SISSI – Web Page’s Structure Identification from Syntactic and Semantic Information

Ana Patrícia Fernandes Salvado

Mestrado em Informática

Dissertação orientada por:
Prof. Doutor Carlos Alberto Pacheco dos Anjos Duarte e Prof. Doutor Luís Manuel Pinto da Rocha Afonso Carriço

2018
SISSI – Web Page’s Structure Identification from Syntactic and Semantic Information

Ana Patrícia Fernandes Salvado

Mestrado em Informática

Dissertação orientada por:
Prof. Doutor Carlos Alberto Pacheco dos Anjos Duarte e Prof. Doutor Luís Manuel Pinto da Rocha Afonso Carriço

2018
Acknowledgments

First of all, I would like to thank my supervisors for all the patience, the guidance and help, whenever I need it.

I want to thank my family, especially my parents and brother for the support and constant encouragement throughout the development of this work. Their motivation helped me to stay focused.

I thank those who have been by my side in the last four years, Inês de Matos, João Vicente, Carlos Duarte and Tiago Moucho for being supportive and helpful through these years and for the entertaining when necessary.

I thank to Elgin Akpinar for helping me understand his segmentation tool and for being available to answer all my doubts.

Finally, I want to thank to accessibility experts Dr. Jorge Fernandes and Dra. Cláudia Cardoso, for being available to participate in an interview.
Resumo

A WWW (World Wide Web) é uma plataforma que contém uma grande quantidade de informação disponível. Hoje em dia, as pessoas dependem muito da Internet para ajudar nas mais pequenas tarefas do dia-a-dia, como ir às compras, ver o correio eletrónico, falar nas redes sociais ou até no trabalho. Este aumento no uso da Internet foi impulsionado pelo uso de sistemas de gestão de conteúdo, permitindo a utilizadores normais, criar e gerir o seu próprio website. Existindo esta quantidade de informação, eventualmente apenas em formato digital, é necessário considerar o seguinte: todas as pessoas são diferentes e podem “ver” a Internet de diferentes perspetivas. Sendo uma plataforma universal, o seu conteúdo deveria ser acessível a todos de maneira igual. Infelizmente, não tem havido muita consideração para com pessoas com necessidades especiais. Pessoas com alguma deficiência, quer seja motora, sensorial ou mental, deveriam conseguir ter a mesma experiência que o resto dos utilizadores. Devido ao facto de estarem condicionados, é preciso ter em conta no desenvolvimento Web, soluções que permitam a essas pessoas terem a mesma experiência de navegação. Por esta razão o conceito de acessibilidade na Web foi crescendo ao longo dos últimos anos. Foram desenvolvidas diretrizes e padrões de modo a que as páginas Web desenvolvidas sejam acessíveis a todos. Para ajudar no desenvolvimento de páginas acessíveis, a tarefa de confirmar se as diretrizes de acessibilidade estavam a ser cumpridas foi automatizada. Uma ferramenta capaz de fazer este tipo de análise é o QualWeb, desenvolvido na faculdade de Ciências da Universidade de Lisboa por um grupo de investigadores, no qual eu colaborei.

Como referido anteriormente, as páginas Web têm cada vez mais conteúdo, pelo que é importante para os seus consumidores conseguir distinguir a informação relevante. Ao ser apresentada uma página a uma pessoa com necessidades especiais, esta pode não ser capaz de fazer a separação de conteúdos, o que pode prejudicar a interpretação da informação.

Com este trabalho pretendo desenvolver um mecanismo capaz de identificar a estrutura de uma página Web, através da sua informação semântica e sintática. Neste caso vou apenas focar-me em menus porque para além de serem um elemento importante nas páginas Web, podem facilitar a navegação para uma pessoa com necessidades especiais.

Para atingir este objetivo, precisei de usar uma ferramenta de segmentação. Dentro de todas as que foram pesquisadas na minha investigação de trabalho relacionado, a melhor foi o VIPO, que relaciona os aspetos visuais da página Web e o código fonte para dividir a página em diferentes secções. O meu capítulo do trabalho relacionado também contém os principais conceitos para entender o que foi desenvolvido, bem como uma investigação de outras ferramentas de avaliação automática da acessibilidade de páginas Web, focando-se mais na que vou utilizar, o QualWeb. Como mencionado também refiro algumas ferramentas de segmentação, descrevendo cada uma delas e mostrando numa tabela as suas características, sendo o VIPS estendido a melhor classificada para o que pretendia.

Depois da análise do trabalho relacionado, fui perceber o funcionamento do VIPS estendido, interpretando os vários passos do algoritmo. Uma das vantagens do VIPS estendido é que para além de fazer a segmentação de uma página, ou seja, dividi-la em diferentes secções, é ainda capaz de categorizar cada secção, atribuindo um papel à mesma. O funcionamento do VIPS estendido é descrito no capítulo 3, bem como a razão do foco na avaliação dos menus de páginas Web e em sistemas de gestão de conteúdos durante o processo de otimização do VIPS estendido.
Inicialmente, avaliei 30 páginas geradas por um sistema de gestão de conteúdos, designado WordPress, e determinei que elementos eu considerava serem menus nessas páginas. Posteriormente, verifiquei se os elementos que eu considerava menus, também eram assinalados pelo VIPS estendido com esse mesmo papel. Os resultados obtidos não foram satisfatórios, o que me levou a melhorar as regras que classificavam as diferentes áreas da página. Após vários testes, cheguei a um resultado satisfatório, que não consegui melhorar mais. Neste ponto, a identificação de menus era bastante positiva, embora o número de falsos positivos também fosse bastante elevado na minha opinião. Com isto em mente, desenvolvi um filtro para detetar os falsos positivos. Através da observação dos resultados de vários testes, pude determinar que vários tipos de elementos eram considerados menus quando não o deveriam ser. Sabendo a tag do elemento posso evitar que elementos com essas tags sejam considerados menus. Desta maneira, os falsos positivos foram reduzidos e resultados foram melhorados ligeiramente.

Estes resultados foram baseados na minha definição de menu e nas observações feitas no conjunto de 30 páginas Web geradas pelo sistema de gestão de conteúdos WordPress, pelo que precisava de validar o mecanismo desenvolvido. Para isto construí um conjunto de outras 30 páginas Web, composto por 15 de páginas geradas por WordPress e outras 15 não geradas por WordPress. Os resultados obtidos nas 15 páginas geradas pelo WordPress foram iguais aos anteriores, o que é positivo, querendo dizer que as regras desenvolvidas são suficientemente genéricas. Em relação às 15 páginas não geradas pelo WordPress, os resultados superaram as expectativas, porque foram melhores que os anteriores.

Como mencionei, o melhoramento das regras baseou-se na minha interpretação de menu, pelo que foi necessário testar se estas podiam ser consideradas gerais, ou seja, se outras pessoas consideram a mesma definição para menu. Para isto, realizei dois testes: o primeiro entrevistando peritos em acessibilidade e outro através de um questionário a utilizadores comuns. Os resultados obtidos divergiram tanto para peritos como para utilizadores. Um aspeto que foi unânime entre os três peritos entrevistados foi o facto de a definição de menu ser suscetível a interpretações, dependendo por vezes do contexto ou opinião pessoal. Um outro aspeto foi o facto de à medida que a entrevista progredia, mais elementos eram considerados menus ou pelo menos surgia a dúvida se seriam ou não. Não houve um consenso na opinião dos peritos, visto que um deles era muito restritivo, seleccionando apenas o menu encontrado no cabeçalho ou rodapé das páginas e outro perito, considerava tudo o que fossem listas como menus.

Em relação aos utilizadores, houve alguns menus selecionados constantemente (menus de navegação) e outros que geraram mais dúvidas, sendo um deles o elemento que contém os ícones das redes sociais.

Após a classificação dos menus estar de acordo com o que pretendia, o próximo passo era implementar o VIPS estendido no QualWeb. Infelizmente, a ferramenta do VIPS estendido que estava a utilizar necessitava do ambiente onde estava implementado (no Eclipse) e portanto perguntei ao responsável pelo desenvolvimento do VIPS estendido se seria possível criar uma versão nova, em que o VIPS seria disponibilizado num Web service e desta maneira seria possível incorporá-lo no QualWeb. Com isto feito, as novas técnicas de acessibilidade, que são aplicadas a menus, puderam ser desenvolvidas. Para validar o desenvolvimento fiz uma análise pericial às 30 páginas Web de treino para conseguir comparar com os resultados das novas técnicas. Através dessa comparação, determinei que as técnicas foram implementadas corretamente.

Palavras-Chave: Segmentação Web, Web Semântica, Acessibilidade Web
Abstract

The World Wide Web (WWW) is today a widespread all-purpose technological platform, with a large quantity of information available, that is supposed to be accessible to everyone. However, this accessibility is not yet a reality. There are some people with disabilities that can’t have the same Web experience as the others. Web designers are becoming more aware of the problem that is the lack of accessible Web pages. The progressive adoption and implementation of specific standards and guidelines on the Web is increasing accessibility, ensuring that everyone has the same Web experience.

These specific standards can be evaluated with Web accessibility evaluation tools, allowing Web designers and developers to evaluate their Web page’s accessibility. With this assessment, it is possible then to change the inaccessible source code.

However, some of those guidelines need semantic knowledge, which is something that has not been implemented in the Web accessibility evaluation tools. With this purpose, my work focuses on developing and implementing a mechanism able to identify the semantic and syntactic structure of a Web page. This mechanism will rely on a segmentation tool, which separates a Web page into regions. The mechanism attributes to each region a role, which semantically represents its usage and purpose on the Web page. With this categorization, the accessibility standards and guidelines can be applied to the right elements.

This document starts with a detailed introduction and goals for this work. Following, a review of the related work already done will be given, which include the concepts to understand the developed work as well as the description of the tools used. Then it will explain in detail the segmentation tool used and the initial analysis made, which was not satisfactory. The improvements to the segmentation tool and the results of each analysis will be presented in the next chapter. Next, the comparison of my evaluation with experts and users evaluations will be given. Finally, the integration of the developed mechanism into the accessibility evaluation tool will be presented, as well as the new developed techniques. At the end, a conclusion will summarize the work done and will include some future work relating to this topic.

Keywords: Web Segmentation, Semantic Web, Web Accessibility
Content

Chapter 1 Introduction ........................................................................................................... 1
  1.1 Motivation .................................................................................................................... 1
  1.2 Goals .......................................................................................................................... 3
  1.3 Work Plan .................................................................................................................... 3
  1.4 Document Organization .............................................................................................. 5
Chapter 2 Related Work ....................................................................................................... 7
  2.1 Web Evolution .............................................................................................................. 7
    2.1.1 Semantic Web ........................................................................................................ 8
  2.2 Web Accessibility ........................................................................................................ 8
    2.2.1 Accessibility Standards ......................................................................................... 8
    2.2.2 Web Accessibility Evaluation Tools ....................................................................... 10
    2.2.3 QualWeb ............................................................................................................... 12
  2.3 Content Management Systems ..................................................................................... 14
  2.4 Web Page Segmentation .............................................................................................. 15
    2.4.1 Web Segmentation Tools ..................................................................................... 16
  2.5 Summary ...................................................................................................................... 20
Chapter 3 Analysis .............................................................................................................. 21
  3.1 Extended VIPS ............................................................................................................ 21
    3.1.1 Extended Visual Block extraction ............................................................................ 22
    3.1.2 Separator detection ............................................................................................... 25
    3.1.3 Content structure construction .............................................................................. 25
    3.1.4 Rule Generation .................................................................................................... 26
    3.1.5 Role Detection ...................................................................................................... 28
  3.2 Web Service VIPS ....................................................................................................... 28
  3.3 Content Management Systems ..................................................................................... 30
  3.4 Menus .......................................................................................................................... 30
  3.5 Summary ...................................................................................................................... 33
List of Figures

Figure 1 – Code used to test post-processing evaluation ........................................... 10
Figure 2 – QualWeb’s Architecture ........................................................................... 13
Figure 3 - Extended VIPS Architecture. Source: M. E. AKPINAR and Y. YEŞILADA, “Discovering Visual Elements of Web Pages and Their Roles: Users’ Perception,” vol. 29, no. 6, 2017. .................................................................................. 22
Figure 4 – The layout structure and vision-based content structure ............................. 23
Figure 5 – The Web page example ............................................................................ 23
Figure 6 – DOM tree Web page’s example ............................................................... 24
Figure 7 – Visual element tree of Web page’s example .......................................... 24
Figure 8 – Horizontal separator detection process .................................................. 25
Figure 9 – Output of VIPS segmentation process ..................................................... 26
Figure 10 – Rule definition in Jess rule engine ......................................................... 27
Figure 11 – Output of VIPS segmentation process with role assignment .............. 28
Figure 12 – Request call example ............................................................................ 29
Figure 13 – Response to the request call example .................................................. 29
Figure 14 – Rule definition in NodeJS rule engine .................................................. 29
Figure 15 – Correct menu example, using WAI-ARIA attributes ............................. 31
Figure 16 – Incorrect menu example ....................................................................... 31
Figure 17 – Rule file properties ................................................................................ 36
Figure 18 – List role definition ................................................................................ 36
Figure 19 – LinkMenu role definition ...................................................................... 37
Figure 20 – Rules added to the MenuItem role ........................................................ 37
Figure 21 – Rule modified for the MenuItem role ..................................................... 37
Figure 22 – Rule modifications for the LinkMenu role ........................................... 38
Figure 23 – Example of an element in the JSON response ..................................... 61
Figure 24 – Flowchart of the process of inserting the Web service VIPS algorithm in QualWeb ...................................................................................................................... 62
Figure 25 - Flowchart of the process of selecting the menus from the VIPS JSON response .................................................................................................................................. 63
Figure 26 – Flowchart of the process of the H97 technique .................................... 66
Figure 27 – Flowchart of the process of the H48 technique ..................................... 67
List of Tables

Table 1- Initial Schedule ................................................................................................................. 4
Table 2 – Summary of Segmentation Tools ......................................................................................... 19
Table 3 – Extended VIPS first evaluation results .............................................................................. 32
Table 4 – Extended VIPS second evaluation results ......................................................................... 39
Table 5 – Web service VIPS results ................................................................................................... 41
Table 6 – Results of the evaluation of new Web pages set with the 2nd filters .................................. 46
Table 7 – Results of the evaluation of new Web pages set with the 3rd filters ................................. 46
Table 8 – Result comparison of the expert’s evaluations with the Web Service VIPS with 2nd filters .................................................................................................................................. 49
Table 9 – Result comparison of the expert’s evaluation with the Web Service VIPS with 3rd filters .................................................................................................................................. 51
Table 10 – Chi-square Test of Experts with Web Service VIPS ...................................................... 51
Table 11 – Mann-Whitney Test of Experts Consensus with VIPS ................................................. 52
Table 12 – Agreement percentage between Experts’ opinion and Web Service VIPS by category .................................................................................................................................. 53
Table 13- Chi-square test of Users with Web service VIPS .............................................................. 55
Table 14 - Agreement % between the first group of the survey and the Web service VIPS by category .................................................................................................................................. 56
Table 15 – Techniques implemented in QualWeb ............................................................................. 59
Table 16 – Results of the initial expert assessment .......................................................................... 68
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WWW</td>
<td>World Wide Web</td>
</tr>
<tr>
<td>CMS</td>
<td>Content Management System</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium</td>
</tr>
<tr>
<td>WCAG</td>
<td>Web Content Accessibility Guidelines</td>
</tr>
<tr>
<td>HTML</td>
<td>HyperText Markup Language</td>
</tr>
<tr>
<td>CCS</td>
<td>Cascading Style Sheets</td>
</tr>
<tr>
<td>AJAX</td>
<td>Asynchronous JavaScript and XML</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>JSON</td>
<td>Javascript Object Notation</td>
</tr>
<tr>
<td>RIA</td>
<td>Rich Internet Application</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>WAI</td>
<td>Web Accessibility Initiative</td>
</tr>
<tr>
<td>WAI-ARIA</td>
<td>Web Accessibility Initiative Accessible Rich Internet Application</td>
</tr>
<tr>
<td>APIs</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>BPS</td>
<td>Browser Processing Simulator</td>
</tr>
<tr>
<td>IS</td>
<td>Interaction Simulator</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>DOM</td>
<td>Document Object Model</td>
</tr>
<tr>
<td>EARL</td>
<td>Evaluation and Report Language</td>
</tr>
<tr>
<td>CSV</td>
<td>Comma-separated values</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>VIPS</td>
<td>Vision-Based Page Segmentation</td>
</tr>
<tr>
<td>DoC</td>
<td>Degree of Coherence</td>
</tr>
<tr>
<td>PDoC</td>
<td>Premitted Degree of Coherence</td>
</tr>
<tr>
<td>W AfA</td>
<td>Web Authoring for Accessibility</td>
</tr>
<tr>
<td>ACTF</td>
<td>Accessibility Tools Framework</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

This chapter summarizes the main concepts that will be discussed throughout the document. It begins with a contextualization of the main topics and its importance. Then, it describes the main goals to achieve, the planning and finally, the document organization.

1.1 Motivation

The WWW (World Wide Web) has grown considerably in the last years, mostly due to a massive increase in the people that use it, about 3.74 billion users, as well as the increasing number of sites that appear every day, totalling over 1.24 billion websites in the world\(^1\). One of the main tools responsible for this growth are CMSs (Content Management Systems). CMSs offer the capability to create, manage and publish a website, even for those users that do not have programming abilities. This is a great advantage, but it introduces some barriers regarding accessibility [1], [2].

Nowadays, people rely on the Web to assist in almost every task that they need to do throughout the day, in their workplace, checking the email, shopping online, communicating in social media, playing for entertainment, and so on. As we need to access a huge amount of information that is becoming computerized, we must consider that no one is equal and different people “see” the Internet from distinct perspectives.

As a free and universal information platform, the Web should be accessed by everyone the same way. Unfortunately, people with special needs have been mostly disregarded. Someone with a disability, such as blindness, or any visual disease, deafness, physical or mental problems, should be able to understand the Web page’s content with the same quality as the rest of the population. Because they are conditioned in the use of the Internet, Web development must consider solutions for them to navigate in the same way as others. For this reason, the concept of Web accessibility has grown through the years. This means that Web pages need to be perceived by everyone, including anyone with special needs.

An organization known as W3C (World Wide Web Consortium) exists to standardize the Web experience, ensuring among other things, that websites are considered accessible to all. This is done by proposing guidelines and patterns, known as WCAG (Web Content Accessibility Guidelines) to develop widely accessible Web pages. Although these guidelines are important and indubitably produce websites more accessible, they might not be enough [3]. These guidelines

\(^1\) https://hostingfacts.com/internet-facts-stats-2016/ Last accessed on November 2017
can be used in the development process or afterward, using Web accessibility evaluation tools to check the conformance of a Web page as well as to identify any changes to improve accessibility.

