Masters Degree in Informatics Engineering

Dissertation

Final report

An Approach for Characterizing HTML Defects

Joaquim José Agostinho Mendes

jjmendes@student.dei.uc.pt

Advisor:

Prof. Dr. Nuno Laranjeiro

cnl@dei.uc.pt

Coimbra, 8th September, 2017

FCTUC DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

FACULDADE DE CIÊNCIAS E TECNOLOGIA

UNIVERSIDADE DE COIMBRA

Masters Degree in Informatics Engineering
Dissertation
Final report

Joaquim José Agostinho Mendes
jjmendes@student.dei.uc.pt

Advisor:
Prof. Dr. Nuno Laranjeiro
cnl@dei.uc.pt

Jury
Main Opponent:
Prof. Dr. Tiago Cruz
tjcruz@dei.uc.pt

Opponent:
Prof. Dr. Raul Barbosa
rbarbosa@dei.uc.pt

Coimbra, 8th September, 2017

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Abstract

Hypertext Markup Language (HTML) is nowadays being massively used as an interface to provide services to users. Web developers are producing new sites and changing them at high pace, while trying to support the latest HTML standards. In this context, it is quite common to find websites that, due to small details, do not comply with the standards and many times fail to be correctly processed by browsers. At the same time, developers also tend to overlook compliance with standards, when their web pages are visually rendered correctly, since current browsers are highly tolerant to mistakes. Errors are also introduced by tools that generate HTML code, making files produced by them automatically non compliant to the standards, as well as from a variety of other sources. Considering this dynamic environment and the increasingly large diversity of browsers, with frequent updates, the sporadic appearance of errors in web pages is a quite common, sometimes severe, hard-to-track problem. In this project, we intended to decide upon a set characteristics deemed important for an accurate portrayal of the HTML available on-line. To obtain indicators of representative errors made by web developers. To determine a group of metrics capable of representing the complexity of an HTML file. And, finally, to create a tool that will be used to obtain large-scale and up-to-date information regarding all of these properties. The information provided by the tool would then be used by developers to build more reliable websites.

Keywords: HTML; Standards; Validation; Characterisation

Acknowledgements

Whenever a significant chapter in life is about to be closed, we tend to look back and reflect upon the events and people that brought us to where we are.

The biggest benefit of these last few years spent here, has certainly been the friendships created between a close group of colleagues. They don't require naming, as they know perfectly who they are. Without these industrious folk, I would have never reached the point where I am at. Certainly not as competently. From tutoring, to technical help, to all sorts of assistance, car rides, time spent talking, to simply hanging out. For all of these, my sincere thanks.

My appreciation must also be expressed towards everyone who has helped me go through what can only be described as the most difficult period in my life.

Finally, words cannot convey the gratitude I feel towards my parents. If I have accomplished anything so far, it has all been thanks to them.

Contents

List of Tables

List of Figures

Acronyms

ASP Active Server Pages. 14

CSS Cascading Style Sheets. 5, 11, 24, 38

CSV Comma Separated Values. 28

DBMS Data Base Management System. 59

DMOZ The Directory of the Web. 9, 10, 13, 15, 19

DNS Domain Name System. 12

DTD Document Type Declaration. 6, 7, 8, 9, 10, 11, 12, 13, 15

FTP File Transfer Protocol. 9

GDP Gross Domestic Product. 8

GIF Graphics Interchange Format. 40

HTML Hypertext Markup Language. v, 1, 2, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, 19, 24, 25, 27, 28, 29, 30, 31, 35, 36, 38, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 53, 55, 56, 59

HTTP Hypertext Transfer Protocol. 9, 15, 27, 28, 31, 39, 44, 46, 50

HTTPS HTTP Secure. 9, 15, 24, 25, 39, 46

IP Internet Protocol. 7, 8, 11, 12, 13, 15

IRC Internet Relay Chat. 9, 40

JAR Java ARchive. 28

JPEG Joint Photographic Experts Group. 40, 44, 50

MAMA Metadata Analysis and Mining Application. 9

PHP Personal Home Page. 14

PNG Portable Network Graphics. 40

PPP Purchasing Power Parity. 8

SGML Standard Generalized Markup Language. 6

SMS Short Message Service. 40

SVG Scalable Vector Graphics. 40

URL Uniform Resource Locator. 2, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 22, 27, 28, 29, 31, 35, 55, 57

VMM Virtual Machine Monitor. 30

W3C World Wide Web Consortium. 5, 7, 8, 9, 11, 12, 14, 15, 25, 27, 28, 35, 53, 55

WARC Web ARChive. 20, 27, 28, 53

WCAG Web Content Accessibility Guidelines. 7

WDG Web Design Group. 10, 11, 14, 15, 24

XHTML Extensible Hypertext Markup Language. 5, 9, 12, 35

XML Extensible Markup Language. 5

1. Introduction

This dissertation proposal was created as part of the discipline of Thesis/Internship for the Master in Informatics Engineering of the Department of Informatics Engineering, a section of the Faculty of Sciences and Technology of University of Coimbra. Apart from the author, this work has received the supervision of Professor Nuno Laranjeiro, which contributed with the role of advisor.

In the following sections we explain the circumstances that led to the initial motivation for this project, how we believe it is a worthwhile endeavour, the goals we set out to achieve and a description of this document's structure.

1.1 Motivation

Hypertext Markup Language (HTML) has been under development since 1990 and, along with other web technologies and languages, is nowadays being used to not only hold information but also to support real businesses, that rely on the correctness of the HTML documents to reach out to clients. Thus, this language is being used as an endpoint interface, through which clients can have access to operations of all kinds, including entertainment or business.

The usual aggressive time-to-market constraints of the global web environment, bring in the need for fast development of new websites, but also the need for fast application of changes to existing sites, so that new or different functionality can be accommodated. Also, the creation of new versions of the HTML standards, is an extra factor, leading developers to change their web applications, so that the latest standards are minimally supported by at least some browsers. In this scenario, developers concentrate on assuring correctness of the existing functionality and overlook the necessary time for verification and validation activities.

As several studies have shown, web pages' standards compliance success rate has been known to be extremely low throughout the times. Nowadays it is quite common to find websites that, due to small details, do not comply with the standards. This, coupled with current browsers' high fault tolerance leads to many web pages being displayed differently than expected by their developers [1]. If a browser is tasked with rendering a web page that does not meet the standards, it will first attempt to infer how it should have been written. Since each browser has a different behaviour, the same web page may be displayed differently on separate browsers [1]. This apparent functionality, despite some times the existence of critical errors in their pages, has led developers to disregard compliance with standards, from the moment that web pages are apparently rendered correctly [2]. Considering the dynamism of the above-mentioned environment and also the increasingly large diversity of browsers, with very frequent updates, it is common to observe failures in web pages. These can be caused by residual HTML mistakes and are a quite common, sometimes severe, hard-to-track problem [2]. Another reason for perpetuating the existence of errors is the fact that validation is not a required step to create a web page, thus, many developers simply opt out of performing it after perceiving their pages' apparent functionality. Pages built with legacy versions of HTML are not often updated to the newest versions and, in some cases, even the tools used to build them generate invalid code, unbeknown to the developers.

1.2 Goals

In this report, we present the initial design of a tool that should support our study by gathering updated and large-scale information regarding characteristics of HTML pages (e.g., type of HTML in use, number of outgoing links) and their conformance to the standards. First, we analysed the current state of the art and decided on the necessary tools between those available, as well as chose what metrics and characteristics to analyse. Then, we built a proof-of-concept prototype in order to carry out an initial assessment of a small set of web pages and identify an initial set of typical problems.

To the best of our knowledge and although similar studies have been carried out in the past (e.g., a decade ago), there is currently no up-to-date information based on large-scale data that can be used by developers or researchers, particularly when considering information regarding HTML 5. Possessing this kind of information is vital in many scenarios, in particular in aiding web developers in understanding the reliability of their websites and also in helping building more reliable sites, that are prepared to identify the presence of small common mistakes (e.g., introduced by an update or change to the site). A set of metrics that can convey the complexity of an HTML file, something we wished to make use of in our characterisation, is also seemingly lacking from current studies.

Our end goals can be summarised in three main points:

- Making a large-scale characterisation of the quality of the HTML currently available on-line, as well as establishing a set of metrics to determine the complexity of an HTML file.
- Through the analysis of common error indicators, producing a fault model for HTML validation.
- Creating a tool that can be used to retrieve and process all the information mentioned in the previous points.

As an expected outcome of this work, we also intended to create an academic paper with our findings and submit it to an international conference.

1.3 Report structure

This section intends to explain the organisation of the following chapters in this report. Chapter 2 presents information gathered from the state of the art review that was performed. A brief description of HTML, an explanation of the common methodology extracted from the analysed studies, a detailing of the process and results obtained for both types of research, as well as a short summary of our findings.

Chapter 3 proposes a system overview, detailing the architecture that we intended to follow. We also find a listing of our tool's functional and non-functional requirements, as well as a short explanation of the prioritisation method employed. Next is an analysis of the alternatives and explanation for our choices for Uniform Resource Locator (URL) sources. The same type of details are given for the remaining external programs needed, web crawlers and HTML validators.

Chapter 4 presents information regarding our tool's way of functioning, a detailing of the external dependencies it currently makes use of and a mention concerning some of the configurations performed.

In Chapter 5, we detail and discuss the set of results that were collected with the use of our tool.

Our final thoughts on the project are shown in Chapter 6, followed by Chapter 7, which details our work plan for both semesters, as well as comparing them to what was actually executed. Finally, in Chapter 8, we discuss what could be done to improve upon this project.

2. Background and related work

Naturally, every study relies on work previously done by other researchers. This chapter focuses on the meaningful studies and materials that approach the relevant subjects to our own efforts. Its intention is to give the reader a current state of the art of the topics we propose to address. First, we present a short description of Hypertext Markup Language (HTML) and its many iterations. This is followed by a description of the methodology employed and results obtained, by the studies that analysed the standards compliance of HTML available on-line. A set of studies which, despite having their main focus on another subject, still contain pertinent results and are included next. Finally, a summary of this information is presented.

2.1 Hypertext Markup Language

Ever since the creation of the World Wide Web by Tim Berners-Lee [3], in 1990, that HTML has been one of its crucial components. It is the basis of content delivery and structure of the information currently available on-line. Providing means to access entertainment, research, information and countless businesses.

HTML itself has been evolving throughout the last 27 years, adding new functionality with each of the its multiple versions. The standard is currently defined by the World Wide Web Consortium (W3C), which has also introduced Extensible Hypertext Markup Language (XHTML) and its multiple versions. Initially, HTML could be used to implement all aspect of a web page, however, the web standards model has evolved into the separation of three basic pillars. HTML provides structure and content, Cascading Style Sheets (CSS) delivers style and presentation, while JavaScript grants functionality. The main reasons for this model are code efficiency, ease of maintenance and compatibility [4].

On its 4.0 version, HTML introduced small variations of the same standard, each with its own differences, used mainly to ease transition from previous versions of HTML or legacy applications. These different versions, also commonly referred to as flavors, are *Strict*, *Transitional* and *Frameset* [5]. *Strict* corresponded to the full HTML standard, it contained all up to date features. However it did not allow the use of style or presentation elements, as these were already considered deprecated at the time. Framesets were also not allowed to be used and will trigger a validation error if included in a document of this type. The *Transitional* flavor, as its name makes apparent, was designed to serve as a transition between older versions of HTML, thus, it allowed for the use of style and other deprecated elements, with the exception of framesets. Finally, the *Frameset* flavor supported all the elements of *Transitional* with the addition of framesets.

Between the release of HTML's fourth and fifth version, XHTML was introduced. It is a variant of HTML that shares the syntax of Extensible Markup Language (XML) and is designed to be used by any XML tool [6]. Because of the stricter syntax, documents must be well formed and tags explicitly closed, it also provides a better error handling. XHTML also shares the same variations as HTML 4, *Strict*, *Transitional* and *Frameset*, they are identical to the ones previously described, apart from their syntax.

HTML was originally a Standard Generalized Markup Language (SGML), as defined by ISO 8879 [7]. An SGML is a language that produces structured documents that are able to be read by both machines and humans. Its documents should able to be read independently "of processing, system, or device dependencies". They consist of markup elements that detail how the software should render or display its content. One of these elements, the Document Type Declaration (DTD), must contain the document type definition identifier, which states the set of rules that must be followed in order to present it correctly. HTML 5, however, has strayed a bit from its SGML roots, opting to turn the document type definition identifier, among others, into optional fields. Thus, it is no longer considered an SGML according to its standard.

2.2 Related studies

In literature, it is possible to find two distinct groups of studies that tackle the issue of standards compliance of HTML on the Web. A group that focuses on evaluating HTML and its quality regarding the standards and a second group that does this as a byproduct of a different objective, accessibility being the most common. The studies in the first group share the same goal, the validity of HTML available on-line and its standards compliance. They make use of a similar method, gather a list of web pages, parse and analyse them, usually producing an extensive set of results. Their main points of variation are: the source of the studied web pages; how they were selected; methods of filtering the Uniform Resource Locator (URL) list, if any; the tools used to extract and parse the information and the amount of detail outputted. This set of studies closely resembles what we wish to accomplish.

The second group is comprised of studies that approach HTML validity somewhat superficially. Their main focus is usually the user accessibility level of the studied web pages, taking into account the Web Content Accessibility Guidelines (WCAG) [8]. Some study a particular feature of HTML, such as its DTD, or perform a broad characterisation of web pages that include one, or a few, standards compliance features.

2.2.1 Standards compliance studies

Chen et al. [2] are among the first group of studies previously mentioned. Their process involved gathering a list of web sites through the use of three different methods, random Internet Protocol (IP) addresses, Alexa.com [9] and search engines' URL search function. First, a set of 1,100 IP addresses was obtained through the use of a custom built program that generated random addresses and probed their port 80 in search of a Web server. In case of a positive response, they would be added to the list. Second, they made use of the top 10,000 most popular web sites list from Alexa.com [9]. Finally, in order to include a number of inner pages in their study, they used the URL search function of three different search engines, Google, Yahoo! and Teoma, to obtain 31,540 pages. The search strings used to find this last set of web pages were randomly generated by a program, with sizes of up to 8 characters. After a two month time period, a second random IP address list was generated in order to further diversify the web sites analysed. This brought the total number of web pages to 44,340.

A Java program was then developed to sequentially pick a URL from the gathered list, put the web page through JTidy [10], so it could be parsed and its information gathered, send the URL to the W3C validator [11] and analyse the results. The process would then be repeated for the next web page on the list.

According to the results, only 5% of the web pages were considered standards compliant. A set of ten most common problems were identified from all four groups of collected URLs:

- Missing DOCTYPE specification
- Missing attribute *alt* within element *IMG*
- Missing encoding specification
- Missing attribute *type* with element *SCRIPT* or *STYLE*

- Invalid attributes *height* and *width*
- Margin problem: invalid extra attributes *topmargin*, *leftmargin*, *marginheight*, and *marginwidth* in element *BODY*
- Missing quotation for attribute values
- Invalid attributes *border*
- Invalid attributes *background*
- Ampersands in URLs

This list of problems was consistent among the URLs collected using the different methods, differing slightly on the order. The issues that occurred the most were the absence of the DTD, on 60% of the total number of pages, the omission of the "*alt*" attribute, affecting 40% of the pages, and the lack of "*type*" attribute, which was detected on 30% of the pages. The last two problems represented 99% of the total number of missing attributes errors.

