Universität Regensburg

Design and Evaluation of a Semi-autonomous Accessilility Tool for Web Development

Masterarbeit im Fach Medieninformatik

am Institut für Information und Medien, Sprache und Kultur (I:IMSK)

Vorgelegt von:	Mathias Götz
Adresse:	Halfingerstr. 24, 81825 München
E-Mail (Universität):	mathias-christian.goetz@stud.uni-regensburg.de
E-Mail (privat):	mathias.goetz@outlook.com
Matrikelnummer:	2244750
Erstgutachter:	Prof. Dr. Christian Wolff
Zweitgutachter:	Prof. Dr. Niels Henze
Betreuer:	Prof. Dr. Christian Wolff, Markus Müller (TÜV SÜD Product Service GmbH)
Laufendes Semester:	6. Semester Medieninformatik M.Sc.
Abgegeben am:	23.06.2023

Inhaltsverzeichnis

Task **5**

1. Introduction **6**
1.1. Problem statement . . . 6
1.2. Approach outline . . . 6
1.3. Problems and results . . . 7
1.4. Structure of this work . . . 7

2. Background **8**
2.1. Digital Accessibility . . . 8

3. Related Work **9**
3.1. Basic guidelines? . . . 9
3.1.1. EAA . . . 9
3.1.2. EN 301 549 . . . 9
3.1.3. W3C WCAG . . . 9
3.2. Automation of Accessibility . . . 10
3.2.1. Testing . . . 10
3.2.2. Implementation . . . 10

4. System and design **11**
4.1. Basic Algorithm Details . . . 11
4.2. Selected Accessibility Problems . . . 11
4.2.1. Language missing . . . 12
4.2.2. Region missing . . . 15
4.2.3. H1 Missing . . . 15
4.2.4. Heading skipped . . . 16
4.2.5. Heading Empty . . . 16
4.2.6. Heading Possible . . . 16
4.2.7. Text Small . . . 17
4.2.8. Contrast . . . 17
4.2.9. Link Empty . . . 18
4.2.10. Link Suspicious . . . 18
4.2.11. Alt Link Missing . . . 19
4.2.12. Link redundant . . . 19
4.2.13. Link internal broken . . . 20
4.2.14. Alt missing . . . 20
4.2.15. Alt suspicious . . . 20
4.2.16. Alt duplicate . . . 21
4.2.17. Alt redundant . . . 21
4.2.18. Table layout . . . 22
4.2.19. Text justified . . . 22
4.2.20. Button empty . . . 22
4.2.21. Label missing . . . 23

4.2.22. Title redundant . . . 23
4.2.23. Event handler . . . 24
4.2.24. Noscript . . . 24

5. System Validation **25**
5.1. Implementation . . . 25
5.1.1. Basic Algorithm Details . . . 25
5.1.2. Algorithm parts . . . 25
5.2. Evaluation . . . 28
5.3. Algorithm Limitations . . . 30
5.4. Results . . . 30

6. Discussion **31**
6.1. Results . . . 31
6.2. Limitations . . . 31

7. Conclusion **32**
7.1. Summary . . . 32
7.2. Outlook/Future Work . . . 32

Literaturverzeichnis **33**

A. Appendix **34**

Erklärung zur Urheberschaft **35**

Erklärung zur Lizenzierung und Publikation dieser Arbeit **36**

Zusammenfassung

Abstract

Digital Accessibility is the principle of removing barriers in the usage of digital services and websites, to make them usable for everyone, regardless of their abilities. Accessibility has a focus on the inclusion of disabled users, but able-bodied people can also profit from the implementation of accessibility. The current standard for Accessibility testing EN301 549/WCAG 2.1 includes 4 principles with 78 success criteria (A&AA). Websites currently implemented and developed without a focus on accessibility have a lot of criteria to implement, to offer accessibility. This amount of work puts off most companies and developers, especially when there has been no focus on accessibility from the start of the development process. Automated Testing Solutions exist, and offer a way to check the Accessibility basics of any Webpage without a lot of effort, but the implementation still needs to be done by hand. This thesis focuses on analyzing these criteria, and finding a way to implement parts of the required accessibility criteria automatically. This automation toolset is then tested with 123 public accessible Weppages. Results show that using the tool improves conformance with the given criteria and therefore accessibility as a whole by XX%.