These tools can be manual or automated. Being automated, they have some advantages. They:

- don’t need manpower;
- are objective;
- are much faster than a human expert evaluator.

But they also have disadvantages, mainly because their results are not contextualized. They only tell us that the Web page respects the guideline or doesn’t. Sometimes the tool just does not know how to decide, needing human intervention to get a contextualized analysis. There are several types of tools, each of them with different components, interfaces and functionalities. An example of these Web accessibility evaluation tools is QualWeb\textsuperscript{2}, developed in the Faculdade de Ciências da Universidade de Lisboa, by a research group with which I collaborate. The aim of this tool is to evaluate Web page accessibility, applying the guidelines of WCAG. Performing this automatic analysis can always assist the developers improving a Web page’s accessibility.

Usually, Web pages have different types of content, which could be either relevant or irrelevant to the main topic of the page or to the user. The relevant content is known as the informative content, e.g. menus, or the data. The irrelevant content, on the other hand, is usually called noisy information (e.g. advertisement). As mentioned, every person sees a Web page in his own way. A page presented to a non-disabled user perceives certain spatial and visual cues that can help him divide the page into certain areas that he knows, understands and is familiar with, even when he expects some other format or functionality from that component. For this user, it’s easy to understand the relevance and usefulness of the information he is looking for. In the case of a disabled person, he may not be able to do that separation or to make sense of what he sees or listen. Therefore, it’s much harder for him to understand the whole page. Frequently, this person requires the use of additional tools so he can understand the content of the page, its structure, and the relevant information. Among others, these tools or mechanisms can be a screen reader, braille display or voice browser. Another practice that facilitates the understanding of a page’s content by a disabled person is to include alternative forms of media on the Web page. As an example, if a Web page has a video, it should also have a text or sign language that describes the video, for blind or deaf people to perceive what’s in it.

In addition to the guidelines used by Web page evaluation tools to analyze the accessibility of a page, there is a process that can also improve that evaluation, called Web page segmentation. This process does the separation of content in a Web page, allowing the identification of different regions in the page. In addition to being able to be applied in Web accessibility, it can assist in repairing a Web page or in the identification of relevant information.

Here the challenge is how to automatically identify the different roles of the regions of a segmented page because websites do not follow the same format and structure. So, it is extremely difficult for a mechanism to understand what the content of the HTML tags represents in the Web page (e.g. a menu can use <div> tags) and the relationship between them.

\textsuperscript{2} http://www.qualweb.di.fc.ul.pt/
1.2 Goals

This work aims to design and develop:

A mechanism able to identify the semantic structure of Web pages through their semantic and syntactic information.

Then, using this mechanism, improve a Web accessibility evaluation tool, namely, QualWeb, through the implementation of WCAG techniques, concerning menus.

To achieve this goal I use a segmentation tool which relates the visual aspect of the Web page to its source code, generating a semantic structure of that Web page. The Web pages used to test and improve the mechanism will be CMS generated. Not only they are based on templates, which can help in patterns identification, but also highly used and with concerns regarding accessibility.

Sometimes the coding sequence of a page doesn’t reflect correctly the visual presentation of the page, e.g., related elements could be next to each other in the source code and apart in the page layout. In the semantic structure obtained it is necessary to understand the roles played by the different regions that compose it. After assigning a role for each of the composing regions it becomes possible to implement WCAG 2.0 techniques that depend on semantic information on QualWeb. In particular in this work, the techniques were addressed to menus.

The main goals of this thesis are:
1. Understand the source code generated by CMS in order to recognize menu patterns produced and their usual elements.
2. Analyse the existing segmentation tools and access the one with better results in extracting a Web page’s semantic structure.
3. Find and improve a mechanism able to correctly attribute menu roles to regions extracted by the segmentation tool.
4. Implement in QualWeb the new techniques incorporating the segmentation tool and the role attribution mechanism.
5. Evaluate the new techniques inserted in QualWeb using a CMS generated Web pages testing set.
6. Analyse the role attribution mechanism with the opinions of regular users and experts.

1.3 Work Plan

This section describes the several tasks needed to be done to accomplish the stated goals. Table 1 contains the task list and the corresponding duration (in months). The first six tasks were performed according to the schedule. To perform task 7, I needed to wait that a new version of the tool I was using was developed. This took about two months to be developed and was only released in the middle of August. Task 10 also took two additional weeks to be done, due to the unavailability of some resources.
Table 1 - Initial Schedule

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Duration (months)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Research on related work</td>
<td>2</td>
</tr>
<tr>
<td>2. Test existing tools</td>
<td>0.5</td>
</tr>
<tr>
<td>3. Preliminary Report</td>
<td>0.5</td>
</tr>
<tr>
<td>4. Analyse the performance of the segmentation tool</td>
<td>1.5</td>
</tr>
<tr>
<td>5. Improve the mechanism for identifying the semantic and syntactic</td>
<td>2</td>
</tr>
<tr>
<td>structure of a Web page</td>
<td></td>
</tr>
<tr>
<td>6. Evaluation and validation of the developed mechanism</td>
<td>2</td>
</tr>
<tr>
<td>7. Implementation of the mechanism into QualWeb</td>
<td>0.5</td>
</tr>
<tr>
<td>8. Implementation of the new techniques into QualWeb</td>
<td>0.5</td>
</tr>
<tr>
<td>9. Validation of the new techniques implementation</td>
<td>1</td>
</tr>
<tr>
<td>10. Comparative analysis with users and experts</td>
<td>0.5</td>
</tr>
<tr>
<td>11. Thesis</td>
<td>2</td>
</tr>
</tbody>
</table>

T1 – Research on related work: The objective of this first step is to understand what has already been done in this area, how it has been done, existing tools and how they work.

T2 – Test existing tools: This task aim is to understand how the tools that already exist do the segmentation of a Web page.

T3 – Writing the Preliminary Report: After all investigation of the related work and collected information about existing tools, the writing of the preliminary work synthesized the state of that research, as well as defined the goals, the planning, and the work that has been done.

T4 – Analyse the performance of the segmentation tool: This task consisted on perceiving how well the segmentation tool performed.

T5 – Improve the mechanism for identifying the semantic and syntactic structure of a Web page: This task consisted on developing an improved mechanism able to identify the semantic and syntactic structure of a Web page.

T6 – Evaluation and validation of the developed mechanism: The objective of this task is to evaluate the mechanisms developed through a series of tests and perceive what could be further improved.

T7 – Implementation of the mechanism into QualWeb: When the mechanisms were according the requirements, it was implemented on the Web page evaluation tool, QualWeb.

T8 – Implementation of the new techniques into QualWeb: Development of the new techniques to be added into QualWeb.

T9 – Validation of the new techniques implementation: After the implementation of the new techniques, an analysis was performed to confirm if the techniques were well implemented.

T10 – Comparative analysis with users and experts: This task consisted in evaluating the menu identification mechanism through users’ and experts’ opinion.

T11 – Writing the Thesis: This task occurs after all the previous work been done and reflects the evaluation of the new mechanisms, as well as all the relevant information concerning the development process.
1.4 Document Organization

This document has been organized in the following chapters:

- Chapter 2 – Related Work: This chapter gives an overall idea of the state of the art in regard to Web page segmentation. It starts with an overview of the Web evolution, taking an in-depth approach to the semantic Web. Next, it will refer Web accessibility standards and tools. Then it will introduce CMSs and one related existing tool. Finally, it mentions the Web page segmentation state of the art and tools;
- Chapter 3 – Analysis: This chapter describes the mechanism chosen to segment and classify the Web pages, as well as its design and architecture. It will also approach the reasons for choosing Content Management Systems and the introduction to the menu element;
- Chapter 4 – Implementation: This chapter describes the improvements made to the mechanism able to identify the semantic and syntactic information on a Web page, specifically the rule file of the segmentation tool.
- Chapter 5 – Evaluation: This chapter compares my menu definition approach and the opinion of experts in accessibility and regular Web users. The chapter also presents the validation of the results obtained.
- Chapter 6 – QualWeb: This chapter lists the relevant techniques to this work, and explains the process of the implementation of the new techniques added to the evaluation tool QualWeb, as well as the integration of the new mechanism VIPS into QualWeb. Then a validation of those techniques will be presented.
- Chapter 7 – Conclusion: This chapter concludes the work and presents possible future work using the developed mechanism.
Chapter 1.

Introduction
Chapter 2

Related Work

This section will focus on the related work that has been developed mostly in the area of Web segmentation, as well as all the relevant information to understand the content of this document. A small introduction will be given about the concept of the Web and its evolution. Next, concepts related to Web accessibility will be introduced. It will mention the Web accessibility standards, the Web accessibility evaluation tools, and a more detailed description of a specific one. Then, the concept of CMSs (Content Management Systems) is introduced. Finally, an introduction to Web segmentation and a summary of the work done so far is presented. The last section summarizes this chapter.

2.1 Web Evolution

There has been a change in the way developers create Web pages, mainly due to the emergence of new technologies but also because the amount of data placed online every day. Static Web pages have been replaced by dynamic pages, where the user can control and have power over the information in the website, see real-time information, change the content and allow more interaction with other users and websites. Such evolution was only possible because of the improvements of HTML (HyperText Markup Language) and CSS (Cascading Style Sheets), currently in versions 5 and 3 respectively. Apart from these basic languages that already existed in Web1.0 (static Web), other languages and technologies have assisted in the growth of Web2.0 (dynamic Web). On the client side, Javascript or jQuery are used for dynamic content, AJAX (Asynchronous Javascript and XML) for updating the page without having to reload it. On the server side, PHP, Python, Ruby or Perl, are used to output information stored in databases dynamically, therefore allowing Web services and even Web pages, to share part of their information with each other through compatible formats, like XML (Extensible Markup Language) or JSON (Javascript Object Notation).

Within the Web2.0 a new trend has arrived, known as RIA’s (Rich Internet Applications). These kind of applications are no longer static, accordingly to [4], but instead complex applications presenting a similar behavior to desktop applications. The use of scripting languages (e.g. JavaScript, AJAX, etc.) granted such applications a higher complexity. Using these scripting languages means that a Web page can be modified several times, without leaving the browser client side. Consequently, the same URL (Uniform Resource Locator) can correspond to different states of the same page.
Web3.0 (semantic Web), already arose and is being discussed. It isn’t a separated Web, although, in fact, it is an extension of the current one, in which information is given a well-defined meaning, better enabling computers and people to work in cooperation [5].

2.1.1 Semantic Web

Currently, the content placed online is user-driven, that is, for people to do something with the information accessed. But at the current pace, it is becoming more and more difficult for a human to manage the available amount of data, and retrieve any useful information from that. With the semantic Web, the idea is to turn the information machine-driven, so that information may be automatically processed by machines, in a way that the information retrieved could be precise and useful to the human. The challenge of the semantic Web is to provide a language that expresses both data and rules for reasoning about the data and that allows rules from any existing knowledge-representation system to be exported to the Web, as it is mentioned in [5]. In [6] they sustain this claim, saying that agents can only perform its tasks when a standard is well established.

2.2 Web Accessibility

It is well known that the Web is in our daily lives and many activities rely on it. If the WWW is universal, this means that it should be accessible to everyone, which is not the case. Some people with a disability, suffer social exclusion due to the difficulty for them to navigate in the Web. This group of people, with special needs, is about 15% of the world’s population3.

With Web dynamics, users and pages themselves can frequently alter their content and consequently, the Web page can be substantially different from what was initially presented. Web page dynamics actually allows the same page to show different content to a user with or without his interaction.

For a disabled person, this is hard to assimilate, and, at the present, the majority of websites aren’t accessible. This is not a problem that has emerged recently, it exists since the Internet’s beginning. Some people aware of it have tried to diminish or find ways to solve this problem. For that reason, the W3C (World Wide Web Consortium) was founded by Tim Berners-Lee in 1994. This consortium develops protocols and guidelines to ensure the growth of the Web, stating that websites that follow those standards could be accessed and visualized by any person or technology4. The W3C has an initiative called WAI (Web Accessibility Initiative) focused on making the Web accessible to people with disabilities. In the next subsection, I’m focusing on a particular standard from the WAI guidelines and a technical specification published by W3C.

2.2.1 Accessibility Standards

The WAI guidelines are directed to make Web pages more accessible and I’m going to mention one standard I think it is important to my work. I will mention also a technical specification, specialized in dynamic content. The first one, the most used is WCAG 2.0 (Web Content Accessibility Guidelines), which, as the name implies, focuses on making content more accessible, not only for people with disabilities but also to the general user. It has four major

---

4 https://www.w3.org/ Last accessed on November 2016
principles, which are: perceivable, operable, understandable and robust, and within these principles twelve guidelines are inserted\(^5\). For each guideline, there are examples of how it should be implemented, how the guideline could be tested and what is necessary for it to be successful. There are three levels of success, A, AA, AAA, but even the content having the highest level of conformance may not be accessible to all individuals.

WCAG2.0 is divided in four principles, which helps developers address parts of the code:

- **Perceivable** – Information and user interface components must be apprehended by users;
- **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 reliably by a wide variety of user agents, including assistive technologies.

Within these principles, as I mentioned, there are guidelines with success criterion, and for a good implementation, some techniques are suggested. These techniques can be implemented in a Web accessibility evaluation tool, and each technique can have three outcomes: pass, fail or warning. The pass and fail results do not need any interpretations because they just respect the standard recommendation. The warning result is more difficult to assess because it could mean that there is some value or element that cannot be evaluated or considered right or wrong. In most cases, the problem is related to the semantic, thus, requiring human intervention. An example of this is the techniques h37 that say: “Using alt attributes on img elements”. An image has an alt attribute, but for this technique to pass, the text on the alt attribute has to describe the image. Often, the evaluator does not verify that just asks the developer to manually inspect the element.

The technical specification that is worth mentioning is WAI-ARIA\(^6\) (Accessible Rich Internet Application), also targeted to making Web content accessible, is specialized in dynamic content and advanced user interface controls. These guidelines have appeared due to the lack of accessible functionalities in some websites, which are difficult for disabled people to use, especially those that rely on screen readers or those who can’t use a mouse. The WAI-ARIA solution pretends to provide a framework so the developers can add attributes to identify features for user interaction, how they relate to each other as well as their state. Also, they describe new navigation techniques to mark regions and common Web structures (e.g. menus, primary content, banner information, etc.). An example of this is the use of WAI-ARIA to identify a region of a page and enable keyboard users to easily move between regions, rather than having to press the <Tab> button several times to get there. It also includes technologies to map controls, Ajax live regions and events to accessibility APIs (Application Programming Interfaces), including custom controls used for rich Internet applications.

WCAG guidelines are focused on helping Web developers, Web accessibility evaluation tool developers and anyone that needs a standard for Web accessibility. In the next subsection, I will discuss the use of these guidelines in Web accessibility evaluation tools, even though content that fully conforms with these guidelines, may not be fully accessible to every person with a disability [3], [7].

---

\(^5\) [https://www.w3.org/WAI/intro/wcag.php](https://www.w3.org/WAI/intro/wcag.php) Last accessed on November 2016  
\(^6\) [https://www.w3.org/TR/WCAG20-TECHS/wai-aria_notes.html](https://www.w3.org/TR/WCAG20-TECHS/wai-aria_notes.html) Last accessed on November 2016
Chapter 2. Related Work

2.2.2 Web Accessibility Evaluation Tools

The websites accessibility has improved through the years, but it isn’t enough. Many developers are not yet aware of the fact that the code they wrote doesn’t make Web pages accessible for those who have some kind of disability. For many companies it is too expensive or takes too many time to make Web pages accessible, thus, they exclude this step in the development process.

Those who want to evaluate the accessibility of their website have two options, based on users or in experts [4]. The approach based on users is performed by real users, so they can test the website for themselves and verify what is wrong or imprecise, with the website. This could be very confined because the users just evaluate the accessibility of the website based on their own needs. This is very subjective and could not consider a major group of people, with other restrictions or needs. The evaluation based on experts can be done in two ways, manually or automatically [4]. The manual evaluation, as the name says is a manual inspection of the website by an expert developer, thus a more accurate assessment of the implementation of the guidelines is performed, than with the approach described above. Just by itself, this type of evaluation is not enough because humans are error prone. Not to mention that it is a slow process, even though the result is a very detailed assessment. So, it should be combined with other solutions. The automatic approach relies on a framework or a tool that has a software to evaluate accessibility standards. It is much more scalable than the others, but relies mostly on the implementation made of the standards and guidelines, and the knowledge that is embedded. Therefore, to have accessible Web pages, the most used guideline is the WCAG2.0, and being correctly implemented, the results can give an accurate estimative of the accessibility degree of the website. With these results, both users and developers can perceive how accessible the Web page actually is. A user can then decide whether to use or not the page.

Because of the importance of this, there are already some tools online that can perform this automatic evaluation of Web pages, each of them with its functionalities and characteristics. One of the functionalities tested in each tool was the post-processing analysis. This analysis consists on understanding if the evaluation done by the tool is before or after the source code has been processed by the browser. Therefore, to realize if the changes that JavaScript, for instance, does in a page are taken into account or, on the other hand, not considered at all. To perceive this, the code depicted in Figure 1 was created to test the evaluators for post-processing analysis. The objective of this code is to introduce an element image without an <alt> attribute into the HTML, through a JavaScript function. If the evaluator tool detects this fault (an image without an <alt> attribute) it means that the tool does post-processing analysis.

```html
<html>
<head>
<script>
    function ola()
    {
        var img = document.createElement("img");
        var body = document.getElementById("cenas");
        body.appendChild(img);
    }
</script>
</head>
<body id="cenas" onloaded="ola()">  
<p>Hello</p>
</body>
</html>

Figure 1 – Code used to test post-processing evaluation
```
I will summarize some of the most important tools that I found and their characteristics:

- **A-Checker**: This evaluator checks HTML pages for conformance with accessibility standards, not only the two version of the WCAG guidelines but other ones. It has many filters, where it is possible to choose between three levels of conformance (A, AA, AAA), different formats of the report (view by guideline or by line number), and validate code (HTML and CSS). It doesn’t have the option of post-processing. Relative to the input methods, it can receive URL links, file upload and pieces of HTML code. The results can be shown in different tabs - known, likely and potential problems - and if the option to validate HTML or CSS is checked, also shows those tabs. Points out the line of the evaluated element, how to repair and the technique being used.

