Updating accessibility monitoring tools with sampling capabilities

Pedro Simão Pais Santos

Mestrado em Engenharia Informática
Especialização em Sistemas de Informação

Dissertação orientada por:
Prof. Doutor Carlos Alberto Pacheco dos Anjos Duarte

2022
Acknowledgments

I thank...
... my mentor Carlos Duarte for all the support given during the project, knowledge passed and specially with all the help writing this thesis
... my coworkers João Vicente and João Andrade for all the support setting up the AMA tools and knowing their intricacies
... my family for all the support and patience in the whole thesis and not letting me give up in the most rough moments
... my friend Anya, for all the support, joy and helping me write the thesis by reading and helping me improve and also for not letting me quit when things felt rough
... and my friends Cristina and Pedro, for all the support and knowledge given while writing the thesis and their opinion, which led me to improve more the thesis.
Nothing is impossible, the word itself says, ‘I’m possible!’ - Audrey Hepburn
Resumo

Desde a sua criação, a internet cresceu em popularidade, passando de uso militar para agora ser usada pela população em geral para fazer tudo - desde ler notícias até pagar contas. À medida que passou para o uso comum, o seu conteúdo passou a ser diversificado, assim como os seus utilizadores. Mas nem todo o conteúdo neste momento é aberto a todos, sendo certos utilizadores incapazes de usufruir desses conteúdos, como por exemplo pessoas cegas, surdas e pessoas com outras deficiências que inibem as suas vidas.

Com esses problemas inerentes de acessibilidade, a União Europeia criou directivas para que os seus websites mais importantes fossem acessíveis e obrigou os países a implementarem essas directivas nas suas leis e também criar uma comissão para que as mesmas fossem impostas sobre os websites de cada país. No caso de Portugal, essa comissão é a Agência para a Modernização Administrativa ou AMA. A AMA monitoriza os websites em duas fases (como a lei estipula): Uma monitorização simplificada em que são usadas ferramentas automáticas que conseguem avaliar a acessibilidade dos websites através de varios testes; E uma monitorização mais aprofundada, em que os especialistas em acessibilidade escolhem uma amostra de páginas de cada website e avaliam a acessibilidade web de cada uma separadamente.

A AMA usa bastantes ferramentas para fazer a sua monitorização de acessibilidade, entre as quais o Admin Monitor Suite (AMS). O AMS é uma ferramenta automática de monitorização de websites que permite saber muita informação sobre as páginas de um website, o que permite depois aos especialistas fazer a monitorização aprofundada de uma melhor forma com toda essa informação. Toda a informação obtida no AMS veio de uma interação com o servidor da AMA, o Monitor-Server. Este permite fazer muitas coisas entre as quais fazer crawl, avaliações através de uma ligação ao avaliador QualWeb entre outras, que ajuda imenso a ter todo o tipo de informação no AMS e noutras ferramentas da AMA que estão ligadas ao Monitor-Server.

De modo a permitir uma melhor segunda fase de avaliação de acessibilidade, foi alterada a ferramenta da AMA mais usada, o Admin Monitor Suite (AMS) e, por consequência, o servidor onde vai buscar toda a informação, o Monitor-Server, de modo a dar mais informação aos especialistas para que a sua amostragem de websites possa ser ainda melhor e mais bem apoiada. Toda essa informação adicional que é mostrada, veio
do avaliador que a ferramenta Monitor-Server usa, o QualWeb, o qual também foi alterado de modo a ter informação de tags e roles no seu relatório, de modo a poder ser utilizado para as alterações no AMS explicadas a seguir. De modo a poder ser guardada essa nova informação, também teve de se alterar a base de dados ligada ao Monitor-Server para poder acrescentar a nova informação no relatório do QualWeb. Isso foi feito através de duas novas colunas numa das tabelas da mesma de modo a poderem ser adicionadas as contagens das tags e das roles. Essas alterações no AMS incluíram o adicionar de número de elementos e número de tipos de elementos no separador da ferramenta AMS onde aparecem todas as páginas de todos os websites e as suas avaliações de modo a dar mais informação aos especialistas. Para além disso e para ajudar mais, foram adicionadas mais informações sobre cada website em específico no separador Website do AMS sobre as páginas do mesmo. Nesse separador, os especialistas podem saber quantas tabelas, checkboxes, títulos, entre outros existem (ou não) no website, e em que páginas se encontram esses elementos. Para além disso, também foi adicionado ao AMS uma simples função de amostragem que permite aos especialistas escolher partes do website como tags, e roles e retirar uma amostra de websites com esses mesmos tags e roles com o tamanho pretendido. Essa amostra será importante de modo a que os peritos consigam comparar a sua amostra e melhorá-la se acharem que a amostra escolhida tem algumas páginas que fazem sentido adicionar.

De modo a fazer essa função de amostragem ainda melhor e até uma melhor ajuda em termos de amostragem, um estudo sobre algoritmos de sampling foi feito. Os algoritmos escolhidos para esses testes foram Simple Random Sampling, Url Sampling, Stratified Sampling, HintSVM e Representative Sampling, sendo que os dois últimos não conseguiram ser testados.

Nesses algoritmos foram feitos vários tipos de testes de modo a poder ser comparada a sua performance, mas mais importante a sua eficácia em termos de encontrar amostras que cumpririam todos os critérios para uma amostra na lei. Em todos os testes foram feitas 10 mil amostras por website de modo a que os resultados fossem o mais fiáveis possível.

Esses testes consistiam num teste de performance pura, onde 10 mil amostras foram feitas e onde foi medido o tempo que demorava uma amostra a ser feita de início ao fim. Após todas as amostras terem sido feitas, seria feita uma média do tempo de todas as amostras de modo a ver a sua performance. Nesse teste específico, o melhor desempenho foi obtido pelo algoritmo Simple Random Sampling.

Depois foram feitos os testes de eficácia. Esses testes de eficácia foram distribuídos em vários grupos dependente da alínea(s) expressas na lei. Foram feitos o teste de páginas que cumpriam as alíneas a e c, que dizem respectivamente que uma amostra deve ter a página inicial, a página de início de sessão, o mapa do sítio, a página de contacto, a página de ajuda e a página de advertência jurídica e que também deve ter as páginas de declaração
ou política de acessibilidade e páginas que contêm mecanismo de retorno de informação. Estas duas alíneas foram testadas em conjunto tendo em conta que estas páginas nunca irão mudar numa sample feita e portanto fazia sentido juntar para procurar páginas que elas próprias nunca mudam. Nesse teste, o melhor na maioria dos websites foi o Url Sampling, que obteve a maior percentagem de páginas que cumpriam as alíneas a e c.

Foram feitos também dois testes para que se pudesse aferir o quão abrangente a sample era em termos de roles e tags do website em si, de modo a ver se cumpria a alínea d). Em ambos os testes, os algoritmos conseguiram ter bastantes das tags e roles do website, sendo que em todos os websites o URL Sampling foi o melhor deles, estando sempre igual ou melhor que os outros dois algoritmos em termos de percentagens de roles e tags num website.

Foi também feito um teste em que se media a percentagem de serviços numa amostra. Em todos os casos, os algoritmos não conseguiram ter grandes percentagens de serviços. O melhor dos algoritmos mesmo assim foi o URL Sampling.

Finalmente foi feito um teste em que se mediu a percentagem de amostras que tinham uma página que tivesse um ficheiro associado. Isso foi feito, comparando o fim do url e se o mesmo tinha uma extensão de um ficheiro. Nesse teste, o algoritmo URL Sampling foi o melhor, sendo que independentemente do website todas as samples tiveram um ficheiro associado.

Após todo o trabalho desenvolvido, conseguiu-se concluir que o URL Sampling é o melhor algoritmo a ser implementado nas ferramentas da AMA de modo a poder ajudar mais os especialistas no seu trabalho diário, apesar das suas falhas. Para trabalho futuro, faz sentido também corrigir as falhas do algoritmo URL Sampling através de escolha de inteligência artificial, a qual ajudaria a colmatar as falhas encontradas no mesmo.

Palavras-chave: acessibilidade web, amostragem, Agência para a Modernização Administrativa, monitorização web
Abstract

Since its creation, the internet has grown in popularity going from military use to being commonly used by the general population to do everything - from reading news to paying their bills. Some but not all content is open to all of them, leaving a lot of disabled users not able to enjoy those contents, as for example blind people, deaf people and those with other disabilities that inhibit their lives. To resolve these issues, the European Union created a directive in 2016 where all countries in the EU must monitor the accessibility of their public sector body websites by creating a national comity. AMA (the Portuguese commission) does their monitoring in two phases (as per law): simplified monitoring and in depth monitoring.