The study is lacking on some aspects, it could benefit from a more diverse information collection and a higher number of analysed web sites. Apart from the use of home and inner pages, no attempt at filtering or removing bias from the URL list seems to have been made. The analysis was also solely focused on errors, validator warnings were not included.

A study from Ofuonye et al. [1] expanded on [2] by increasing the web site pool to 100,000 and comparing the results with three additional of metrics. The Gross Domestic Product (GDP) at Purchasing Power Parity (PPP) per capita [12], used to rate the wealth and standard of living of a country, the e-readiness ranking [13], that measures a regions' Internet and computer related technological status, as well as the geographic region of each web page. The source for its web sites was the top 100,000 most popular web sites from Alexa.com for which, according to the site, the chance of a typical Internet user visiting a page outside of the ones listed was 0.00125%.

The W3C validator was used to assess the URLs' standards compliance, the GDP at PPP per capita was extracted from [14] and the e-readiness ranking was obtained from [15]. Geographical location was determined by comparing a database purchased from IP2Location, which maps IP addresses to its nation of origin, with the IP addresses resolved from the Linux *host* program.

The conclusions seemed to corroborate the ones obtained by [2], showing that only 3% of the web pages result in a successful validation, a decrease from 5% after only five years. Results revealed that there was a significant difference in the number of successfully validated web pages when comparing between their country of origin, from 8% in the United Kingdom to 0.5% in the Republic of Korea. There was also a continental trend, with Oceanian and European countries displaying the highest percentage of validation, 6.6% and 5.7% respectively, and Asian and African territories on the opposite side of the spectrum, with 1.7% and 0.4%. The validation rate increased proportionately to the GDP at PPP per capita, this was also true when comparing with the e-readiness ranking. This revealed a trend of better web page standards-compliance for richer and more technologically advanced countries. The only exception to this trend was the United States, which seemed to drag behind those numbers. Of the types of errors obtained during validation, only three affected more than 1% of the web pages. The most common were, the lack of a DTD, missing character set and a conflict between the character set declaration and the "*meta*" attribute. The most frequent fatal errors were the use of characters from an unspecified character set, unreachable web pages and problems with the character set. All the web pages that triggered these types of fatal errors prevented the validator from completing its analysis.

Some weaknesses can be pinpointed to this study. The lists from Alexa.com can be biased towards large, international sites with region specific alternatives, Google and its regional variants represented more than 12% of the top 500 most popular sites at one point in time [16]. At the time, this list was only representative of Internet Explorer users that had Alexa's toolbar installed, therefore neglecting a large user base from other browsers. There were no inner pages included, only the home pages were analysed in the study, using the assumption that the remainder of the site follows the same level of quality. There is no description of any attempt at filtering or removing bias from the URL list.

The Metadata Analysis and Mining Application (MAMA) [16] is an extensive study on the meta data gathered using URLs from three different sources, The Directory of the Web (DMOZ) [17], Alexa.com top 500 most popular web sites and a listing of W3C member companies. A series of filtering techniques were used on the 4.5+ million URLs obtained, resulting in the 3.5+ million that were analysed. In order not to skew the results, several techniques were used on the list of obtained URLs. Each domain was represented by a maximum of 30 web pages, so that the list's content would be as diverse as possible. There is a common practice for individuals and companies to buy domain names as a speculative business move, waiting for the name to rise in value before selling it. Most of these domains will also point to the same page or template in order to generate additional revenue from advertisements This practice is called domain parking. There was an attempt to remove as many of these cases as possible by filtering out links from the most well known domain parking companies. Some web sites return a custom web page whenever the user is faced with an error, such as a 404. These also tend to be identical between them, as they are produced from the same template and only the error message differs. There was also an attempt to remove these custom error pages as much as possible. Links that used protocols apart from Hypertext Transfer Protocol (HTTP) and HTTP Secure (HTTPS), such as File Transfer Protocol (FTP) and Internet Relay Chat (IRC), were discarded. To increase the number of URLs, the DMOZ web directory listing was scanned twice, one apart. The URL list from both scans was joined and duplicates removed.

MAMA was written in Perl, a set of 4 to 8 machines running Linux distributions were tasked of fetching the information from each of the URLs previously gathered and, after analysing it, they passed that data to a separate machine that ran a MySQL database. For the validation effort, the W3C validator was installed on two different machines, MAMA retrieved a URL from the database, it would send it to one of the validator machines, parse the result and store the information retrieved.

The results obtained were comprehensive. A thorough characterisation of the HTML documents was produced, with a wide range of information, from the most to least used attributes, types of encoding used, to the total number of hyperlinks and images, among many others. The markup validation portion of the study revealed that only 4.13% of the URLs tested were considered valid. Curiously, there was a very large difference when considering the HTML editors used to create the pages tested, revealing that 81.91% of pages created with Apple's iWeb were valid. In contrast, the second most successful editor was Microsoft FrontPage with only 0.55% success rate. Surprisingly, more than 50% of web pages that contained the W3C validation badge were in fact not valid at all, however, the validation rate of W3C companies web sites was quite higher than the global average, 20.15%. The amount of documents that included the DTD was 51%, with Strict flavors displaying a higher validation rate when comparing to Transitional and Frameset, with 17.5%, 8.4% and 7.2% respectively. XHTML flavor, with 13.4% validation, also fared better than HTML, with 6.6%. MAMA also showed that the transitional version of HTML 4.0 greatly outnumbered the other flavors (Strict and Frameset), at the time by 10 to 1. Of the 27 possible types of warnings, 14 were found in the study, of these, only five occurred in more than 1% of the total web pages. These were, *Unable to determine parse mode (XML/SGML)* (45.17%), *No DOCTYPE found* (39.12%), *No character encoding found* (13.71%), *Character encoding mismatch (HTTP header/META element)* (3.25%) and *Namespace found in non-XML document* (1.88%). The largest amount of warnings per web page was five, with two being the most common. Of this last case, 92.84% were combinations of the two most popular warnings, related to parse mode and lack of DTD. Errors were more prevalent, the 17 most common errors occurred on more than 10% of the total web pages. In order to avoid a long listing, only the first five are mentioned here, *There is no attribute X* (64.23%), *Required attribute X not specified* (57.37%), *No document type declaration; implying X* (39.09%), *End tag for element X which is not open* (35.11%) and *Document type does not allow element X here* (35.03%). The average number of errors per page was 46.70, the highest number of different errors in the same page was 39 and the biggest amount of errors on a single page was 37,370.

Despite the large scale of the study, additional measures could be taken in order to improve it. Limiting the number of URLs with sub domains in mind, instead of domain based. Considering removing the limit of 30 web pages per domain for sites that have a large amount of user generated content, as their quality varies greatly from page to page. Balance the number of home and inner pages by including at least one of each for every domain, as DMOZ is heavily biased in that regard, since 75% of its links are home pages. Further filter domain name parking and custom error pages. The validation phase was done roughly two months after the initial characterisation analysis, which means that some URLs could have changed their content significantly, ideally this should be done at the same time.

Dagfinn Parnas [18] used the Open Web Directory, the precursor of DMOZ, to gather over 2.5 million URLs. These were further refined to 2.4 million by removing similar URLs, those that were called by scripts which only differed in the parameters and URLs that did not point to HTML documents. Of the remainder, roughly only 2 million were put through validation, since a significant portion either could not be downloaded or had an invalid / custom DTD.

The URL list was split into packets of 8000 links and a script with the following behaviour was developed: acquire a packet, use Wget to download the HTML file from the URL, parse it with the use of the Web Design Group (WDG) HTML validator [19], repeat until all URLs are completed. Two instances of the script were simultaneously ran by 30 different machines, each instance redirected its output to a unique file. Invalid DTDs were not included in the group of invalid pages for two reasons. First, the difficulty in distinguishing between incorrect and custom DTDs. Second, since the parser used would stop after finding an incorrect or custom DTD, any errors beyond that point would not be found.

Results included 0.71% of pages being valid, a number that rose to 2.58% if DTD was not taken into account. The most frequent errors were *No DTD declared*, followed by *Non-standard attribute specified* and *Required attribute not specified.* The most common number of errors per page was 4, with an average of 5.2. No information regarding validator warnings was supplied nor any other filtering techniques for the URL list were specified.

Similarly, Rene Saarsoo [20] gathered 4.36 million pages from the Open Web Directory, however, only 1.27 million ended up being analysed in his study. Filtering techniques used included, removing duplicates, removing links to non HTML pages, excluding pages with 0 bytes in size and links that returned non 200 OK status codes. This further reduced the number of pages to 1,002,350.

In order to automate the process of analysing the web sites, he created a program in Perl, it made use of the HTML::Parser and CSS::AC libraries. This program was ran by 17 different machines, 5 instances on each, totalling 85. After separating the list into an equal number, GNU Wget was used to download the content of the links, the file was then put through the WDG HTML Validator and the W3C CSS Validator [21]. Finally, the relevant data was passed to an 18th machine, that ran a MySQL database.

Results revealed that only 2.6% of the web pages were considered valid and that the DTD was included in 39.08%. There was an average of 6 errors per page, with 3 being the most frequent. The most common errors were, *There is no attribute X*, *Required attribute x not specified*, *Missing document type declaration*, *End tag for element x which is not open* and *Element x not allowed here.* About a quarter of the collected pages were left out of the study as they were erroneously found to have 0 HTML elements. The WDG HTML Validator was preferred over the W3C HTML Validator in order to more closely compare to [18].

The focus of the work in [22] was the home page of Croatian top level domains. The list of web pages contained within it was supplied by the Croatian Academic and Research Network, a total of 75,357.

An automated program, dubbed Web Miner System, was written in Java to fetch a URL from the list and, in case it was available, would download, validate the HTML file and write the results in a database. The IP address and geographical location were also added. If a web site was not available, the error would be written to the database and the next URL would be processed.

Of the 55,089 web pages scanned, only 14% were considered valid, a higher number when comparing with results obtained in global studies. However, when taking into account only the most popular web pages from the study, this number would actually lower to 9%. Transitional flavors of HTML continued having a higher rate of adoption when compared to their strict versions.

Only the home page from the web sites was analysed. There was no mention of sub domains or any filtering technique used for the web site list. Validation errors and warnings were also not analyzed.

The authors in [23], collected a subset of approximately 92,000 HTML documents, selected from a list of over 2,6 million that were gathered with the use of the Inktomi web crawler. These pages were parsed by a modified version of weblint [24], written in Perl, that included the errors in a file instead of line-by-line. All non HTML and binary documents gathered were excluded.

A diverse set of properties was analysed, including, document size, tag / size ratio, attribute usage, among others. Validation-wise, results revealed that over 40% of the documents had at least one error. The top 12 most common errors detected occurred in more than 10% of the analysed web pages. For brevity, only the first five are mentioned here, *Missing <HEAD>*, *Outer tags should be <HTML> . . </HTML>*, *Heading-only tag found outside of heading*, *Netscape-specific tag* and *Missing <BODY>*.

Due to the fast changing nature of the Internet, this study is obviously far from the current state of the Web. No information regarding the selection of the studied web pages is available, nor any attempts to remove bias from the URL list.

2.2.2 Validation as a secondary focus

In order to gather information regarding the prevalence of different types of HTML and XHTML being used throughout the Web, the W3C HTML validator was used to inspect the top 100,000 web sites list from Alexa.com in [25]. Only the home pages were put through validation and included in the study. IP2Location was used to match the geographical origin of the web sites with their IP addresses.

The number of web sites that used Transitional flavors of the standards was concluded to be overwhelmingly higher, 74.7%, when comparing to the total sum of the Strict and Frameset flavors for all versions of HTML identified in the study, 2.5%. The amount of web sites that did not even include the DTD (which is required for it to be considered standard compliant) continued to be quite high, 22.0%. This absence of DTD fluctuated between 14.1% to more than 40%, depending on the country being analysed. There was a slight relation between popularity and web sites that included the DTD, with less popular web sites being also less likely to include it. The version and flavor of HTML in use also differed when analysing the results per country, but with a general tendency for a higher percentage of use for the most recent Transitional version (HTML 4.01 at the time). This is explained by the Transitional flavor providing a broader feature set and compatibility without as many restrictions as the Strict version.

Apart from the DTD, no validation errors, warnings or any other type of standard validation was performed. Inner pages were also not included, only the home pages were analysed. There was also no indication of filtering or any other procedure to remove bias from the list obtained from Alexa.com.

In [26], the main focus was the functionality and accessibility of web sites devoted to an environmental information. In an attempt to include only well-established web sites, as well as a geographically disperse sample, it collected 251 sites that shared the same theme from the Google Directory. Of these, 226 were available at the time of the study. Audit IT [27] and Alexa.com were used to calculate the traffic and popularity of each site while the validation was performed by the WebXACT [28] HTML validator.

Despite the focus on accessibility, similar conclusions to previous studies were reached regarding the low percentage of successful validation to standards. Only 4% of the total web sites managed to do so with no errors. HTML 4.0 / 4.1, was found to be the most popular of HTML variants, with a hefty 75%. Although it provides some relevant results, the study suffers from a very limited sample, only 226 web sites with a single common theme, environment, and a superficial validation analysis. There are no mentions of filtering of the list of web sites, nor any error and warning studies.

Ganzeli et al. [29] had an initial goal of analysing all web sites under the .br domain, which was later shifted to focus on a smaller subset, the .gov.br web sites. Despite not being the only component of this study, it still gathered information regarding HTML validity. The list was obtained by a series of Domain Name System (DNS) based queries to the top level domain and by performing a walk through, which resulted in 18,229 web sites. A program with modular architecture, ConNeCTOR, was developed to perform the analysis required for this work, the HTML validation was performed with the help of the W3C HTML Validator and accessibility evaluated by the ASES Web Sites Accessibility Simulator and Validator [30].

Diverse metrics were gathered, such as the size of pages, time difference between the servers' internal clock and the UTC time zone, amount of websites that supported IPv6 and HTML validation rates. The study showed that the percentage of valid web pages under the .gov.br domain were just over 6% in 2011, an increase from 5% in the previous year. Although the number of web pages analysed was quite high, the sample could be more diverse if it was not restricted to the Brazilian governmental domain. No apparent filtering of the list of web sites was done.

Beckett [31] collected a total of 39,162 domains that were listed by recursively searching through the records of the UK Domain Name System, with the help of the *host* program. In order to remove bias from multiple domains pointing to the same page, their HTTP headers were removed and their content was checksummed using MD5. This was used to identify duplicates and remove them. Domains with no WWW sites were also discarded.

While most of this study was focused on accessibility, of the 13,312 .co.uk web pages remaining from the initial list, only 6.50% were considered valid by the standards. The most used server was determined to be Apache, of which, version 1.1.1 was the most popular. The most prevalent DTD was HTML2, whose total sum of variants reached 66.88%. If a DTD was not found, the study assumed a default for the then most recent version, HTML3.2, which may have skewed its numbers slightly. There were no validation errors or warnings detailed.

2.2.3 Summary

Table 2.1 presents the main characteristics from each study, such as the URL source, type of domain and page being analysed, techniques used to remove bias from the list, number of pages processed and presence of error and warning analysis. Whenever such information was not specifically stated on the corresponding study, "N/A" was used. Each row represents a different characteristic, while its columns a separate study. It is also organised by relevance, with the four further right columns displaying information regarding the group of studies that have validation as a secondary focus and the remaining to the ones dedicated to standards compliance.

The **source of the URLs** used varies from one to several different origins. **Alexa.com and DMOZ** web directory seem to be a particularly popular choice, most likely due to the popularity ranking of the first and due to the large, but humanly curated list of the latter. Other URL gathering techniques such as **random IP address and search string generation** seem to have been **less successful** in providing a large amount of results. **Less than half** of the studies specifically state that they **analysed sub domains and inner pages**, focusing instead on top level domains and the home page, therefore assuming it represents the quality of the remainder of the site.