- **TAW dis**: This tool analyses Web pages based on the W3C recommendation, specifically, WCAG guidelines in both versions (WCAG 1.0, WCAG 2.0). It can evaluate not only the HTML and CSS languages but also JavaScript. Using some filters, it is possible to choose between three levels of conformance (A, AA, AAA). It does not have a post-processing analysis option. Concerning the input methods, it only receives URL links. The results are presented tabbed by the four principles of the WCAG 2.0 guidelines. On each tab, ordered by success criteria and technique this tool lists every detected issue, along with the line number and a sign with the evaluated status. Further down, it shows the page code, being a bit difficult to understand how to solve the problems.

- **Total Validator**: Total Validator, among other things, is an HTML validator delivered as a desktop application. It considers both versions of the WCAG guidelines (1.0, 2.0) and all levels of conformance (A, AA, AAA). It does not have a post-processing analysis option. It provides the option to upload a file as input and the output can be automatically saved to a file. The code that has been evaluated for each technique is displayed on the output file.

- **WAVE**: Wave was developed by WebAIM and is a free service. The only form of input is by URL link. The results are highlighted on the page that is evaluated, and a sidebar with the WAVE detailed results will be shown, allowing the user to select a specific issue that will be automatically shown in the page. The standards used are the W3C, including WCAG and WAI-ARIA, in all levels of conformance. This tool does post-processing analysis. There is also a Chrome plugin available.

- **AccessMonitor**: This evaluator checks the HTML content through both WCAG guidelines (1.0, 2.0) and was developed by FCT (Fundação para a Ciência e Tecnologia - Portugal). It does not have a post-processing option. As input methods, it can receive URL links, file upload and pieces of HTML code. In the results page, an accessibility degree is granted, and a summary table with all tests performed by level and their status is displayed. A second table is displayed further

---

9 [https://www.totalvalidator.com/](https://www.totalvalidator.com/)
10 [http://wave.webaim.org/](http://wave.webaim.org/)
detailing the results, including the code and the respective documentation by WCAG.

- **A-Tester**\(^\text{12}\): Another evaluator tool, which is based on WCAG 2.0 guidelines and the level of conformance AA. The input method is a URL link. This evaluator does not have filters nor post-processing option. The presentation of results is very simple, displayed in two sorted tabs, one by the original result and the other by WCAG success criteria. For every issue, it shows two frames, one with the code segment and the other with a page segment highlighted.

- **Accessibility Developer Tools (Google)**\(^\text{13}\): This tool is delivered as a plugin for Chrome, developed by Google Accessibility. The page that is evaluated is the browsed page, and the results are shown in a sidebar pane in the Elements tab from the developer tools menu option. It does post-processing analysis, and the guidelines followed aren’t described.

- **QualWeb**\(^\text{14}\): This tool was developed at FCUL, and it evaluates Web pages according to WCAG 2.0 guidelines. The input method is a URL link and has the four following filters: delivery context (mobile or desktop), JavaScript pre or post-processing analysis, CSS checking of validation techniques (on or off) and finally, the results presentation method (success criteria or techniques). The resulting area is divided into two sections. The first one is a statistical table showing a summary of the number of techniques that have passed, failed and issued warnings, as well as the total number of elements evaluated. This section also calculates the percentage of a page’s accessibility based on the previous results. The second section describes in detail the technique or criteria that is being evaluated, corresponding code and the outcome (pass, fail or warning), suggesting a way to solve the issue.

No matter how sophisticated these evaluators are, none of them can repair the code that is not compatible with the accessibility guidelines, neither validate the semantic analysis of the techniques, thus, requiring the user to manually check or correct the elements.

In the next subsection the Web accessibility evaluator tool, QualWeb will be described in detail. For the purpose of my thesis, it will be used for testing my work by implementing on it, the developed features.

### 2.2.3 QualWeb

As mention before, QualWeb [4] is a tool to evaluate the accessibility of a Web page, using specific standards issued by W3C, the WCAG 2.0 guidelines. This assessment will provide an approximated level of accessibility of the page, expressed as a percentage.

The evaluation process starts by providing a Web page URL to the tool. The execution can be customized by choosing:

- the delivery context for the analysis (Desktop or Mobile);
- the CSS content evaluation (On/Off);

\(^\text{12}\) [http://www.evaluera.co.uk/](http://www.evaluera.co.uk/)

\(^\text{13}\) [https://chrome.google.com/webstore/detail/accessibility-developer-t/fpknkjcljcfencdbgkenhalqipecmnb](https://chrome.google.com/webstore/detail/accessibility-developer-t/fpknkjcljcfencdbgkenhalqipecmnb)

Chapter 2. Related Work

- JavaScript evaluation (Pre or Post-Processing);
- which presentation method to use (Success Criteria or Technique).

After the process execution, the results of the evaluation are displayed in two sections. In the first one, the total number of evaluated elements is shown, and a percentage of those who had passed, failed or issued warnings. The level of accessibility referred above is calculated based on the percentage of the passed evaluated elements against the total. In addition to this summary, it is also possible to see the techniques in more detail. The second section shows the technique description, the affected element or the lack of it, the status (pass, fail or warning) and shows a piece of the code where the element resides. These guidelines only refer to HTML and CSS tags. The current version of QualWeb implements WCAG 2.0 guidelines and includes 34 HTML and 13 CSS techniques, providing the essential guidelines to check the accessibility of a website.

This framework is composed by three major groups: the evaluator core, the browser processing simulator (BPS) and the interaction simulator (IS). The architecture of QualWeb is depicted in Figure 2.

**Evaluator Core**

The BPS (browser processing simulator) receives the URI (Uniform Resource Identifier) of the page that needs to be evaluated and processes it. If the page has more than one state, the DOM (Document Object Model) is sent to the IS (interaction simulator), which simulates the multiple states of the Web page. The IS simulates a new state for any change of the page’s DOM, with JavaScript, for example, without modifying the URL. Then, after all states being simulated,
Chapter 2. Related Work

the IS returns them to the BPS for processing. Posteriorly the processed DOM is sent to the core for evaluation. So it’s the DOM that’s evaluated by the core, which assesses the quality of the RIA. A CSS pre-processor is used to obtain all CSS from the page, extracting it from the DOM tree, whether it is internal, external or in-line. Next, the available WCAG 2.0 techniques are used to perform the evaluation. There are three output formats for the results: EARL (Evaluation and Resource Language), JSON and CSV (Comma-separated values). The EARL is a standard format for accessibility reports. The CSV is adequate for statistical purposes. Finally, the JSON format saves all the results so they can be used by web services.

**Browser Processing Simulator**

This module simulates a Web browser processing through PhantomJS. The PhantomJS is a headerless browser, meaning that it hasn’t a graphical representation, so it is a command-line interface and uses WebKit. WebKit is an open source software, for Web page’s rendering in Web browsers. Therefore, it’s possible to rapidly process Web pages sequentially with this software, promoting large scale evaluations, as it doesn’t need to deal with a GUI (Graphical User Interface).

**Interaction Simulator**

To cope with RIA’s challenge, an interaction simulator was integrated into QualWeb. This component is responsible for simulating the user’s actions and triggering interactive elements in the page. As a result, the RIA’s different states are obtained in DOM format. To perform this simulation, JQuery is used to interact with the clickable elements (ex.: buttons, divs, inputs, etc.). Through these clickable elements, the program reaches the children states from each state. Then they are processed by BPS, and for every child state, a verification occurs to determine whether that child state has already been processed. If not, the BPS processes the child state and goes to the next. At the end, a graph with all interaction of the RIA states is obtained.

The QualWeb supports three types of evaluation:

- **E1** – Evaluation performed before the Web browser processing, where both simulators, BPS and IS are turned off. This evaluation is called pre-processing, and it’s the most used by this kind of tools.
- **E2** – Evaluation performed after browser processing, without considering the different Web page’s states, where the IS is turned off, but the BPS is active. This evaluation is called post-processing and is a type of evaluation less considered by most evaluation tools.
- **E3** – Evaluation performed after browser processing, considering the different states of the pages, where both IS and BPS are active. Currently, this evaluation is not fully functional.

2.3 **Content Management Systems**

With the fast evolution of the WWW and the emergence of several website development tools, the number of pages created increased exponentially.

Any person can now create a website and manage it, even without any knowledge about programming or Web development. This is only possible due to the existence of Content
Management Systems, called CMSs. There is a great variety available, each one with their functionalities, designs and platforms. Apart from their differences, they all have the same goal: to facilitate the creation, management, maintenance and publishing of websites. Often these tools have default pages, or templates, already structured with specific functionalities and layouts, according to their usage so the user can take advantage of them. These templates are created by Web experts, following certain rules and guidelines, to ensure that they work well and have all the components requested by the users. The user can then take the template and insert his own content, edit it, and managing it as he pleases. Because the user can change the source code, the websites uploaded to the Web, are not only many times incorrectly programmed, but also not according to Web accessibility guidelines.

Additionally, these tools have some advantages. They provide user support, security, additional plug-ins, well defined documentation, and so on. As a consequence, these tools make a Web page more practical and modular, as well as easier for a normal user to manage and maintain.

Some of these tools already implement several WCAG guidelines, as well as WAI-ARIA specifications, so the generated websites can be accessible to all. Besides all the advantages, there is a barrier regarding accessibility as it was mentioned in [1], [2]. The reason for this is that these tools, as mentioned above, allow the user to change the source code of the page, not just to insert content, but to change its structure. In ideal conditions, the generated templates perform well, but when the user modifies the source code without any knowledge, the accessibility of the page can diminish. There is, however, an effort to mitigate this problem as on [8].

There are already many tools to manage content, both on desktop applications and on the Web. For the purpose of this work, I’ll refer just one, known as WordPress\footnote{https://en.wordpress.com Last accessed on January 2018}. The reason to choose this tool is that it is the most used to build websites, in an estimated 29% of the whole Web. A study has been performed in [8] that demonstrates that WordPress is one of the CMS with a large number of accessible Web pages, being other advantage for choosing this CMS. Through Web page segmentation it is possible to segment the page and perform an analysis of each section of the page to classify its role. After this classification, the results can be assessed in a Web accessibility evaluation tool to understand the Web page’s accessibility.

WordPress is a platform with a free version, available on the Web and as a desktop application. To get more functionalities or support to the user it also has a paid option. Initially, this tool was meant for blogging websites, but today, it expanded and serves all purposes. Like other CMSs, it allows the user to create, edit, manage and personalize the content. It uses plugins to create and organize the information and allows the owner to see stats of his website. This tool in particular already has taken steps to improve the accessibility of the Web pages, adopting the WAI-ARIA specifications, as well as the WCAG guidelines. The available templates already use WAI-ARIA attributes and they are reviewed to ensure they are accessible.

### 2.4 Web Page Segmentation

Web page segmentation is the process of dividing a page into visual and semantically coherent segments, also known as blocks [9].

This division can be based on several approaches [10], such as:
• Structure - This approach uses only the HTML tags as they are in the Web page source code or in the DOM of the page after browser processing;
• Layout – Focuses on the visual properties of the page;
• Image – Takes an image of the page and applies techniques of image processing to detect blocks;
• Text – Retrieves segments from Web pages based on the text properties, such as excerpt similarity, clustering and so on;
• Fixed-length – Removes all the semantic information (e.g. tags) from the page and then uses fixed-length algorithms to segment the Web page;
• Hybrid – This uses both structure and layout approaches so it can take advantage of the syntactic and visual information. The hybrid approach can combine any other two approaches from those mentioned above.

Later in this chapter, a review will be given of recent developments in this area and the type of approaches that those segmentation tools use.

As is described in [9], identifying segments can be very useful for different areas. These areas can be, mobile Web [11], [12], [13], [14], [15], [16], [17], voice Web, Web page phishing, duplicate detection, information retrieval [11], [16], [17], image retrieval, information extraction [11], [12], [16], [17], [18], [19], [20], user interest detection/sense making tasks, visual quality evaluation, Web page clustering [12], [19], [21], caching, archiving [22], publishing dynamic content, semantic annotation and finally, Web accessibility.

The identification of Web page segments can help rearrange the content for it to be properly displayed on small screens, like mobile devices. The information can be filtered by screen readers, thus facilitating the content interpretation for blind users. Even search engines can use segmentation to find implicit information in order to provide better search results, excluding the irrelevant content.

Several segmentation algorithms have already been proposed, which use different techniques: graph based algorithms, ranking algorithms mainly based on PageRank, heuristics-based, rule-based, text-based, image processing, psychology based, machine learning, clustering-based and pattern matching, and some custom algorithms [9]. The only conclusion that can be made, at this point, is that only a few of these algorithms are mostly used, but it doesn’t mean they work better than the others.

These algorithms can be grouped into two groups: top-down page segmentation or bottom-up page segmentation [23]. The top-down page segmentation group [10], [11], [12], [16], [17] considers the Web page as a block and divides this block into smaller blocks using different components, obtained throughout the content of the page. The bottom-up page segmentation group considers the leaf nodes of the DOM tree representation as atomic content units and then tries to group these units into coherent blocks.

### 2.4.1 Web Segmentation Tools

In this subsection, I will revisit the tools that I have searched to understand the evolution state of the Web segmentation development.

D. Cai et al., 2003 proposes in [11] the VIPS (Vision-based Page Segmentation) algorithm, which is one of the most used top-down approaches. This algorithm also serves as a base for other algorithms. The VIPS algorithm is based on the visual representation as well as on the DOM of the page. Its purpose is to extract the content structure of a Web page. This algorithm has three
steps: block extraction, separator detection and content structure construction. In the block extraction, the Web page is recursively divided into blocks by using a number of heuristics; in the separator detection, the blocks are put into a pool for visual separator detection; and the last step, the content structure construction, is basically the construction of the Web page structure in a tree. This approach can benefit information retrieval, information extraction and automatic page adaptation.

E. Akpinar et al., 2012 proposes in [16] a new version of VIPS algorithm, the extended VIPS, which improves the second step, the visual block extraction. They claim that there are some missing definitions and ambiguities in the algorithm, and if the block structure of a Web page is constructed false, the separators and content structure are constructed false, too. No other changes were made to the algorithm. There is a follow up work developed in [17] to improve even further the extended VIPS, which introduces the role attribution after the segmentation process. Both these approaches can benefit information retrieval, mobile device adaptation and information extraction, among others.

P. K. S. Kadam et al., 2016 proposes in [18] the utilization of visual features of a Web page to extract data from the Web pages. To make the system efficient, the authors propose, combining the visual features with the non-visual information, like symbols and tags. The authors approach is based on three steps, the first one uses the VIPS algorithm to extract the content structure of the Web page and construct a tree with visual blocks for each responsive page. The second step consist on locating the information region in that tree, based on the position features. And the third one, is to extract the information records of the data region, based on the layout and appearance characteristics. This work is for data extraction, but a limitation is that it can only process Web pages that just contain one data region.

A. Sanoja et al., 2013 proposes in [10] the Block-o-Matic algorithm, which is an hybrid approach. It analyses the page from the structural perspective using the VIPS algorithm and evaluates the visual information with the Geometric Layout Model. This model considers three aspects: the content, the geometrics and the logic of the page. The segmentation process also has three phases, which are the page analysis, the comprehension and the reconstruction. This tool also allows a manual segmentation. The result of the comparison between the manual and automatic segmentation has to be the same. It is a top-down approach, because it walks through the DOM starting on the root of the tree until it gets to the leaves and it makes decisions along the way.

K. Wah et al., 2014 proposes in [24] a vision-based Web page segmentation algorithm with Gomory-Hu Tree. This approach also starts with the VIPS algorithm to segment the page and then, the number of different blocks is achieved. The content blocks that are retrieved have, not only main content information, but also noisy information, such as advertisement and so on. What it proposes is to find the similarity weight to retrieve the main content of that page using the Gomory-Hu tree.

S. Singhal et al., 2013 proposes in [12] an hybrid Web page segmentation approach, based on VIPS. After processing each node in the DOM tree, it separates the relevant information and assembles coherent regions. This could be helpful to information extraction, Web mobile and clustering, because only the important information is obtained. It is also a top-down approach.

A. Sanoja et al., 2012 proposes in [22] another hybrid tool, that implements a modified version of the VIPS algorithm, aiming at enhancing the precision of visual block extraction and the hierarchy construction. It analyses Web pages based on their DOM tree information and their visual rendering, through image segmentation.
Chapter 2. Related Work

L. Yi et al., 2003 in [19], focuses on detecting and eliminating noisy content of a Web page, to improve the performance of Web mining. The cleansing technique used is based on the observation that commercial websites tend to follow a certain layout or presentation style. The parts of a Web page whose layout contents also appear in many other pages are more likely to be noisy information.

H. Ahmadi et al., 2008 proposes in [13] a method to adapt Web pages so they can fit into smaller screens and to increase browsing efficiency, as well as user satisfaction. It uses the structural and visual layout information on the page to detect related content. This approach can be divided in two phases: related content detection and layout adaptation. There are a set of heuristics to identify the main content of the Web page (relying on a model of a Web page, containing top, menus, main content, and bottom). Then the blocks in the main content are identified.

Y. Chen et al., 2003, 2015 proposes in [14], [15] a page adaptation technique that splits existing Web pages into smaller, logically related units. It is an algorithm that identifies the higher-level content blocks such as the header, footer, sidebars and body. After this, they use two approaches to divide even more the main content into a number of blocks. The first approach is the explicit separator detection which can partition the block even further using three types of separator: &lt;hr&gt; tag, border properties of &lt;table&gt;, &lt;td&gt; and &lt;div&gt; elements, and images. The second approach is the implicit separator detection: implicit blank areas created intentionally by the authors to separate the content. Both approaches are focused on small screens. The disadvantage of this method is that it follows only one type of template.

B. Liu et al., 2003 tries in [25] tries to mine topic-specific knowledge on the Web, with the goal of helping people learn in-depth about one specific topic. It starts by identifying blocks in a Web page, using the VIPS algorithm. Then, through heuristics, the blocks are categorized into three categories: navigation bar, navigation list and content blocks. These categories are then used to filter unnecessary contents. This approach just considers these three categories, which is very restrictive.

M. Kovacevic et al., 2002 proposes in [21] to build a representation for a Web page in which objects are placed on a well-defined tree hierarchy, according to where they belong in an HTML page structure. It tries to recognize some common areas in the Web page, such as the header, footer, left and right menus and the centre of the page. A group of heuristics has been defined to identify these common areas. Those heuristics are based on a rigid abstraction of the visual representation of the page. This approach also focuses on a Web page template.

A summary of these segmentation tools is depicted in Table 2. As we can see many of these approaches are based on the VIPS algorithm mostly because the algorithm divides the Web page in a visual level, retrieving the content structure based on the visual presentation. This algorithm simulates how the user sees a Web page, and with any doubt, that is a great advantage.