In order to help them do it in a better way, some changes were made to their tools to better translate information for more effective monitoring. Those included: the addition of an element and type count, which counts the number of elements and type of elements of a page, cards for showing all the roles and tags in a website; also some cards for certain roles and tags like tables, Iframes, images and some others. Finally, a sampling card was also added, where experts could sample a specified number of pages based on certain tags and/or elements present in the website chosen by them.

Sampling testing was also done to help the experts have even better information to compare to while they are monitoring. Simple Random Sampling, Url Sampling and Stratified sampling were tested and compared. In the majority of the testing done it was found that in performance Simple Random Sampling was faster but in accuracy it was found Url Sampling was the best in every category. Therefore, Url Sampling was the chosen algorithm, since the performance gains did not justify the lack of accuracy of Simple Random Sampling.

Keywords: web accessibility, sampling, Agência para a Modernização Administrativa, Web monitoring
# Contents

List of Figures xiv
List of Tables xv
List of Listings xvii

1 Introduction 1
  1.1 Context 1
  1.2 Motivation 2
  1.3 Objectives 2
  1.4 Contributions 2
  1.5 Document Structure 3

2 Background 5
  2.1 Admin Monitor Suite (AMS) 5
  2.2 Monitor-Server 6
  2.3 QualWeb 8

3 Related Work 15
  3.1 Accessibility 15
    3.1.1 Automatic Evaluation 15
    3.1.2 Manual Evaluation 19
    3.1.3 Monitoring of Accessibility 20
  3.2 Sampling 21
    3.2.1 Sampling Algorithms 21
    3.2.2 Use of Sampling on Accessibility 23

4 Updating AMA Tools 25
  4.1 Requirements 25
  4.2 QualWeb 25
  4.3 Monitor-Server 27
  4.4 AMS 28
5 Sampling

5.1 Implementation ................................................. 33
  5.1.1 Justification of language and IDE .................. 33
  5.1.2 Data Treatment for testing ............................ 33
  5.1.3 Algorithm Choice and implementation explanation .. 35
5.2 Results .......................................................... 38
5.3 Discussion ..................................................... 42
  5.3.1 Performance tests ....................................... 43
  5.3.2 Accuracy tests ........................................... 44
  5.3.3 Conclusions of Testing ................................. 45

6 Conclusions ....................................................... 47

Bibliography ....................................................... 50
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>User Tab of AMS</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>Website tab of AMS</td>
<td>6</td>
</tr>
<tr>
<td>2.3</td>
<td>AMA Tools architecture</td>
<td>7</td>
</tr>
<tr>
<td>2.4</td>
<td>Monitor Server database</td>
<td>8</td>
</tr>
<tr>
<td>2.5</td>
<td>QualWeb Architecture</td>
<td>9</td>
</tr>
<tr>
<td>2.6</td>
<td>Example of a EARL Report file</td>
<td>10</td>
</tr>
<tr>
<td>3.1</td>
<td>QualWeb UI</td>
<td>16</td>
</tr>
<tr>
<td>3.2</td>
<td>QualWeb Results UI</td>
<td>17</td>
</tr>
<tr>
<td>3.3</td>
<td>Tenon.Io UI</td>
<td>18</td>
</tr>
<tr>
<td>3.4</td>
<td>Tenon.Io Results UI</td>
<td>18</td>
</tr>
<tr>
<td>3.5</td>
<td>Tenon.Io Result File</td>
<td>19</td>
</tr>
<tr>
<td>3.6</td>
<td>A picture of a Red Apple</td>
<td>19</td>
</tr>
<tr>
<td>3.7</td>
<td>A picture of a Red Apple falling into water</td>
<td>19</td>
</tr>
<tr>
<td>4.1</td>
<td>New QualWeb Architecture</td>
<td>26</td>
</tr>
<tr>
<td>4.2</td>
<td>Counter module code</td>
<td>27</td>
</tr>
<tr>
<td>4.3</td>
<td>AMS Page Tab with changes</td>
<td>28</td>
</tr>
<tr>
<td>4.4</td>
<td>Roles and tags card</td>
<td>29</td>
</tr>
<tr>
<td>4.5</td>
<td>Image and Iframe categorization card</td>
<td>29</td>
</tr>
<tr>
<td>4.6</td>
<td>Sampling card</td>
<td>30</td>
</tr>
<tr>
<td>5.1</td>
<td>Query from database</td>
<td>34</td>
</tr>
<tr>
<td>5.2</td>
<td>File after the query was done</td>
<td>34</td>
</tr>
<tr>
<td>5.3</td>
<td>Data treatment code and final result</td>
<td>35</td>
</tr>
<tr>
<td>5.4</td>
<td>Simple Random Sampling code</td>
<td>36</td>
</tr>
<tr>
<td>5.5</td>
<td>Url Sampling code</td>
<td>37</td>
</tr>
<tr>
<td>5.6</td>
<td>Stratified Sampling code</td>
<td>38</td>
</tr>
<tr>
<td>5.7</td>
<td>Performance test results for algorithms</td>
<td>40</td>
</tr>
<tr>
<td>5.8</td>
<td>Sampling Tag accuracy test results</td>
<td>40</td>
</tr>
<tr>
<td>5.9</td>
<td>Sampling Role accuracy test</td>
<td>41</td>
</tr>
<tr>
<td>5.11</td>
<td>Sampling service page accuracy test by service</td>
<td>41</td>
</tr>
<tr>
<td>5.10</td>
<td>Sampling service page accuracy test by page</td>
<td>42</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>5.12 Sampling Mandatory Pages accuracy test</td>
<td>42</td>
<td></td>
</tr>
<tr>
<td>5.13 Sampling pages with file accuracy test</td>
<td>43</td>
<td></td>
</tr>
</tbody>
</table>
List of Tables

3.1 Summary table showing details about the sampling algorithms . . . . . . 24
Listings

2.1 Example of a non EARL Report file .................................................. 11
4.1 Pseudocode of Sampling Card .......................................................... 30
Chapter 1

Introduction

1.1 Context

Since its creation, the internet has grown in popularity going from military use to now being commonly used by the general population to do everything - from reading the news to paying their bills. As it transitioned to common use, its content started to be diversified, like its users. Some but not all content is open to all of them, leaving a lot of disabled users not able to enjoy it, for example, blind people, deaf people, and those with other disabilities that inhibit their lives.

In Europe, web accessibility finally became a pressing concern. To resolve this, the European Union created a directive in 2016 [3] that makes all countries in the EU monitor the accessibility of their public sector body websites by a comity created by each of them.

In the following years, the EU deliberated about the approved directive and wanted to build on it. With that in mind, they then issued the European Accessibility Act (EAA) [4] that is only to be active in 2025, in a quest to make most websites accessible to the general public, adding e-commerce accessibility monitoring to the already existing ‘public sector body’ accessibility monitoring.

As said before, every EU country has its own commission that monitors the accessibility of government websites. The results of that monitoring are then reported back to the EU. The national monitoring commission in the case of Portugal is the Agência para a Modernização Administrativa (AMA). AMA has some accessibility experts that do the monitoring. In their monitoring activities, they are supported by some tools. In that group of tools used by the AMA are Admin Monitor Suite (AMS for short), MyMonitor, Observatório Português da Acessibilidade Web, and Monitor-Server which will be described in more detail later.

All of the tools mentioned, and more used by AMA, have been improved in collaboration with Faculdade de Ciencias da Universidade de Lisboa (FCUL), to make them better and easier to use by the AMA experts. All of the results of those evaluations are sent to the EU, but the AMA also has an accessibility website, called Observatório Português
da Acessibilidade Web[^1] where these results mentioned earlier can be checked out by the general public. The above-mentioned website uses the results of the AMS tool under the hood of the website to show them. The AMS tool uses the results stored in Monitor Server, which are calculated by FCUL’s QualWeb accessibility evaluation engine (the program responsible for evaluating each website accessibility score) which is always being improved as the tools the AMA uses.