Preprocessing techniques for removing bias from the URL lists used are not frequently stated, removing duplicates is the most common. However, we still found a good number of procedures mentioned such as: setting a **limit of pages** analysed **per domain**, **removing** cases of **domain parking**, filtering **custom error pages**, removing **non-HTML links**, excluding **pages with 0 bytes in size and non 200 OK status**

codes.

None of the studies mentioned seems to **analyse dynamically generated HTML or non-indexed web pages**, this is likely due to the higher complexity required to do so. These pages may require authentication or the use of forms with a specific interaction or value to be accessed / generated. According to [29], analysing static HTML is still important, as using Active Server Pages (ASP), Personal Home Page (PHP) or other dynamic generation methods are simply different techniques of producing HTML and not a replacement.

There was also a **single HTML validator** in use **per study**, the tools mentioned were the W3C, WDG and WebXACT HTML validators, NSGMLS and Weblint. Making use of more than one may provide complementary information that would otherwise not be obtained. The one most commonly used was the W3C HTML Validator.

Even though the nature of these studies prevents them from being completely reproduced, due to the changing nature of the URL lists based on popularity, the changes on the analysed web pages or even the updates on HTML itself, **most studies omit many details** either from their methodologies or regarding the tools that were used. This severely limits their reproducibility.

Table 2.1: Comparison between the analysed studies.

Study	[2]	[1]	[16]	[20]	[18]	[23]	[22]	[25]	[26]	[29]	[31]
URL Source(s)	Random IP addresses, Alexa.com, random URL searches	Alexa.com	DMOZ, Alexa.com, W3C member companies	Open Web Directory	Open Web Directory	Inktomi Web Crawler	Croatian Academic and Research Network	Alexa.com	Google Directory, Yahoo, DMOZ	DNS-based queries and walkthrough	UK Domain Name System
Top level domains	Yes	Yes	Yes	Yes	Yes	N/A	Yes	Yes	Yes	Yes	Yes
Sub domains	N/A	N/A	Yes	Yes	Yes	N/A	N/A	No	N/A	N/A	N/A
Homepage	Yes	Yes	Yes	Yes	Yes	N/A	Yes	Yes	Yes	Yes	Yes
Inner page	Yes	No	Yes	Yes	Yes	N/A	No	No	N/A	Yes	No
Duplicates removed	N/A	N/A	N/A	Yes	Yes	No	N/A	N/A	N/A	N/A	Yes
Cap per domain	N/A	N/A	30 pages per domain	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Domain parking	N/A	N/A	Yes	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Custom error pages removed	N/A	N/A	Yes	N/A	N/A	Yes	N/A	N/A	N/A	N/A	N/A
Removed non HTML links	N/A	N/A	N/A	Yes	Yes	Yes	N/A	N/A	N/A	N/A	N/A
Removed non HTTP / HTTPS protocols	N/A	N/A	Yes	N/A	N/A	No	N/A	N/A	N/A	N/A	N/A
Removed 0 bytes in size	N/A	N/A	N/A	Yes	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Removed non 200 OK status codes	N/A	N/A	N/A	Yes	N/A	Yes	N/A	N/A	N/A	N/A	N/A
Dynamic HTML	No	No	No	No	No	No	No	No	No	No	No
Deep web analysed	No	No	No	No	No	No	No	No	No	No	No
Common errors analysis	Yes	Yes	Yes	Yes	Yes	Yes	No	DTD only	No	No	DTD only
Common warnings analysis	No	Yes	Yes	No	No	No	No	No	No	No	No
W3C Validator	Yes	Yes	Yes	No	No	No	N/A	Yes	No	Yes	No
WDG Validator	No	No	No	Yes	Yes	No	N/A	No	No	No	No
Weblint	No	No	No	No	No	Yes	N/A	No	No	No	No
WebXACT	No	No	No	No	No	No	N/A	No	Yes	No	No
NSGMLS	No	No	No	No	No	No	No	No	No	No	Yes
Global study	Yes	Yes	Yes	Yes	Yes	Yes	".hr" top level domain only	Yes	Environmental themed only	".gov.br" top level domain only	".co.uk" top level domain only
Total pages analysed	44,340	100,000	4,700,000	1,270,000	2,500,000	2,600,000	75,357	100,000	251	13,206,182	39,162
Number of pages after filtering	44,340	100,000	3,509,180	1,002,350	2,049,351	92,000	55,089	100,000	226	13,206,182	13,312

3. Methodology and approach overview

In this chapter, based on our state of the art analysis, we show what we believe are the main characteristics of a system that can achieve the results we are aiming for. In the following sections we detail our process of choosing particular aspects of said system, the sources for the Uniform Resource Locator (URL) list, the web crawler used for retrieval of information and the Hypertext Markup Language (HTML) validator used to check for standards compliance.

3.1 Proposed system overview

From the analysis of the previously mentioned studies, one can extract the general idea that a similar research might employ. Figure 3.1 shows the overview of such a system. A URL list is obtained by, ideally, using several different sources, popularity based, human curated or whole Web crawls should be used in the interest of diversifying and better representing the current state of the Web. After filtering said list in order to remove bias, it is sent to a Web crawler that will fetch the raw HTML files and store them in a file system or database. A script or some other program would then be used to automate the process of retrieving each file, analysing and parsing its information, as well as putting them through validation with one or more tools, process the results and store them. After this task, a data analyser is used to process results, find patterns and other relevant information.

Ideally, in order to process the largest amount of information in a smaller time period, the crawling and validation tasks should be able to be run concurrently.

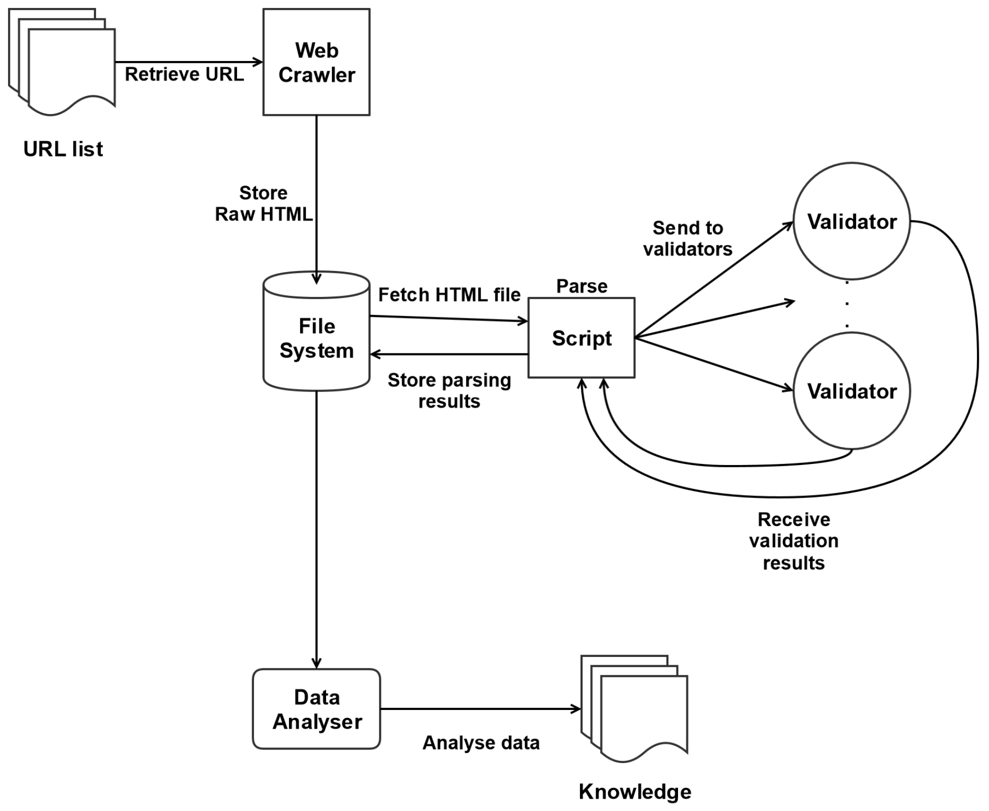

Figure 3.1: Overview of the proposed system.

3.2 Requirements

Based on the previous information, we reached to a set of functional and non-functional requirements that are listed in Table 3.1. Functional requirements are identified with FR##, non-functional requirements follow the NFR## pattern.

For prioritisation, we employed the MoSCoW method [32], which makes use of a simple four step scale of priorities. "Must have" and "Should have", represent the highest priorities, for features that must absolutely be included and important features, respectively. On the lower end, "Could have" is used for characteristics that would improve the application but are not a necessity. Finally, "Won't have" is used for features that are interesting but will not be included in this version of the project. The full description of each requirement is present in Appendix A.

3.3 URL selection

In order to obtain a representative sample of the state of the Web, we aimed to collect a mix of diverse and frequently accessed pages. This was achieved by making use of two sources of URL, one based on popularity among users and the other on a large quantity of diverse URLs. Very early on, we decided against doing our own Web crawl for the purpose of collecting URLs, as it seemed to provide no extra benefit, thanks to

Table 3.1: Functional and non-functional requirements list.

Identifier	Name	Category	Prioritisation
FR01	WARC file manipulation	Data handling	Must have
FR02	HTML file manipulation	Data handling	Must have
FR03	Validator integration	Data handling	Must have
FR04	Valid results	Data handling	Must have
FR05	DBMS integration	Data handling	Could have
NFR01	Resume functionality	Recoverability	Should have
NFR02	Processing data times	Performance	Must have
NFR03	Hardware dependent performance	Scalability	Should have
NFR04	Ease of use	Usability	Could have

the multiple, frequently updated sources available.

3.3.1 Popularity based approach

Of the pages that track web sites' popularity, three were initially chosen due to being the most prominent, SimilarWeb.com [33], Quantcast.com [34], and Alexa.com [9].

The first one was rejected because of the small amount of freely available information, only the top 50 most popular web sites for each category and / or region do not require a subscription.

Quantcast.com freely distributes the top one million most popular web sites list for a variety of countries, however, only the United States' list is easily available for processing. The list is also partially censored by Quantcast, as information from clients that wish to remain anonymous will be replaced with *Hidden profile*. Although, at the time of writing[1], the total number of removed profiles within the list was limited to 4,723 in one million, or 0.473%, if narrowed down to the first one thousand results, that percentage rises to 14.3%, which is a significant number of omissions among the most popular web sites.

Alexa.com displayed none of the previous drawbacks, a list of the top one million most popular web sites globally is available freely, uncensored and in an easy to process format. Some of the disadvantages of using Alexa's list, mentioned in [1], have also been minimised. The toolbar that users install on their browsers, in order to gather Alexa's data, is now available for Mozilla Firefox and Google Chrome, in addition to Microsoft's Internet Explorer. Alexa.com also gathers information directly, through the use of a crawler, as well as adding their own code to willing web sites. The list being comprised of home pages only has also been taken into consideration. We configured Nutch to add one of the inner pages for each site to the fetch list during the retrieval of their HTML files.

3.3.2 Mass quantity approach

Concerning the gathering of a large amount of URLs, three sources were considered, Majestic, CommonCrawl and The Directory of the Web (DMOZ).

[1] 20th October, 2016

Majestic [35] claims to be the world's largest link index database, with more than 4 trillion unique URLs found to date. However, despite sporting an extremely vast data set, it was rejected as its information is kept behind a paywall.

Common Crawl [36] is a project that aims to make large web crawls available to the public, free of charge. CommonCrawl's data is also very extensive, representing more than seven years of monthly web crawls, each with over one billion URLs. Its information is delivered in Web ARChive (WARC) format, which is shared by the crawler chosen to gather data from the popularity list of URLs.

The Directory of the Web [17] states that it is the largest human-edited directory available on-line, it was our choice for second source of URLs. Since it is curated by humans, its' URLs are organised in categories related to their content. This extra information was considered potentially useful for our study. It has also been used in several of the previous studies in this area. Dead links are also routinely removed from the listing. This causes its total number of available URLs to fluctuate between three, to four and a half million.

3.4 Web crawlers

Web crawlers, also known as spiders or bots, are computer programs that traverse the World Wide Web and retrieve and store information from it. This can range from a certain type of file to whole web sites. The retrieved information can then be used for statistical analysis, indexed so it's employed in a search engine, used as an on-line / offline mirror, for archival purposes, maintenance tasks, among many others [37].

Most web crawlers begin crawling with a seed of one or more web sites, the desired information is retrieved from these sites and any existing links are added to the fetch list so that they also get processed and analysed. This general behaviour usually repeats until a certain condition, such as maximum depth or number of pages is reached [38]. They can behave in many different ways, some of which detailed in [39]:

- **Breadth-first**. Try to cover as many pages as possible by crawling web pages and following their outgoing links first. This is usually the case for search engines and other applications that require large amounts of data from diverse locations.
- **Re-crawling pages for updates**. Go through a list of previously scanned pages and fetch the ones that have been updated since the last scan.
- **Focused crawling**. Instead of attempting to be as broad as possible, crawl only a particular subset of web pages, or the ones dedicated to a specific subject.
- **Random walking and sampling**. A path of random steps through the Web, used for estimations. The Web can be represented as a graph, with pages taking the role of vertices and hyperlinks acting as edges. A random walk will start in a set group of pages and follow one of the existing links to another page. This process is then repeated [40].
- **Crawling the hidden web**. Focus on non indexed data that, despite being available, requires certain parameters, inputs or credentials to be retrieved.

Since crawling web pages is an easily distributable task, crawlers usually process several web pages concurrently. The author in [37] details the reasons that can make web crawlers harmful, these include their recursive nature, the multitude of requests that one can make in a short time, their ability to run in a continuous mode and inefficient and bad implementations that download file types that they cannot process. Reduced network resources and high server load are problems caused by from their use. To counter this, some websites intentionally developed spider traps, also known as black holes, directories with infinite depth or other similar technique, that can make a crawler continuously try to reach its end, effectively trapping it and reducing its performance, output or crashing it [38].

The Robots Exclusion Standard [41] was created as a way to minimise these problems and has seen widespread adoption. A file named *robots.txt*, that describes the files and directories which should or should not be targeted by crawlers is placed in the root directory of a web site. Access can be fine tuned from a single file or directory, to the whole web site, different rules for specific web crawlers can also be set. The permissions described in *robots.txt* are not binding, they are simply guidelines for polite crawlers to know how to behave. Another politeness metric, not detailed in the standard, is the amount of time between two consecutive requests to the same server. Some web crawlers have adopted to follow the Crawl-delay directive [42] which, despite being a nonstandard extension, can be added to the *robots.txt* file in order to specify the amount of time between different accesses.

As expected, there are various web crawlers available, both commercially and open source. As any other application, they sport a variety of features that should be taken into consideration when choosing the one that best suits ones' needs. According to [43], the main properties that must be fulfilled are:

- **Robustness**. Ability to avoid or withstand spider traps, as well as enduring interaction with badly implemented web sites.
- **Politeness**. Following the guidelines set by each web site, using the Robots Exclusion Standard, and avoid causing a high server load for the targeted pages.
- **Distributed**. Ability to be run from multiple machines in a coordinated manner.
- **Scalable**. Capable of increasing its throughput when provided with additional resources.
- **Performance and efficiency**. Make good use of its provided resources.
- **Quality**. When faced with a choice between multiple pages, the crawler should be able to choose the most important or relevant to the task.
- **Freshness**. The crawler is able to fetch newly updated pages in an acceptable time frame.
- **Extensibility**. Its architecture is modular enough to enable the development of added features or the change of existing ones.