Some of them also rely on a single template for the Web page segmentation process. Personally, I believe this is the greatest disadvantage in this theme. For solving problems like information retrieval, information extraction, accessibility or automatically adapt pages layout, we can’t rely on only one kind of module. It cannot be applicable to all Web pages since it only considers a specific type of template. If a page does not comply with that template, the segmentation process does not succeed. Therefore its usability is not universal. So far, the only work found related to Web page segmentation concerning the use of CMS is described in [17].


<table>
<thead>
<tr>
<th>Ref.</th>
<th>Year</th>
<th>Group (Top-Down/Bottom-Up)</th>
<th>Approach</th>
<th>Algorithm</th>
<th>Uses VIPS</th>
<th>Uses Template</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>[16]</td>
<td>2012</td>
<td>Top-Down</td>
<td>Visual representation and DOM</td>
<td>Rule-Based</td>
<td>Yes</td>
<td>No</td>
<td>Information retrieval, Information extraction, Mobile page adaptation</td>
</tr>
<tr>
<td>[17]</td>
<td>2017</td>
<td>Top-Down</td>
<td>Visual representation and DOM</td>
<td>Rule-Based</td>
<td>Yes</td>
<td>No</td>
<td>Information retrieval, Information extraction, Mobile page adaptation, Web accessibility</td>
</tr>
<tr>
<td>[18]</td>
<td>2016</td>
<td>Top-Down</td>
<td>Visual content features</td>
<td>Rule-Based</td>
<td>Yes</td>
<td>Yes</td>
<td>Information extraction, Automatic page adaptation</td>
</tr>
<tr>
<td>[10]</td>
<td>2013</td>
<td>Top-Down</td>
<td>Hybrid</td>
<td>Rule-Based</td>
<td>Yes</td>
<td>No</td>
<td>Information retrieval, Mobile Web, Page classification, Web accessibility, Archiving, Phishing, Clustering, Information extraction, Automatic page adaptation, Duplicate detection</td>
</tr>
<tr>
<td>[24]</td>
<td>2014</td>
<td>Top-Down</td>
<td>Hybrid</td>
<td>Clustering</td>
<td>Yes</td>
<td>No</td>
<td>Ranking, Duplicate detection, Content extraction</td>
</tr>
<tr>
<td>[12]</td>
<td>2013</td>
<td>Top-Down</td>
<td>Hybrid</td>
<td>Rule-Based</td>
<td>Yes</td>
<td>No</td>
<td>Mobile Web, Information extraction, Clustering, Automatic page adaptation</td>
</tr>
<tr>
<td>[22]</td>
<td>2009</td>
<td>Top-Down</td>
<td>Dom and Image</td>
<td>----</td>
<td>Yes</td>
<td>No</td>
<td>Archiving, Page comparison, Crawling, Information retrieval</td>
</tr>
<tr>
<td>[19]</td>
<td>2003</td>
<td>Top-Down</td>
<td>Hybrid</td>
<td>Cleaning</td>
<td>No</td>
<td>No</td>
<td>Web mining</td>
</tr>
<tr>
<td>[14]</td>
<td>2003</td>
<td>Top-Down</td>
<td>Hybrid</td>
<td>----</td>
<td>No</td>
<td>Yes</td>
<td>Mobile Web, Automatic page adaption</td>
</tr>
</tbody>
</table>
2.5 Summary

This chapter provided an overview of the topics that are important for the understanding of this thesis, as well as, a state of the art review. It starts by introducing Web evolution, centred on the semantic Web. Then an introduction to Web accessibility describes the standards used to make accessible Web pages and the tools that perform an assessment of Web page accessibility. One of the tools is described in more detail because it is going to be the platform that will incorporate the framework that is being developed in this project. Afterwards, a short description of the CMSs is given, as well as an introduction to the specific one studied in this work. The final section summarizes my research about the present development status of Web page segmentation.
Chapter 3

Analysis

This section will focus on the analysis of the tools chosen to develop this work. It will also introduce the menu element concept. A description of the extended VIPS segmentation tool algorithm will be given. Next, a new version of the extended VIPS, namely Web service VIPS is introduced. Then, the criterions for the use of CMSs are presented. Finally, an introduction of the menu definition will be given, as well as the first analysis using the extended VIPS algorithm. The last section summarizes this chapter.

3.1 Extended VIPS

Through my analysis and the obtained results in Table 2, I determined that VIPS is the most commonly used algorithm and was the better segmentation tool option for what I intended. Therefore I chose to use the extended VIPS version, developed by Elgin Akpınar and Yeliz Yesılıada [17]. Not only was the tool with better results in Web page segmentation, but also this version of VIPS can assign a role to each block of the page, as well as the elements inside the blocks. This feature was a great advantage and the main reason for choosing extended VIPS instead of other algorithms. Another criterion to support my choice was the fact that VIPS simulates how a user understands the Web layout structure based on his visual perception [16], which is the semantic structure of a Web page.

The extended VIPS algorithm aims to find visual blocks in a Web page through visual representation (visual cues and tag properties of the nodes) as well as the page’s source code (DOM tree) [17]. Before describing the segmentation process of the extended VIPS algorithm I need to introduce some concepts:

1. Blocks – These blocks can either be a basic object (the leaf nodes in the DOM tree) or a set of basic objects. They compose the final semantic structure generated by extended VIPS. The nodes in that structure do not necessarily correspond to the nodes in the DOM tree. There is a finite set of blocks which do not overlap each other.

2. Separators – A separator represents the horizontal or vertical lines and spaces in a Web page that visually cross with no blocks. Each separator has a correspondent weight, which represents the relation between the visual elements surrounding that separator. There are a finite set of separators both horizontal and vertical.

3. Degree of Coherence (DoC) – It is an integer value, between 1 and 10, assigned to each block in the structure to measure how coherent the content is in the block, based on visual perception. In the semantic structure, the DoC of the child is not smaller than that of its
parent. It is also possible to achieve different granularities of the content structure by pre-defining the PDOC (Permitted Degree of Coherence).

The segmentation process is divided into three main steps, visual block extraction, separator detection and content structure construction, described below. These three steps are considered a single round of the segmentation process. In each round, the Web page is segmented into several big blocks and the hierarchical structure of this level is recorded. For each of those blocks the three steps are recursively executed until the granularity requirements are achieved, that is, until there are enough small blocks whose DoC values are greater than a pre-defined PDoC. After the segmentation process, there are two additional steps: rule generation and role detection. In Figure 3, the system’s architecture is illustrated.

![Overall System Architecture](image)

**Figure 3 - Extended VIPS Architecture. Source: M. E. AKPINAR and Y. YEŞILADA, “Discovering Visual Elements of Web Pages and Their Roles: Users’ Perception,” vol. 29, no. 6, 2017.**

### 3.1.1 Extended Visual Block extraction

The goal of this step is to find all suitable visual elements in the current Web page. In general, every node in the DOM tree can represent a visual block. However, as established, many Web pages do not follow W3C specifications, so the DOM tree cannot always reflect the true relationship of the different DOM nodes. Moreover, although the DOM provides significant information about the structure of the Web page, it is not enough. For that reason, the extended VIPS algorithm uses the combination of both DOM structure and visual rendering, which provides much richer information for the segmentation process, mainly to this first step.

The extended VIPS algorithm starts by constructing the DOM structure and then labels all the nodes in the Web page. This label is based on the element’s visibility in the page layout, the line-breaks they produce and their children nodes. With these considerations and node attributes (e.g.: tag, color, text, font size, etc.) the algorithm decides if the node should be represented as a single element in the visual element tree or needs to be further segmented.

To each visual block extracted, a DoC value is set. At the end, all appropriate nodes are found and put into a pool, which contains the visual blocks. The Web page illustrated in Figure 5 is an example website to demonstrate the extended VIPS segmentation process.

---

Chapter 3. Analysis

It is possible to observe in Figure 4 the final result after the first level of segmentation, where three blocks are extracted from the first block (VB.1), which is the root element, ‘body’. This extraction is made based on heuristic rules [16], that determine if the present node should be further segmented or not. After the first level, three blocks, VB.1.1, VB.1.2 and VB.1.3 are put into a pool. At the end of the process, each visual block is listed in a visual element tree, as shown in Figure 7. It is possible to compare the visual element tree state after the first level of segmentation in Figure 7, with the DOM tree extracted from the Web page’s example mentioned above in Figure 6. It is possible to observe that some nodes in DOM tree do not appear in the visual elements tree.

Figure 5 – The Web page example

Figure 4 – The layout structure and vision-based content structure
Chapter 3. Analysis

Figure 7 – Visual element tree of Web page’s example

Figure 6 – DOM tree Web page’s example
3.1.2 Separator detection

The separator detection step aims to detect the separators between the visual elements, extracted in the previous step. These visual blocks are put into a pool. As mentioned separators are horizontal or vertical lines and spaces in a Web page that do not visually cross with the blocks in the pool. From a visual perspective, separators could help distinguish different semantic areas within a Web page.

This algorithm starts with only one separator corresponding to the borders of the pool. For each block in the pool, its relation with each separator is evaluated. This relation could be:

a) the block is contained in the separator, then the separator is divided;

b) the block crosses the separator, then the separator’s parameters are updated;

c) the block covers the separator, then the separator is removed.

Finally, the separator that stands at the border of the pool is removed. Then, based on rules that consider the visual difference between its adjacent blocks, a weight is attributed to each separator.

Figure 8 is an example of how horizontal separators are detected. For the purpose of this example instead of a line, consider the separators as the blue opaque block represented initially with S1 in the right superior corner of the block. The thickness of the block does not represent its weight. The process starts by selecting the first separator, which corresponds to the whole pool. Then when the first block VB.1.1 is selected, according to the relation between the separator and the block evaluation mentioned above, the separator is divided because the block is contained in the separator S1. This evaluation is going to be made for each visual block in the pool, VB.1.1, VB.1.2 and VB.1.3. At the end, the separator that stands at the border of the pool, S1, is removed. So there are only two horizontal separators after the first level of separator detection step.

3.1.3 Content structure construction

After the blocks extracted and the separator’s weights are set, the content structure construction step can be initiated. The construction process starts from the separators with the lowest weight. The blocks beside these separators are merged forming a new block.

This is a recursive process, continuing in ascending order of weights of separators. For each new block a DoC is generated based on the maximum weight of the separators in the block’s region. At the end of this process, each leaf node is evaluated to check if the granularity requirement is met. For every node that fails, the visual block extraction step is called to construct the sub content structure within the node. When all nodes meet the granularity requirement or all the leaves are reached, this process ends and the Web page’s semantic structure is obtained.

This generated hierarchical tree structure is different from the DOM structure of the Web page, as I already mentioned. This is due to the VIPS algorithm, which eliminates invalid nodes and groups the adjacent virtual text nodes and inline nodes into a single visual element so the visual element’s tree is typically simpler than the DOM tree.
In Figure 9 the output of VIPS segmentation process is shown. This result contains the generated semantic tree, where each level represents segmentation results in different levels of granularity. As illustrates, VB.1 corresponds to the body node of the Web page and is the root node of visual block’s tree. It has three children (VB.1.1, VB.1.2 and VB.1.3) as the first level of segmentation. Each of these children also has children blocks, which corresponds to the second level of segmentation. The Web page can be recursively segmented into several levels depending on the granularity requirement. The output also contains the tag of each element, the font size, the corresponding DoC and the Path in the semantic structure to the visual element.

<table>
<thead>
<tr>
<th>Block</th>
<th>Tag</th>
<th>DoC</th>
<th>Font</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>VB.1</td>
<td>BODY</td>
<td>/HTML/</td>
<td>2</td>
<td>/HTML/BODY</td>
</tr>
<tr>
<td>VB.1</td>
<td>HEADER</td>
<td>2</td>
<td>16</td>
<td>/HTML/BODY/HEADER</td>
</tr>
<tr>
<td>VB.1.1</td>
<td>A</td>
<td>4</td>
<td>16</td>
<td>/HTML/BODY/HEADER/NAV</td>
</tr>
<tr>
<td>VB.1.2</td>
<td>NAV</td>
<td>4</td>
<td>27.6</td>
<td>/HTML/BODY/HEADER/NAV/NAV</td>
</tr>
<tr>
<td>VB.1.2</td>
<td>COMPOSITE</td>
<td>2</td>
<td></td>
<td>/HTML/BODY/COMPOSITE</td>
</tr>
<tr>
<td>VB.1.2</td>
<td>DIV</td>
<td>11</td>
<td>16</td>
<td>/HTML/BODY/COMPOSITE/DIV</td>
</tr>
<tr>
<td>VB.1.2</td>
<td>DIV</td>
<td>10</td>
<td>16</td>
<td>/HTML/BODY/COMPOSITE/DIV/DIV</td>
</tr>
<tr>
<td>VB.1.3</td>
<td>FOOTER</td>
<td>2</td>
<td>16</td>
<td>/HTML/BODY/FOOTER</td>
</tr>
<tr>
<td>VB.1.3</td>
<td>A</td>
<td>11</td>
<td>16</td>
<td>/HTML/BODY/FOOTER/NAV</td>
</tr>
<tr>
<td>VB.1.3</td>
<td>SPAN</td>
<td>4</td>
<td>16</td>
<td>/HTML/BODY/FOOTER/P/SPAN</td>
</tr>
</tbody>
</table>

Figure 9 – Output of VIPS segmentation process

After the semantic structure is completed, there are two additional steps in the extended VIPS version regarding the automatic role detection of visual elements described above.

3.1.4 Rule Generation

A role is a structural type of a visual element which semantically shows its usage and overall interaction purpose in a Web page [17].

In order to describe and capture the roles, an ontology called eMine\(^{17}\) was created based on the WAfA (Web Authoring for Accessibility) ontology [17]. WAfA is a comprehensive knowledge base created to capture a shared understanding of visual elements of Web pages [26]. However, in order to address some problems in the WAfA ontology, eMine was extended and improved relating to coverage and definitions of concepts defined. The eMine ontology was extended with the help of concepts from WAI-ARIA and HTML5 which provides a rich set of roles.

Even if WAI-ARIA and HTML5 provide much stronger and better list of roles, they still do not cover deep understanding of the visual elements. In case that the source code is annotated correctly, that could facilitate the identification of the element’s role. But, when the source code does not contain this assistive annotation role, it is still necessary a mechanism able to understand the semantic role of that visual element. Therefore, the eMine ontology not only defines and lists the roles of visual elements, as well as describes their properties. This ontology is then transformed into a rules file to be used in the role detection step.

Knowledge Representation

Each Web element has a set of attributes which affect its representation in the Web page. However, it is possible to categorize the roles of visual elements by understanding these attributes

\(^{17}\) [http://emine.ncc.metu.edu.tr/ontology/emine.owl](http://emine.ncc.metu.edu.tr/ontology/emine.owl) Last accessed on January 2018
and their differences. With this in mind, a set of properties that affects how the visual elements are used and represented in a Web page were selected [17]:

- Underlying tag (HTML/HTML5);
- Children and parent elements in the underlying DOM tree;
- Size of the element;
- Border and background color of the element;
- Position of the element;
- Some attributes including onclick, for, onmouseover, etc.;
- CSS styles (font size, color, etc.) of the elements;
- Some specific keywords which appear in the textual content of the element;
- Some specific keywords which appear in the id, class, role, src, background-image attributes of the element.

These properties are used to describe the visual elements. For example, in Figure 11, the role of V.B.1.1.2 is LinkMenu which represents a menu. According to the LinkMenu definition in the eMine ontology, a menu element is likely to have the following properties:

- It may have ul, ol, dl or nav tags;
- Its ID, class, source, value or name attribute may contain "menu" or "nav" keys;
- It usually has different background color than its parent;
- It appears in a Header, Footer, Sidebar or Nav element;
- It contains MenuItem children;
- It appears at the top of a Web page;
- It is not atomic, it contains children Web elements (MenuItems);
- It may be in different sizes, such as full width, etc. but its height is generally short.

This does not guarantee that the role identification is 100% accurate because sometimes developers do not assign the correct tag to the element or some of their attributes could be missing or wrong. The properties mentioned above just increase the probability of being the correct role for the visual element. For example, the visual element with the id value of “nav” is more likely to be a LinkMenu than a visual element with the id value of “footer”. The combination of all properties is used to identify and classify the visual element’s role.

**Heuristic Rules**

After defining the eMine ontology, it is automatically processed to construct heuristic rules in order to be applied to visual elements in the semantic structure obtained in the previous step. The ontology is then converted to Jess, to be used in the role detection step. An example of how the rules are defined in the Jess file is depicted in Figure 10. This rule is applied to a block object and is divided into two parts: condition and the action if the condition is satisfied. The condition is triggered if the visual element’s id contains the word “nav”. If the condition is fired, then the likelihood score for LinkMenu is incremented by 5, which is the weight for this rule.

![Figure 10 – Rule definition in Jess rule engine](image-url)
### 3.1.5 Role Detection

For detecting the roles, the extended VIPS uses the rule engine, Jess, for Java, as mentioned above [17]. After the process of segmentation, the visual elements in the semantic tree are considered to be individuals. Each of the individual visual element is converted to facts in Jess rule engine. Every fact has a template, which defines its name and the set of attributes, which corresponds to the visual element properties. These templates, global variables and defrules are stored in the CLP (Jess file) file and they are generated using the eMine ontology.

In the end, for each visual element, there are a set of likelihood scores for each heuristics role. The role with the highest value is the most likely role for the visual element. This process is repeated for each visual element in the Web page. After each iteration, the semantic tree is updated to represent the roles assigned to each visual element. The output of segmentation process and role annotation is illustrated in Figure 11.