### 1.2 Motivation

As mentioned before, the AMA monitors website accessibility in Portugal. Their monitoring is done in two phases (as per law): a simplified monitoring phase with the use of automatic tools that assess the accessibility of the websites themselves and give the results to the experts; and an in-depth monitoring phase where the experts sample a select number of websites and then choose a sample of pages of each website to monitor their accessibility manually. The last sampling process is where we try to help them by adding new information before the sampling process and helping them with that. Therefore, the motivation for this project is to help out Portugal’s accessibility comity AMA to better sample the pages for the best accessibility evaluation, by making accessibility sampling better and fixing its flaws.

### 1.3 Objectives

As mentioned before the AMA uses a variety of tools to monitor accessibility. This project is intended to respond to the new challenges that are presented to accessibility tools today and therefore make AMA have the best tools possible for the future. With that said, the project sought to focus on the following aspects:

- Change the AMA tools, adding the elements’ role and tag count, to help in the sampling process
- Finding an algorithm or creating one to make automated or assisted sampling possible

### 1.4 Contributions

As a result of this project there are the following contributions:

- A new QualWeb module that counts all the roles and tags of elements on a web page.

[^1]: https://observatorio.acessibilidade.gov.pt
• New functionalities on the AMA accessibility evaluation and management software:
  – A role and tag count
  – A card with all the information of a page
  – A card where sampling can be done

• Sampling algorithm testing so the algorithm can be used in AMA tools to help experts.

1.5 Document Structure

This document is divided into the following chapters:

• Chapter 1, (the current one) that introduces accessibility, the motivation of the work, objectives, and the document structure.

• Chapter 2, where previous work done related to the project is talked about and discussed.

• Chapter 3, where we explain a bit of what is done in accessibility evaluation.

• Chapter 4, where all updates done to the current automatic tools used by the AMA are explained.

• Chapter 5, where everything about the implementation of the sampling algorithms chosen is explained and the results it had on accessibility assessment are shown and discussed.

• Chapter 6, where we draw the final conclusions.
AMA, as mentioned earlier, does the accessibility monitoring for Portuguese public sector body’s websites. They use a couple of tools to assess web accessibility: AMS, Access Monitor, My Monitor, Observatorio, and Study Monitor.

Access Monitor lets anyone evaluate a website by inputting a website URL, HTML code (via file or pasting it in), and getting the results out for that website or code accessibility-wise. My Monitor allows experts to monitor an entire website while using it and do various accessibility tests on it. Observatorio is a website where there are statistics about the evaluated websites inputted by experts and where everyone can see all the results of the accessibility tests done by AMA on the pages chosen by the experts and which usability and accessibility stamp was given by the AMA to the specific website. Study Monitor makes it possible to use the techniques used by Access Monitor to make sectorial studies of Web accessibility.

We will explain next, in greater detail, the two main tools changed as part of this work: Admin Monitor Suite (AMS) and Monitor-Server.

2.1 Admin Monitor Suite (AMS)

Admin Monitor Suite, or AMS for short, is an important AMA tool that allows its experts to do some automatic testing on websites and get all of the accessibility information about that website. It was created with the Angular framework and uses the TypeScript language to generate an accessible and user-friendly UI for the experts to use in their accessibility monitoring activities.

AMS lets the AMA experts do a lot of things, from adding monitoring data to the Observatório da Acessibilidade Web to crawling a full website. The first tab that AMS has is to show the stats of the database and manage all of the stats for the Observatório da Acessibilidade Web. Another function of AMS is to manage users. On this page (figure 2.1) they can add users or remove users and edit permissions on this page.

To monitor accessibility, the experts first have to go to the website page to define the
website domain in AMS (figure 2.2). On the website page there is a place where besides choosing the website domain, we could request a crawl based on the website domain choosing the crawl depth. The website page also shows all the stats of the website, being able to see each page’s stats in greater detail from there linking to the page tab.

If the experts did not do the crawl on the website domain page, they can add the pages associated with that specific domain to be evaluated on any page by clicking the plus button and clicking to add a page.

When it is added to the evaluation queue, experts can see what is the state of the evaluation page and the evaluation score if the evaluation is done. As the experts ask for a page, AMS sends a page evaluation request to Monitor-Server that will process the request and then send the information to AMS, as will be explained better later.

Besides the aforementioned features, the experts can also crawl a website domain without needing to evaluate it, with the same parameters as the ones mentioned earlier when mentioning the website domain page. After the crawl is done by the experts, they can then select pages in the page tab to evaluate.

### 2.2 Monitor-Server

Monitor-Server is the backbone of all of the AMA tools. It is the server that does all of the work behind the scenes. It is composed of some controllers, modules, a router, a library, and middleware (figure 2.3).
When anyone uses one of the AMA tools mentioned earlier it sends an HTTP request to Monitor Server that receives it on the router. The router then sends it to a controller. Depending on the request sent it will go to a specific controller. For example, if there is a request related to the Observatório it will go to the observatório controller. Then those controllers will send it to the respective module. Then the module will do its operations to respond to the request received. To complete the operation requested, the module could request data from the database, for example, the results of the websites already in the database; or send a request to the QualWeb evaluator through a layer of middleware. When that request is sent it then asks QualWeb to evaluate and send the results of a specific evaluation in a JSON File. That file is then sent to the module that requested it but also writes those new evaluation results to the database to be later read if needed.

Depending on which module is contacted for a request to access the database, a different table on the database will be accessed. The database consists of 9 main tables.
Chapter 2. Background

(figure 2.4): CrawlDomain, Directory, Domain, Entity, Evaluation, Page, Tag, User, and Website. Each of these has a specific purpose and responds to specific requests from the AMA tools. For example, if in AMS we select the page tab, a request to get the information on the page table on the database is done and all the information of that table is then treated on AMS to fill the table on the page tab. Some of the mentioned tables have connections, for example, an evaluation of a certain page of a website cannot happen if there is no page on the page table with their URL. For that to happen the website table firstly has to have the domain URL and a website name in its table to allow pages from that domain to be inserted into the pages table to then be evaluated.

![Monitor Server database diagram]

Figure 2.4: Monitor Server database

2.3 QualWeb

QualWeb is an evaluator of web accessibility implemented in JavaScript using the NodeJS framework. QualWeb consists of several modules, with almost all of them related to the
evaluation process (figure 2.5). They interact with one another when needed to give the evaluation wanted by the user.

![QualWeb Architecture Diagram]

**Figure 2.5: QualWeb Architecture**

The modules include implementations of WCAG, ACT, Best Practices (BP), HTML and CSS techniques, a command-line interface, or CLI for short. The CLI allows you to do evaluations from the command line of your PC, choosing what you want to evaluate, as well as output converters to the Evaluation and Report Language (EARL). This XML-based format, created by the W3C, facilitates processing by programs and promotes the use of the results by other programs that also use it, in addition to making it possible to compare results from programs that perform the same tasks (figure 2.6).

WCAG Techniques are a series of W3C standards that govern the accessibility of a website. If these criteria are not met the website will be deemed by experts as not accessible.

ACT Rules are a series of rules that can be transposed from the W3C standards like WCAG. They were created to make the W3C standards mostly assessable automatically and manually.

Best Practices is a module that evaluates a website based on good accessibility prac-
Chapter 2. Background

Figure 2.6: Example of a EARL Report file

tices. If a website fails some of the best practices here, it's not a major issue compared to WCAG or ACT, the website just does not use the best accessibility practices to make the website more accessible.

The operation of this version of QualWeb is schematized in figure 2.5. To evaluate in the CLI, the user executes a command in which he chooses: the URL to evaluate, which technique packages to evaluate (WCAG, ACT, BP), and the format of the evaluation output. In the website version of QualWeb, you put the URL on the URL bar and choose the options on the website. After that, the URL is processed by the 'Dom' module, where the corresponding page is "visited" using a 'Phantom' headless browser, and a structure is built which represents the content of the web page. This structure contains the pre and post HTML processed. Next, the structure is sent to the evaluation module that then interacts with the chosen evaluation modules where the website is evaluated by the several rules of each module. Finally, the content of the evaluations is put together in a report and sent to the user in the chosen format. Either of the reports (EARL formatting or not) will be in JSON format. Those JSON reports will have the page information from all of the page code to its URL, the information about the number of tests done, how many passed and failed, where the tests were not failed but not passed: the warnings, and the number of tests that were not applicable on the page. On top of that, there is all the information
on all the tests done by module chosen and by test in the module as shown in listing 2.1.