Task

Hintergrund

Der Kerngedanke (Digitaler) Accessibility ist es, allen Menschen gleichermaßen ohne fremde Hilfe Zugang zu digitalen Systemen zu ermöglichen. Daher gibt es bereits bestehende Guidelines, die bspw. für Webseiten umgesetzt werden können, um Menschen mit Einschränkungen den Zugang zu ermöglichen oder zu erleichtern. Diese Methoden werden allerdings in einem Großteil der bereits bestehenden Systeme nicht umgesetzt (Beispiel/Quelle). Um Entwicklern bei der Umsetzung zu helfen, existieren Tools um die Accessibility (teil)automatisiert zu testen. Diese können allerdings nur die Korrektheit einer zuvor manuell durchgeführten Implementierung überprüfen.

Zielsetzung

Um das Problem der fehlenden Accessibility in einem Großteil der bereits bestehenden Webseiten zu lösen, soll ein Algorithmus entwickelt werden damit bereits bestehende Accessibility Guidelines (teil-)automatisiert umgesetzt werden können. Dieser Algorithmus soll dann mit einem Datensatz aus bestehenden Webseiten validiert werden, indem die Accessibility der Webseiten vor und nach dem Einsatz des Algorithmus bewertet wird.

Konkrete Aufgaben:

- Aufbereiten Relevanter Literatur
- Erarbeiten eines Algorithmus für (teil)automatisierte Umsetzung von Accessibility für Webseiten anhand bestehender Guidelines
- Prototypisches Umsetzen dieses Algorithmus
- Validierung des Algorithmus an bestehenden Webseiten

1. Introduction

Untere Kapitel zusammenfassen

1.1. Problem statement

In Germany alone XX.X million people suffer from disabilities.[source]. With the digital world playing an increasingly big role in everyday life, those people need to have the same access to digital technologies as everyone else. To make the internet usable, and have these peopele be able to take part in digital life, digital accessibility is needed. Accessibility is an important part in web development,. but most developers still do not take account of it [cite].

- Anschauliches Beispiel

This thesis sets out to show if automating accessibility for webpages can be a viable option to make the web more accessible. A lot of previous research shows automated testing of accessibility for websites, but the projects automating the implementation of accessibility are scarce.

- Add EAA

1.2. Approach outline

The thesis starts by approaching a subset of accessibility problems, with a theoretical algorithm for implementation. This algorithm is then implemented in python. To see if this prof of concept can be used in the real world, a study with the top 100 webpages in germany is conducted, and the results are then discussed.

1.3. Problems and results

1.4. Structure of this work

2. Background

TASK: Alle Hintergrundbegriffe herleiten/definieren die zum Verständnis der Thesis nötig sind.

2.1. Digital Accessibility

What is Accessibility? Digital Accessibility is the principle of removing barriers in the usage of digital services and websites, to make them usable for everyone, regardless of their abilities. Accessibility has a focus on the inclusion of disabled users, but able-bodied people can also profit from the implementation of accessibility. [Citation]

The current standard for Accessibility testing EN301 549/WCAG 2.1 includes 4 principles with 78 success criteria (A&AA). These success criteria which need to be tested by hand, as a human judgement is needed.

Websites currently implemented and developed without a focus on accessibility have a lot of criteria to implement, to offer accessibility. This amount of work puts off most companies and developers.

As these guidelines have a main focus on Web, the mobile sector (Apps) is mainly left out, with WCAG stating "Mobile accessibility is covered in existing W3C WAI accessibility standards/guidelines. There are not separate guidelines for mobile accessibility.". This means, the guidelines are very broad, to fit in everything.

- Way more information - How many peaople are affected?