<table>
<thead>
<tr>
<th>Block</th>
<th>Tag</th>
<th>DoC</th>
<th>Font</th>
<th>Path</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BODY</td>
<td></td>
<td></td>
<td>/HTML/BODY</td>
<td>Body</td>
</tr>
<tr>
<td>VB.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VB.1.1</td>
<td>HEADER</td>
<td>2</td>
<td>16</td>
<td>/HTML/BODY/HEADER</td>
<td>Header</td>
</tr>
<tr>
<td></td>
<td>VB.1.1.1</td>
<td></td>
<td></td>
<td>/HTML/BODY/HEADER/NAV</td>
<td>Logo</td>
</tr>
<tr>
<td>VB.1.1.2</td>
<td>NAV</td>
<td>4</td>
<td>27.6</td>
<td>/HTML/BODY/HEADER/NAV/NAV</td>
<td>Header/LinkMenu</td>
</tr>
<tr>
<td>VB.1.2</td>
<td>COMPOSITE</td>
<td>2</td>
<td></td>
<td>/HTML/BODY/COMPOSITE</td>
<td>Container</td>
</tr>
<tr>
<td></td>
<td>DIV</td>
<td>11</td>
<td>16</td>
<td>/HTML/BODY/COMPOSITE/DIV</td>
<td>Container</td>
</tr>
<tr>
<td></td>
<td>DIV</td>
<td>10</td>
<td>16</td>
<td>/HTML/BODY/COMPOSITE/SPANN</td>
<td>Footer</td>
</tr>
<tr>
<td>VB.1.3</td>
<td>FOOTER</td>
<td>2</td>
<td>16</td>
<td>/HTML/BODY/FOOTER</td>
<td>Footer</td>
</tr>
<tr>
<td>VB.1.3.1</td>
<td>A</td>
<td>11</td>
<td>16</td>
<td>/HTML/BODY/FOOTER/P/A</td>
<td>Title, Icon</td>
</tr>
<tr>
<td>VB.1.3.2</td>
<td>SPAN</td>
<td>4</td>
<td>16</td>
<td>/HTML/BODY/FOOTER/P/SPAN</td>
<td>Link</td>
</tr>
</tbody>
</table>

*Figure 11 – Output of VIPS segmentation process with role assignment*

The extended VIPS is open source and was implemented in the ACTF (Accessibility Tools Framework)\(^\text{18}\) of Eclipse Foundation. ACTF is a framework that serves as an extensible infrastructure, which allows developers to build a variety of utilities which can help evaluate and improve the accessibility of applications and content for disabled people. The IWebBrowser interface allows the access to the Web page element’s properties, as well as to control the Web browser, i.e., changing the textual content of an element.

However, for the purpose of my work, which included implementing VIPS in QualWeb in order to develop new WCAG 2.0 techniques, this version of VIPS failed. It was not possible to integrate the extended VIPS into QualWeb because its implementation depends on ACTF and it requires a browser to render the source code and get visual attributes of Web elements. With this in mind, I suggested to its authors the implementation of VIPS as a Web service, in which given an URL, the returned result would be a JSON with the semantic structure and roles.

### 3.2 Web Service VIPS

The Web service VIPS was developed by Elgin Akpinar and follows the same process as Extended VIPS but was developed in nodeJS and uses PhantomJS, which is a headless WebKit, allowing the access to Web page’s content and its DOM. A request example is depicted in Figure 12 and the response in Figure 13.

\(^\text{18}\) [https://www.eclipse.org/actf/](https://www.eclipse.org/actf/) Last accessed on November 2017
Chapter 3. Analysis

The rule file has the same configuration as in the previous version (Extended VIPS). But instead of Jess as rule engine, the node-rules module was used to detect the roles. This library enables to load defined rules from a text file. The way to program the rules changed, as it is shown in Figure 14, but the logic is the same as explained before. It is divided into two parts: condition and consequence. If the condition is fired, then the consequence is executed. In this case, if the visual element contains “nav” in the id attribute, then the likelihood value is incremented by 5.

```
{ "condition": function(R) {
  R.when(this && this.hasId && this.hasId.indexOf('nav') > -1);
},
"consequence": function(R) {
  this.reason['LinkMenu'].push('hasId:nav');
  this.roles['LinkMenu'] += 5;
  R.next();
},
"priority": 1,
"name": 'LinkMenu01'
}
```

Figure 12 – Request call example

```
POST http://localhost:8080/
{
  "url": "http://elginapinar.com",
  "width": 1920,
  "height": 1080,
  "agent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:27.0) Gecko/20100101 Firefox/27.0",
  "explainRoles": false
}
```

Figure 13 – Response to the request call example

Figure 14 – Rule definition in NodeJS rule engine
3.3 Content Management Systems

I decided to use the WordPress CMS, because it is the most used to build websites, with around 29%\(^{19}\) of the whole Web being WordPress websites. Additionally, it is one of the CMSs with a larger number of accessible Web pages, as it is referred in [8]. The WordPress generated templates employ WAI-ARIA attributes and HTML5 tags and they review every single template in order to be accessible according to the WCAG 2.0 heuristics. The main problem is the fact that users can change the template’s source code. By doing this, the template may no longer be accessible. Although this is a possibility, the templates can still be a better option rather than user-generated Web pages, because as they are based on templates, this could facilitate patterns discovery. This way, it is possible to improve the VIPS rule definition file.

I chose to analyze menus, because not only they are an important element in a Web page, but also to disabled people they can assist in the Web page’s navigation. Unfortunately, sometimes Web pages are not well programmed neither have into account the W3C guidelines, so screen readers can’t transmit the correct information to the end user. In an attempt to mitigate this problem, I wanted to insert new techniques into QualWeb, related to menus, so it could evaluate these elements and understand if they are accessible on the page.

3.4 Menus

The WCAG menu definition\(^{20}\) says that menus are used for navigation and to provide functionality which are critical parts of a Web page operability. I realized that the menu definition was a little vague and susceptible to different interpretations. It was not always easy to identify a menu in a Web page. My menu definition is that it is a fixed element in a Web page, composed by links, e.g. images, words or icons, allowing the user to navigate inside the page or to another Web page. I identify some visual features that helped me distinguishing menu elements on the Web page:

- A menu is a set of links;
- The link’s textual content is generally short, i.e. 1-5 words;
- The links can be or include images or icons;
- These links can navigate the user to another Web page’s section or to a different page;
- The background color of the menu could be different from the rest of the Web page;
- The size of the menu text could be different from the surrounding text;
- Menus can appear at any position on the Web page, but usually at least one menu appears at the top.
- Usually, a menu has a fixed position on all the website’s pages.
- Usually, a menu has fixed content, regardless of the day or the user that access the Web page.
- The bread crumb element is a menu, i.e. the sequential path since the first page, until the page where the user is.
- The icons in a slideshow or carousel elements can be considered menus, as well as the icons allowing the user to navigate between Web page’s sections.

---

\(^{19}\) https://en.wordpress.com Last accessed on January 2018

\(^{20}\) https://www.w3.org/WAI/tutorials/menus/ Last accessed on November 2017
My biggest difficulty here was the correct identification of menu elements. I exclude sets of news or images, even though they may check a few features mentioned above. In my opinion, they do not allow a person to navigate in the Web page, neither have a fixed content. Although I tried to establish these features to assist in the menu identification, I still have doubts in the element’s classification. Sometimes, the context where the menu is located in the Web page may assist in the identification of menu elements.

To be able to identify the features mentioned above, I started to analyze WordPress templates, regarding only to menus elements, focusing on the source code and how elements were displayed in a Web page. The templates developed by WordPress are accessible, accordingly to WCAG 2.0 guidelines, nevertheless, users can change these templates, disregarding good programming practices, which not guarantee that WordPress Web pages are accessible. WordPress uses HTML5 tags. Usually, menus have the tag `<nav>` and also use WAI-ARIA attributes so it is possible to specify different roles, i.e. `<nav role="navigation">`, which represents a navigational menu. Figure 15 shows one correct example from the several WordPress Web pages analyzed. The `<nav>` tag is used to point that is a navigational element and presents the role navigation, which can assist user agents that do not support HTML5 understanding the code structure. In Figure 16 an incorrect example is shown, where the tag used to represent a menu is a `<div>`.

![Figure 15 – Correct menu example, using WAI-ARIA attributes](image15.png)

![Figure 16 – Incorrect menu example](image16.png)
After these observations, I selected 30 generated WordPress Web pages randomly, listed in Appendix A, to serve as my training set. These 30 Web pages were retrieved from the official WordPress website, in the examples section. I inspected the source code as well as the visual aspect of each page, and identified the menus in those pages. For each Web page, I ran the extended VIPS and registered the results. Of all the 30 Web pages, there was one which could not be run by VIPS, so only the 29 remaining Web pages were in fact evaluated.

After, I compared the LinkMenu roles from extended VIPS with the elements I considered menus. I divided the data collection into three different groups:

- **True Positives (TP):** Element that I considered menu as well as the extended VIPS. The role attributed to this element is LinkMenu;
- **False Positives (FP):** Element that I did not consider to be menu unlike the extended VIPS, which attributes the LinkMenu role to the element or more than one role, i.e. LinkMenu/Container;
- **False Negatives (FN):** Element that I considered menu that is not detected by the extended VIPS. The role attributed to this element is not LinkMenu;

From the 29 Web pages evaluated, I detected 129 menus, 45 correctly identified (TP) by VIPS and 84 that extended VIPS did not detect (FN). Besides, 16 false positives (FP) were detected. This means that extended VIPS has misclassified elements, as menus. With these data, I was able to calculate the precision, recall and the F-measure. These measurements allowed me to assess the extended VIPS accuracy, as well as to compare different VIPS versions. As is depicted in Table 3, this evaluation has a very low recall, although the precision is quite suitable. Unfortunately, the F-measure, which is the relation between precision and recall, was quite low as well, less than 0.5. Therefore I concluded that regarding menus, the extended VIPS ignores many of them (low recall), but what it detects as menu is almost always well identified (high precision). In comparison to my identification, the extended VIPS results were not suitable. After this evaluation, I decided to improve the rule file, in order to get better results.

<table>
<thead>
<tr>
<th><strong>Extended VIPS Evaluation (129 Menus)</strong></th>
<th><strong>TP</strong></th>
<th><strong>FN</strong></th>
<th><strong>FP</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data</strong></td>
<td>45</td>
<td>84</td>
<td>16</td>
</tr>
<tr>
<td><strong>Precision</strong></td>
<td>0.737</td>
<td>0.348</td>
<td>0.472</td>
</tr>
</tbody>
</table>

Table 3 – Extended VIPS first evaluation results

---

21 [https://wordpress.org/showcase/](https://wordpress.org/showcase/) Last accessed on January 2018
3.5 Summary

This section contains my analysis work, which consisted of doing research on the existing segmentation tools and CMSs, so I could decide which ones are best suited to implement and use in QualWeb.

The segmentation tool will allow me to divide the page into regions and to those regions, I can assign a specific role. This is important because there are elements in a Web page that can represent many purposes, for example, the element <div> can either serve as a list or contain a banner. The role lets me know what the different regions or elements of the page actually correspond to. In WCAG 2.0 there are techniques that require semantic knowledge to be evaluated. With the role attribution, I can evaluate some of those techniques. For each different role, I can then evaluate the corresponding technique. I chose to use CMS’s rather than a traditional website to assist in pattern detection because as I mentioned before these Web site’s construction is based on templates, so the Web pages are similar to each other.

Finally, I introduce the element that I’m focusing on, which is the menu element.
Chapter 4

Implementation

This section describes the modifications made to improve the extended VIPS rule file and the analysis made to understand the obtained results. The second set of analysis, comparing the previous results with the ones obtained from the new version Web service VIPS, is then shown. The revisions made to the role detection algorithm to improve the results are also presented. The final section summarizes this chapter.

4.1 Rule File Extended VIPS

After my first evaluation with extended VIPS, I decided to change the rules for menu detection in the rule file (eMine.clp) in order to improve the results. I started by understanding how the rule file was composed as well as the originally defined rules for the menu role. As mentioned in Chapter 3, in the extended VIPS approach there are a set of properties, depicted in Figure 17, for each role and if an element has these properties, the likelihood score for that role is incremented. At the end, the system assigns the role with the highest likelihood score.

As I inspected the file, I noticed that the menu role, namely LinkMenu, was similar to the List role. The initial properties of LinkMenu role are:

- It may have ul, ol, dl or nav tags;
- Its id attribute may contain “menu” or “nav” keys;
- It usually has different background color than its parent;
- It appears in a Header, Footer, Sidebar or Nav element;
- It contains MenuItem children;
- It appears at the top or left side of a Web page;
- It is not atomic, it contains children Web elements (MenuItems);
- It may be in different sizes, such as full width, etc. but its height is generally short.

And the properties of the List role are:

- It may have one of ul, ol or dl tags;
- It may contain Link children;
- It generally has a list-style CSS attribute;
- It has the same background as the parent;
- It generally appears in a Sidebar, Container or another List;
- It is not atomic, it contains children Web elements;
It has medium size.

The List and LinkMenu roles are in Figure 18 and Figure 19 respectively.

As mentioned, these two roles are similar and it is possible that List elements may be created with properties that are assumed for LinkMenu elements. This is due to the fact that HTML and CSS are flexible and developers may not follow any pattern when programming websites. Similarly, both MenuItem and Link, which are two other element’s roles that identify links, are used in the properties of LinkMenu and List respectively. Both of them are created with the `<a>` tag, which makes it also difficult to distinguish.

In the first evaluation results, shown in Table 3, the recall value was too low, because, in my opinion, extended VIPS was not taking into account several menus. Some of them were also considered Lists, as expected due to the roles definition.
Chapter 4. Implementation

4.2 Modifications of the Extended VIPS Rule File

I reviewed the source code of the 29 WordPress Web pages training set, previously mentioned. With the information gathered and the previous understanding of the roles definition, I began by inspecting the LinkMenu role. To each propriety in the LinkMenu role, I tested different likelihood scores to inspect the weight of that property for a correct menu identification. With each modification I ran the extended VIPS and compared the number of menus correctly identified, the incorrectly ones and the false positives. After this step, there were still a few properties not implemented in the LinkMenu role, which could help improving the classification. Once again, I tested the properties one at a time, changing the likelihood score each turn to see how it improved or not the identification of menu elements. I also felt the need to modify other roles, because if the correct identification of a menu depends on their composed elements, their children, they need to be correctly identified. The method used was the same as before.

Finally, I reached a state that I could not improve the result any further. Therefore, the final definition of LinkMenu role is shown next. Also, I changed the definition of the MenuItem role. I thought that by improving the detection of the elements inside the menu that could facilitate the detection of the menu itself. The original MenuItem definition did not take into consideration the has_id variable (id, class or source values), so I added two new rules shown in Figure 20:

I also changed the likelihood score from 2 to 8, in case that the parent of the element is a LinkMenu, presented in Figure 21.

The role attribute from WAI-ARIA was not implemented in extended VIPS, so I asked Elgin to include it, so the system can take advantage if the element is already annotated with menu role.
The modifications made to the LinkMenu definition was the increase of the likelihood score from 5 to 9 if id, class or source values contains the substring “menu” or “nav”. Also added the new rules highlighted in Figure 22:

- If the size of the element has a medium height, the likelihood score increases by 4, the same that the other relative size rules;
- If it has children link or icon, the likelihood score increases by 2, the same as MenuItem;
- If the element position is on the right, the likelihood score increases by 2, the same as the other position rules;
- If the id, class, src, etc., values, contains the substring “item”, the likelihood score increases by 5;
- If the background color is the same, the likelihood score increases by 2;

![Figure 22 – Rule modifications for the LinkMenu role](image)

### 4.2.1 Second Evaluation of Extended VIPS

As Table 4 shows, the second evaluation with the final modification of the LinkMenu definition has better results than the first one. For the same 129 menus, 93 were correctly identified (TP) by VIPS and 36 were not detected by extended VIPS (FN). There was an increase in the false positives count, which was expected with the new rules added to the extended VIPS. As the algorithm is able to detect much more menu elements, the probability of detecting other elements that are not menus also increases, so the probability of misclassifying the elements can increase too. This second evaluation detected 72 false positives (FP). With my changes the recall has doubled, meaning that the extended VIPS detects more menu elements, without hurting the precision, which has decreased, but not significantly. The F-measure has a better result, consequently, 0.63.

This corresponds to the best results because further increases to the recall meant lower precision and F-Measure values, caused by the increasing false positives.
Table 4 – Extended VIPS second evaluation results

<table>
<thead>
<tr>
<th>Extended VIPS Evaluation (129 Menus)</th>
<th>TP</th>
<th>FN</th>
<th>FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>45</td>
<td>84</td>
<td>16</td>
</tr>
<tr>
<td>Precision</td>
<td>Recall</td>
<td>F-Measure</td>
<td></td>
</tr>
<tr>
<td>0.737</td>
<td>0.348</td>
<td>0.472</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Extended VIPS Evaluation with new Rules (129 Menus)</th>
<th>TP</th>
<th>FN</th>
<th>FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>93</td>
<td>36</td>
<td>72</td>
</tr>
<tr>
<td>Precision</td>
<td>Recall</td>
<td>F-Measure</td>
<td></td>
</tr>
<tr>
<td>0.563</td>
<td>0.720</td>
<td>0.631</td>
<td></td>
</tr>
</tbody>
</table>

4.3 Web Service VIPS Evaluation

After this second evaluation, I needed to test the VIPS algorithm in the Web evaluation tool, QualWeb. Unfortunately, the extended VIPS implementation was not able to be used outside the Eclipse environment. This is due to the fact that extended VIPS depends on ACTF and it requires a browser to render the source code and get visual attributes of Web elements. So, the Web service VIPS was developed. This Web service receives a Web page URL and returns a block structure in JSON. The implementation process into QualWeb is described in Chapter 6.

As the Web is dynamic, Web pages change their content, including their menus, so there was a different set of menus, due to the fact that these evaluations were performed with two months apart. A few elements classified as menus, no longer existed on those pages and there were some new elements I identify as menus in the WordPress training set. This is the reason for the different number of menus evaluated in comparison to the second analysis, mentioned above.

The training set WordPress Web pages evaluation results of Web service VIPS implemented on QualWeb are presented in Table 5. Comparing the results of both, extended VIPS and Web service VIPS without new rules, it is possible to see that they differ from each other. This is due to several reasons. First of all, in the extended VIPS, ACTF uses an older version of Internet Explorer and the Web service VIPS uses PhantomJS, which is a GUIless browser. There are some differences in how they render a Web page. Also, PhantomJS enables access to a richer set of visual attributes after the page is rendered. There are other differences in the categorization of Web elements as well as in segmentation heuristics. These differences are an improvement of the extended VIPS algorithm, however, the rule file does not suffer any changes.

The extended VIPS without rules results, depicted in Table 3, are worse when compared with the Web service VIPS. The recall has increased, as well as the precision and consequently the F-measure, which is already above 0.5. The number of menus found by the algorithm has
incremented and the misclassified one decreased slightly. The false positives are similar, although there are more in this evaluation.

Then I modified the rule file (rule-generator.js), with the changes I made in the previous version (extended VIPS). The results are also in Table 5. As is shown the results were not as improved as the previous ones, however, they are consistent. There are more menus found, which increases the recall, and decreases the precision, as expected. The F-measure also increases, but as the previous F-measure was already high, the improvement is not as significant as expected. The false positives have also increased, from 21 to 63, which is normal since we are catching more elements. Although there are many false positives the difference is smaller than the evaluation from the extended VIPS, which initially had 16 FP and at the second evaluation had 72 FP.