In our study's case, the Quality and Freshness features are not particularly relevant as it will be based on a single crawl, from a previously gathered URL list. The amount of data expected to be processed is also an important factor, [44] defines three categories, mirroring, medium and large collections. Due to the expected amount of pages to be processed by this study, and considering possible future work, it should fall into the medium to large categories.

Table 3.2 shows a comparison between the analysed crawlers for this study, each column represents a feature, while each row is representative of a single crawler. Whenever a feature is not immediately apparent, either through documentation or its code, *N/A* is used[2].

As previously mentioned, there is a wide variety of open source web crawlers, [44] and [43] were used as the basis of our search. As shown in the previous chapter, Table 3.2 displays a comparison between the features of the crawlers analysed for this study.

Of the various tools analyzed, Ebot, Heritrix, Ht://Dig, Pavuk, Selenium, WebSPHINX and WIRE were discarded due to not being updated within the last two years. Crawler4j and Scrapy cannot be used in a distributed environment, which was deemed a beneficial feature in case our crawling scope were to increase. DataparkSearch, HTTrack and Norconex make no mention of this feature either and were thus not considered as well. Coupled with more than a year without being updated, Bixo seems to lack the

[2]Last updated as of 7th of November, 2016

Table 3.2: Comparison of features between analysed crawlers.

	Last updated	Multi thread	Distributed	Supports robots.txt	Can crawl pages that require authentication	Language
Bixo	10/2015	Yes	Hadoop	Yes	N/A	Java
Crawler4J	09/2016	Yes	No	Yes	Yes	Java
DataparkSearch	11/2016	N/A	N/A	Yes	Yes	C
Ebot	04/2011	Yes	Yes	N/A	N/A	Erlang
Heritrix	01/2014	Yes	Hadoop	Yes	Yes	Java
Ht://Dig	06/2004	N/A	N/A	Yes	Yes	C++
HTTrack	09/2016	Yes	N/A	Yes	Yes	C
mnoGoSearch	12/2015	Yes	Yes	Yes	Yes	C
Norconex	08/2016	Yes	N/A	Yes	Yes	Java
Nutch	10/2016	Yes	Hadoop	Yes	Yes	Java
Open Search Server	10/2016	Yes	Yes	Yes	Yes	Java
Pavuk	02/2007	Yes	N/A	Yes	Yes	C
Scrapy	01/2016	Yes	No	Yes	Yes	Python
Selenium	04/2013	N/A	N/A	N/A	N/A	Python
WebSPHINX	03/2007	Yes	No	Yes	Yes	Java
WIRE	02/2010	No	N/A	Yes	N/A	C++

ability to crawl through secure pages that require authentication and was felt as a disadvantage since including them in our study was a possibility that was being considered. Despite sporting all of the considered features, mnoGoSearch was also discarded due to being built in C language, as Java alternatives would be easier to modify in case such need ever arose. The final two alternatives, Nutch and OpenSearchServer, share all of the analysed features, programming language and even the date of their last update.

As [44] describes, “the range of open-source tools available to make a choice is broad and there is no clear software that is more suitable than others”. Ultimately, Apache Nutch was our choice. It is a feature rich implementation, easier configuration, has a big community, good support and very active development, due to being a top tier Apache project. In terms of performance, it is multi threaded by default, “highly scalable and also dynamically scalable through Hadoop” and “can be run on a cluster of up to 100 machines” [43]. It is one of the choices of Serrão [44] for large scale crawls, which also adds that “to provide reliable, fast and scalable computing” it is “the best answer”. This is backed by high profile users such as Wikia Search, Creative Commons and Common Crawl, among others. Additional features include support for the Robot Exclusion Standard, the ability to crawl web pages locked behind authentication when supplied with the correct credentials and extensive configuration options.

As for the choice between the 1.x and 2.x branches of Nutch, we opted for the first one, the stable version, due to its higher number of features, plug-ins, bug fixes and better performance versus a higher storage flexibility and a more complex configuration for 2.x. In a comparison between both branches, Klaussner [45] determined that version 1.x beat 2.x on all tasks, performance wise, by a considerable margin.

3.5 HTML validators

HTML validators are tools that compare HTML contained within a file with its standards. Mirroring the functionality of a compiler, they will warn the user where errors and warnings are found, if any. Unlike a compiler, however, neither their use is mandatory, nor the absence of errors is required to produce a functional HTML file. Thus, it is very common for HTML files available on-line to contain a significant amount of errors and deviations from the standard.

Table 3.3 shows a comparison between open source HTML validators and their most relevant features. Each column represents a feature, each line a single validator. Whenever a feature was not able to be determined, *N/A* was used. Apart from HTML Validator Firefox Add-on and Total Validator, only the on-line versions of the crawlers were tested. In order to cover as many web pages as possible, support for HTML 5 and HTTP Secure (HTTPS) were considered important features. As a method of comparing their output, the same test web page (http://www.google.pt) was validated with each one and the number of errors and warnings registered.

Table 3.3: Comparison of features between analysed HTML validators.

	Last updated	Supports HTML5	File size limit	Can parse secure pages	Maximum number of errors / warnings	Control web page errors / warnings
Anybrowser Weblint Gateway	1997	Up to HTML 3.2	97Kb	No	N/A	N/A
CSE HTML Validator	09/2016	Yes	292Kb	Yes	26 errors 25 warnings	19 errors (1 CSS related) 4 warnings
Dr. Watson	07/2016	Yes	70Kb	No	N/A	0 errors 0 warnings
HTML Tidy	04/2016	Yes	>1Mb	Yes	N/A	0 errors 36 warnings
HTML Validator Firefox Add-on	06/2016	Yes	>1Mb	Yes	N/A	16 errors 0 warnings
HTML Validator Pro	N/A	Yes	>1Mb	Yes	N/A	27 errors 0 warnings
Total Validator	08/2016	Yes	>1Mb	Yes	N/A	19 errors (1 parsing related) 1 warning
Validome	N/A	N/A	N/A	N/A	N/A	N/A
Web Page Purifier	09/2016	Up to HTML 4.0	N/A	No	N/A	N/A
WGD HTML Validator	02/2012	Up to HTML 4.01	>1Mb	Yes	50 errors	46 errors 0 warnings
W3C HTML Validator	11/2016	Yes	>1Mb	Yes	N/A	20 errors 2 warnings

Validome [46] was the first of the group to be excluded as its home page was inaccessible at the time of writing and was thus unable to be tested. Anybrowser Weblint Gateway [47] is based on very old technology, only allowing for HTML up to version 3.2 and non secure pages to be tested. Despite this, it did include an interesting feature, allowing to perform validation against the standard or including browser specific tags (Internet Explorer and Netscape Navigator). Web page Purifier [48] also suffered from the same problems, with the added difficulty of needing full control of the web pages being tested, requiring a specific file in their root directory for it to work. Web Design Group (WDG) HTML Validator [19] also dragged behind the current version of HTML, supporting up to version 4.01 only, as well as featuring a maximum error limit, ignoring any occurrence past the 50th detection. The high number of errors detected on the test page seems to have been caused by lack of HTML 5 support. As for Dr. Watson [49], although it supports validation of HTML 5, it detected no errors on the test web page, despite several being present. It also did not support the validation of pages delivered through HTTPS and had a maximum size limit of 70Kb. CSE HTML Validator [50] displayed a somewhat short page size and error limits, 292Kb and 26 errors and 25 warnings. It was the only validator to specify if an error was Cascading Style Sheets (CSS) related.

The remaining five tools all supported pages made with HTML 5 and delivered through HTTPS, their maximum size limit was over 1Mb and there did not appear to be a maximum number of errors and warnings. HTML Tidy [51] did not appear to distinguish between errors and warnings, opting to classify all occurrences as errors. HTML Validator Firefox Add-on [52] required every frame to be validated separately and appeared to not be able to be used in an automated way. Total Validator's Basic version [53], the only one available for free, did not include the ability to test multiple web pages. Finally, HTML Validator Pro [54] incorrectly flagged unambiguous ampersands [55] found within hyperlinks as errors, thus totalling 27 on the test page. The World Wide Web Consortium (W3C) HTML Validator [11] displayed none of these drawbacks and a few additional features. It is possible to select a default document type and character encoding, automatically detect them or specify a choice in case they are not mentioned. A listing of all images and their textual alternatives can be outputted, as well as displaying heading-level and structural outlines for the page. Cleaning up the errors from the HTML being analysed is also possible. The simultaneous use of multiple validators was initially considered, but as this was determined not to be beneficial, our choice was to make sole use of the W3C HTML Validator.

4. Tool

The following chapter presents the state of the current tool implementation. First, we give an explanation of its behaviour and how it processes information. Next is a description of the external tools and libraries that are necessary for the tool to obtain results. Following is a brief discussion and listing of the information currently obtainable by the tool.

4.1 Tool behaviour

Figure 4.1, at the end of this chapter, shows the overview of the current tool implementation. In short, an external web crawler, in our case Apache Nutch, is used to gather web page data from our chosen source and export it into Web ARChive (WARC) files. Our tool will then read through them and parse and store all the relevant meta data and Hypertext Markup Language (HTML) information. Next, it will put the web pages' HTML code through validation, with the use of an external tool, process the results and add them to the previously collected information. Finally, all the data is outputted and is ready to be analysed.

As previously mentioned, Uniform Resource Locator (URL)s are gathered from the chosen source and crawled with the use of Apache Nutch, their information is outputted and stored in WARC files. This is a separate step and is done without direct involvement of our application. The tool starts by reading the contents of a specific folder and making a first pass through all the WARC files, connecting original URLs to their redirection targets. As shown in Figure 4.2, also present at the end of this chapter, the main class will then distribute work through all of the available threads. Each one will then begin processing web page information, retrieving their useful meta data, in case it is available for the files in question, such as the URL being processed, content type, Hypertext Transfer Protocol (HTTP) code, type of server the page is being hosted at and its raw HTML code. Remaining meta data is discarded.

Next, the HTML code is parsed with the use of Jsoup and a series of HTML related information is extracted and stored with the previously collected data. Examples of this information are the type of HTML in use, total number of HTML elements, among others listed in the section 4.5. While retrieving information from its HTML elements, the size of the web page's textual content, HTML elements, among others, are fetches and a series of related ratios are calculated.

Afterwards, an HTTP connection is established with the local installation of the World Wide Web Consortium (W3C) HTML validator and the HTML code of the web page being currently processed is sent for validation. If said web page is made with HTML 5, the validator will send it to a second, HTML 5 specific validator. The results are received by the original thread and deserialized with the help of Gson, the number and type of errors and warnings are stored. All of the information retrieved from the web page is then added to the output file and the process repeats until all WARC files are processed. Finally, the main class will generate the all the totals based on the information retrieved.

For reference, this is performed on a virtual machine with a 4 core, 2Ghz processor, with 8Gb of RAM memory and 200Gb of hard disk space. Table 4.1 details the versions of all of the software involved.

Table 4.1: Versions of all the software used.

Software	**Version**
Ubuntu	16.04.1 LTS 64bit
Java	OpenJDK 1.8.0_111
Apache Nutch	1.12
Apache2 Web Server	2.4.18
W3C Validator	1.3
W3C HTML5 Validator	vnu.jar_17.2.1
Gson	2.0.0-beta3
Retrofit	2.1.0
Jsoup	1.10.1
Commons-CSV	1.1

4.2 Dependencies

The current version of the tool is dependant on a few external tools. The selected web crawler, Apache Nutch, is tasked with crawling through the URL list and output the retrieved data, which includes the HTTP headers, into WARC format. The W3C HTML validator available on-line is actually made up of two different validators that are used depending on the type of file in need to be analysed. One exclusively validates files made with HTML 5, it is available in Java ARchive (JAR) format and was simply run through a command on the terminal that specifies its listening port. The other validator processes HTML files of all remaining types and required a more extensive configuration. It is run through the use of Apache web server. In order to obtain consistent results throughout the whole duration of this study, both versions of the validators were installed and ran locally.

There are also four Java libraries currently in use by the prototype. Retrofit [56], a type-safe HTTP client, is used to simplify connections established between the tool and the web server running our local version of the HTML validator. Jsoup [57], an HTML parser, is used for extracting information from the HTML code of the target web pages, such as number and occurrence of the different tags. In order to simplify the process of reading and writing information onto Comma Separated Values (CSV) files, Apache Commons-CSV [58] was used. Finally, Gson "a serialization/deserialization library that can convert Java Objects into JSON and back" [59] is used to ease the task of processing results sent by the validators.

4.3 Configuration

Some configuration steps were made, particularly in regards to the used crawler, Apache Nutch. Its Java max heap size was increased to 6Gb to increase performance. It was made to store HTTP headers and requests, as well as to not impose any size limits on HTTP content retrieved. It was configured to include parsing during the fetching step and to follow up to five redirections. This number was chosen as to not significantly increase the time spent on the fetching phase. In order to follow a bigger number of redirections, the crawler's filtering rules were relaxed slightly, allowing for the fetching of web pages with the characters *?* and = in them. Some URLs from our chosen sources pointed directly to non-HTML files, in order to further increase the filtering done by the crawler, the following file types were added to its ignore list:

– .mp4, .rar, .swf, .pdf, .owl, .xls, .ppt, .doc, .docx, .dmg

Finally, an additional 5Gb of memory were added to the operating system's swap.

4.4 Challenges

Every project has its set of challenges that need to be overcome in order for it to come into fruition. Naturally, this work was not an exception. While some had straightforward solutions, a few of them required a large time investment.

Some of the URLs from our sources did not point directly to the desired web pages, instead making use of redirections to reach them. Each redirection caught by the crawler produces a separate output file with a usually empty web page with the next step of this path. Since our goal was to only include the target destination web page in our analysis and not the remainder, a pre-processing step was added. This was done so it is possible to connect the original URL to its final redirection target, as well as removing the undesired files.

Most of the data collection tasks were reached through an iterative process. An extremely simple web page, which contained the feature being collected, would be manually created and put through our tool's analysis. After a successful result, the page would be modified with more instances or different uses of said feature and tested again. Verification would then evolve into using actual randomly selected web pages of varying complexity through the tool and manually comparing the results with their HTML code. The same technique was used to verify the HTML validation information returned by our application, simple pages with a single error were manually created and tested. The next step was to use pages with multiple errors of the same type, errors of more than one type and, finally, actual randomly selected web pages. Our tool's HTML validation error and warning counts and types were then compared to the output produced by manually validating them with the W3C HTML validation tool [11] installed locally.

Handling the HTML validation error and warning messages was a time consuming process. The majority of said messages include up to six different text variables that are usually reserved for the HTML element or attribute that triggered the problem, among many other possibilities. As an example, "*Duplicate attribute id.*" and "*Duplicate attribute href.*" are obviously referring to the same error type. Thus, the usage of a regex was deemed an appropriate solution. The full list of error and warning messages for the validation of HTML older than version 5 was easily accessible within the source code of its validator. It was a simple process of creating a regex rule for every one of the 447 messages that included any type of variable text. The messages reserved for validation errors and warnings for HTML 5 however, were not such an easy task, they were not easily discernible in its validator source code. Therefore, the process of creating their rules required us to run our tool with a set of pages created with HTML 5, waiting for the output and using it to identify messages related to the same type of problem and creating their respective regex rules. Naturally, on the first pass, most error messages were considered unique as no regex rules had been created for them yet. With each pass, the number of web pages included in the processed sets was increased in order to obtain the highest number of different messages as possible, which in turn, also increased the time spent processing them. Gradually, the set of regex rules reserved for validation error and warning messages for HTML 5 grew more robust.