3. Related Work

TASK: Auf andere Paper, Forschungsarbeiten, Tools eingehen die sonst in genau diesem feld unterwegs sind - und Begründen warum sie hier einfließen oder auch nicht

3.1. Basic guidelines?

3.1.1. EAA

- Describe european accessibility act

3.1.2. EN 301 549

- describe (shortly) what EN 301 549 is

3.1.3. W3C WCAG

WCAG is a set of success criteria created by the W3C. Criteria can be class A, AA, or AAA - with A & AA required for compliance, and AAA as "going the extra step". The current Version is WCAG 2.1, released in 2018 with WCAG 2.2 scheduled to arrive in December 2022 [-> maybe here already use 2.2? Drafts are available]. WCAG contains of 4 main classes of principles:

- Percievable:
  Information and user interface components must be presentable to users in ways they can perceive.
- Operable:
  User interface components and navigation must be operable.
- Understandable:
  Information and the operation of user interface must be understandable.

- Robust:

  Content must be robust enough that it can be interpreted by by a wide variety of user agents, including assistive technologies.

The current version 2.1 contains 78 success criteria (A&AA) to be implemented and tested.

- More information - Direct quote from WCAG

3.2. Automation of Accessibility

3.2.1. Testing

3- 4 Paper beschreiben die sich mit automatisierung von testing accessibility befassen, dazu eine übersicht der relevanten tools geben (Deque, Lighthouse etc.)

3.2.2. Implementation

Das eine Paper beschreiben das sich mit dem automatisierten Refactoring beschäftigt. - Viel mehr gibt es nicht!

4. System and design

TASK: Theoretisches System-Design, herleiten von Lösungen für die definierten Probleme, Hier nur pseudocode-Algorithmus vorschläge wie diese Problem e gelöst werden können. Herleiten von relevanz wieso gnau diese Parts der WCAG Angeschaut werden (Bsp. sehr schnell/einfch umzusetzen, sehr wichtig, sehr verteiltes problem)

Herleiten woher die Guidelines Stammen, Tabelle

-> Onenote

4.1. Basic Algorithm Details

The algorithm finds accessibility flaws within a specified website. For the predefined accessibility problems an algorithm is defined to fix the problem automatically. The algorithm is designed to be used by developers, so every change made to the website that changes the appearance in any way, can be checked by the developer. Originally, the idea was to make the algorithm tailored to users, by fixing every webpage before it is opened. This idea can still be persued using the parts of the algorithm that do not need user input or supervision.

- Results: 2 differnt tools, checking and fixing

4.2. Selected Accessibility Problems

To find a selection of accessibility problems, the WebAim million study has been taken into account. This study looks at the top million websites worldwide, and groups the most influential and frequent issues. [citation] These issues are differentiated in errors and warnings. For the purpose of this thesis, the team behind the study has been contacted, to get the list of problems with a overall percentage of

which problems are the most frequent. The problems are categorized, these categories can be idenitified using the wave analytics tool. All categories can be mapped to W3C WCAG Guidelines. (Verlinkungen!) The following table shows the errors and warnings from the WebAim 2022 study, together with total percentage of the million websites showing these problems. For the sake of this study, problems occuring more than 10% of total webpages are considered. With this, 24 different accessibility errors and warnings remain. Of those 24 errors and warnings, X can automatically be fixed without developer input, and X need confirmation or mor information from the developer, because they either need a decision or alter the websites layout. (Mapping auf WCAG?)

- Sub Problems Categorized

4.2.1. Language missing

Relevance

Language tags are a kind of meta-tag, informing the user in which language the text of a webpage is presented. The usage of language tags is mainly important for users of screen-readers, with the screen readers using different voices and pronounciations for different languages. If a language tag is not set or set incorrectly, the pronunciation of text may not make sense, and therefore be difficult to understand for users.

Approach

This issue can be fixed fully automatically, by looking at a text part of the website and determining the language using language-recognition libraries. As this only adds a tag to the html-header, no layout change or other side-effects occur.

