Mannhart. 2011. Web application for analysis, manipulation and generation of accessible PDF documents. In *International Conference on Universal Access in Human-Computer Interaction*. Springer, 121–128.
[17] Alireza Darvishy, Thomas Leemann, and Hans-Peter Hutter. 2012. Two software plugins for the creation of fully accessible PDF documents based on a flexible software architecture. In *International Conference on Computers for Handicapped Persons*. Springer, 621–624.
[18] Heather Devine, Andres Gonzalez, and Matthew Hardy. 2011. Making accessible PDF documents. In *Proceedings of the 11th ACM symposium on Document engineering*. 275–276.
[19] Luchin Doblies, David Stolz, Alireza Darvishy, and Hans-Peter Hutter. 2014. PAVE: A web application to identify and correct accessibility problems in PDF documents. In *International Conference on Computers for Handicapped Persons*. Springer, 185–192.
[20] Freedom Scientific. [n.d.]. JAWS® - Freedom Scientific. https://www.freedomscientific.com/products/software/jaws/.
[21] Yoichi Haga, Wataru Makishi, Kentaro Iwami, Kentaro Totsu, Kazuhiro Nakamura, and Masayoshi Esashi. 2005. Dynamic Braille display using SMA coil actuator and magnetic latch. *Sensors and Actuators A: Physical* 119, 2 (2005), 316–322.
[22] Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika* 30, 1/2 (1938), 81–93.
[23] Donald E Knuth and Michael F Plass. 1981. Breaking paragraphs into lines. *Software: Practice and Experience* 11, 11 (1981), 1119–1184.
[24] Template Lab. [n.d.]. CONTRACT FOR SELLING A CAR. http://templatelab.com/vehicle-purchase-agreement/?wpdmdl=32339
[25] 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.
[26] Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, Vol. 10. 707–710.
[27] Jennifer Mankoff, Holly Fait, and Tu Tran. 2005. Is your web page accessible? A comparative study of methods for assessing web page accessibility for the blind. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 41–50.
[28] Hisashi Miyashita, Daisuke Sato, Hironobu Takagi, and Chieko Asakawa. 2007. Making multimedia content accessible for screen reader users. In *Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A)*. 126–127.
[29] Julius T Nganji. 2015. The Portable Document Format (PDF) accessibility practice of four journal publishers. *Library & Information Science Research* 37, 3 (2015), 254–262.
[30] NVDA team. [n.d.]. NVDA. https://www.nvaccess.org/.
[31] University of Minnesota. [n.d.]. ADMISSION APPLICATION UPDATE FORM. https://admissions.tc.umn.edu/PDFs/Application_Update.pdf

[32] State of Nebraska Department of Motor Vehicles. [n.d.]. Bill of Sale. https://dmv.nebraska.gov/sites/dmv.nebraska.gov/files/doc/dvr/forms/billofsale.pdf?trkid=1&cka=88&cko=219&cks1=push&cks2=24248663&cks3=psh-cpa-pushnami-trumpboasts-fri-720am-cr1

[33] Christopher Power, André Freire, Helen Petrie, and David Swallow. 2012. Guidelines are only half of the story: accessibility problems encountered by blind users on the web. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 433–442.

[34] Brian Sierkowski. 2002. Achieving web accessibility. In *Proceedings of the 30th annual ACM SIGUCCS conference on User services*. 288–291.

[35] Mireia Ribera Turró. 2008. Are PDF documents accessible? *Information Technology and Libraries* 27, 3 (2008), 25–43.

[36] W3C. 2016. PDF Techniques for WCAG 2.0. https://www.w3.org/TR/WCAG20-TECHS/pdf

[37] Cheng Xu, Ali Israr, Ivan Poupyrev, Olivier Bau, and Chris Harrison. 2011. Tactile display for the visually impaired using TeslaTouch. In *CHI'11 Extended Abstracts on Human Factors in Computing Systems*. 317–322.

[38] Levent Yobas, Dominique M Durand, Gerard G Skebe, Frederick J Lisy, and Michael A Huff. 2003. A novel integrable microvalve for refreshable braille display system. *Journal of microelectromechanical systems* 12, 3 (2003), 252–263.

Received February 2020; revised March 2020; accepted April 2020