After these evaluations, I still thought that I could improve the accuracy of the menu detection algorithm even further. As I had too many false positives at the end of the second evaluation, I focused on reducing those. I noticed some patterns when observing the elements that the extended VIPS considered wrongly as LinkMenu. So I grouped the affected elements by tag, and clustered the false positives in 12 manageable tag groups:

- Article;
- Header;
- Aside;
- Composite;
- Div;
- Footer;
- A;
- Section;
- P;
- Text;
- Ul;
- Table/Thead/Tbody;
- H1/H2/H3/…/H6;

From here, I selected the tags that should not be menus, as <p>, <table>, <thead>, <tbody>, <ul>, all the H’s as well as combination of roles, e.g. LinkMenu/Container. With these tags, I developed a filter to run after the extended VIPS detects that the element has a LinkMenu role. This filter does a second role evaluation, where it inspects the element tag and compares it with the tags mentioned above. The filter function is described in detail in Chapter 6.

In this third analysis, I managed to reduce the false positives from 63 to 45. The results are also depicted in Table 5. There is also a decrease in the menus correctly identified, due to the fact that some menu elements use the tags implemented on the filter. These could mean that they are not well implemented. Consequently, there is an increase in the number of menus that were not classified as LinkMenus. The recall has diminished again, from 0.63 to 0.55, but the precision is a little better, from 0.57 to 0.62.

After these results, I inspected the elements retrieved from the filter and I noticed many elements with the tag <ul> caught in it. I analyze the source code again and noticed that a lot of menus were represented with just an <ul>, without a <nav> tag or not so often, even without a <div> tag. Therefore, I removed the condition with the <ul> tag from the filter algorithm. I also added two other tags to the filter that I could determine that were not used to classify menus, <text> and <section> and did not affect the number of menus found. I ran the Web service VIPS for a fourth evaluation. The results are also shown in Table 5.
### Table 5 – Web service VIPS results

<table>
<thead>
<tr>
<th>Web Service Evaluation (132 Menus)</th>
<th>TP</th>
<th>FN</th>
<th>FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>60</td>
<td>72</td>
<td>21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Web Service Evaluation with new Rules (132 Menus)</th>
<th>TP</th>
<th>FN</th>
<th>FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>83</td>
<td>49</td>
<td>63</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Web Service Evaluation with new Rules and 1st Filters (132 Menus)</th>
<th>TP</th>
<th>FN</th>
<th>FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>73</td>
<td>59</td>
<td>45</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Web Service Evaluation with new Rules and 2nd Filters (132 Menus)</th>
<th>TP</th>
<th>FN</th>
<th>FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>80</td>
<td>52</td>
<td>52</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Web Service Evaluation with new Rules and 3rd Filters (132 Menus)</th>
<th>TP</th>
<th>FN</th>
<th>FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>69</td>
<td>63</td>
<td>32</td>
</tr>
</tbody>
</table>

### Results

<table>
<thead>
<tr>
<th>Web Service Evaluation (132 Menus)</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>0.741</td>
<td>0.455</td>
<td>0.564</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Web Service Evaluation with new Rules (132 Menus)</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>0.568</td>
<td>0.629</td>
<td>0.597</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Web Service Evaluation with new Rules and 1st Filters (132 Menus)</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>0.619</td>
<td>0.553</td>
<td>0.584</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Web Service Evaluation with new Rules and 2nd Filters (132 Menus)</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>0.606</td>
<td>0.606</td>
<td>0.606</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Web Service Evaluation with new Rules and 3rd Filters (132 Menus)</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>0.683</td>
<td>0.523</td>
<td>0.592</td>
</tr>
</tbody>
</table>
With this evaluation I intended to recover the values obtained in the second evaluation but with less false positives. This was accomplished, with just less 3 TP, as well as FN. The false positives incremented regarding the third evaluation but, they are lower than in the second evaluation. Comparing these results to the previous evaluation, means that the precision has dropped slightly, but the value of both recall and F-measure has increased.

A third attempt to improve the filter was made, after a pattern discovered in the VIPS classification of false positives. In Chapter 5, section 5.2 there was the need to categorize each menu. This labelling is better explained in that Chapter, but there were a small part of the menus difficult to classify with the established categories. These menus were then classified with three new categories, which are: text, image and button. All the menus assigned with these new categories (text, image and button) were false positives, meaning that only the VIPS considered them as menus. For that reason, I examine the source code of those elements to understand why they were considered menus and how I could improve the filter to get better results. In total there were 16 elements with those labels. Two of them were using the tags <article> and <composite>, and a third element was using the <address> tag. So I introduce these tags in the filter and those three elements were excluded from the false positives elements detected by VIPS. There were 4 elements using the tag <ul>, which I removed from the filter, because as I shown before, it improves the results, due to the fact that a menu needs a <ul> tag. The remaining 9 false positives were mainly <div> elements, but a few of them did not contain any children <ul>, <dl> or <ol> tags, which is a required element for menus, so a new rule was added to the filter: If an element does not contain any of the tags, <ul>, <dl>, <ol>, <a> or <img> then the element is not classified as LinkMenu. After inserting this rule in the filter, 3 of the 9 remaining false positives were solved and the other 6, as they contained the mentioned tags, were still classified as false positives.

After this third modification to the filter, I ran the Web service VIPS for a fifth evaluation and the results are shown in Table 5. Similarly to the third evaluation, there was a decrease in the menus correctly identified, once again due to the fact that the more filters are implemented, there is a bigger chance that more menus, using those tags, are catched by the filter. As mentioned above, these could mean that the menus are not well implemented. As there is a decrease in the TP elements, the number of FN elements increased, which decreased the recall value. Additional, there was a considerable decrease in the false positives (less 20 FP), which increased the precision. Overall, the F-Measure went from 0.61 to 0.59, which is not a substantial difference, since the number of FP improved.

The remaining tags in the list presented above which do not integrate the filter have not been tested to see if the results could be further improved. The final filter algorithm contain the following tags: <p>, <table>, <thead>, <tbody>, <h1>, <h2>, <h3>, <h4>, <h5>, <h6>, <text>, <section>, <article>, <address> and <composite> (“element determined by VIPS as a related group of elements). Additionally, the filter evaluates if the role obtained is composed by more than one role, e.g. LinkMenu/Container. The filter also inspects the children of elements classified as LinkMenu to ensure that contains one of the following tags: <ul>, <dl>, <ol>, <li>, <a> or <img>.
4.4 Summary

This section contains the several improvements and evaluations made to both extended VIPS and Web service VIPS with the WordPress Web page’s set, as well as, the modifications made to the rule file and the introduction of a filter to improve the menu role detection.

It starts by describing the rule file of the extended VIPS regarding the LinkMenu role. As the evaluation made at this point had poor results, namely the low recall (extended VIPS was disregarding several menu elements), there was a need to modify and improve the rule file. With the new rules implemented, the evaluation results were significantly better with an improvement in the F-Measure from 0.47 to 0.63. The recall also increased from 0.35 to 0.72. Consequently, the precision value suffered because the number of false positives is greater.

Then a new version of VIPS is introduced, the Web service VIPS, due to the fact that the previous one (extended VIPS) was not compatible to work in QualWeb. After implementing this new version into the QualWeb, the same evaluations were performed.

The results with this version were better, because were also a change in the browser engine. Nevertheless, there were a lot of false positives detected, and to try to mitigate this problem a filter was developed. There were three evaluations with different versions of the filter. The second and third filters were the ones with better results, but the one with lower number of false positives was the Web service VIPS with the third filters implemented (only 32), although the F-Measure has a small decrease.
Chapter 5

Evaluation

This section contains an assessment of a new set of WordPress Web pages, as well as non WordPress Web pages, to validate the previous proposed menu identification procedure. Additionally, the results of the interviews and the surveys made to understand the concept of menu element in a Web page by accessibility experts and regular Web users are presented, and a comparison between these results and my own menu definition is shown. At the end of this chapter, a summary is given.

5.1 Validation of the Implemented Algorithm

The rule file was improved based on the evaluations and observations made from a single set of WordPress Web pages. To ensure that the system performs well in other Web pages, I tested it on a new set of WordPress Web pages (15 pages listed in Appendix B) and a new set of non WordPress Web pages (15 pages listed in Appendix C) to verify if the developed mechanism also performs well with pages developed with a different construction from the WordPress. All the tested Web pages where randomly selected. This aims to validate the algorithm developed in the previous chapter.

The evaluations performed in this chapter use the two last versions of the algorithm developed, which is the Web service VIPS with new rules and the second filters and the third filters implemented. This could help determine, in a different set of Web pages, which of the implementations achieve better results.

Table 6 contains the results of the Web service VIPS with new rules and the second filters implemented. As it is shown, the results of the evaluation made to the new WordPress Web pages set are very similar to the ones obtained with the first set of WordPress Web pages. Although the precision has decreased slightly, this compensated by a recall increase. The F-Measure result was almost the same has the previous result, 0.60.

Although these results are satisfying, the ones obtained from the non WordPress Web pages were better. The number of false positives was also low, only 22, which increments the precision. As the false negatives were also low, both the precision and the recall values increase their values, almost 0.7 and consequently, the F-Measure is also a high value, nearly 0.7 comparing with the 29 training WordPress Web pages set.
Table 6 – Results of the evaluation of new Web pages set with the 2nd filters

<table>
<thead>
<tr>
<th>Web service VIPS with 2nd Filters</th>
<th>WordPress Web Pages Evaluation (52 Menus)</th>
<th>Non WordPress Web Pages Evaluation (75 Menus)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>TP</td>
<td>FN</td>
</tr>
<tr>
<td>Results</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>36</td>
<td>16</td>
</tr>
</tbody>
</table>

Table 7 contain the results of the Web service VIPS with new rules and the third filters implemented. With these new filters implemented, there are a decrease in the F-Measure results, as observed with the training set in the previous chapter. In the results of the new WordPress Web pages set, unlike the training set example, the precision instead of increasing, decreases slightly, as the recall, with a more significant decrease. The results of the non WordPress Web pages were more similar to the results observed with the training set, since precision has increased significantly in this case and the recall decreased.

Both these evaluations were not extensive since, as mentioned, only 15 Web pages were tested for WordPress and 15 for non WordPress. However, they provide an indication that the developed algorithm for menus identification is able to generalize not only among WordPress developed pages, but also to other Web pages.

Table 7 – Results of the evaluation of new Web pages set with the 3rd filters

<table>
<thead>
<tr>
<th>Web service VIPS with 3rd Filters</th>
<th>WordPress Web Pages Evaluation (52 Menus)</th>
<th>Non WordPress Web Pages Evaluation (75 Menus)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>TP</td>
<td>FN</td>
</tr>
<tr>
<td>Results</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>19</td>
</tr>
</tbody>
</table>
5.2 Evaluation of my Menu Definition

As I mentioned above, the rule file was developed and improved with my assumptions of what a menu element is. After obtaining the results and with my definition more solidified, I felt the need to investigate if my definition of menu was generalized. As the rule file was developed based only on my opinion, the menu set resulting from Web service VIPS may not consider the menu definition of other users, which could consider other elements as menus. To gather this information, I performed two different analysis. The first one consisted of interviewing accessibility experts and ask them to analyse the same 29 training set WordPress Web pages to determine the menu elements in those pages. The second analysis consisted of doing an online survey, allowing regular users to give me their menu definition opinion by identifying sections of a page as menus. I compare both of these analysis to the Web service VIPS results with the second filters and the third filters implemented.

5.2.1 Accessibility Experts Analysis

I interviewed 3 accessibility experts to understand their opinion regarding the menu definition. Two of those experts work in FCT, in the ACCESS Unit for the Information Society department (further referred to as expert 3 and expert 4). The other one is one of the extended VIPS developers (referred as expert 2).

I ask them to observe the 29 WordPress Web pages and for each of them, point out the elements they thought were menus. This Web page observation was only visual and if necessary by interacting with the page, but without inspecting the Web page’s source code. This analysis was based in interviews to better understand how the experts react to the pages, their questions or doubts and if necessary to ask them why they choose a specific element.

The first thing I noticed was that the opinions regarding the menu definition diverged. The second note observed was the fact that as the interview goes on, the interviewees seem to extend their menu definitions, embracing more elements as menus. The third thing I noticed was that menu identification is susceptible to interpretations, not concerning rules or a real definition. Each interviewee has their own opinion regarding the menu definition, all different from one another. In the following I am Expert 1. For the other 3 experts, I summarize their menu definition:

- Expert 2: Only considers the top and bottom menus, which are commonly found in the header and footer. In his own words: “A menu is a specific Web element which navigates a person to other pages within a Web site. In general, its visual representation is identical in all Web pages in a Web site. Most of its items contain internal links. Its function is to show which pages you can navigate and alternatively which page you are currently in. The textual content in a menu item is generally short (1-3 words in general)”. The set of social media links, which appears in several Web pages shown, was considered by this expert, toolbars.
- Expert 3: This expert had the definition more similar to my own. For this expert, a menu is a set of links, which could allow a person to navigate in the page or to other pages. The expert considered not only the usual navigational menus and social media menus, but also sets of images with text, representing articles or news. Occasionally, he pointed out sets of buttons or sets of texts. Almost at the end of the interview, he also started to identify slideshows as menus.
- Expert 4: The last expert was the one which considered more elements as menus. For this expert a menu is a list of elements, links or text, which could show the information right
away or could navigate you to other Web page. Groups of images, sets of news or articles and lists are defined as a menu, as well as social media menus and slideshows.

To evaluate these results and compare their classifications with my own, I identified the set of all potential menus, which is composed by all of the menus mentioned by at least one expert as well as the menus given by the Web service VIPS, which amounts to 219 potential menus for the Web Service VIPS evaluation with the second filters applied and 200 with the third filters. For each potential menu, a classification was attributed regarding the expert’s opinion and the Web service VIPS result if the element was a menu (1) or not (0). With these results, I was able to determine the F-Measure comparing each expert with the VIPS evaluation.

**Web Service VIPS with Second Filters Evaluation**

The results of the comparison of VIPS evaluation with each expert is shown in Table 8. The Web service VIPS considered 132 elements as menus, whereas the expert 1 considered 127 menus, the expert 2 pointed out 53 menus, the expert 3 126 menus and the expert 4 found 157 menu elements.

My evaluation (expert 1) is represented in the first row and the results are the same as mentioned previously, with minor differences, due to the dynamic nature of the Web (elements classified as menus disappear from some pages, and/or were replaced). The F-Measure value is the highest of the four experts’ results, being 0.6.

Expert 2 only considers menus the top navigation bar in the Web page’s header, and occasionally a similar menu on the footer. For that reason, the obtained results are very different from the other experts. The F-Measure is quite low in comparison with the others, because for his menu definition, VIPS catches too many elements. In this case, the precision is very low as well, because many of the elements that VIPS defines as menus, are not considered menus by this expert. The recall value in this case is much higher than any of the other because, the majority of the elements considered by the expert 2 as menus has the same classification as VIPS. Consequently, with the low precision and high recall, the F-Measure, 0.48, is much lower than the rest.

Unexpectedly, the expert 3, which was the one that seems to have the most similar menu definition to my own, had the second lowest F-Measure, 0.54.

The results of expert 4 were the most comprehensive sample of all four, because, in his opinion all lists were menus. He not only pointed out the menus, but as well the lists elements, which increased the precision because VIPS also has some difficulty on separating both. Even so, the F-Measure value, was the closest to my, 0.59.
Table 8 – Result comparison of the expert’s evaluations with the Web Service VIPS with 2nd filters

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>FN</th>
<th>FP</th>
<th>TN</th>
</tr>
</thead>
<tbody>
<tr>
<td>My Evaluation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expert 1 (127 Menus)</td>
<td>78</td>
<td>49</td>
<td>54</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expert 2 (53 Menus)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expert 3 (126 Menus)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expert 4 (157 Menus)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Web Service VIPS with Third Filters Evaluation

The results of the comparison of VIPS evaluation with each expert is shown in Table 9. The Web service VIPS considered 103 elements as menus, less 29 elements. For the experts, the elements considered menus are the same as before.

Once again my evaluation (expert 1) has similar results with the previous analysis with the evaluation made with the third filters implemented, with minor differences, due to the dynamic nature of the Web (elements classified as menus disappear from some pages, and/or were replaced). The F-Measure value is once again the highest of the four experts’ results, being 0.62. The increase of the F-Measure compared with the previous one is due to the decrease in the false positives found, which went from 54 to 32.

The expert 2 results remain very different from the other experts. The F-Measure is the lowest of all four because of its menu definition. In this case, the precision is very low as well,
but has a higher value than before. The recall value is the highest of all because the majority of the elements considered by the expert 2 as menus has the same classification as VIPS. Consequently, with the low precision and high recall, the F-Measure, 0.51, is lower than the rest.

The expert 3, which is the one that has the most similar menu definition to my own, also shows the increase on the precision value and the decrease on the recall, but the F-Measure value change from 0.54 to 0.55.

The results of expert 4 do not show any significant difference in the F-Measure result, being 0.59. The same pattern is found in relation to precision/recall values.

All the four F-Measures have highest values, due to the increase of precision values and the decrease of the recall values. The precision has bigger values because there are many less false positives found due to the third filters implemented on the Web service VIPS. Consequently, the true positives suffers slightly, because VIPS could also misclassify elements, which increase the false negatives.

After this step, I used the Chi-square test to determine, for each expert if his classification is similar to the VIPS classification. This was the test selected because the values used to classify the potential menus are binary (0 or 1). The hypotheses are:

H0: The expert’s and VIPS’s opinion are independent.
H1: There is a link between the expert’s and VIPS’s opinion.

The results of the Chi-square Test are presented in Table 10.