Wave Category	Class	Percentage
contrast	Error	83,9%
link_empty	Error	50,1%
label_missing	Error	46,1%
alt_link_missing	Error	38,7%
alt_missing	Error	33,8%
button_empty	Error	27,2%
language_missing	Error	22,3%
heading_empty	Error	13,0%
link_redundant	Warning	71,1%
noscript	Warning	48,5%
heading_skipped	Warning	40,5%
title_redundant	Warning	38,4%
text_small	Warning	29,2%
region_missing	Warning	26,0%
h1_missing	Warning	23,8%
heading_possible	Warning	19,7%
alt_redundant	Warning	18,1%
link_suspicious	Warning	18,0%
alt_duplicate	Warning	14,5%
table_layout	Warning	13,9%
alt_suspicious	Warning	13,5%
link_internal_broken	Warning	12,0%
event_handler	Warning	11,4%
text_justified	Warning	10,8%

Tabelle 1.: All selected accessibility problems

Guideline Name	Type	Described in
	Basic Page Data	
language_missing	2*Basic Page information	
region_missing		
	Text Content	
h1_missing	4*Headings	
heading_skipped		
heading_empty		
heading_possible		
text_small	2*Text Content	
contrast		
link_empty	5*Links	
link_suspicious		
alt_link_missing		
link_redundant		
link_internal_broken		
	Images	
alt_missing	4*ALT Text	
alt_suspicious		
alt_duplicate		
alt_redundant		
	Layout/Structure	
table_layout	2*Layout	
text_justified		
button_empty	Button	
label_missing	2*Assistive Tags	
title_redundant		
event_handler	2*Scripting	
noscript		

Tabelle 2.: All selected accessibility problems categorized

4.2.2. Region missing

Relevance

Website regions, for example ”header”,”footer”, ”navigationör ”contentäre parts of a webpage, that exist on every subpage of the website. Because of that, it makes no sense for screenreaders to read it every time, when the user already heard it for the last page, and only reads out the ”content”that is different on every subpage.

Approach

To automatically determine page regions, it is possible to look at different subpages of the same website, and trying to determine which elements are on every subpage. With this information, it is possible to determine which parts are content and which are parts of the layout.

4.2.3. H1 Missing

Relevance

The heading structure is an important part of conveying the website structure for screenreader users and users with other disabilities. Screenreaders may group the content part in chapters using these headings. If the heading is missing, it can be hard for some users to understand the website structure.

Approach

All heading levels can be determined inside the html-structure. If there are H2-elements, the top H2 element can be converted to a H2 element, while keeping the appearance (size, text style) of a H2 element. This changes nothing layout-wise, while providing more structure.

4.2.4. Heading skipped

Relevance

Like with H1 missing, when a heading level between H1 and H6 is skipped, like for example there is no H3 element, it just continues from H2 to H4 makes it difficult to understand the website structure for some users. That may be a conscious layout decision, but makes the structure harder to understand for screenreader users.

Approach

Like with H1 missing, it is possible to determine al heading levels automatically. If a level is skipped, it is possible to change the level, so there are no gaps, without changing the layout. This makes the structure easier to understand for for ex. screenreader users, while keeping the same appearance.

4.2.5. Heading Empty

Relevance

As described in earlieer chapters, headings add structure to websites. When a heading element is present that has no content, it may still be read to screenreader users. That may cause confusion on thew website structure.

Approach

As an empty heading will not be seen for users that view the website regulrily, there is no information lost when the element is deleted. So these empty headings can just be deleted, so screenreader users are not confused and have the same experience as regular users.

4.2.6. Heading Possible

Relevance

A part of text styled like a headline can make no difference to a seeing user. On screenreaders, infortmation like text size or effects like ”boldör ïtalicäre mostly lost.

This is why a seeing user may see no differnce between a real heading and regular text that is styled like a heading. Users of screenreaders will not see theis informationm, and it will be treated like regualr text. If it is intended to be seen like a heading, this information will be lost for these users.

Approach

Only parts like text size and style can be determined automatically, not the intention behind these style choices. If a full paragraph is styled like a heading, and contains onyl one sentence or less, it can be suspected to be a heading. If such occurrences are found, the assumption of it being a heading can be made.

4.2.7. Text Small

Relevance

Small text might not be seen by users with a vision impairment. On the other hand, text with a size of 0 will not be seen by regular website users, but might be read out to screenreader users. This can lead to information being lost for different user groups, and therefore a different usage expreience for these groups.

Approach

Text size inside a html document can be programmatically determined. If the size is 0, it can just be deleted, without losing any information, as it would not be seen anyway. If it is too small to be seen for users with lower vision, the text-size can be increased automatically. This can change the layout, so this change is not without side-effects.