Due to the limited resources of the virtual machines tasked with running our tool, several performance considerations were taken. It would not be possible to keep all the gathered data in memory, particularly due to the sheer size of the intended web page set. Thus, after fully processing a web page, its information is, instead, added to temporary files on the hard drive. This option has two side effects, on the one hand, it requires a separate step on the output data in order to generate the totals. On the other hand, it facilitates the implementation of a resume feature, so the processing does not have to restart in case it is stopped before it is finished, due to an unforeseen problem such as a power failure. In order to speed up the processing of web pages, the tool was changed from its initial linear processing to a configurable number of threads, each processing its own web page. This change significantly increased the performance of the application.

When the time came to abandon the testing grounds of a local virtual machine and create a fresh installation of the work environment, with crawler, validators and our tool, on a remote machine, things stopped working. None of the HTML 5 web pages were able to get through the validation process. This ended up revealing itself as a major problem and one of the main causes for missing the planned schedule. The first attempts to fix the problem revolved at looking at our tool's code, debugging the areas where the problem might arise from, but it persisted. Fresh installations were attempted, double checking each step, configurations for all the applications involved were confirmed and new ones were attempted, but to no avail. All error messages pointed to a character encoding problem. A version mismatch was noticed between the validators being used, a more recent version having been released since the installation on the testing area. The problem continued after using the same version on both machines. All remaining software versions were checked, Java in use, crawler, operating system and its language settings, as well as script and library dependencies for the validators, all were identical. Yet, the same behaviour lingered. The only apparent difference between machines now seemed related to the Virtual Machine Monitor (VMM), the original installation using Oracle VM VirtualBox [60], the remote one employing Xen Project [61]. A third party cloud solution with a different virtualization environment was used to repeat the process of fresh installations, which still lead to the same outcome. Ultimately, the problem was bypassed by making a clone of the original virtual machine and importing it to our remote one.

4.5 Retrievable information

Currently, data gathered by our tool can be arranged into four different categories: meta data, information regarding HTML elements, data related to HTML validation and file size information.

The first category, meta data, includes the analysed web page's URL, as well as the original URL, in case there were any redirections. Whenever included, the HTTP code is also retrieved. The content type header, information that describes the type of content sent by the server reply, is stored. Server type and version, although not guaranteed to be available, is also collected when possible. Finally, the charset for the included data is stored.

Regarding HTML elements, the following information is saved. The type of HTML declared to be in use, or the absence of its declaration. The total number of HTML elements present in the analysed page, as well as a list containing their names and number of times each one of them occurs. Going down a level, for every HTML element encountered, the list of every attribute and number of times they occur is also stored. The total number of hyperlinks in the page, discerning between bookmarks, internal and external links. Hyperlinks that point to a place within the same page, to a place in the same domain and to a place outside the domain, respectively. The total number of images, as well as a list of their types and number of times each one is detected. Lastly, the total number of comments found in the HTML code is also accounted for.

HTML validation of the analysed web pages also produces important information. Of note, the total number of validation errors and warnings for each page is stored. Likewise, a list of all the different types of errors, warnings and the number of times each one occurs is also kept. Finally, a list of the message variables is gathered for each error and warning type so it is possible to detect the frequency with which element or attribute is responsible for a particular problem.

A series of file and element sizes are tracked in order to understand how much of an HTML file is taken by each one. Naturally, the file size is collected. The size taken by text content, HTML elements and comments is gathered s well. This information is then used to extract the text content to HTML elements, HTML elements to file size, text content to file size and HTML comments to file size ratios.

All this information is outputted to two separate sets. The first, a general information, includes all the gathered data and is meant to provide a global view of the web pages analysed. The second group, contains all the characteristics and attributes collected previously, but is restricted to data originated from web pages constructed with HTML 5.

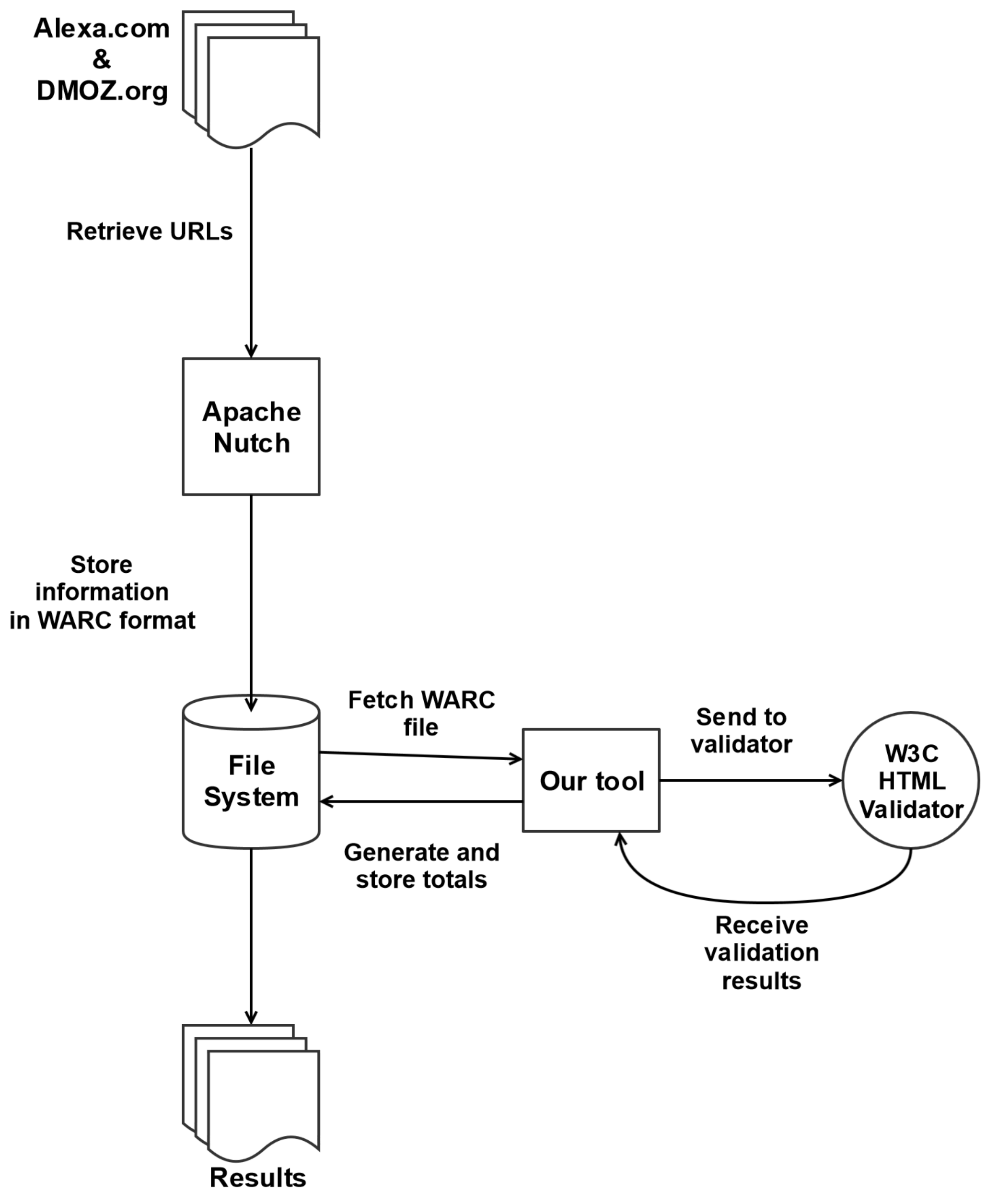

Figure 4.1: Level 1 of the current implementation.

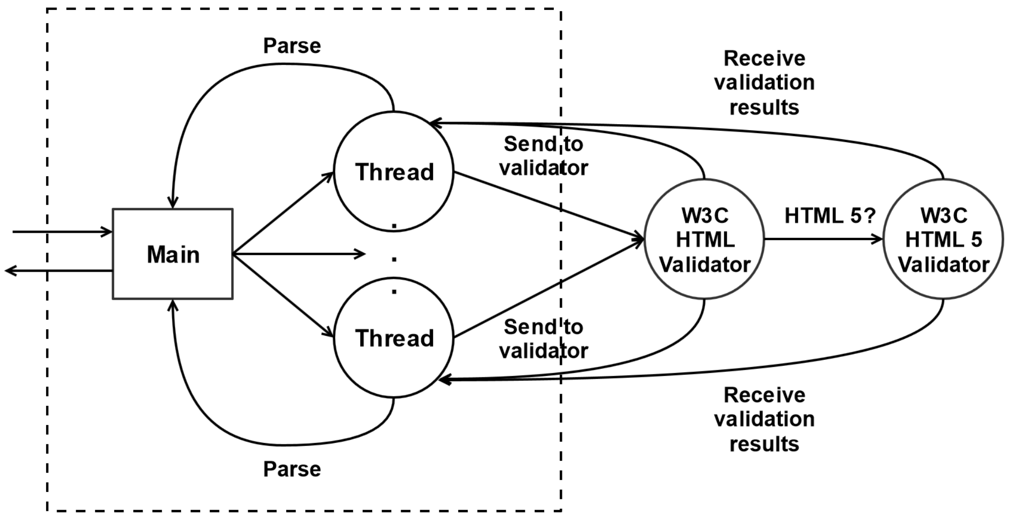

Figure 4.2: Level 2 of the current implementation.

5. Results

In this chapter, we present our most relevant results gathered from the output of our tool. First, this is done for all the pages analysed and, then, to pages made specifically with Hypertext Markup Language (HTML) 5.

The source of Uniform Resource Locator (URL)s used was the top 50,000 most popular web pages list, according to Alexa.com, collected on the 30th of August, 2017. Of the original 50,000 URLs, only 46,682 web pages were successfully crawled due to connection timeouts, temporarily being down for maintenance or other assorted reasons. The list was further shrunken as the files crawled from 1539 of them were discarded for having 0 bytes. The external validator failed to successfully analyse a total of 1399, leading them to also be excluded. Finally, 2 web pages were eliminated from the list due to their HTML failing to be parsed. This produced a final count of 43742 web pages.

5.1 General findings

First, we detail information from web pages made with all versions of HTML, from a variety of different aspects, document types, HTML elements, validation data, file size information and a short summary of what the average web page consists of.

5.1.1 HTML version

As can be seen in Figure 5.1, the overwhelming majority of web pages analysed were made with the use of HTML 5. This comes as no surprise, as the World Wide Web Consortium (W3C) [62] upgraded the HTML 5 Draft to Recommendation on the 28th of October, 2014 [63], giving Web and IDE developers plenty of time to start using it. Extensible Hypertext Markup Language (XHTML) 1.0, with its different variants, is also a popular choice, comprising 14.58% of gathered web pages. Due to the source of our URLs being based on user popularity, there is a certain level of quality expected from the web pages contained there. This doesn't prevent a significant number, 4115, from still not making use of the document type declaration in order to state the type of HTML in use on their files. It is, however, a much lower figure when compared to previous studies such as [1], staying under 9.5%. With each, newer version of HTML, comes an increased percentage of web pages that make use of it. Two exceptions are clear though. A surprising number of pages powered by HTML 2, 1097, continues to hold its ground despite the constant advances on the Web, perhaps, still made popular by their unwillingness to change. XHTML 1.1, on the other hand, seems to have seen its older version, XHTML 1.0, preferred by developers. An indication that the small set of differences between both of them as not been sufficient to entice developers into using it.

Finally, the group entitled *Other* gathers mixed types such as XHTML+RDFa, mobile oriented, custom type declarations and erroneous ones.

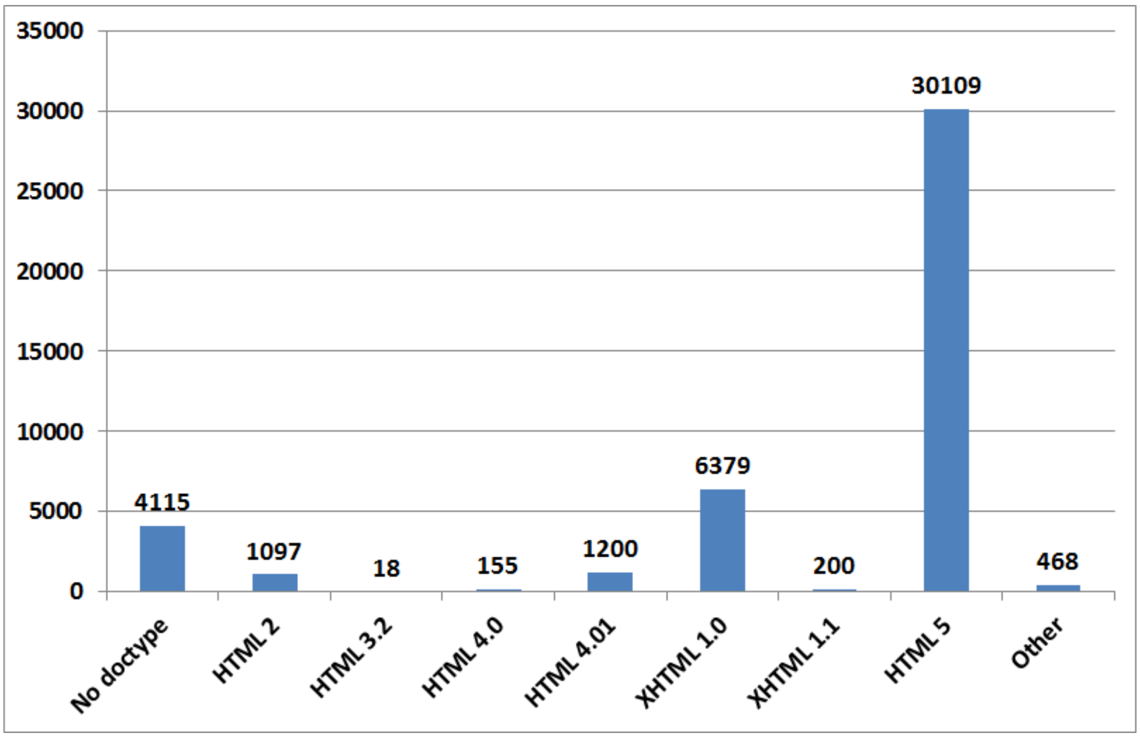

Figure 5.1: Breakdown of the different types of HTML found in use.

According to a previous study [25], when faced with a choice between Strict, Transitional and Frameset variants of HTML, developers will opt to make use of the more relaxed set of rules, transitional flavour. This seems to be confirmed when looking at Figure 5.2, which presents a more detailed description of their proportions for HTML 4.0 and 4.01. Version 4.0 had an 87% occurrence of its Transitional variant and a mere 1% of the Strict version. No pages making use of the Frameset variant were detected. For document type declarations that did not explicitly state their variant in use, 12%, they were simply labelled HTML 4.0. Still on Figure, 5.2, the same trend occurs for HTML 4.01. Transitional is used on slightly more than three quarters, followed by Strict with 21% and a few pages make use of the Frameset variant. Again, pages with no explicit variant were labelled simply as HTML 4.01.