Listing 2.1: Example of a non EARL Report file

```json
{
  "type": "evaluation",
  "system": {
    "name": "QualWeb",
    "description": "QualWeb is an automatic accessibility evaluator for webpages.",
    "version": "3.0.0",
    "homepage": "http://www.qualweb.di.fc.ul.pt/",
    "date": "2022-03-01 15:55:09",
    "hash": "cf840a393ba0",
    "url": {
      "inputUrl": "https://act-rules.github.io/pages/about/",
      "protocol": "https",
      "domainName": "act-rules.github.io",
      "domain": "io",
      "uri": "/pages/about/",
      "completeUrl": "https://act-rules.github.io/pages/about/"
    },
    "page": {
      "viewport": {
        "mobile": false,
        "landscape": true,
        "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/91.0.4469.0 Safari/537.36",
        "resolution": {
          "width": 1366,
          "height": 768
        }
      },
      "dom": {
        "html": "(codigoHtmlDapagina)",
        "title": "About Us | ACT-Rules Community",
        "elementCount": 181
      }
    },
    "metadata": {
      "passed": 35,
      "warning": 11,
      "failed": 2,
      "inapplicable": 70
    }
  }
}
```
Chapter 2. Background

"modules": {
    "act-rules": {
        "type": "act-rules",
        "metadata": {
            "passed": 22,
            "warning": 7,
            "failed": 1,
            "inapplicable": 42
        },
        "assertions": {
            "QW-ACT-R72": {
                "name": "First focusable element is link to non-repeated content",
                "code": "QW-ACT-R72",
                "mapping": "8a213c",
                "description": "This rule checks that the first focusable element is a link to non-repeated content in the page",
                "metadata": {
                    "target": {
                        "element": "*"
                    },
                    "success-criteria": [],
                    "related": [],
                    "url": "https://act-rules.github.io/rules/8a213c",
                    "passed": 0,
                    "warning": 0,
                    "failed": 1,
                    "inapplicable": 0,
                    "outcome": "failed",
                    "description": "The first focusable element does not skip to the main content"
                },
                "results": [
                    {
                        "verdict": "failed",
                        "description": "The first focusable element does not skip to the main content",
                        "resultCode": "RC5",
                        "elements": [
                            {
                                "htmlCode": "<a href="/pages/about">"</a>",
                                "pointer": "html > body:nth-child(2) >""
                            }
                        ]
                    }
                ]
            }
        }
    }
}
Chapter 3

Related Work

In this chapter, we will talk about previous work done that is related to web accessibility and sampling.

3.1 Accessibility

Web accessibility is the inclusive practice of making a website as barrier-free as possible to all users, disabled or not. For a website to be accessible, the World Wide Web Consortium, or W3C, a consortium that manages how the contents of the internet are shown since the dawn of the internet, issued a set of guidelines for websites to become accessible and also created 3 levels of conformance to the guidelines (WCAG or Web Content Accessibility Guidelines). Those tell the user how accessible the website is: if the site is level A conformant, it is accessible but still lacks in accessibility terms; if it is level AA, it is more accessible than A but there is still room for improvement; if it is level AAA it is accessible or close to being fully accessible. All those guidelines regulate web accessibility.

Some of the monitoring of those guidelines depends a lot on the context, which does not allow them to be checked automatically. With that said there are a lot of automated tools (QualWeb and Tenon.io ) that can test your website for partial conformance to W3C standards. Those tools can also help in the monitoring of web accessibility by the EU countries’ committees.

The monitoring of web accessibility is done with three techniques: automatic, manual, and user evaluation, well explained in [12]. The first two will be explained in greater detail in the next two subsections.

3.1.1 Automatic Evaluation

The automatic evaluation is done through automatic tools [6], which evaluate the accessibility of a website, such as QualWeb or Tenon.io. Since they have problems in being able to fully evaluate the accessibility, there is a manual evaluation made by specialists so that
we can have a more accurate assessment of the accessibility of a website [13] (explained in the next subsection in more detail). If necessary, user evaluation is also used, which is done by testing the website accessibility with users with disabilities to know their opinion about it and thus the experts can reach a conclusion on whether a website is accessible or not [13].

Companies want to make a website accessible in an efficient manner, minimizing specialized personnel [13]. Automatic tools are increasingly used, and they try to solve the problem mentioned above. With this in mind, the tools needed to make the assessment as efficient as possible. These tools not only faced the challenge of knowing which models to use (seeing as page based is only viable to small websites, as bigger websites take longer and have a need to analyze page by page), but also the computational resources needed for it, especially in the faster page based models. If you choose website-based, it brings the challenge mentioned earlier, which is to choose the best sampling method that does the best choices without leaving out key pieces of the website to be checked by the tool.

As mentioned at the start of the section, we are going to explain some accessibility evaluation testing tools. Automatic tools work like this: The user gives a URL of a website to either an application/extension or a website and that communicates to a server. That server gets the URL, gets all the DOM tree information of the website, and creates a "copy" on the server that then is evaluated by the evaluator that the server has and then sends the results to the website or the application/extension that then either nicely shows them or gives the user a file with all accessibility results.

QualWeb works similarly. In the web version, it uses a UI (shown in Figure 3.1) that lets you choose from two separate QualWeb modules (that can be picked both at once as well): Accessibly Conformance Testing rules, or ACT rules for short and Web Content Accessibility Guidelines Techniques or WCAG Techniques for short.
The module ACT Rules is selected by default and it tells the QualWeb evaluator that is within the server that you want QualWeb to evaluate according to the ACT rules. The ACT rules are a model in which the W3C accessibility recommendations are turned into objective accessibility rules, some of which can be automatically checked. The rules are checked by the whole community and then approved to be used in automated tools like QualWeb.

There is also the WCAG techniques module for QualWeb which makes the QualWeb evaluator check for the Web Content Accessibility Guidelines (WCAG) techniques, which are used for checking conformance with the WCAG success criteria. The WCAG techniques are a series of guidelines created by the Web Accessibility Initiative (WAI) of W3C that web content creators use when trying to create an accessible website. If both modules are chosen it will compare the website with both criteria.

After the user chooses the sets of rules that will be used for QualWeb to evaluate the pages of the website chosen and pressing the Check button, the QualWeb client sends the request to the QualWeb evaluator server. The server not only creates the website “copy” (as mentioned earlier) but also evaluates that specific website by comparing the “copy” with the rules specified by the user and sending the results back to the client UI. The client UI will then use all of that data and make it readable for the user, giving them the number of rules that the website has passed on, the number of rules that failed, and the number of rules that the QualWeb evaluator couldn’t evaluate (mostly context-driven rules) shown as warnings for the user to check. It also shows the rules that do not apply. The UI with the results (Figure 3.2) also gives the user the chance to filter by the outcome, type, Accessibility principle, and W3C conformance level.

Tenon.io is another accessibility testing tool. Tenon.io has a different UI (Figure 3.3) to QualWeb but has also some other differences. In Tenon.io not only can a website be
assessed, but also HTML line(s) of code. The user inputs the website URL or the HTML line(s) that they want Tenon.Io to evaluate. Then it sends the code or the website DOM tree to Tenon’s evaluator that will evaluate it according to WCAG, not giving the user the chance to choose like QualWeb.

![Figure 3.3: Tenon.Io UI](image)

After the evaluation is done, Tenon’s Server gives it to its UI (Figure 3.4) that shows in a pie graph each type of error with the percentages of errors of that type, a list with all the errors, giving an example of the mistake, a link to a web page that explains the error in greater detail and also how important it is to solve by showing a priority gauge that goes from 0 to 100 percent.

![Figure 3.4: Tenon.Io Results UI](image)

It also has a file that the user can download, with all the errors with detailed information as shown in Figure 3.5 (image presents the tenon.io CSV transformed into a table in MS Excel)
3.1.2 Manual Evaluation

As mentioned at the start of the section, we are going to explain how Manual Evaluation works. Manual Evaluation is a step in accessibility evaluation. It happens since automated tools cannot say if certain accessibility guidelines are followed, mostly guidelines related to context. An automated tool can say if there is an alt text for a screen reader to read, but it cannot say if the alt text is conformant to what is in the picture. For example, if there is a picture of a red apple (Figure 3.6) and a picture of a red apple that drops in water (Figure 3.7), if both have the alt text of “red apple”, the automated tool says that the picture has an alt text but that does not mean the picture’s alt text is right. It is for that specific reason there is manual evaluation.