4.2.8. Contrast

Relevance

Good contrast is needed for visually impaired users. If the contrast is too low, they may not be able to read the text presented on a webpage.

Approach

Ther contrast can be calculated automatically, when looking at the text and background colors. If the contrast is not sufficient, the colors can be changed automatically. As the background is more likely to be an intentional layout decision, the text color can be changed. This still carries the risk to destroy intentional layout decisions.

4.2.9. Link Empty

Relevance

An empty link has no text, and will therefore not be shown correctly. This problem exists for regular as well as screen reader users.

Approach

As the purpose of the link is not known, it can either be deleted or the link destination programmatically determined. In this case, the Link destination will be determined, with the goal to add a meaningful description. When opening the link, one can either take the site title or description to see, what the contents of the link are. This information can then be used to create a link description to use as new link text. This may change the layout, so the developers approval is needed.

4.2.10. Link Suspicious

Relevance

Link suspicious can be when the link text consists of ”Click”, ”Hereör only the URL, that means the link purpose can not be discerned only from the link text. This can be needed for users of assistive technologies, that may read out the links in a different way than it is presented to seeing users.

Approach

If the link text can not be used to convey the correct link information, the text can be programmatically changed, as described in X.X.

4.2.11. Alt Link Missing

Relevance

ALT-Link missing describes the problem arising when the only content of a link is an image, without alternative text. That means, the link will be seen to screenreader users like it has no content, when there is no text and no alternative text for the image to read out.

Approach

This problem can either be fixed by adding alternative text (see X.X) or by determining the links destination (see X.X).

4.2.12. Link redundant

Relevance

A redundant link may not disturb regular users, but to screen reader users the link may be read twice. This is especially problematic when the links are next to each other, because one link will be read out right after the other. This adds time for screenreader users, while offering no extra information.

Approach

As links can be identified, link sources can be checked for adjacent links. If adjacent links go to the dsame destination, one can be deleted without losing information. For the selection of which link is more useful and should be kept, developer input may be needed.

4.2.13. Link internal broken

Relevance

Internal links provide a way to jump to a point inside of the webpage. If these links are broken, they only add confusion and do not offer any value to the user.

Approach

As it is not possible to get the original intention of the link, when the corresponding target does not exist on the webpage, no functionality is lost when the broken link is removed. These links start with a ””, and if the correspnding ID is not present on the webpage, it is removed.

4.2.14. Alt missing

Relevance

ALT-Tags are important for blind users or users with a sighting impairment. These Users can only percieve parts of images, or not percieve them at all. For this, every image in a website has to have an alt tag, containing a description of the image, to give another method to get the information otherwise only conveyed by the picture.

Approach

Inside HTML code, it is easy to find <img>-tags, that do not contain an alt= element. An AI description service can then be used to transcribe the contents of these pictures. Any context between pictures and text is lost, because these libraries only transcribe what they can see in the image.

4.2.15. Alt suspicious

Relevance

Like ALT-Text missing, a not suspicious ALT-Text is needed to convey the same information the image shows. Suspicious ALT-Text might be alt text that is very short, only states Ïmageör just the Filename.

Approach

The ALT text can be checked for length, Keywords like Ïmageör if it ends in .JPG/.PNG. If that is the case, the Algorithm from 4.2.14 can be used, to ghenerate a new ALT-Text.

4.2.16. Alt duplicate

Relevance

Duplicate alt texts can be a problem, if different images have the same alt-text. That means, for users with impaired vision, two different pictures have the same descriptions. In this case, they miss out on information seeing users can perceive.

Approach

Images can be checked, if the adjacent images contain the same alt-text. If this is the case, the alt-text for one or both images can be changed automatically using Computer Vision, or the developers input.

4.2.17. Alt redundant

Relevance

Redundant ALT-Text means, the contents of the alt text are also seen in nearby text. This means, the same information is seen more than once. This can add confusion.

Approach

The images alt-text can be checked. If it appears in nearby text, it can be removed without losing any information. It can also be changed to Ïmage providing context to XYZör completely regenerated using a ComputerVision Library.