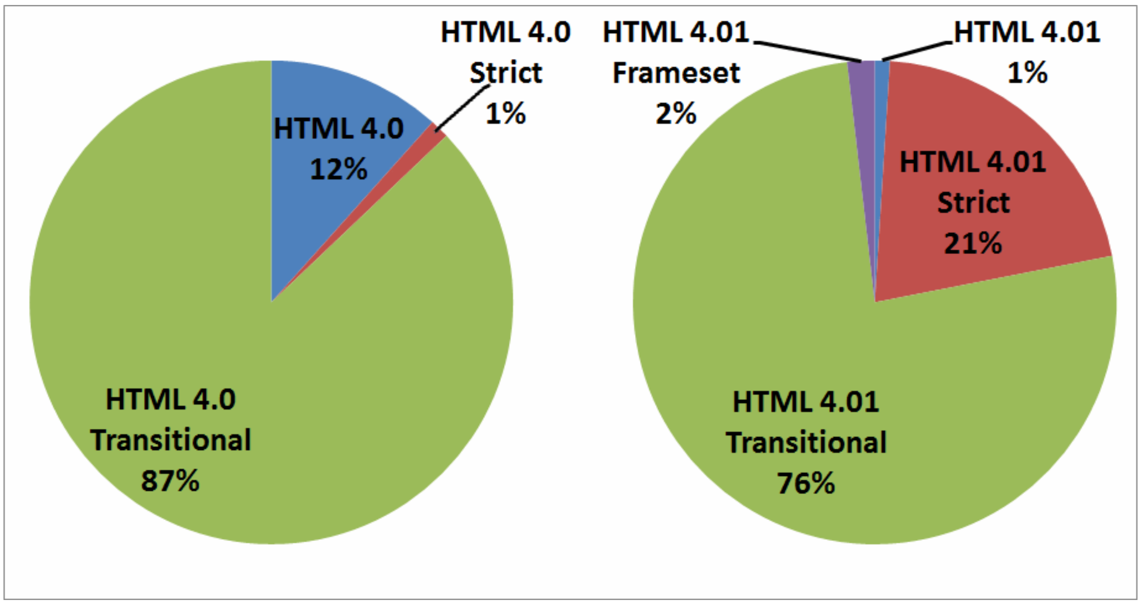

Figure 5.2: Flavour for HTML 4.0 (on the left) and HTML 4.01 (on the right).

Looking at Figure 5.3, the same trend continues with XHTML 1.0 variants. Transitional, once again, dominates the chart, followed by Strict. No web page made use of XHTML 1.0 Frameset and a single case of non explicit variant was present. XHTML 1.1, however, turned things around, non explicit variant declarations comprised the majority of cases detected, while Strict and Transitional were residual, with a mere 1% each. Frameset continued to be the least used variant.

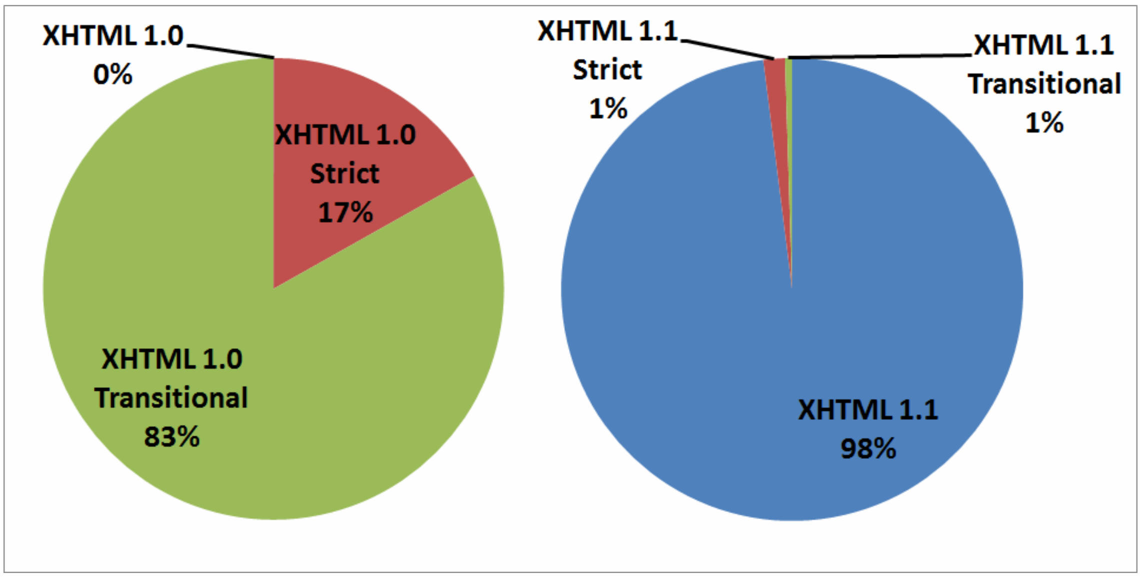

Figure 5.3: Flavours for XHTML 1.0 (on the left) and XHTML 1.1 (on the right).

5.1.2 HTML elements

Table 5.1 lists all HTML elements found, with a percentage of occurrence higher than 1%. The most recognisable feature of a web page is also its most common element, the hyperlink. Further information regarding this element is detailed further in this section. The second most common element, <div>, is invisible to the regular user. It is used by developers to group multiple elements together and create sections that can be formatted at the same time through the use of Cascading Style Sheets (CSS). With the current paradigm of using HTML as base, CSS for presentation and JavaScript for functionality, it is only normal to see it thoroughly used, closely following the first spot. The element <span>, in the 4th position is also used for the same effect and thus remains away from public scrutiny. Several elements are commonly used due to their nature, <li>, <ul>and <option>are all used to represent single items on a list. This means that anytime a list is created, multiple occurrences of these elements will be spotted, inflating their numbers. The same can be said of element <td>, which represents a single table cell.

A group of text manipulation elements, <p>, <i>and <br>, are expected to be frequently used as even a small amount of text can contain multiple occurrences of them all.

Table 5.1: Common HTML elements.

Element	Quantity	Percentage
<a>	9242300	22.37%
<div>	9117367	22.07%
<li>	4812468	11.65%
<span>	4037096	9.77%
<img>	2069446	5.01%
<p>	1134487	2.75%
<script>	865273	2.09%
<td>	854573	2.07%
<ul>	793052	1.92%
<option>	708678	1.72%
<i>	701897	1.70%
<br>	620273	1.50%
<meta>	561510	1.36%
<link>	468476	1.13%
<h3>	421333	1.02%

A brief mention regarding HTML comments, the average number found in our data set was 32 per page. Several pages displayed no comments of any type, but one of them managed to reach the hefty sum of 6692.

Hyperlinks

In order to sustain its status of world wide web of information, the Web heavily relies on hyperlinks. As noted on Table 5.1, it was the most common HTML element found on all of the analysed pages in our study, a total of more than 9.25 million. Naturally this brings the average number of hyperlinks per page to 211, a confirmation of the connected nature of the Web. One page, *http://www.sarkarinaukriblog.com/*, went above its call of duty by producing 15322 hyperlinks, while there were multiple cases of pages with no connections whatsoever.

We also made the distinction between internal and external links, as well as bookmarks. We considered internal links as hyperlinks that pointed to a place within the same domain as the page it was place in. External links were considered as any hyperlink that pointed to a domain outside the page it originated from. Finally, any hyperlink that pointed to a place within the same page it originated from, besides being labelled as an internal link, was considered a bookmark.

The average internal links per page was 188, while the average external links per page was a much smaller number of 23. When compared to the general average number of hyperlinks per page, 211 as mentioned above, this means that most pages contain a majority of same domain links versus a small number external ones. This seems in line with what can be gathered through the average user experience, with most pages linking thoroughly to their own content and finishing with a handful of referrals to outside areas. A high percentage of external links seems reserved to content aggregators or other similar web pages. Bookmarks are seemingly uncommon, with a mere average of 6 occurrences per page. Finally, the highest numbers encountered on a single page were 12138, 7442 and 2872 for internal, external and bookmark hyperlinks, respectively. All of them had pages without a single occurrence.

Table 5.2: Protocols used in hyperlinks.

Protocol	**Quantity**	**Percentage**
http	5751284	66.70%
https	2701250	31.32%
javascript	149253	1.73%
mailto	8823	0.10%
magnet	6187	0.07%
tel	4115	0.05%
other	2053	0.03%

In regards to the protocols used within hyperlinks themselves, they are mostly split between Hypertext Transfer Protocol (HTTP) and HTTP Secure (HTTPS), with 66.70% and 31.32%, respectively. Table 5.2 shows these results in more detail. This is obviously expected, as they are the protocols on which the Web most relies on. Though not really a protocol, JavaScript appears at a distant 3rd place, with 1.73%, but it is a testament to the frequency with which developers rely on this technology to build their web pages. Curiously, JavaScript seems to deserve the award for most misspelled hyperlink destination. Apart from the correct spelling, an, impressive 28 different erroneously typed attempts were found. Half of them were present in hyperlinks multiple times, totalling 209, or 0.14% of the total. One might wonder, how many different ways of spelling JavaScript there are. Although the most common protocols, HTTP and HTTPS, also sported a few miss typed variants, they were nowhere near the amount encountered with JavaScript, especially considering their staggering percentage of use. This may be indicative of the way developers construct their creations, copying and pasting their content when their hyperlinks point to other pages and typing them out instead, when they lead to JavaScript content. Thus leading to the occurrence of more mistakes. This also serves as further proof that developers don't properly test their web pages, as a single click would make their errors apparent. Perhaps further hyperlink study is required, including a dead link analysis.

The remainder of protocols in use can be seen as a popularity snapshot of current times, email links are still very common, bit torrent magnet links also make their appearance. New or relatively recent forms of communication such as WhatsApp [64], Tencent QQ Messenger [65] or even Skype [66] are among the top 10 most used protocols, although representing a very minor percentage. While older messaging means, such as Short Message Service (SMS) or Internet Relay Chat (IRC), still make their presence, they have certainly been left behind by most users.

Images

Images are a common element for most web pages. Used to quickly convey information, employed as decoration or, in older HTML standards, even as a means of aligning other elements in your page. It comes as no surprise that more than 2 million images were found in our selection of 43472 web pages. The average number of images per page was slightly above 47. On both ends of the spectrum, several pages revealed themselves completely void of pictures, while *https://www.buy123.com.tw/*, managed to boast an impressive number of 3829.

Table 5.3: Types of images in use.

Image	**Quantity**	**Percentage**
.jpg	889333	42.97%
.png	550475	26.60%
.gif	301470	14.57%
empty	134834	6.52%
hyperlink	134369	6.49%
.svg	37098	1.79%
.jpeg	20452	0.99%
other	1415	0.07%

As shown in table 5.3, the results of our study of image formats used in web pages are somewhat expected. The podium is occupied by three well known names. Although it sports multiple versions of its file extension, Joint Photographic Experts Group (JPEG) shows up in first place with its Jpg variant, with almost 43%, followed by Portable Network Graphics (PNG) with 26.6% and Graphics Interchange Format (GIF) follows suite with 14.57%.

Their use can be explained through their image properties. JPEG is a lossy format, which produces images that trade their quality for their smaller file size, thus leading to shorter web page load times. A feature, apparently, highly sought out by developers. PNG, on the other hand, is a loss-less format, which means the opposite, no reduction in image quality and a, usually, larger file size. So developers might use them more sparingly, saving them exclusively for the areas on their pages where that they want to stand out, leading to less overall usage. Finally, GIF images are known for their ability to be used as means to provide short animated content.

A somewhat large percentage, 6.52%, of <*img*> HTML tags were surprisingly empty, for no apparent reason. A similar proportion, 6.49%, pointed to web locations with no discernible image types from their hyperlinks. Apart from a brief mention of Scalable

Vector Graphics (SVG), for their vector style pictures, the remainder of image types are underused when compared to the ones previously mentioned.

5.1.3 Validation

When it comes to HTML validation, the results are not optimistic. There is an average of 81 errors and 26 warnings for each web page. The highest number of errors found on a single page was 5961, while the biggest amount of warnings amounted to 3981. Of all the web pages analysed, 29.34% do not trigger any kind of validation warning, 6.05% are error free and a mere 2268, 5.18% successfully go through validation without revealing any errors or warnings.

Table 5.4 displays all validation errors with an occurrence above 1%. The column labelled HTML 5 has an asterisk (*) if the error in question is specifically related to HTML 5.

Table 5.4: Most common validation errors.

Error	**Percentage**	**HTML5**
Attribute X not allowed on element Y at this point.	10.44%	*
Reference to entity X for which no system identifier could be generated.	10.33%	
Required attribute X not specified.	7.62%	
An img element must have an alt attribute, except under certain conditions.	6.92%	*
End tag for X omitted, but omittag no was specified.	6.58%	
Document type does not allow element X here; Missing one of Y start-tag.	5..91%	
Element X not allowed as child of element Y in this context.	4.37%	*
Element X undefined.	3.95%	
The X attribute on the Y element is obsolete. Use CSS instead.	3.36%	*
There is no attribute X.	3.14%	
Document type does not allow element X here.	2.76%	
Duplicate id X.	2.68%	*
Element X is missing required attribute Y.	2.34%	*
An attribute value specification must be an attribute value literal unless SHORTTAG YES is specified.	2.05%	
End tag for element X which is not open.	1.81%	
General entity X not defined and no default entity.	1.64%	
X is not a member of a group specified for any attribute.	1.53%	
End tag X.	1.49%	*
Id X already defined.	1.42%	
Bad value X for attribute Y on element Z.	1.39%	*
An attribute value must be a literal unless it contains only name characters.	1.14%	

The most commonly found error, with 10.44%, is triggered when developers attempt to use an attribute that is not defined for the accompanying element. Moving on to the second most common mistake, with 10.33%, the error *Reference to entity X for which no system identifier could be generated.* is returned by the validator whenever unencoded special characters or undefined entities are used [67]. To finish the top 3, with 7.62%, *Required attribute X is not specified* is self explanatory, a required attribute for the element in question was omitted.

Table 5.5 contains the most common HTML validation warnings found with percentage higher than 1%. Again, the column labelled as HTML 5 has an asterisk (*) if the warning is specific for HTML 5.

Table 5.5: Most common validation warnings.

Warning	**Percentage**	**HTML 5**
Reference not terminated by REFC delimiter	17.95%	
NET-enabling start-tag requires SHORTTAG YES	17.16%	
Reference to external entity in attribute value	14.28%	
Character X is the first character of a delimiter but occurred as data.	13.88%	
The first occurrence of id X was here.	8.54%	*
Cannot generate system identifier for general entity X.	5.23%	
The border attribute is obsolete. Consider specifying img { border: 0; } in CSS instead.	3.37%	*
Section lacks heading. Consider using h2-h6 elements to add identifying headings to all sections.	3.34%	*
Attribute X is not serializable as XML 1.0.	2.41%	*
Article lacks heading. Consider using h2-h6 elements to add identifying headings to all articles.	1.92%	*
The name attribute is obsolete. Consider putting an id attribute on the nearest container instead.	1.84%	*
Consider using the h1 element as a top-level heading only.	1.49%	*
Multiple comments in comment declaration.	1.23%	
The X role is unnecessary for element Y.	1.16%	*

With 17.95%, the most frequent validation warning is mostly caused by the lack of a semi-colon, in order to terminate an open special character. *NET-enabling start-tag requires SHORTTAG YES*, with 17.16%, is caused by the use of self closing tags in HTML documents that don't expect them. For the third most common warning, an non terminated special character placed inside an attribute is the most likely cause. Of note, the top four most frequent warnings are not HTML 5 specific.

5.1.4 Sizes

File sizes are important for web pages that want to be delivered quickly to its user. In spite of other types of content being mostly responsible for the total download size, such as video, image, sound or other types of media, HTML files can also help to increase it. The average file size found on our data set was revealed to be 101.176 Kb. The biggest one was found to be 5287 Kb, over 5 Mb. The smallest file had 9 bytes, most likely the product of a problematic redirection.