Figure 3.6: A picture of a Red Apple

Figure 3.7: A picture of a Red Apple falling into water

Manual evaluation is a type of accessibility evaluation that is done by accessibility experts (a person who has a lot of experience in evaluating website accessibility). A website is given to the experts so they can analyze it via numerous tests. Here are some of the most important tests done by experts to ensure that a website is accessible:
• Evaluation of a website with a screen-reader and checking all of the alt texts are according to the picture that is shown on screen to give a blind person an experience as close as a sighted person would have.

• Evaluation of a website’s traversability. A website is traversable when all of the website menus and content are accessible via pressing the tab key.

• Testing a website’s experience ensuring that the experience of a website is the same when the website has no sound, by turning off the browser’s sound.

As shown above, manual evaluation is crucial for accessibility evaluation, hence why the 2016 EU directive [3] says that for each member state the comity to ensure that the website of public entities and state websites are fully accessible must use automatic and manual evaluation methodologies.

3.1.3 Monitoring of Accessibility

The monitoring of accessibility is done as said before with these three techniques: automatic evaluation, manual evaluation, and user evaluation. Depending on the situation and the problem to address, a different technique is used, either dependent on the costs or even what is the objective of the monitoring process. In the case of a monitoring entity, the EU law states that there are two types of monitoring: automatic by using a tool on some web pages and manual evaluation by experts. With that said the Portuguese approved law for accessibility monitoring has also added some evaluation with users with disabilities to monitor all accessibility. The AMA’s monitoring firstly uses a manual selection of websites from the required ones, since it would be impossible to select all websites to monitor. Monitoring checks a sample of pages of some of those websites with automatic tools in two ways: the first checks the homepage and all pages linked from the home page; the second checks up to 2000 web pages of the website collected in a breadth-first manner. Then a sample of those websites is selected for manual evaluation. For these websites, they sample an even smaller number of pages, ranging from 20 to 30 pages (unless the website is too small) since it would be impossible to evaluate every page of a website by hand, with some websites having a lot of pages. They use the following criteria to sample the pages of a website for manual evaluation [4] (if the pages in the criteria exist):

a) The sample has to have the homepage, login page, website map, contact page, help page, and copyright page

b) The sample has to have 1 pertinent page at least for each service that the website provides, including the search service

c) It has to have the accessibility declaration or accessibility policy and pages that have an information return mechanism.
d) Example of pages that have a different appearance or that have a different type of content

e) At least one downloadable file that is pertinent or at least a file for each website service, if that is applicable

f) Any other pages that the monitoring experts think are needed

After the sample is done according to the criteria specified above, experts have to add a further 10% of pages to those already sampled to complete the above sampling process.

Companies, on the other hand, want the most accessible website they can have at the lowest cost. Since they do not need to have manual monitoring they use some automatic tools to monitor their websites and make them more accessible.

### 3.2 Sampling

Sampling, as mentioned above, starts to be super important to make automatic tools that can be efficient and effective in their evaluation while still having a good analysis speed. The big problem with several sampling methods is knowing which pages of a website should be sampled to be able to evaluate correctly without seeing the entire website. That specific evaluation should have roughly the same results as a page-based full scan, so as not to leave important details of the website behind as mentioned earlier. Not only that but sampling is also really useful in manual evaluation as mentioned in the previous chapter.

In the next subsections, we are going to show how some sampling algorithms work and how sampling is used in accessibility currently.

#### 3.2.1 Sampling Algorithms

There is a multitude of Sampling Algorithms of all shapes and sizes, that are used in a lot of programs and that could be useful for accessibility purposes. There are several groups where they can be categorized according to their respective sampling method. They are divided into probability and non-probability sampling. The main difference between both of them is the fact that non-probabilistic will have a predefined sample that will not change, whereas probabilistic sampling depends on the probability and has an ever-changing sample.

Simple Random Sampling (SRS) chooses a number of random items in a pool given by the user and gives the results, either in file or UI depending on where it is used. This algorithm is mostly used for scenarios where choosing something at random is needed, like for example random number generators, where we choose a number between a certain range specified by a user, in dice number choice in online games like Dungeons
and Dragons and other games where dice is used, in physics calculations in games, to choose participants in samples for testing and even in other programs and algorithms.

Representative Sampling (RS) [10] samples by choosing the most representative items of a sample given to the user in a greedy first search way. That means that the algorithm works by choosing items of a pool in a way that the algorithm thinks is representative of the corresponding pool at that moment. For example, if a sample is formed by 10 girls and 10 boys and we want to sample 4 people. To make a representative sample by the algorithm’s standards we had to sample something like 2 girls and 2 boys since the original pool has a 50/50 split of girls and boys. It is used in situations where representativeness matters, like scientific and marketing research in the choice of groups.

URL Sampling (US) [15] uses a technique called Clustering [9, 8, 7], that groups the various pages according to a specific parameter. URL Sampling groups the web pages given to it by a crawler, by URL link of those web pages and takes samples from the various groups so that their accessibility is then analyzed.

Stratified Sampling (StS) [5] samples by dividing the population into sub populations based on certain criteria. It is similar to representative sampling, but differently from RS, the criteria for the search can be chosen by the user, whereas in RS it’s chosen on the fly by the algorithm. For example, let us say that there is an election and we want to sample a certain number of votes of a town by age of the voter to do a poll just as the voting has ended. What the algorithm does is firstly divide the population into strata. Afterwards, it samples based on the percentages that each stratum has on the population. For this example, let’s say we have 18-25 age voters as 25 percent, 33 percent for 26-40, and the rest as 41-60. With that said if we sample 100 votes from this town, the algorithm would choose 25 voters from the 18-25 range, 33 from the 25-40 range, and 42 voters from the 40-60 range. As shown in the example this sampling algorithm samples based on the importance of the sampling group on the whole pool as said before.

Systematic Sampling(SyS) [5] is a sampling algorithm where items are chosen at regular intervals in the pool. For example, if we have a pool of 1000 numbers and we want to choose 10 numbers, this algorithm will choose the items that are 1000/10 spaces apart from each other. So in this case, the algorithm would choose (assuming we start on the first person and it’s not counted towards the sample) the numbers 100, 200, 300, 400, 500, 600, 700, 800, 900, and 1000. This is used to select groups of people for research where there is a low risk of data manipulation over random sampling.

Snowball sampling (SnS) [5] samples based on the passing of information/recruitment of participants. Let us say we want to research shoplifting and want 50 participants. Not many people (if any) will want to be associated with such research. In that situation, we talk to a shoplifter, telling him that he will remain anonymous and will be compensated for participating and would receive an amount for every other person he recruits. And this sampling works in this way. When we get the 50 people wanted for the study it stops.
This algorithm is used in the research of subjects where they might not attract a lot of participants.

The Hint SVM sampling (HS) \[11\] method evaluates the accessibility of websites by clustering the pages using "hints" to be able to group them more accurately. The algorithm uses support vector machines to divide the cluster of pages based on the "hints" given to it from the outside and will make the best groups possible with the "hints" given. This algorithm has an issue that is if the "hints" given are not the best it might not perform as well in terms of grouping, but if using the right "hints" it might solve the problem in the best way possible. Zhi Yu et al \[14\] uses this same algorithm to great effect on accessibility evaluation by making the groups based on the informativeness of each page, which is an amazing “hint” to use.

Table 3.1 summarizes the information of all the sampling methods presented.

### 3.2.2 Use of Sampling on Accessibility

Manual sampling has been a big part of accessibility monitoring for years since it's resource-consuming to monitor every website and every page of each website. With that said there have been recent works that are trying to help research a better way to do it and the creation of algorithms that are suited to this monitoring.

Zhi \[14\] has several contributions to the research in this area, based on the work of Chun-Liang Li, Chun-Sung Ferng, and Hsuan-Tien Lin \[11\] where they developed an algorithm based on hints: Hint Support Vector Machine. That algorithm would use hints and a hint line to try and sample as close as what the user wanted to sample using support vector machines. Support vector machines allow you to create vectors that define a space the algorithm would use to see what the wanted samples are.

Zhi \[14\] then uses the aforementioned algorithm to see if it can be used effectively in accessibility monitoring with the use of active learning to make the hints more accurate. After some testing that Hint SVM with Active learning could be extremely accurate and that active learning could make not only Hint SVM better but also other algorithms like representative sampling and uncertainty sampling (to a lesser extent)

Zhang \[15\] has also had some contributions with sampling with Url Clustering in accessibility. Zhang explored how the granularity of the URL clustering would affect the precision of the groups. He then used the groups done in clustering to sample some web pages from the aforementioned groups and compared them with uniform random sampling. The results showed that URLSamp was superior to uniform random sampling.