For the Web service VIPS with the second filters, the significance level of alpha was 0.05. For the experts’ 1 and 3, as the p-value is greater than the alpha, the H0 hypothesis cannot be rejected, meaning that the experts and VIPS opinions are independent. For the experts’ 2 and 4 as the p-value is lower than the alpha, the H0 hypothesis should be rejected and accept the H1 hypothesis.
### Table 9 – Result comparison of the expert’s evaluation with the Web Service VIPS with 3rd filters

<table>
<thead>
<tr>
<th>Expert</th>
<th>Data</th>
<th>TP</th>
<th>FN</th>
<th>FP</th>
<th>TN</th>
</tr>
</thead>
<tbody>
<tr>
<td>My Evaluation Expert 1 (127 Menus)</td>
<td>Data</td>
<td>71</td>
<td>56</td>
<td>32</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>Precision</td>
<td>0.689</td>
<td>Recall</td>
<td>0.559</td>
<td>F-Measure</td>
</tr>
<tr>
<td>Expert 2 (53 Menus)</td>
<td>Data</td>
<td>40</td>
<td>13</td>
<td>63</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>Precision</td>
<td>0.388</td>
<td>Recall</td>
<td>0.755</td>
<td>F-Measure</td>
</tr>
<tr>
<td>Expert 3 (126 Menus)</td>
<td>Data</td>
<td>63</td>
<td>63</td>
<td>40</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>Precision</td>
<td>0.612</td>
<td>Recall</td>
<td>0.500</td>
<td>F-Measure</td>
</tr>
<tr>
<td>Expert 4 (157 Menus)</td>
<td>Data</td>
<td>76</td>
<td>81</td>
<td>27</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Precision</td>
<td>0.738</td>
<td>Recall</td>
<td>0.484</td>
<td>F-Measure</td>
</tr>
</tbody>
</table>

### Table 10 – Chi-square Test of Experts with Web Service VIPS

<table>
<thead>
<tr>
<th>Chi-square Test</th>
<th>Expert 1</th>
<th>Expert 2</th>
<th>Expert 3</th>
<th>Expert 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-value (Web Service VIPS with 2nd filters)</td>
<td>0.685</td>
<td>0.000</td>
<td>0.097</td>
<td>0.003</td>
</tr>
<tr>
<td>P-value (Web Service VIPS with 3rd filters)</td>
<td>0.100</td>
<td>&lt;0.0001</td>
<td>0.580</td>
<td>0.095</td>
</tr>
</tbody>
</table>
For the Web service VIPS with the third filters only the expert 2 as a p-value lower than the alpha so the H0 hypothesis is rejected and H1 hypothesis is accepted, meaning that there is a link between the expert’s and the VIPS’s opinion. The remaining values are bigger than the alpha, so the H0 hypothesis cannot be rejected.

Then, to see if the collective experts’ opinion resemble the VIPS opinion, I created an expert consensus. Before determine that consensus, I use the Fleiss’ Kappa test to assess the reliability of agreement between the four experts’ opinion. The kappa value was 0.38, which corresponds to fair agreement.

Without disregard the previous test, I created a consensus between the four experts which was determined by the experts’ majority and when it was a draw, the considered value was 0.5. To test if the consensus classifications is similar to the VIPS classification, the test used was the Mann-Whitney, because the values are between 0 and 1. The p-value obtained when comparing the expert consensus with both versions of Web service VIPS (with 2nd and 3rd filters) is presented Table 11. The hypotheses for this test are:

H0: There is no difference between the experts’ consensus and VIPS results.
H1: There is a difference between the experts’ consensus and VIPS results.

Both the versions of the Web service VIPS have a p-value higher than the significance level alpha = 0.05, which means that the hypotheses H0 cannot be rejected. This represents that there are no difference between the experts’ consensus and the VIPS results.

<table>
<thead>
<tr>
<th>Mann-Whitney Test</th>
<th>Web Service VIPS with 2nd filters</th>
<th>Web Service VIPS with 3rd filters</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-value</td>
<td>0.086</td>
<td>0.216</td>
</tr>
</tbody>
</table>

After this analysis, it was very clear that different experts considered different types of elements a menu. From here, it is worth investigating how VIPS handles the different element types. For this, a classification was attributed to each of the potential menus to group them. The 4 major categories are listed below, together with their sub-categories:

- **Navigation** – Menu which allows the user to navigate in the Web page or site. Usually in the header or in the footer of the page.
  - Navigation Header (NH);
  - Navigation Footer (NF).
- **Slideshow** – The slideshow or carousel menus (S).
- **Lists** – The menus composed by a list, which could be either of images, text, buttons, articles or news.
  - List of Images (LI);
  - List of Texts (LT);
  - List of Buttons (LB);
  - List of News (LN);
  - List of Articles (LA);
  - List of Images and Text (LIT).
- **Social Network** – The menu composed by icons representing the social media (R).
There were also some elements in the potential menu list which do not fit in any category, so I added 3 more categories to classify those elements:

- Image (I);
- Text (T);
- Button (B).

An agreement percentage analysis is depicted in Table 12. This analysis compares each expert opinion with Web service VIPS with the second and third filters, by category. This pretends to perceive the relation between experts and Web service VIPS. This agreement percentage is obtained by summing the number of menus classified equally by both expert and Web service VIPS. It was necessary to use percentages instead of another statistic test due to the fact that when menus were divided by category, the sample is much smaller, which do not allow the application of another statistic analysis. This comparison does not reflect the veracity of the evaluation, it only compares the experts and the Web service VIPS opinion.

For both versions, it is possible to see that the navigational menus are classified by the experts and the Web service VIPS equally about 70% of the times. It is the menu category when experts and VIPS agree more.

**Table 12 – Agreement percentage between Experts’ opinion and Web Service VIPS by category**

<table>
<thead>
<tr>
<th>Agreement %</th>
<th>Web Service VIPS with 2nd filters</th>
<th>Web Service VIPS with 3rd filters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category (Menu count)/Expert</td>
<td>Expert1</td>
<td>Expert2</td>
</tr>
<tr>
<td>N (63) (62)</td>
<td>79</td>
<td>79</td>
</tr>
<tr>
<td>NH (38) (37)</td>
<td>82</td>
<td>84</td>
</tr>
<tr>
<td>NF (25) (25)</td>
<td>76</td>
<td>72</td>
</tr>
<tr>
<td>S (7) (7)</td>
<td>57</td>
<td>57</td>
</tr>
<tr>
<td>L (108) (96)</td>
<td>49</td>
<td>47</td>
</tr>
<tr>
<td>LI (10) (5)</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>LT (50) (44)</td>
<td>50</td>
<td>32</td>
</tr>
<tr>
<td>LN (21) (20)</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>LTI (7) (7)</td>
<td>29</td>
<td>71</td>
</tr>
<tr>
<td>LA (7) (7)</td>
<td>71</td>
<td>86</td>
</tr>
<tr>
<td>LB (13) (13)</td>
<td>54</td>
<td>77</td>
</tr>
<tr>
<td>R (24) (24)</td>
<td>33</td>
<td>67</td>
</tr>
<tr>
<td>I (7) (4)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B (2) (0)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>T (7) (6)</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
For the comparison between expert 1 and the Web service VIPS with the second filters, the most agreed elements are the navigational menus and the list of articles. Expert 2 has more categories above 70%, which contain the navigational menus, the lists of text and image, the list of articles, the list of buttons and the social network. For expert 3, just the navigational menus are above the 70% agreement level as well as for the expert 4.

After the categorization of the potential menus there was a noticeable pattern, relating the three new categories added to classify the remaining elements. The Image, Text and Button categories were only elements classified as menus by the Web service VIPS, in other words false positives. For that reason I decided to improve the filters of the Web service VIPS (3rd filters) and the modifications are explained in the previous chapter. With the third filters implemented on the Web service VIPS the Button category no longer exists and the remaining two categories (Text and Image) have less elements.

The level of agreement for the Web service VIPS with the 3rd filters implemented is very similar to the previous one.

5.2.2 Users Analysis

For this analysis I developed two surveys to be answered by regular Web users. Each survey was divided into two groups. The first group consisted in finding menus in a section of a Web page. The Web pages evaluated by the users, were also the 29 WordPress Web pages I initially used (training set). The second group of the survey consisted in answering if an element depicted in an image represented a menu. Because the potential menus of the first part of the survey are presented in the context of their Web page and the ones from the second part are presented without context, I have prepared two versions of the survey to determine if the context influences the users’ answer. Each survey evaluated about 15 Web pages, and the potential menus illustrated in the second part of the first survey, were retrieved from the Web pages sections in the first group of the second survey, and vice-versa.

This survey was published through an e-mail to all students, teachers and any employee in Faculdade de Ciências – Universidade de Lisboa, (survey 2) as well as my social networks (survey 1).

I gathered 181 responses, 62 for the survey 1 and 119 for the survey 2. For the first group of both surveys, I noticed that the navigation menus in the top of a Web page were always pointed out by the users. Relating the social media menus the opinions diverged: half of the users considered them menus and the other half did not. Lists of images were also often considered menus.

As not all of the menus in the potential menu list were evaluated by the users, I only could compare the ones shown in the first group with the Web service VIPS result. I classified each of those potential menus with the percentage of users who had selected that menu and then compared it with Web service VIPS with second and third filters results. The resulting data of this comparison are numbers between 0 and 1 (quantitative) and, in order to analyse these data, I transformed it into qualitative data (0 and 1). This calculus was made by finding the mean between all the users’ classified menus, in this case 0.7. Then, for each potential menu classified by the user, the following condition was applied: if the user’s classification is higher than the mean value (0.7), then that menu is classified with 1, if it is lower than the mean, the classification attributed is 0. With this new classification, the chi-square test was applied to understand if there is a relation between the users and the Web service VIPS opinion. The results of the Chi-square Test are presented in Table 13.
Chapter 5. Evaluation

Table 13- Chi-square test of Users with Web service VIPS

<table>
<thead>
<tr>
<th>Chi-Square Test</th>
<th>Web Service VIPS with 2(^{nd}) filters</th>
<th>Web Service VIPS with 3(^{rd}) filters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 1</td>
<td>0.000</td>
<td>0.017</td>
</tr>
<tr>
<td>Group 2</td>
<td>0.676</td>
<td>0.516</td>
</tr>
</tbody>
</table>

The hypotheses are:

H0: The users’ and VIPS’s opinion are independent.
H1: There is a link between the users’ and VIPS’s opinion.

For both versions of the Web service VIPS the significance level of alpha was 0.05. For the first group of the survey, the Web service VIPS with the second filters results had a p-value lower than the alpha, so the H0 hypothesis should be rejected, and accept the H1 hypothesis, meaning that there is a relation between the users’ and VIPS opinions. For the Web service VIPS with the third filters the p-value was also below the significance level of alpha, so the H0 hypothesis is rejected and H1 hypothesis is accepted, meaning that for both versions there is a link between the users’ and VIPS’ classifications.

For the second group of the survey, the Web service VIPS with the second filters results had a p-value higher than the alpha, so the H0 hypothesis cannot be rejected, meaning that the users answers of the second group of the survey are independent from Web service VIPS. For the Web service VIPS with the third filters the p-value is also higher than the significance level of alpha, so the H0 hypothesis cannot be rejected. For both of the versions of Web service VIPS, the opinion of the second group of users is independent of Web service VIPS.

Then I compared the menus in the second group evaluations of the surveys with the same menus in the first group to determine if context could change the user’s opinion. The test used determine this relation was the Mann-Whitney statistical test, because the values of both samples are quantitative (between 0 and 1). The hypotheses are:

H0: There is no difference between the opinion between the first and the second group of users.
H1: There is a difference between the opinion between the first and the second group of users.

The p-value value is 0.372, being higher than the significance level alpha = 0.05, which means that the hypotheses H0 cannot be rejected. This represents that there are no difference between users’ and the VIPS results. The sample is small (26 results) but this could mean that the context does not change the users’ opinion regarding menus.

Finally, the last test relating the users’ evaluations is understanding the agreement level with the Web service VIPS. This test is shown in Table 14. It was only tested for the first group because the second group has a very small sample. For some categories of the first group, the number of menu elements to evaluate is also low, so this test is just an approximation. The most agreed elements between the users and both versions of Web service VIPS is the Lists of text and images. Once again the navigational menus also have a high percentage of agreement between the users and the Web service VIPS.
Table 14 - Agreement % between the first group of the survey and the Web service VIPS by category

<table>
<thead>
<tr>
<th>Category (Menu count)/Expert</th>
<th>Agreement %</th>
<th>Web Service VIPS with 2nd filters</th>
<th>Web Service VIPS with 3rd filters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Group 1</td>
<td>Group 1</td>
</tr>
<tr>
<td>N (41)</td>
<td></td>
<td>83</td>
<td>71</td>
</tr>
<tr>
<td>NH (29)</td>
<td></td>
<td>79</td>
<td>66</td>
</tr>
<tr>
<td>NF (12)</td>
<td></td>
<td>92</td>
<td>83</td>
</tr>
<tr>
<td>S (3)</td>
<td></td>
<td>0</td>
<td>33</td>
</tr>
<tr>
<td>L (22)</td>
<td></td>
<td>59</td>
<td>59</td>
</tr>
<tr>
<td>LI (4)</td>
<td></td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>LT (9)</td>
<td></td>
<td>67</td>
<td>67</td>
</tr>
<tr>
<td>LN (0)</td>
<td></td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>LTI (3)</td>
<td></td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>LA (1)</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LB (5)</td>
<td></td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>R (8)</td>
<td></td>
<td>56</td>
<td>63</td>
</tr>
<tr>
<td>I (0)</td>
<td></td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>B (0)</td>
<td></td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>T (0)</td>
<td></td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>

5.3 Summary

This chapter presented all the evaluations made after the extended VIPS was implemented into QualWeb. It started by depicting the validation of the results obtained in the Chapter 4. Then it presented the comparison between my menu definition with experts and users’ opinion relating that definition.

The validation step in this chapter had the purpose of verifying how well the developed mechanism performs with different set of pages. It was tested a new set of WordPress Web pages and a new set of non WordPress Web pages in both version of Web service VIPS, with the second and the third filters implemented. In both versions of VIPS the set with better results was the non WordPress Web pages, but the WordPress set also had good results.

After confirming that the developed mechanism can be generalized to different sets of Web pages, I felt the need to verify if my menu definition was also general. Since my rule file was developed based only in my opinion, the Web service VIPS output may not consider the menu definition of other users, which can differ from my own. To understand this, two different analysis were performed. The first one with accessibility experts and the second one, with Web users. The first conclusion noticed was that there were not a consensus between the experts’ opinions. From the four experts opinion gathered, three of them obtained good F-Measures results and the fourth...
expert had the lowest F-Measure, due to the fact that his menu opinion was very restrictive. Even though, there were some similarity between the opinion of two experts and the Web service VIPS result. Even though, there was not a level of agreement between the experts was only fair, a consensus analysis was made to perceive if the experts’ consensus opinion could relate to the Web service VIPS classification and I found there were no difference between the experts’ consensus and the Web service VIPS results. After this analysis, it was clear that different experts had different opinions regarding menu elements, so a category was attributed to each potential menu to understand which menus were classified equally by both experts and Web service VIPS. The category with better results was the navigational menus.

Finally, a users’ analysis was performed. The users opinion was similar to the Web service VIPS classification when the menus were analysed in context (shown in the website), but when the menus were not seen context (shown an image with the menu) there was a difference between the users and the Web service VIPS classification. Then I tested if the context could influence the users opinion (small sample), but the analysis result stated that the context does not change the users’ opinion regarding menus. I also evaluate the agreement level by category and once again, the navigational menus were the ones with better results.
Chapter 6

QualWeb

This section explains the integration of the Web service VIPS algorithm into the accessibility evaluation tool, QualWeb. It starts by introducing the WCAG 2.0 techniques that concerning menu elements and the relevant techniques to be introduced into QualWeb. Then a comparison between a QualWeb evaluation and a manual evaluation of the same elements will be shown to validate the implementation. The last section summarizes this chapter.

6.1 WCAG 2.0 Techniques applicable to Menus

The WCAG 2.0 guidelines aim to improve the accessibility of a Web page, focusing on the information present in a website, including text, imagens, forms, sounds, among others. Table 15 presents the techniques already implemented in QualWeb. One thing they all have in common is the fact that the techniques are applied to specific tags, e.g. `<img>` or `<table>`. If the element is a table, then the table related techniques are applied. However, if the technique needs to be applied to a group of related links, which are not specific elements, neither have a specific tag, then the technique cannot be applied in the current version of QualWeb. There has to be a previous understanding of what is a group of links in a page, i.e., Menus or Lists. To obtain this knowledge the Web service VIPS algorithm is called before calling the technique. Having the element’s role attributed then it is possible to have a target to apply the technique.

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H2</td>
<td>Combining adjacent image and text links for the same resource</td>
</tr>
<tr>
<td>H25</td>
<td>Providing a title using the title element</td>
</tr>
<tr>
<td>H30</td>
<td>Providing link text that describes the purpose of a link for anchor elements</td>
</tr>
<tr>
<td>H32</td>
<td>Providing submit buttons</td>
</tr>
<tr>
<td>H33</td>
<td>Supplementing link text with the title attribute</td>
</tr>
<tr>
<td>H36</td>
<td>Using alt attributes on images used as submit buttons</td>
</tr>
<tr>
<td>H37</td>
<td>Using alt attributes on img elements</td>
</tr>
<tr>
<td>H44</td>
<td>Using label elements to associate text labels with form controls</td>
</tr>
<tr>
<td></td>
<td>Description</td>
</tr>
<tr>
<td>---</td>
<td>-------------</td>
</tr>
<tr>
<td>H46</td>
<td>Using noembed with embed</td>
</tr>
<tr>
<td>H53</td>
<td>Using the body of the object element</td>
</tr>
<tr>
<td>H57</td>
<td>Using language attributes on the html element</td>
</tr>
<tr>
<td>H63</td>
<td>Using the scope attribute to associate header cells and data cells in data tables</td>
</tr>
<tr>
<td>H64</td>
<td>Using the title attribute of the frame and iframe elements</td>
</tr>
<tr>
<td>H65</td>
<td>Using the title attribute to identify form controls when the label element cannot be used</td>
</tr>
<tr>
<td>H67</td>
<td>Using null alt text and no title attribute on img elements for images that AT should ignore</td>
</tr>
<tr>
<td>H71</td>
<td>Providing a description for groups of form controls using fieldset and legend elements</td>
</tr>
<tr>
<td>H76</td>
<td>Using meta refresh to create an instant client-side redirect</td>
</tr>
<tr>
<td>H89</td>
<td>Using the title attribute to provide context-sensitive help</td>
</tr>
<tr>
<td>H91</td>
<td>Using HTML form controls and links</td>
</tr>
<tr>
<td>H93</td>
<td>Ensuring that id attributes are unique on a Web page</td>
</tr>
</tbody>
</table>

As part of my work, I identified all the HTML related WCAG techniques that could be applied to menu elements:

- H30 – Providing link text that describes the purpose of a link for anchor elements.
- H48 – Using ol, ul and dl for lists or groups of links.
- H77 – Identifying the purpose of a link using link text combined with its enclosing list item.
- H83 – Using the target attribute to open a new window on user request and indicating this in link text.
- H91 – Using HTML form controls and links.
- H93 – Ensuring that id attributes are unique on a Web page.
- H94 – Ensuring that elements do not contain duplicate attributes.
- H97 – Grouping related links using the nav element.

From these eight techniques, three of them were already implemented in QualWeb (H30, H91 and H93), one required semantic to understand the purpose of links (H77) and two of them were not exclusively for menu elements (H83 and H94). The remaining two techniques (H48 and H97) were selected to implement in QualWeb.
Chapter 6. QualWeb

6.2 Integration of Web service VIPS

The process of evaluate a Web page starts by doing a request to QualWeb, where is passed a Web page URL.