Their HTML elements accounted for the bulk of the file size, with an average of 92.715 Kb per page. This means that the time when pages conveyed information mostly through the use of text may be somewhat in the past. The page with the heaviest set of HTML elements managed to reach 5.156 Mb. The smallest amount used by its elements was 9 bytes, which points back to the previously mentioned page.

Text content was, as noted, much lighter when compared to the amount spent by HTML elements, an average of 8.460 Kb per page. Some pages revealed no text content at all, while others managed to pack almost 2.5 Mb of text information.

Looking at comments is also an interesting proposition, as it allows one to peer into the habits of developers. The average size taken by HTML comments was 1.81 Kb. While some may see the largest amount of space taken by comments in a single page, over 2.5 Mb, as excessive, the number of pages with no comments at all revealed itself quite large, 7496, over 17% of the total. Documentation, as in programming, reveals itself a tricky subject for web developers.

Table 5.6: Ratios between components and file size.

	Elements to Size	**Text to Size**	**Comments to Size**
Average	90%	10%	2%
Maximum	100%	98%	95%
Minimum	2%	0%	0%

We now point to Table 5.6 for a set of ratios between the previous values, they also reveal interesting information. While all other components can be non existent in an HTML file, obviously, its HTML elements are never completely absent, or you would not have a web page. This analysis also reveals that one page had text content worthy of 98% of its total file size. The same feat was also matched, this time by a page with 95% of its content reserved for HTML comments.

5.1.5 Servers used

We were unable to detect the type of server most of our web pages were hosted with, 70.33%. Of the remainder, Figure 5.4 presents a detailed account of the servers found. Clearly, nginx is the most popular option, with 46.62%. Its adoption rate is almost as large as every other server combined. In terms of popularity, different versions of Apache and Microsoft-IIS close the top 3. All other entries were residual at best.

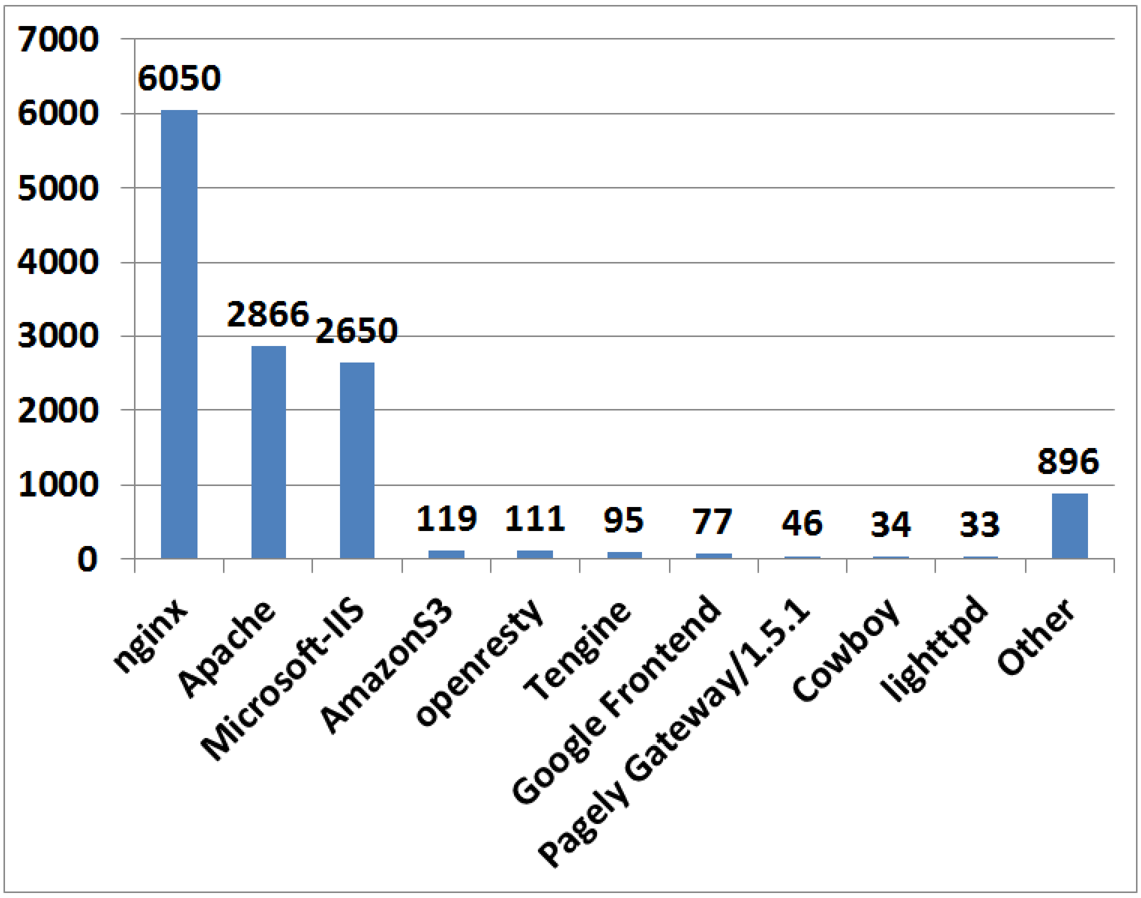

Figure 5.4: Different server types in use.

5.1.6 Average web page

Finally, based on the information gathered from the previous sections, one might construct the average page from our data set. It is a page made with the use of HTML 5, it has close to 950 HTML elements, of which <*a*> is the most common one. It sports more than 200 hyperlinks, of which the majority point towards its own domain and to content delivered through HTTP. It is decorated with almost 50 images, close to half of them belonging to the JPEG format. Its HTML code contains over 30 different comments and information on its server is unannounced. Its HTML validation report comes back with 83 different errors and 26 separate warnings. Finally, its size spans just over 100 Kb, most of it due to its HTML elements, while its text content represents only 10% of its total. HTML comments take their slice of the file size, but dont go over 2.30% the total amount.

Table 5.7: The average web page.

Characteristic	Value
HTML version used	HTML 5 (68.83%)
Number of HTML elements	944
Most frequent HTML element	<a>(22.37%)
Number of hyperlinks	211
Number of internal hyperlinks	188
Number of bookmarks	6
Number of external hyperlinks	23
Most common protocol used in hyperlinks	http (62.12%)
Number of images	47
Most common image type used	.jpg (42.97%)
Number of HTML comments	32
Server	Not declared (70.33%)
Number of HTML validation errors	83
Number of HTML validation warnings	26
HTML file size	101.176 Kb
Size of text content	8.460 Kb
Size of HTML elements	92.715 Kb
Size of HTML comments	1.810 Kb
Text to HTML elements ratio	13.52%
HTML elements to file size ratio	89.61%
Text to file size ratio	10.39%
HTML comments to file size ratio	2.30%

5.2 HTML 5 findings

In this section, we repeat the previous analysis with information specific to web pages made with HTML 5. From the final 43742 web pages that got processed, our tool managed to find 30109 that were constructed with HTML 5.

5.2.1 HTML elements

While glancing at Table 5.8, one might think it is identical to the one in the previous section, but it is not. The previous number one most common element, the <a>tag, has traded places with the former number two, <div>. While all the HTML elements that are found with a percentage bigger than 1% are exactly the same as in the list presented in the general findings, the order in the second half of the table is completely different. Since a significant part of all the web pages analysed is made with HTML 5, it is normal for them to present similarities.

Regarding the number of HTML comments, as in our general findings, several pages did not contain a single one. The highest number of comments in a single page reached 6692, while the average was 35.

Table 5.8: Common HTML elements in HTML 5 web pages.

Element	Quantity	Percentage
<div>	6945490	23.23%
<a>	6513336	21.79%
<li>	3541961	11.85%
<span>	3023289	10.11%
<img>	1417142	4.74%
<p>	815604	2.73%
<script>	639166	2.14%
<i>	596541	2.00%
<ul>	589797	1.97%
<option>	486285	1.63%
<meta>	454553	1.52%
<link>	380853	1.27%
<td>	366833	1.23%
<h3>	336303	1.12%
<br>	326813	1.09%

Hyperlinks

On the HTML 5 side of hyperlinks, the average number per page as gone up slightly, from 211 to 216. Again, several pages show up with no links whatsoever, as well as the maximum number of hyperlinks in a single page remains 15322.

Table 5.9: Protocols used in hyperlinks in HTML 5 web pages.

Protocol	Quantity	Percentage
http	3792320	61.49%
https	2258977	36.63%
javascript	103145	1.67%
mailto	6668	0.11%
tel	3647	0.06%
other	2182	0.04%

Table 5.9 displays, once again, the list of most common protocols used in the hyperlinks found on our data set. When comparing to the general HTML results, they prove to be similar. HTTP and HTTPS continue to dominate hyperlink destinations, although HTTPS manages to steal 5% from its non secure version. JavaScript continues to be heavily miss typed, with 19 different attempts for a total of 180 occurrences. Magnet links dropped off the radar with a smaller amount of 546 appearances.

Images

The average number of images, present in HTML 5 web pages only, remained the same as in the general findings, 47. The minimum amount of images found on a page was 0, while the maximum remained 3829. The total number of images gathered from this particular set was 1417142.

Table 5.10: Types of images in use for HTML 5 web pages.

Image	**Quantity**	**Percentage**
.jpg	627580	44.29%
.png	388571	27.42%
.gif	137554	9.71%
empty	104283	7.64%
hyperlink	104538	7.38%
.svg	33142	2.34%
.jpeg	16245	1.15%
other	1129	0.07%

Apart from a small fluctuation of percentages, the order of the list shown in Table 5.10 remains the same as the one shown in Table 5.3. It is easy to understand why, the reasons previously detailed for the use of the top 3 most common image formats remain valid independently of the type of HTML in use.

5.2.2 Validation

HTML validation results for HTML 5 web pages are a mixed result. On the one hand, the average number of errors and warnings is lower, 54 and 11, respectively. The highest number of errors on a single page is almost one sixth of the result obtained in the general findings, 1001. The highest number of warnings on a single page was also significantly lower, from 3981 to 878. The total number of pages that do not have a single validation warning is also slightly higher, up to 30.22% from 29.34% previously. On the other hand, in regards to errors, the amount of pages that do not sport a single error dropped down to 4.29% and, most importantly, pages which successfully go through validation without any error or warning message reached a low of 3.06%. The result obtained on the general findings being 5.18%.

A listing of all the error types that occurred more than in 1% of cases is shown on Table 5.11. The most frequent validation error found on the previous data set is also the most common for HTML 5 web pages. *Attribute X not allowed on element Y at this point*, with 22.63%, caused by using an attribute for an element that does not have such a definition. On the second spot of most frequent errors, *An IMG element must have an alt attribute, except under certain conditions* is pretty clear, developers chose not to include a required attribute for an <img>tag, 15.05%. Finally, with 9.53%, the *Element X not allowed as child of element Y in this context* error is triggered when an erroneous nesting of elements is placed withing the HTML code.

Table 5.11: Most common validation errors for HTML 5 web pages.

Error	**Percentage**
Attribute X not allowed on element Y at this point.	22.63%
An img element must have an alt attribute, except under certain conditions.	15.05%
Element X not allowed as child of element Y in this context.	9.53%
The X attribute on the Y element is obsolete. Use CSS instead.	7.00%
Duplicate id X.	5.87%
Element X is missing required attribute Y.	5.13%
End tag X.	3.22%
Bad value X for attribute Y on element Z.	3.04%
Bad value X for attribute Y on element Z: expected a digit but saw A instead.	2.05%
The X attribute was specified, but the element is not a property of any item.	1.89%
No space between attributes.	1.69%
Bad value X for attribute Y on element Z: The string A is not a registered keyword.	1.54%
The X element is obsolete. Use CSS instead.	1.54%
Bad value X for attribute Y on element Z: Browsing context name must be at least one character long.	1.27%
Duplicate attribute X.	1.22%
Bad value X for attribute Y on element Z: Illegal character in query: A is not allowed.	1.14%
Unclosed element X.	1.05%

Table 5.12 contains the list of the most frequent HTML 5 validation warnings, with a percentage higher than 1%. The most relevant warning, *The first occurrence of id X was here* with 28.68%, relates to the usage of the same id on multiple locations. The warning will point out the place where said id was first used. The cause of the second highest problem, sporting 11.25%, is simple, there are no heading elements in use in at least one page section. Finally, with 11.21%, another warning with an obvious cause, the obsolete border attribute was used within an element.

Table 5.12: Most common validation warnings for HTML 5 web pages.

Warning	Percentage
The first occurrence of id X was here.	28.68%
Section lacks heading. Consider using h2-h6 elements to add identifying headings to all sections.	11.25%
The border attribute is obsolete. Consider specifying img { border: 0; } in CSS instead.	11.21%
Attribute X is not serializable as XML 1.0.	8.08%
Article lacks heading. Consider using h2-h6 elements to add identifying headings to all articles.	6.46%
The name attribute is obsolete. Consider putting an id attribute on the nearest container instead.	5.88%
Consider using the h1 element as a top-level heading only.	4.95%
The X role is unnecessary for element Y.	3.90%
The X attribute on the Y element is obsolete. You can safely omit it.	2.55%
Empty heading.	2.47%
Element X does not need a Y attribute.	2.36%
The document is not mappable to XML 1.0 due to two consecutive hyphens in a comment.	2.30%
Attribute with the local name X is not serializable as XML 1.0.	2.14%
Element X cannot be represented as XML 1.0.	1.44%
Text run is not in Unicode Normalization Form C.	1.38%
The value of attribute X on element Y from namespace Z is not in Unicode Normalization Form C.	1.14%

5.2.3 Sizes

The average file size increased slightly when compared to the general findings, from 101.176 Kb to 112.666 Kb. The previously identified web page with the biggest file size, was an HTML 5 one, as the same value is found here, 5287 Kb. The smallest file however, had was comprised of only 94 bytes.

The size of HTML elements averaged at 104.084 Kb, also an increase when compared to the previous value. The largest amount of space taken by a pages' elements was indeed the same one identified on the previous set of pages, 5.156 Mb, while the smallest only required 64 bytes for its elements.

Text content remained, on average, pretty much the same, 8.583 Kb when compared to 8.460 Kb. Regarding the highs and lows of text content, it was identical to the previous analysis. Some pages had no text content whatsoever, while the highest one went almost to 2.5 Mb.

The average size of HTML comments was reduced by a small amount, down to 1.775 Kb, from 1.81 Kb. The largest file size reserved for comments was reduced to 386 Kb, but the number of web pages with no comments shrank to 12.46%. A good improvement when compared to the previous 17%.

Looking at the ratios present in Table 5.13, we notice that the trend of the previous analysis continues. Only a few decimal points have changed between the general findings

Table 5.13: Ratios between components and file size for HTML 5 web pages.

	Elements to Size	Text to Size	Comments to Size
Average	90.93%	9.07%	2.14%
Maximum	100%	98.24%	95.02%
Minimum	1.76%	0%	0%

and the HTML 5 specific values. HTML elements continue to take roughly 90% of the total web page file size.

5.2.4 Servers used

Uncovering the server type with which most of the HTML 5 web pages were delivered remained a difficult task, 72.04% of them did not share this information. As we can see on Figure 5.5, the list remains mostly unchanged, with a single difference, Google Frontend is now ahead of Tengine. Nginx also increased its lead by a slight amount, from 46.62% to 53.02%.