Besides these contributions, there is also Harper’s research in Clustering by Domain tree to try and make sampling better and also to see the impacts of Domain tree clustering in accessibility sampling. Harper uses this to see how the DOM tree clustering algorithm will fair in finding pages based on some chosen criteria and how it would work in different types of website categories.
<table>
<thead>
<tr>
<th>Sampling Algorithm</th>
<th>Description</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Random Sampling (SRS)</td>
<td>A sampling algorithm based on random choice</td>
<td>Fast algorithm</td>
<td>-No choice on which websites are chosen</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-No way to know how the websites are chosen</td>
</tr>
<tr>
<td>Represent. Sampling (RS)</td>
<td>A sampling algorithm based on the significance of pages</td>
<td>-We know how the sampling algorithm chooses</td>
<td>The choice of the representative web pages for the algorithm might not have key web pages to be analyzed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-We choose web pages by representativeness</td>
<td></td>
</tr>
<tr>
<td>URL Sampling (US)</td>
<td>A sampling algorithm based on sampling by URL</td>
<td>Fast algorithm</td>
<td>The group creation is not ideal, might miss some key elements since group creation is done by URL</td>
</tr>
<tr>
<td>Systematic Sampling (SyS)</td>
<td>A sampling algorithm based on intervals</td>
<td>Fast algorithm</td>
<td>The group creation is not ideal, the group chosen might not have key elements, and it might be always the same if the starting point is not changed.</td>
</tr>
<tr>
<td>Snowball Sampling (SnS)</td>
<td>A sampling algorithm based on networking</td>
<td>We can get people for a sample based on the characteristics wanted/needed</td>
<td>There could be biased sampling</td>
</tr>
<tr>
<td>Hint SVM (HS)</td>
<td>A sampling algorithm based on hints</td>
<td>Good group creation</td>
<td>Time needed to sample</td>
</tr>
<tr>
<td>Stratified Sampling (StS)</td>
<td>A sampling algorithm based on the importance of each group of websites</td>
<td>We can choose the groups by defining the importance of each group of pages</td>
<td>If some pages are in 2 or more groups it might miss some crucial pages.</td>
</tr>
</tbody>
</table>

Table 3.1: Summary table showing details about the sampling algorithms
Chapter 4

Updating AMA Tools

As mentioned earlier, one of the goals of this thesis was to change the AMA tools to help experts have an easier monitoring process. In this chapter, we are going to discuss all the changes made to the AMA tools: AMS, and Monitor Server and also to QualWeb to allow meeting the requested changes.

4.1 Requirements

After several conversations with the experts about what their needs are to do a better sampling, the following requirements were set:

- Show how many different roles and tags exist on a website
- Show what types of roles and tags a website and each page has
- Show what pages in a website have video and audio
- Show how many pages in a website have tables, forms, iframes, headers, buttons, checkboxes, and elements with role presentation
- Make all this new information available in a CSV/excel file
- Add a feature that can sample pages by tags/roles chosen by experts

4.2 QualWeb

To add the required information to AMS and Monitor-Server, a change in the architecture of QualWeb had to be made to accommodate the information needed and for the information to be collected in the report asked by Monitor-Server. Instead of adding the new information to any of the existing modules, it was chosen to create a new module called Counter (figure 4.1), since it did not fit in any of the other modules to add this information.
For the module to be added, it was needed to do some changes to the evaluation module, since it is the module that controls the tests that are done by an evaluation. Not only that but it was needed to add the module to the CLI module so it can be called from the command line in case someone else also needs the information in the report.

The Counter module adds the type of elements present on the website: roles and tags as well as the number of each of them to the report. To have that information we go through all of the elements on the website categorizing them and then we use a function called getElementRole for getting the role of the element in the HTML code and getElementTagName for the tags of the HTML elements. As we get the role and tags of the elements we either add a new role or tag to the role/tag list if it is not already there, where it stores a new element in the list with count 1 or we add 1 to the already existing count to the list (figure 4.2).

After that is done, we then send all of the results back to the evaluation module that then creates the evaluation report, which is then processed in AMS or sent to the person that asked for the report.
Figure 4.2: Counter module code

4.3 Monitor-Server

To accommodate the new data from the QualWeb report, changes had to be made to the database, so that the new data could be then usable by AMS for the requirements mentioned earlier.

To add the new information, the evaluation table was changed, since it is the one that is responsible for the data used in the changes in AMS. A field Element Count and a field Role Count were added to the evaluation table. Those fields store, for each page evaluated, the list of tags and roles respectively and their counts from the QualWeb report in JSON format.

Code was also added to deal with the new requests for the new information. When AMS requested the new information it would do a query to the database and get also the element and role counts that then would be treated in AMS.
4.4 AMS

After meetings with the experts, they asked to add some more important information relative to the websites on the AMS tool itself to help them make better decisions. With that information gathered, it was first added a total elements counter and a counter of how many different types of elements existed to the UI of AMS for the experts to get a better understanding of which pages are more relevant to do the manual sampling of the website in question. The total elements counter uses the element counter stored in the database (that has all of the tags and their counts) to sum all the counts of each tag to get to the total that is then shown on the page. To count the types inside of the page to get the type counter, it first turns the element count row into a list that has all of the types of tags on the page and their counts (used to get the total elements count). As each list element is a different type of element, it is counted the number of elements on this list to get the types of elements on the page (figure 4.3).

Figure 4.3: AMS Page Tab with changes

After that, it was also added the new statistics regarding roles and tags, where all of the roles and tags are shown describing how many of each role and tag there are on the page and the total types of roles and tags. AMS gets tags and roles from the database and then processes that data to show each of them. As for the total number of elements, the same process used in the aforementioned Page tab is used. For the total number of roles and tags on the page, the role and element count are used from the database. After that, it’s counted the number of elements inside of the element and role count, which are the number of types of roles and tags on the page. All this information in term helps the experts to see what pages to choose based on the numbers shown. This was added as
a card on the webpage’s page to see all of the specific elements of that specific page in greater detail (figure 4.4).

Other cards were also added to the website where the pages of a website are categorized by specific tags, like tables and Iframes. These cards in the website tab categorize the website pages by their elements and help the experts to choose based on the most important roles and tags for the sample (figure 4.5).

Finally, to help the experts, even more, it was added a sampling card where it could be chosen a number of specified pages that have the selected number of roles and or tags that
are present on the website (figure 4.6). We can choose only tags, roles, or a mix of both. For example, we can choose 5 pages with buttons (that is a tag) and they will show those 5 pages. If there are less than 5 pages with buttons it will choose all of them but not get 5 pages. That works the same for the roles. When you choose roles and tags it will either choose the number of pages selected for each role/tag or will choose the number selected in case the roles/tags are on all of the chosen pages (listing 4.1).

Figure 4.6: Sampling card

Listing 4.1: Pseudocode of Sampling Card

```python
function sampleWebsite(number_pages_sample, desired_roles_sample, desired_tags_sample) {
    role_list = getRolesInWebsite()
    tags_list = getTagsInWebsite()
    number_pages = 0
    sample = {}

    # if we selected roles only
    if (length(desired_tags_sample == 0) {
        # if selected one role
        if (length(desired_roles_sample == 1) {
            if (length(role_list[desired_roles_sample[0]]) <= n_pages_sample) {
                sample.append(role_list[role])
            } else {
                ...
            }
        } else {
            ...
        }
    } else {
        ...
    }
}
```
sample.append(
    random(role_list[role_list[desired_roles_sample[0]],
    n_pages_sample])
}

# if selected more than one role
} else (length(desired_roles_sample > 1) {
    for (role in desired_roles_sample){
        if (length(role_list[role]) <= n_pages_sample) {
            sample.append(role_list[role])
        } else {
            sample.append(
                random(role_list[role_list[role]],
                n_pages_sample))
        }
    }
}

# if tags are selected
else if (length(desired_roles_sample == 0){
    # if selected one tag
    if (length(desired_roles_sample == 1) {
        if (length(tag_list[desired_tags_sample[0]])
            <= n_pages_sample) {
            sample.append(tag_list[tag])
        } else {
            sample.append(
                random(tag_list[tag_list[desired_tags_sample[0]],
                n_pages_sample])
        }
    }
    # if selected more than one role
} else (length(desired_roles_sample > 1) {
    for (role in desired_roles_sample){
        if (length(tag_list[tag]) <= n_pages_sample) {
            sample.append(tag_list[tag])
        } else {
            sample.append(
                random(tag_list[tag_list[tag]],
                n_pages_sample))
        }
    }
}
}
# if roles and tags are selected

else {
    for (role in desired_roles_sample) {
        if (length(role_list[role]) <= n_pages_sample) {
            sample.append(role_list[role])
        } else {
            sample.append(
                random(role_list[role],
                n_pages_sample))
        }
    }

    for (tag in desired_tags_sample) {
        if (length(tag_list[tag]) <= n_pages_sample) {
            sample.append(tag_list[tag])
        } else {
            sample.append(
                random(tag_list[tag],
                n_pages_sample))
        }
    }
}
Chapter 5

Sampling

To help the experts have an easier time sampling the pages, and to receive a sample for them to base theirs on, some sampling algorithms were implemented to compare and see which one was better for sampling in accessibility, to be used afterwards.