4.2.18. Table layout

Relevance

Layout Tables are Tables that do not come with a header. This is done for layout purposes, for example to get two text-areas next to each other. These tables are read out to screenreader users like a real table. This adds confusion, as the tables are not intended to be presented as such, and should only change the layout.

Approach

Layout tables can be found, as they contain no <th> elements. This content can then be changed into regular html elements, without altering the layout.

4.2.19. Text justified

Relevance

Text justification adds white spaces, so that every line of text has the same length[1]. This may look cleaner, but different spacing can make it more difficult to read in large blocks.

Approach

Fully justified paragraphs can be automatically found, checked for length and the justification can be switched to regular text-layout. This makes the text easier to read, but does change the layout.

4.2.20. Button empty

Relevance

Empty buttons have no descriptive Text. So the user does not know what the purpose of the button is, if the context does not provide additional information.

[1] https://practicaltypography.com/justified-text.html

Approach

Empty buttons can be found, but the purpose of the buttons can not be automatically determined. In this case, developer input on the purpose of the button may be needed. With this information, the Button-Text can be changed automatically.

4.2.21. Label missing

Relevance

When a Website has Forms for user input, they can have a label attached to it. This label provides information on the purpose of that label, and is needed for every user to understand the forms purpose.

Approach

The intention behind the label can not be found automatically. It is possible to add the input type, for example ”Textör ”Passwordäs a label, to add a little bit of extra information, to make it easier for the user to understand the purpose of that input field.

4.2.22. Title redundant

Relevance

The title tag for html elements adds extra information to html elements. If it is the same as image alt-texts or the text in general, it adds no extra information.It is then just another element that may be shown to the user, whilst not providing any benefits or new information.

Approach

All html elements can be searched for the title tag. If it contains the same text as image alt text or the adjacent text, it can just be deleted without losing information or changing the layout in any way.

4.2.23. Event handler

Relevance

Event handlers, like önclicköf önmouseoverßtart an action, when the mouse is clicked or just hovers over the content. This can lead to content not being seen by users that access the website only using a keyboard or in other ways that do not include a mouse.

Approach

The make this content accessible, elements like önfocus", that trigger when the element recieves focus using keyboard controls, can be added. For this, every element with only mouse accessible event handlers can be found, and more accessible options like önfocus"can be added.

4.2.24. Noscript

Relevance

The noscript element is just a warning. Noscript elements contain information, that will be presented to the user if JavaScript is disabled. This tries to convey the same information as the JavaScript parts. As only 1.3% of users have JavaScript disabled [2], this may not be needed. Most assistive teconologies can handle JavaScript [Cite], so it does not add accessibility. This just adds extra bulk to the website.

Approach

Noscript elements can be found, and deleted without changing the layout. No information will be lost, as these elements are only shown if javascript is disabled, and try to provide the same information as the JavaScript parts.

[2] https://www.searchenginepeople.com/blog/stats-no-javascript.html

5. System Validation

To validate the algorithm, the Top 100 Websites in Germany are automatically evaluated. To have a level of repeatability, all those Websites are downloaded and saved into a data set. (auf algorithmus eingehen?) All websites are first evaluated for accessibility problems, and these scores are saved into a csv file. After that, the combined algorithms defined in section X (Verlinkung) are used, to fix all implemented accessibility problems automatically. For this, the number of changes is also recorded. After the Algorithm, all changed websites are once again checked for accessibility problems, and the sore is recorded. This means, a before and after comparision for the algorithm can be performed.

5.1. Implementation

5.1.1. Basic Algorithm Details

Umsetzung in python, BS4, ... Für umsetzung - testpage mit allen 24 Proiblemen, evtl. Screenshot? - Reihenfolge spielt eine rolle - Fixer tool ausgehend vom Checker

5.1.2. Algorithm parts

Language missing

Using bs4, the html-tag can be found. This tag can the be checked for the ”langättribute. If the attribute is not present, a paragraph with text can then be loaded into the library XXX, which returns the appropriate iso code. This code can then be added as lang-attribute to the html tag.

Region missing

How is it done?

H1 missing

How is it done?

Heading skipped

How is it done?

Heading Empty

How is it done?

Heading Possible

How is it done?

Text Small

How is it done?

Contrast

How is it done?