This URL is sent to the browser processing simulator (BPS) to be processed. After obtaining the page’s DOM tree, this is sent to the core. For each element in the DOM, the appropriate techniques are called. However, before this process, the Web service VIPS algorithm needs to be called, so it is possible to have an element to apply the new techniques inserted into QualWeb.

QualWeb passes the received URL to Web service VIPS and waits for its response, which comes in the JSON with the format shown in Figure 23. This response is then sent to a new script added to QualWeb to select just the elements with menu role. With the element’s role attribution it is possible to have a target to apply the technique. So, the new techniques are applied to the menu elements retrieved. This process is depicted in Figure 24.

The script mentioned above receives the DOM tree from QualWeb, the JSON response from Web service VIPS and the URL. The JSON is traversed and for each element the script checks its role:

- If the element’s role is LinkMenu and the tag is <Nav>, then the element is considered a menu.
- If the element’s role is List, then the element is considered a list.
- If the element’s role is LinkMenu, then the developed filter is applied.

For each element that needs to be filtered, its tag is checked and if it is equal to the group of tags mentioned in Chapter 4, the element is not considered a menu neither added to the list of menus.

After established the role, the element needs to be added to the correspondent list, which are the menus or the lists, because the new techniques implemented are applied to these two elements. Before saving the element in the menu list, it is checked if a parent of the specific element already exists in the list. This is due to the fact that I do not want to save the submenus of an already registered menu, because sometimes Web service VIPS can also consider the menu’s children as menus, which can lead to several false positives. Another reason to ignore the

Figure 23 – Example of an element in the JSON response
submenus is the fact that they were not considered in my initial analysis. The submenus of a menu were not relevant to understand the menus in a Web page, so to compare the results, the script also does not need to take into account these ones. Figure 23 shows an example of one element in the JSON response. To avoid saving submenus in the menu list, I check the name of the element, in this example, VB.1.1.1 and compare it with all the names saved in the menu list. If the element’s parent already exists in the list, e.g. VB.1.1, then the element is not saved.

![Flowchart](image-url)  
*Figure 24 – Flowchart of the process of inserting the Web service VIPS algorithm in QualWeb*
This script is a recursive function, so it will evaluate all the elements in the JSON. After the menus and lists are found, the techniques will be applied to each corresponding element. The technique H48 is applied to both lists and menus and the technique H97 is only applied to menus. The process of selecting the menus and lists from the JSON is depicted in Figure 25.

Figure 25 - Flowchart of the process of selecting the menus from the VIPS JSON response
6.2.1 Implementation of relevant techniques into QualWeb

To insert a new technique into QualWeb it is necessary to modify some files. To serve as an example, the technique to be added is the H48, and the changes to be made are listed below:

1. `/home/qualweb/framework/lib/qualWeb/evaluator/criteriaProcessor.js` – In `initializeCriteria()` function, add the technique, for example h48 in the corresponding criterion.

2. `/home/qualweb/framework/lib/qualWeb/evaluator/evaluator.js` – In `initializeResLineAux()` add the new var `resultadosLinhaAux["h48"] = [];` as in `initializeResultados()`, add the `resultados["h46"] = [];` variable. In `evaluation()` function add the var `t_h48` which contain the path to the technique.

Ex: `var t_h48 = require (`path` + '/lib/qualWeb/evaluator/techniques/h48.js');?>.

3. `/home/qualweb/qualweb_server/lib/client.js` – In function `start()` add in var `techs` the new technique. Repeat the same process in the file: `/home/qualweb/framework/client.js`.

4. `/home/qualweb/public/javascripts/extra11.js` – Insert in var `tecs` the new technique and respective WCAG 2.0 link, which leads to the technique information, as well as the description, retrieved from that web page.

Ex: 

```
"h48":{"url":"http://www.w3.org/TR/WCAG-TECHS/H48.html",
"des":"Using ol, ul and dl for lists or groups of links","warn":","pass":","fail":","tudo":","countW":0,"countP":0,"countF":0}.
```

5. In function `convertTypeToString()` insert an elseif condition and specify the result description for every outcome from the technique.

After these modifications, it is also required to create a new JavaScript file in `/home/qualweb/framework/lib/qualWeb/evaluator/techniques/h48.js` to implement the technique.

The process of H97 technique is depicted in Figure 26. All the menus found after analysing the Web service VIPS JSON result, are going to be evaluated by this technique. This technique receives the HTML DOM tree of the Web page, the element to be evaluated and the URL of the page. The technique states that a group of related links, which I consider a menu, needs to use the `<nav>` element.

The first condition of the H97 technique is if the tag of the element already is `<nav>`, then the technique pass. Otherwise, the element needs to be found in the HTML DOM tree to understand if its children or parent elements have a `<nav>` tag. There are two ways to find the element in the HTML DOM tree from the JSON element: through its ID or the path from the root to the element. Finding the element by its ID is simple, because we retrieve that from the xpath of the JSON element and search it in the HTML DOM tree. The problem is, as established, the HTML DOM tree and the JSON result could not have the same structure. So the path given in the JSON element xpath might not be exactly the same as the real path in the DOM tree.

My script needs to have into account four types of xpath:

1. Xpath = `//[@id='navigation']`;
2. Xpath = `//[@id='navigation']/DIV/NAV`;
3. Xpath = `"BODY/DIV/NAV"`;
4. Xpath = `"BODY/DIV[2]/NAV"`. 
In the beginning of the function to find the element I check if the xpath has an ID. In the case of option 1, where the xpath contains just the ID, then the function findNodeByID() is called to traverse the DOM tree until the element with the same ID is found. If the element is found, then it is saved, if not the function throws an error.

In the case of option 2, where the xpath contains an ID but also a path to the element, then the function findNodeByID() is called to find the element that starts the path. Next the function findNodeByPath() is called. This function will save each element in the xpath in an array to compare it to the elements in the DOM tree. This comparison is done taking into account the position of the elements in the path. For example, in option 4, the mid element DIV[2] represents the second <div> children of the previous element. If this comparison is right and the element is found then it is saved, if not the function throws an error.

For the third and fourth options, which do not contain an ID and starts always from the root of the tree with an HTML or a BODY element, there is a findNodeByPath2() function that saves each element in the path to an array. Then it will compare the elements in that array with the DOM tree one by one. If the last element in the array matches the element in the DOM tree, then the element is saved.

After finding the element, the function will evaluate the element’s children to understand if any child contains a <nav> tag. The condition will run until reaching a <ol>, <ul>, <dl> or <a> tag, which are the links of the menu.

Checking the element’s children is important because sometimes, the element considered LinkMenu is the parent of the actual menu element, for example a <div> element, with an ID equals to “menu” is considered LinkMenu and contains a child <nav>, the real menu element. As mentioned earlier in this report, the submenus of a menu are not taken into account and are not listed in the menu list, because it could lead to replicated results. Therefore, it is important to check the element’s children in order to validate the technique, because the tag <nav> could be one of the children, instead of the actual element.

If there are no children <nav> elements, then the element’s parents are checked, following a similar reasoning. If a <nav> element for any reason is not classified as LinkMenu, and a child of that element, e.g. a <div>, is considered the menu, the technique should pass. So the function checks the parents of the element to find out if a <nav> exists. If parent <nav> does not exists the technique fails.

The process of H48 technique is shown in Figure 27. This technique states that lists or group of links, which is a menu, need to use the tag <ol>, <ul> or <dl>. After analysing the VIPS JSON result, all the menus and lists found are going to be evaluated by the technique H48. This technique also receives the HTML DOM tree of the Web page, the element to be evaluated and the URL of the page.

The first condition of the H48 technique checks if the tag of the element is one of the mentioned earlier: <ol>, <ul> or <dl>, and if so the technique passes. Otherwise, the element needs to be found in the HTML DOM tree to be checked. The process of finding the element in the HTML DOM tree is the same described for the previous technique. After finding the element, its children need to be traversed to check if one of them contains the tags needed to satisfy the technique. If one of the tags <ul>, <ol> or <dl> is found then the technique H48 passes, otherwise it fails.
The techniques H97 receives the HTML DOM tree, the element to be evaluated and the Web page URL

Has a tag equal to NAV?

Yes ➔ H97 Pass

No ➔ Find the element in the HTML DOM tree

Element found?

No ➔ H97 Fail

Yes ➔ Check if the element has children or parents with the tag NAV

Found children or parent with tag NAV?

No ➔ H97 Fail

Yes ➔ H97 Pass

Figure 26 – Flowchart of the process of the H97 technique
The techniques H48 receives the HTML DOM tree, the element to be evaluated and the Web page URL.

- **Has a tag equal to UL, OL or DL?**
  - No: Find the element in the HTML DOM tree.
  - Yes: **H48 Pass**

- **Find the element in the HTML DOM tree**
  - **Element found?**
    - No: **H48 Fail**
    - Yes: **Check if the element has children with the tag UL, OL or DL**

- **Found children with tag UL, OL or DL?**
  - No: **H48 Fail**
  - Yes: **H48 Pass**

*Figure 27 – Flowchart of the process of the H48 technique*
6.3 Validation of WCAG 2.0 Techniques Implementation

Before I implemented the H48 and H97 techniques, I performed an expert assessment of all the related menu techniques in the 29 WordPress Web pages. I have manually inspected the source code of each page and evaluated each menu to understand if they were accordingly to the guidelines. Regarding the two implemented techniques (H48 and 49), the evaluation itself did not have satisfactory results. The results are presented in Table 16.

For the H97 technique, in 127 evaluated menus, only 39 passes the technique, which means that approximately 31% of the menus respects the H97 technique, meaning that less than one-third of the menus use the tag <nav>. The other 69% do not use the appropriate tag.

However, the evaluation of the H48 technique has better results. Almost 77% of the menus evaluated pass in this technique, which means the majority of lists or group of links uses <ul>, <ol> or <dl> to group their links. The minority, approximately 23%, which fails the technique, usually uses only the <div> tag to represent a menu or a list.

| Table 16 – Results of the initial expert assessment |
|----------------------------------------|----------------|----------------|----------------|
|                                      | **H97**        |                | **H48**        |
| Count                                 | %              | Count          | %              |
| Pass                                  | 39             | 30.71          | 98             | 77.17          |
| Fail                                  | 88             | 69.29          | 29             | 22.83          |
| Total menus                           | 127            | 100%           | 127            | 100%           |

After I implemented the two techniques, I went back to evaluate the 29 WordPress Web pages in QualWeb and registered the results. The Web service VIPS implemented in QualWeb was the Web service VIPS with the second filters implemented. This second evaluation was needed to determine if the new techniques were correctly implemented in QualWeb. For this to be right, the results of both evaluations had to be the same.

Of these 30 Web pages, 7 of them could not be rendered by QualWeb, due to a PhantomJS error and 2 of them could not be segmented by Web service VIPS. Therefore, these pages were not accounted for the analysis results. I compared the expert assessment and the QualWeb evaluation of the remaining 21 Web pages and the obtained results were the expected ones. Apart from the fact that some menus could not be detected by Web service VIPS, as established before, the ones it detects had the same classification as my expert evaluation. The correlation value for the H97 technique was equal to 1 and for the H48 technique, the correlation value was 0.98. This value is due to an element which could not be detected the element in the HTML DOM tree, and for that reason, the element’s children tags could not be inspected to assess the technique.
6.4 Summary

This chapter explains the integration of the used systems, which are VIPS and QualWeb. It also mentions the contributions to the QualWeb evaluation framework tool.

It begins with a summary of the explored techniques and identifies the relevant ones regarding menu elements. Then it explains the integration of web service VIPS into QualWeb, as well as the process of inserting new techniques into the evaluation tool.

Finally, it presents the validation of the new techniques.
Chapter 7

Conclusion

Many works addressing Web page segmentation, as it is described in [11], choose to extract the information structure from only the DOM tree. As established, evaluating just the syntax often is not enough if we can’t make sense of it. The semantic analysis can complement this evaluation, solving ambiguities.

The more techniques implemented in the accessibility evaluation tools, better the results obtained and consequently can lead to more accessible Web pages. Understanding semantics can be helpful for implementing new WCAG techniques in accessibility evaluation tools, which could require previous knowledge of the Web page elements. This knowledge can be obtained with a segmentation tool like extended VIPS, who attributes roles to the elements.

There are some disadvantages relating Web page segmentation tools, being the main one, undoubtedly, the heterogeneity of the pages [9]. Many works addressing this subject assume a default layout or a predefined template. This template consists in, a header at the top of the page, a right or left menu, the content in the centre and a footer at the bottom of the page. This is not a good procedure, because nowadays, people can customize the websites as they please. If a Web page segmentation tool cannot interpret any Web page model it’s not very useful. Therefore, they can’t be used in all websites, that is, they are not universal [13]–[15]. This is an issue, because if we want to improve the way disabled people interact with Web pages, the segmentation tool must be able to analyse all the available websites.

With the extended VIPS [17], I was able to find a tool that uses the Web page’s source code, as well as the visual representation of the page and performed well. In addition to the segmentation, the extended VIPS, also had a role attribution function, which attributed a role to each segmented region of the page. I choose to focus only on menu elements, because they are an important element in a Web page, but also because they can assist a person with special needs in the Web page’s navigation.

Unfortunately, for the menu elements, this role attribution was not very satisfying. I needed to improve the role attribution function in other to obtain better results. I selected 30 WordPress Web pages to serve as my training set and tested VIPS for all of them. After each analysis, I modified the rule file in order to improve the outcome. When I reached a point that could not improve any further the mechanism, I tried to incorporate it into the QualWeb evaluator tool. However, the extended VIPS could not be integrated with other systems. So I suggested to Elgin (one of the developers of the extended VIPS) the creation of a Web service version. A new version of VIPS was created, allowing me to implement it on QualWeb. After the implementation and with the changes to the rule file, I analysed the Web service VIPS results. The VIPS results were satisfactory, but had still a large amount of false positives. I added a false positive filter, which
allowed me to disregard elements by their tags. For example, a single text (<p> tag) cannot be considered a menu, so it does not pass through the filter. After refining the filter, the menu detection was completed. After, a validation of these results was performed, with another set of WordPress pages and with non WordPress pages. Both performed well, but the non WordPress Web pages test set had a better result than the WordPress Web pages set, what was unexpected, since WordPress Web pages follow a template pattern and use HTML5 and WAI-ARIA tags. Overall, the mechanism developed can be generalized to different set of WordPress Web pages but even better to non WordPress Web pages.

Additionally, an evaluation to compare my menu definition with expert’s and user’s definition was performed. With the experts I performed an interview and asked them to analyse the same training set as I did, and identify the menu elements in those pages. The first thing I noticed was that their opinions diverged a lot. For example, one of them just considers the navigational menus in the header or footer, the other considers all lists of elements as menus. They all agreed that is not easy to define menu, because it is a personal opinion and susceptible to interpretation. Also I noticed that the more Web pages they analyse, more elements were considered menus and more questions emerged. To collect the users’ opinion I developed two surveys to determine what elements they considered menus. I had 181 responses and noticed that the navigation menus in the top of a Web page were always pointed out as well as the menus in the footer. Lists were often considered menus, whether they are lists of images, lists of text or both. The social media menus were where opinions diverged. However, the comparison between the user’s results and the Web service VIPS were similar when menus were in context of Web pages and differ when menus were presented as an isolated image. Also, the context does not affect the identification of menu elements by the users. When comparing the users and experts opinion with the Web service VIPS classification, the users opinion is more similar to Web service VIPS classification than the experts.

With the role attribution I was able to add new WCAG techniques to QualWeb, which was not possible before. These techniques are the H97 which deals only with menus and the H48 which deals with menus and list. Both of these techniques are applied to menus and without the classification of the different regions of the page it was not possible to know what a menu was, so the technique could not be applied. After implementing the new techniques, I had to validate those results and apart from an outlier, the techniques returned the expected results.

The results from this work were submitted to the Web 4 All 2018 conference and are under revision.

### 7.1 Future Work

Facing the obtained results some work can be conducted regarding the menu definition analysis:

- Compare the experts and users opinions, in order to provide a consensus of the menu definition.
- Improve the rule file based on the opinion of both experts’ and users’.
- Understand to what extent context affects the user’s opinion regarding menus.
- Perform more expert interviews to obtain more accurate results.
- Understand if the rule file modifications undermine the accuracy of VIPS.
- Improve the false positive filter, adding more elements that could not be menus.
Regarding the QualWeb, there are some improvements that could be done:

- Implement a reparation mechanism, which would be called after the accessibility verification was made. For every inaccessible element, which did not pass in the techniques, the mechanism should repair its source code.
References


[14] Y. Chen, W.-Y. Ma, and H.-J. Zhang, “Detecting web page structure for adaptive viewing...


Appendix A

List of first 30 WordPress Web page’s tested:

- http://curtsspecialrecipe.com/
- http://galp.in/
- http://grainandmortar.com/
- http://www.infografixlab.com/
- http://www.japantimes.co.jp/
- http://johnsardine.com/
- http://www.lbvd.com/
- http://market-pl.com/
- https://www.marketingweek.com/
- http://matterofform.com/
- https://paidtoexist.com/
- https://piano.io/
- http://www.prowebdesign.ro/
- http://racket.net.au/
- http://www.rokivo.com/
- http://www.rudalov.com/
- http://www.sparked.nl/
- https://affinitydemo.wordpress.com/
- https://theundefeated.com/
- http://www.greenwichlibrary.org/
- http://blogs.faz.net/
- http://www.gracenote.com/
- http://www.angrybirds.com/
- http://chicago.suntimes.com/
- https://www.studiopress.com/
- https://webdesignledger.com/
- https://themetrust.com/
- http://www.jennybristow.com/
- https://www.entyce-creative.com/
Appendix B

List of first 30 WordPress Web page’s tested:

- https://www.villagevoice.com/
- https://thewaltdisneycompany.com/
- http://www.thisisyourkingdom.co.uk/
- https://www.obama.org/
- https://www.bloomberg.com/professional/
- http://www.gsd.harvard.edu/
- https://irontoiron.com/
- http://www.thisisyoke.com/
- https://finland.fi/
- http://girlwithacamera.co.uk/
- http://www.tinkeringmonkey.com/
- https://www.nprdpinc.org/
- http://observer.com/
- http://briansmith.com/
Appendix C

List of 15 non WordPress Web page’s tested:

- http://www.bing.com/
- https://www.ebay.com/
- http://awis.com/
- http://www.tstorm.net/
- http://www.themovingblog.com/
- https://www.moving.org/
- http://www.instructables.com/
- https://www.ehow.com/
- https://www.healthcare.gov/
- http://www.trustedreviews.com/
- http://www.imdb.com/
- http://www.foxnews.com/
- https://www.spotify.com/pt/
- http://www.cooks.com/