5.1 Implementation

5.1.1 Justification of language and IDE

All of the algorithms chosen were implemented in Python. This choice was made so Artificial Intelligence could later be used to perfect the choices of any chosen algorithm. As for the IDE, it was chosen Jupyter notebook since it was easier to test each part since it has a property that allows sections of code to be tested without affecting the performance of others, which also allowed to have data and algorithms in the same file without affecting testing of algorithm performance.

5.1.2 Data Treatment for testing

The data used to test the algorithms was taken from the AMA database by using this query shown in figure 5.1. This query gets the PageId and Uri (both of them being important elements to distinguish in a simple way the pages) from the Page Table in the Monitor-Server database. It also retrieved the number of passed tests in A, double A, and triple A testing by the specific page. It was also retrieved in this query the element_count and role_count, which have the list of tags and roles respectively of the page with their counts. The number of A, double A, and triple A tests and the element and role count are all present in the evaluation table. To join all of this information in a way that each page is unique both tables were inner joined with each other by the common element in both tables which is the PageId.

After that, the file that was downloaded in the MySQL program after the query was executed had to be treated in order to be used by the algorithms. That occurs since the
original file (figure 5.2) did not let us study the amount of each role and tag separately for each page. That could be used to judge samples against each other and see how good a sample is, as will be explained further but also used as a parameter in one of the sampling algorithms.

That treatment is done inside of the Jupyter file (which allows testing of different samples of code separately) that had the implementation of the sampling algorithm since if anything was to be changed there it would be easier to change and at the same time, it
could be changed for each algorithm, if there was the need for it. It was done by using the original query file that was in a CSV format and transferring it into a Pandas DataFrame inside the file. Then afterward, we separate the element and tag count parts of the file by making each into a list that then is added to a Pandas DataFrame with the rest of the information. Due to some roles and tags having the same name which would cause conflicts, all of the names of the tags have been changed, with the addition of a \texttt{t} before the tag to fix that problem and to be understandable when performing the tests. That final Pandas DataFrame is the one then used as data. All the data treatment process and final results are shown in figure 5.3.

5.1.3 Algorithm Choice and implementation explanation

The algorithms chosen were Simple Random Sampling to be a control sampling algorithm since there is nothing worse than choosing random pages, Url Sampling, Stratified Sampling, HintSVM, and Representative Sampling. Only these were chosen since they covered the probability sampling group that has all of the sampling algorithms that made sense to test for accessibility sampling. The last two algorithms mentioned couldn’t be implemented in a working manner and there was no working implementation available.

Simple Random Sampling was implemented by using the random module on Python. With that module, a random sample will be chosen randomly from the data frame with the number of pages selected in the sample function as shown in figure 5.4.
Figure 5.4: Simple Random Sampling code

Url Sampling’s implementation was done by using Url clustering and random modules. The implementation firstly converts the percentage of the size of the sample input by the user to get a number of pages to sample (if the size inputed is less than 1). After that, the algorithm clusters the pages by their URL depending on the minimum cluster size that the user selected. To do that clustering, it first goes through the URLs of the dataframe and the choice of the user and tries to see where is a good place to divide URLs in order to create each group. After the groups are created and the basis URL for each group is found, it goes through the dataframe completely to check which URLs fit in which groups. If a URL does not meet any group criteria it goes to the unclustered pages group. After all the clustering is done, the sampling algorithm goes through the created groups on clustering and URLs associated with them to randomly pick one page from each until the size requirement is met. The result of the algorithm sends a list of URLs. Those URLs are then used to search the dataframe created previously for the URLs to also show all the parameters and make it easier to compare (figure 5.5).

Finally, the stratified sampling implementation was found and used[1]. The algorithm works like this: it receives a dataframe with all the information. In this case, it has pageID, URL, and all of the element and tag counts. It also receives which are the chosen parameters by the user from the available ones (URI, page id, number of passed tests of A, AA, and AAA tests, roles, and tags) from the dataframe from which to perform the sample. It

Figure 5.5: Url Sampling code

also receives the size of the sample wanted (could also be in percentage), although it is optional. If no size is specified it will calculate a size using Cochran adjusted sampling formula.

Firstly the number of samples is calculated by the following formula: $\text{cochran}_n = \frac{Z^2 \times p \times q}{e^2}$, where:

- $Z$ is a constant called $z$-value and is used in Cochran formula
- $p$ is the estimated proportion of the population which has one of the parameters.
- $q$ is $1 - p$
- and $e$ is the margin of error

After that number is calculated, its then adjusted by using this formula and then used in the algorithm as the final sample size: $\text{adjustedcochran} = \frac{\text{cochran}_n}{1 + (\text{cochran}_n - 1)}$

The algorithm also might use a seed for the sampling and also receives a parameter that controls if the final dataframe has its original indexes.

It uses those input parameters and first checks if the size is a percentage and converts it to a number of entries from the dataframe. It uses that to then create all the groups depending on the parameters chosen by the user. It also uses the strata and the dataframe to calculate how the population is divided to do the sampling accordingly. It then goes through each of those groups and takes the number of pages per each group bar the statistics created earlier and then presents them to the user (figure 5.6).
5.2 Results

After all the sampling algorithms were up and running as intended, a series of tests were done. All tests consisted of ten thousand runs of each test in order to have more accurate results and with the same parameters. The tests done were:

- A performance test. First, it measures the time of each sampling cycle done on a specific website individually. After that is done it will sum all of the times from each sampling cycle. Finally, it averages up the times of sampling to get the final time of running on average for each algorithm in each website (figure 5.7).

- Several different accuracy tests. These tests were based on the directive that AMA needs to apply in the future when doing their sampling, that being Council regulation (EU) nº 2018/1524.

The first set of accuracy tests measured how many pages were sampled from paragraphs a) and c). Those paragraphs say that: the sample has to have the homepage, login page, website map, contact page, help page, and copyright page in paragraph a); And in paragraph c) the regulation says that it has to have the accessibility declaration or accessibility policy and pages that have an information return mechanism (figure 5.12). In order to measure how accurate the samples were relative to the paragraphs, all the pages on each website were searched and their URLs found (if existing). After that, the URLs in the samples were compared to the ones extracted before. Then it would be counted how many pages did the sample have and then turned into a percentage. After that was done
every sample was averaged out on the ten thousand samples in order to get the average of pages in a sample of each algorithm.

The other tests were based on paragraphs b), d), and e). Those paragraphs say that the sample has to have at least 1 pertinent page at least for each service that the website provides, including the search service in paragraph b), it must have examples of pages that have a different appearance or that have a different type of content in paragraph d); And it must also have at least one downloadable file that is pertinent or at least a file for each website service if that applies to the website being tested as mentioned in paragraph e) (figures 5.8, 5.10, 5.9, 5.13). One of these tests, the Service page accuracy test was not able to be done on the same websites as the others, since some of the websites chosen did not have any type of services. To measure the accuracy of the sampling algorithms in the Service page test, firstly it was found how many services it had. Then all of the URLs linked with that service were found. After it was done, the samples were compared to the pages in each service to see if the service(s) were present and the number of each that was present was tallied. After it was done for every sample, it was averaged out and transformed into a percentage \[
\frac{\text{number of services in same}}{\text{number of services in the webpage}} \times 100
\]
by dividing the number of services found on average compared to the services present on the website (being values close to 0 bad and close to 100.