Link Empty

How is it done?

Link Suspicious

How is it done?

Alt Link Missing

How is it done?

Link redundant

How is it done?

Link internal broken

How is it done?

Alt missing

How is it done?

Alt suspicious

How is it done?

Alt duplicate

How is it done?

Alt redundant

How is it done?

Table layout

How is it done?

Text justified

How is it done?

Button empty

How is it done?

Label missing

How is it done?

Title redundant

How is it done?

Event handler

How is it done?

Noscript

How is it done?

5.2. Evaluation

Webcrawler

To have comparable results, the top 100 Websites in germany [3] have been crawled, using python requests. This leads to the main page saved as html. All websites which could not be crawled or resolved to an ërrorör "browser not supported"page have been removed from the dataset. Websites occuring twice in the dataset, but with different top-level-domains (for example ebay.com and ebay.de) have only been considered once. This leads to a full dataset of 75/100.

[3] https://de.semrush.com/blog/top-der-meistbesuchten-webseiten/

Tabelle 3.: Top 100 websites in Germany (citation)

Ranking	Website	Usable
1	google.com	X
2	youtube.com	X
3	facebook.com	X
4	amazon.de	X
5	wikipedia.org	X
6	bild.de	X
7	google.de	X
8	ebay.de	X
9	t-online.de	X
10	instagram.com	X
11	ebay-kleinanzeigen.de	
12	web.de	X
13	spiegel.de	X
14	tagesschau.de	X
15	focus.de	X
16	dhl.de	X
17	xhamster.com	X
18	samsung.com	X
19	twitter.com	X
20	twitch.tv	X
21	gmx.net	X
22	paypal.com	X
23	wetter.com	
24	otto.de	X
25	welt.de	
26	n-tv.de	X
27	pornhub.com	X
28	derwesten.de	X
29	merkur.de	X
30	wetteronline.de	X
31	chefkoch.de	X
32	sport1.de	X
33	idealo.de	X
34	live.com	X
35	accuweather.com	
36	fandom.com	X
37	teads.tv	X
38	chip.de	X
39	zdf.de	X
40	vodafone.de	X
41	immobilienscout24.de	X
42	ndr.de	X
43	netflix.com	X
44	mobile.de	
45	tz.de	X
46	booking.com	X
47	yahoo.com	X
48	kaufland.de	X
49	whatsapp.com	X
50	pressekompass.net	
51	ebay.com	X

To have comparable results, the checking algorithm created during this thesis is used, as well as the independent automatic test solution axe-selenium-python 2.1.6 [4] with data reprocessing to have comparable results. This checking algorithm uses the axe automated checking tool by deque systems, a leader in accessibility toolkit providers (source)[5]. These two different results are then compared.

5.3. Algorithm Limitations

Problems when using the algorithm on a real dataset, not the testpage that is under my control: - uncaught errors - directly embedded pictures (no src link) - Links not working anymore - small pictures/ wrong format - pictures that are Layout elements, not real pictures (How to handle these?)

5.4. Results

- Zeit für durchlauf auf 75 Seiten - Wieviel zeit ist pro seite einzuplanen - Wieviel änderungen brauchen oversight - Wieviel wurde pro seite verändert (im schnitt) (volle logs im Appendix) - wieviel besser ist es danach? - Wieviel ändert sich am layout der Seite? -

[4] https://pypi.org/project/axe-selenium-python/
[5] https://www.deque.com/axe/

6. Discussion

TASK: Ergebnisse betrachten

6.1. Results

- Prozentuelle verbesserung einordnen - Real World mehrwert vorhanden, weil....

6.2. Limitations

As this is an toolkit created for the purpose of a study, some assumptions are taken, and it does not include the level of human judgement needed to be used in the real world. To use this toolkit in the real world, most of the algorithms can be used to create a toolkit with a level of human supervision is added. This may be done by letting a developer approve changes, with a certain level of explanation, and then performing the changes automatically. This toolkit can then be used by developers or other users with limited technical knowledge, to transform the website to a more accessible one.

As the toolkit is designed to work automatically, not all issues can be detected an therefore fixed automatically.