For the Role and Tag Accuracy tests, firstly it was found which roles/tags were in the website itself and tallied how many of those tags/roles were in each sample, transforming it then to a percentage of roles/tags in the website by dividing the tally by the total number of roles/tags in the website (being values close to 0 bad and close to 100.

Finally, on the page with the file accuracy test, each of the samples was checked for extensions related to a file (such as PDF, Docx, XLSX, and others) at the end of the URL. If it was found that one URL in the sample had the extension it was counted as having a page with a file. After it was all done in each sample it would be tallied how many samples had a file and then divided by the total number of samples, being then transformed into a percentage.

After all the tests were concluded, the results were tallied. These were the results:

As visible in figure 5.7, the performance of all algorithms is fast but gets slower as the number of pages increases, except for Simple Random Sampling. It has been shown that on the FCUL website the average time for sampling spikes in Random Sampling. It also can be seen that in Url sampling, it took longer than expected in terms of average sample time on the Caminha and Transparencia websites.

In the tag accuracy test (figure 5.8), it is seen that URL sampling is the better at getting most tags of a specific website, independent of the number of pages.

In the role accuracy test (figure 5.9), it can be seen that URL sampling is also better at getting most roles of a specific website, independent of the number of pages.

In the Service page accuracy tests (figures 5.10 and 5.11), it is seen that URL sampling
Figure 5.7: Performance test results for algorithms

Figure 5.8: Sampling Tag accuracy test results
Figure 5.9: Sampling Role accuracy test

is the better at getting most services in the smaller websites only, lacking in the bigger ones. Despite that fact, it’s also seen that all of the sampling algorithms severely lack in getting the services out of a webpage.

Figure 5.11: Sampling service page accuracy test by service

Regarding the percentage of “mandatory pages”, meaning the pages that fulfill paragraphs a) and c), it can be seen (figure 5.12) that all of the algorithms lack a lot in finding
them and the best algorithm is URL sampling once again, but it depended in the website in question.

Finally, in the sampling of pages with a file (figure 5.13), URL Sampling always managed to find one page for every single sample, better than the other sampling algorithms.

5.3 Discussion

As mentioned in the results section, there were a lot of tests done on the sampling algorithms to compare each of them in a way that could lead us to a conclusion of what is
the overall best sampling algorithm. In this section, all the comparisons will be done, and taken the conclusion of which is the better sampling algorithm for the job at hand.

### 5.3.1 Performance tests

The first tests done on the algorithms were the performance tests. For these tests, 10000 samples were done for each website separately, which were timed from start to the end of each of them. At the end of the test, an average was done to see which sampling algorithm was the fastest and had the best performance, as it could be the deciding factor if any of the other tests did not get conclusive results. After the tests were run on average each sample was taken at a really fast speed although it had some unexpected spikes in the Caminha and Transparencia websites for URL sampling and FCUL for Random Sampling as shown in figure 5.7. After some retesting on the FCUL, Caminha, and Transparencia websites and the addition of the ESTC website, which had a similar number of pages to the FCUL website, the spike could only be attributed to a spike in CPU usage for one or more runs in this website since all the math that gave us this average of the websites was also checked and it was as intended. The same process happened for the spikes in URL sampling time for Caminha and Transparencia websites and the same reason can’t be attributed to it as a result. As can also be seen, besides the spikes mentioned earlier, Random Sampling takes the same time to run as expected since there is not anything in its code where the number of pages can play a role in its performance. URL and Stratified sampling on the other
hand had their times increase as expected as the number of pages increased, besides the
aforementioned spikes.

5.3.2 Accuracy tests

To test the accuracy of each algorithm and how each of the algorithms compares in com-
pleting each of the tasks that each of the paragraphs implies, they were split into several
different types of tests. We tested a) and c) together as the pages never change if the web-
site does not change, being called the mandatory pages accuracy test. It was also decided
that to see how diverse the sample was to do a role accuracy and tag test. To also see
if the sample complied with the ‘one page per service’ requirement the sampling service
accuracy test was done. Finally, it was tested if each sample had at least one page with a
file.

Tag accuracy tests

As for the tag accuracy test (figure [5.8]), the same phenomena of the roles test also occurs,
although not as noticeable in the tags. Although less noticeable, it boosts Simple Random
Sampling’s results. Compared to the role test, the grouping of URLs that is employed by
Url sampling is not as effective but it is better.

Role accuracy tests

In the roles accuracy test, all the algorithms seemed pretty similar in the percentage of
roles in each sample, with Url sampling being the better one. Those results (figure [5.9])
could be explained by the fact of some pages having a lot of the roles of the specific
website, being some specific pages the ones with the roles needed for the full roles in
the sample. As shown by the results, the Url sampling way of grouping the websites
is better at grouping roles since it’s based on URLs compared to Stratified sampling.
Simple Random gets a great result on this particular test due to the aforementioned role
distribution of a website.

Sampling Service accuracy tests

As mentioned earlier, the websites used for testing did not all have services, meaning
fewer data to be taken from the website. As for the results (figures [5.10] and [5.11]) of
the algorithms, no correlation between the number of services or number of pages could
be made with the performance of the algorithms. All of the algorithms seem to have
issues finding service pages as seen in the graphs. That could be explained in random
sampling’s case with the lack of groups. In the cases of Stratified and Url sampling, it
could be explained by the groups made by each algorithm not being service-oriented and
therefore being harder to find services in the groups done by each of them.
Mandatory Pages accuracy tests

It can be seen that all of the algorithms are not amazing at finding the “Mandatory Pages” (figure 5.12), with the accuracy decreasing as the number of pages increases. This behavior was expected since there is sampling involved and it is never the same sample, therefore it is really hard for a sampling algorithm to get all of the pages in a lot of samples in a row since all of them are probabilistic sampling, and therefore samples are not set before the sampling is done. On most occasions, URL sampling is the better at doing this. It makes total sense since Url sampling groups the page URLs and therefore it is easier for it to find the pages that are the “Mandatory Pages”.

Sampling pages with file accuracy tests

In this test, URL sampling always had a page in their sample as shown in figure 5.13. That was explained by the fact that all URLs corresponding to downloadable files were inside of one of the clusters in URL clustering making them always selectable during testing no matter the number of pages. As for Stratified and Random Sampling, their results do not follow a specific pattern with the number of pages. The oscillation in random’s rating might be related to the random factor of random sampling and it not grouping the websites which makes it select at random the file on a sample, compared with URL sampling. Stratifieds’ randomness in results might be explained by the grouping done by Stratified sampling and how it’s not as straightforward as URL Sampling is in the groups it creates, where there is no specific group dedicated for files, and therefore files are spread out between groups which makes it harder and depending on the chance of the choice made in each group being the file URL.

5.3.3 Conclusions of Testing

After all the testing was done, it was concluded that the sampling algorithms chosen had an issue sampling pages related to website services, as well as “Mandatory Pages”. That last fact is not as problematic, since all of the pages in that sample are easily found by the experts. It can also be concluded that Url Sampling was the best in almost every scenario tested and therefore would be the best algorithm to choose to sample pages for AMA, although it severely lacks on service sampling.
Chapter 6

Conclusions

Website accessibility monitoring is vital for the usability of websites all over the world by everyone. With website accessibility achieved and monitored, everyone on the planet, disabled or not, can access any website and have the same good experience doing everything they desire. The accessibility monitoring entities should have the best tools at their disposal so that their job is easier.

The work developed in this thesis led to the upgrading of the existing tools so that the experts had an even better way to use their tools while doing their job. It added some metrics to the existing tools in order to be easier to access each website’s data helping the experts do their manual sampling in a more informed way. It also has added a simple sampling function that allows the experts to sample a certain amount of pages with certain role and tag parameters. The generated samples could then be compared to the manual sampling they had and might help the experts.

This thesis also contains a study done in sampling algorithms, in order to implement an even better version of the current simple sampling that is currently in the experts’ tools in favor of helping them even further by giving a sample that complies with most of the rules that are to be implemented in the future and help create their samples even faster. Several sampling algorithms were evaluated, where it came out that URL Sampling was the best choice for a sampling algorithm, having a clear advantage over any other algorithm in most of the tests conducted, although still not perfect.

Regarding future work, the algorithm chosen by the study should be implemented and could even be improved by the use of Artificial Intelligence to better its current weaknesses and learn even more about how to cluster in the best way and also choose the best pages.
Bibliography