- One size fits all may be complicated for an unspecified number of webpages - Has to be considered in content creation (ex. News pages)

7. Conclusion

TASK: Gesamte Arbeit zusammenfassen, Ausblick für future Work geben

- Website Accessibility has improved by XX- The script takes XX minutes per website, adding no significant burden to web developers

7.1. Summary

7.2. Outlook/Future Work

For future work, the basics of this algorithm can be used to create a toolkit that integrates developer inputs. This toolkit can then be validated with developers, to show the real-world use and benefits in comparison to manual code changes.

Literaturverzeichnis

A. Appendix

Erklärung zur Urheberschaft

Ich habe die Arbeit selbständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie alle Zitate und Übernahmen von fremden Aussagen kenntlich gemacht.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt.

Die vorgelegten Druckexemplare und die vorgelegte digitale Version sind identisch.

Von den zu § 27 Abs. 5 der Prüfungsordnung vorgesehenen Rechtsfolgen habe ich Kenntnis.

Regensburg, 23.06.2023

Unterschrift

Erklärung zur Lizenzierung und Publikation dieser Arbeit

Name: Mathias Götz

Titel der Arbeit: *Design and Evaluation of a Semi-autonomous Accessilility Tool for Web Development*

Hiermit gestatte ich die Verwendung der schriftlichen Ausarbeitung zeitlich unbegrenzt und nicht-exklusiv unter folgenden Bedingungen:

- ☐ Nur zur Bewertung dieser Arbeit
- ☐ Nur innerhalb des Lehrstuhls im Rahmen von Forschung und Lehre
- ☒ Unter einer Creative-Commons-Lizenz mit den folgenden Einschränkungen:
  - ☒ BY – Namensnennung des Autors
  - ☐ NC – Nichtkommerziell
  - ☐ SA – Share-Alike, d.h. alle Änderungen müssen unter die gleiche Lizenz gestellt werden.

(An Zitaten und Abbildungen aus fremden Quellen werden keine weiteren Rechte eingeräumt.)

Außerdem gestatte ich die Verwendung des im Rahmen dieser Arbeit erstellten Quellcodes unter folgender Lizenz:

- ☐ Nur zur Bewertung dieser Arbeit
- ☐ Nur innerhalb des Lehrstuhls im Rahmen von Forschung und Lehre
- ☐ Unter der CC-0-Lizenz (= beliebige Nutzung)
- ☒ Unter der MIT-Lizenz (= Namensnennung)
- ☐ Unter der GPLv3-Lizenz (oder neuere Versionen)

(An explizit mit einer anderen Lizenz gekennzeichneten Bibliotheken und Daten werden keine weiteren Rechte eingeräumt.)

Ich willige ein, dass der Lehrstuhl für Medieninformatik diese Arbeit – falls sie besonders gut ausfällt - auf dem Publikationsserver der Universität Regensburg veröffentlichen lässt.

Ich übertrage deshalb der Universität Regensburg das Recht, die Arbeit elektronisch zu speichern und in Datennetzen öffentlich zugänglich zu machen. Ich übertrage der Universität Regensburg ferner das Recht zur Konvertierung zum Zwecke der Langzeitarchivierung unter Beachtung der Bewahrung des Inhalts (die Originalarchivierung bleibt erhalten).

Ich erkläre außerdem, dass von mir die urheber- und lizenzrechtliche Seite (Copyright) geklärt wurde und Rechte Dritter der Publikation nicht entgegenstehen.

- ☒ Ja, für die komplette Arbeit inklusive Anhang
- ☐ Ja, für eine um vertrauliche Informationen gekürzte Variante (auf dem Datenträger beigefügt)
- ☐ Nein
- ☐ Sperrvermerk bis (Datum):

Regensburg, 23.06.2023

Unterschrift

Contents of added flash disk

`/1_Thesis`	Thesis as PDF and Latex File
`/2_Code`	Source Code for the Algorithm
`/3_Study/Design`	Source Code for the Study
`/3_Study/RAW`	Raw Data of the Study (Websites before & after etc.)
`/3_Study/CSV`	All evaluation CSV Files created dor the study
`/4_Sources`	Sources of the Thesis
`/5_Slides`	Slides of Start & End Presentations