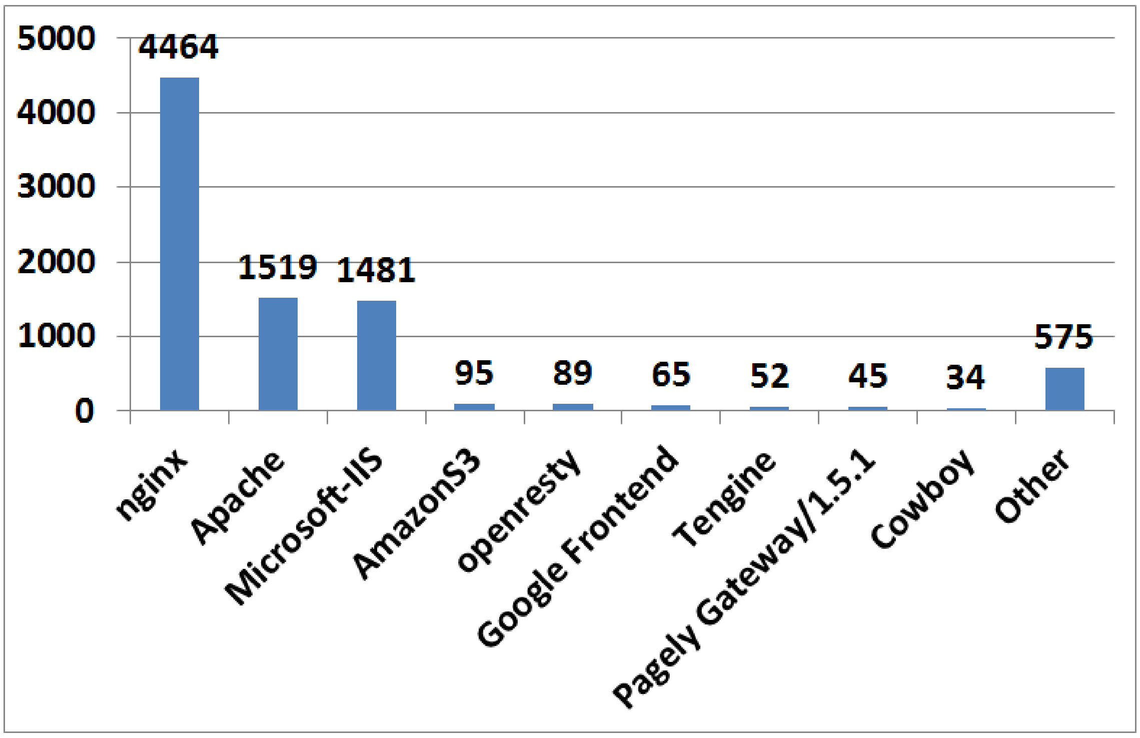

Figure 5.5: Different server types in use with HTML 5 web pages.

5.2.5 Average web page

Gathering all of the previous information, we can once again construct the average web page, this time specifically for one based on HTML 5. It is one composed of close to 1000 HTML elements, of which the <div>is the most common one. 195 of its 216 hyperlinks point towards its domain, while only 20 aim outwards. More than half of them make use of the HTTP protocol. Among its content, 47 images with JPEG format are present. Its server continues to be unannounced. There are 54 HTML validation errors, while validation warnings are few, only 11. The file size for our web page is 112 Kb, of which 90.93% is taken by its HTML elements, text content represents a much smaller slice, 9.07%.

Table 5.14: The average HTML 5 web page.

Characteristic	**Value**
Number of HTML elements	992
Most frequent HTML element	<div>(29.89%)
Number of hyperlinks	216
Number of internal hyperlinks	195
Number of bookmarks	6
Number of external hyperlinks	20
Most common protocol used in hyperlinks	http (58.15%)
Number of images	47
Most common image type used	.jpg (44.28%)
Number of HTML comments	35
Server	Not declared (72.04%)
Number of HTML validation errors	54
Number of HTML validation warnings	11
HTML file size	112.666 Kb
Size of text content	8.583 Kb
Size of HTML elements	104.083 Kb
Size of HTML comments	1.775 Kb
Text to HTML elements ratio	11.35%
HTML elements to file size ratio	90.93%
Text to file size ratio	9.07%
HTML comments to file size ratio	2.14%

6. Conclusion

While there have been previous ventures that performed similar studies, most of them have been done in a somewhat distant time. The rate with which the Internet and the World Wide Web continue to advance and evolve is staggering. Web developers, as well as researchers are thus continually in need of up to date information, that is not always available. Creating a tool that can provide this information is certainly a worthwhile endeavour.

Not surprisingly, much like the variety of content found on the Web itself, so do the problems that might occur in such an enterprise seem infinitely diverse. Our initial goals spanned more than simply creating a tool capable of analysing web pages, their Hypertext Markup Language (HTML) and producing a set of statistics. But this one has been mostly accomplished. Web ARChive (WARC) files are read and used as a source of information, HTML files parsed and their elements analysed. Integration with the external World Wide Web Consortium (W3C) HTML validator has been accomplished and requests are successfully made, answers received and processed. Data is handled within an appropriate time frame, it has some degree of scalability and the resume function provides partial recoverability.

Much more could be done and, despite our best efforts, our tool is not perfect. It has some faults and can certainly be improved in other areas as well. It is, in our opinion, an adequate beginning for a, hopefully, more fully fledged tool that can be used to perform the large scale study that we originally set to perform. The results we present in this report are somewhat superficial, as they mostly represent the average web page. They are also meagre when compared to the initial plan of analysing several million web pages and performing an in-depth study. In spite of our setbacks and limitations and, although simpler when compared to what we originally envisioned, our tool and analysis still produced a considerable amount of information regarding HTML found online. Perhaps a mere stepping stone on the way to a much larger study.

Finally, the source code for our tool is available online at the following location:

– https://github.com/Joca64/HTMLTool/

7. Work plan

This chapter details the original work plans initially set for the project and compares them to what was actually executed, as well as discussing the reasons behind changes and delays.

7.1 First semester

The initial work plan devised for the first semester is presented in Figure 7.1. It consisted of a two month period of reviewing the state of the art regarding Hypertext Markup Language (HTML), web crawlers and HTML validators and APIs. This would be followed by a month long requirements survey for the study approach, which would coincide with the start of a two month task of designing the preliminary system architecture. During the fourth month a small proof-of-concept prototype would also be implemented. Finally, the last month of the semester would be dedicated to the writing of the intermediate report.

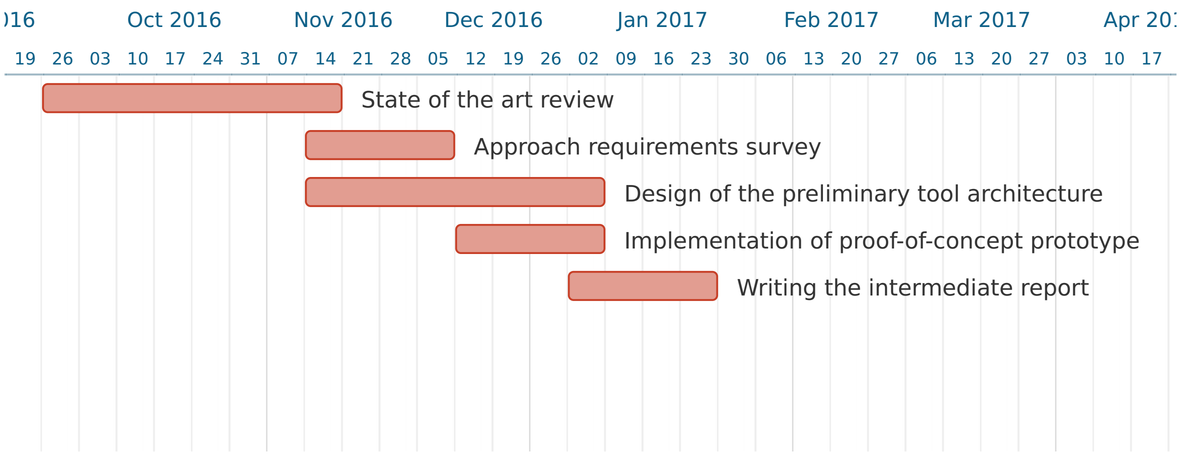

Figure 7.1: Work plan for the first semester.

The state of the art review went according to plan but development was different from what was initially expected. It was decided early on that, due to the nature of the project being a research focused one, there would be no need for a formal definition of the project requirements. Both the requirements survey for the study approach and the preliminary tool architecture design were also mostly done during the state of the art review. Several tasks were added, such as the installation and configuration of the selected crawler, followed by a set of test crawls for roughly 1500 Uniform Resource Locator

(URL)s each, the choice of URL sources, as well as the installation and configuration of local versions of both World Wide Web Consortium (W3C) HTML validators. The writing of the intermediate report was also spread along the whole semester instead of reserving the final month for that task. Weekly meetings with the advisor, Prof. Nuno Laranjeiro, are also not accounted for in Image 7.2.

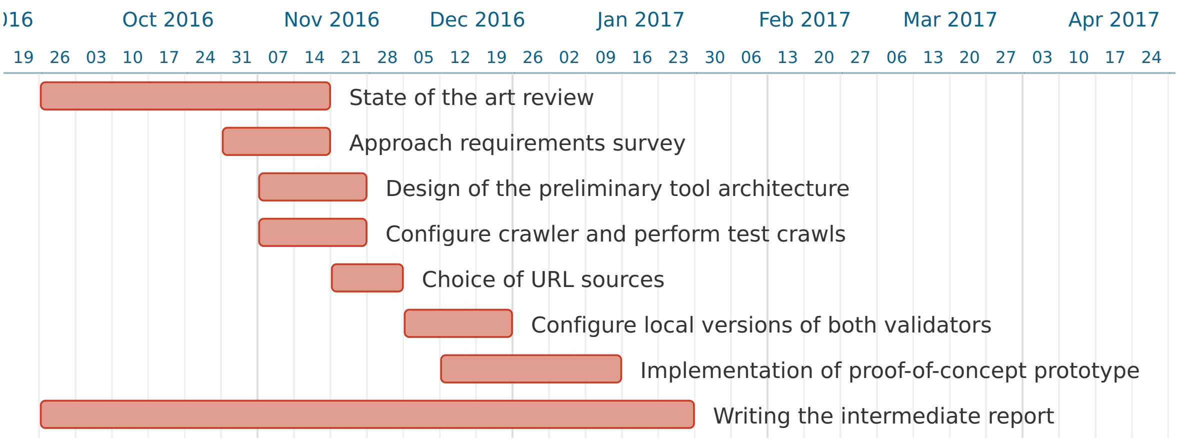

Figure 7.2: Actual work performed on the first semester.

7.2 Second semester

The work planned for the second half of the project is shown on Figure 7.3. The second semester was intended to begin with the creation of the intermediate presentation, alongside with the writing of a short paper, detailing our intention of building a tool for analysing HTML in depth. Deciding upon the characteristics and attributes that our tool should extract was the following task. Finishing our tool's implementation was slated to begin mid February and extend until the end of March. Halfway through this last task, the crawling of the selected web pages should begin so that they could be readily processed once the tool implementation was completed. The month of April would be reserved for the writing of a paper detailing our findings specifically regarding pages built with HTML 5, while the next month would see a similar task, writing a paper detailing our general findings. Writing the final report would be undertaken during the whole duration of the project. Finally, the semester would conclude with the creation of the final presentation.

In contrast, the work actually performed during the second semester is shown in Figure 7.4. The first three tasks, creating the intermediate presentation, writing a short paper and finalising the set of metrics and indicators to be retrieved by the tool went according to plan. Despite the initial time frame proposal for completing the tool implementation being somewhat short, with a moderate amount of delay expected, it extended well beyond what was planned. As discussed in chapter 4, section 4.4, the biggest setback was caused by the move from the development virtual machine to the production equivalent. This change caused the HTML validation of all HTML 5 web pages to fail.

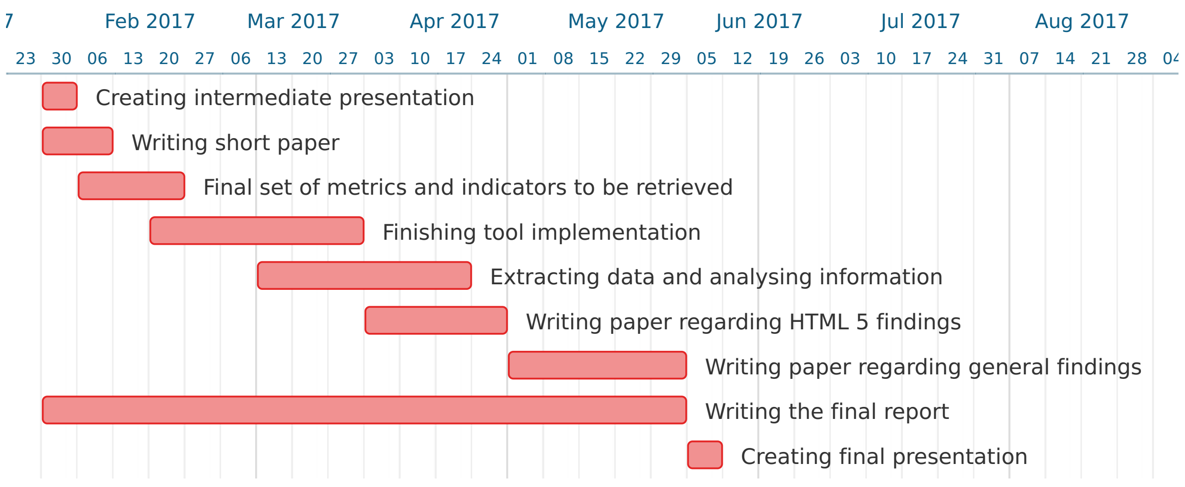

Figure 7.3: Work plan for the second semester.

Subsequently, as previously detailed, several attempts were made in order to fix the problem, from code changes, to program re-installations, configurations checked and new ones attempted, versions of all software confirmed, to using completely different virtualization environments. Finally, the solution employed, making a clone of the development environment and importing it to the production machine, managed to go around the problem and making our setup functional. It is however, somewhat unsatisfactory as it does not address the underlying problem, but time constraints forced us to proceed with this option.

Once the tool was up and running and it began processing actual batches of crawled web pages, another problem surfaced. An exception was being raised and preventing the tool from generating the totals for the information gathered. An apparent rogue character was interfering with our parsing and splitting strings in an unintended way, producing erroneous results. This behaviour was also never detected in the development environment, despite the tool being tested with sets of thousands of web pages, it was obviously not subjected to the same degree of variability as a group of one million pages. Attempts at resolving this problem were made, the set of web pages being tested was reduced to 250,000 in order to accelerate the debugging process, but still lead to a somewhat lengthy wait time between attempts. The pages were made sure to be read in the appropriate character encoding so as to not lead to charset problems while parsing them. The strings causing the problem were filtered and any characters that might incorrectly lead to strings being split, such as line breaks or rogue string terminators were removed. Ultimately, despite our best efforts, the problem still persists.

Due to unfortunate, personal and external circumstances, development for the project was halted during roughly two and a half months, the dates marked on Figure 7.4 in blue. This made it impossible to successfully complete both tasks regarding the writing of papers detailing our findings. Despite the crawling of web pages having been performed in the initially planned time frame, due to these delays, said information was deemed obsolete and a new, much smaller, set of web pages was retrieved. Due to being more relevant in emulating the average user experience, the URLs used, were limited to Alexa.com. The remainder of the time left was spent analysing said information, writing the final version of this report and creating the final presentation.

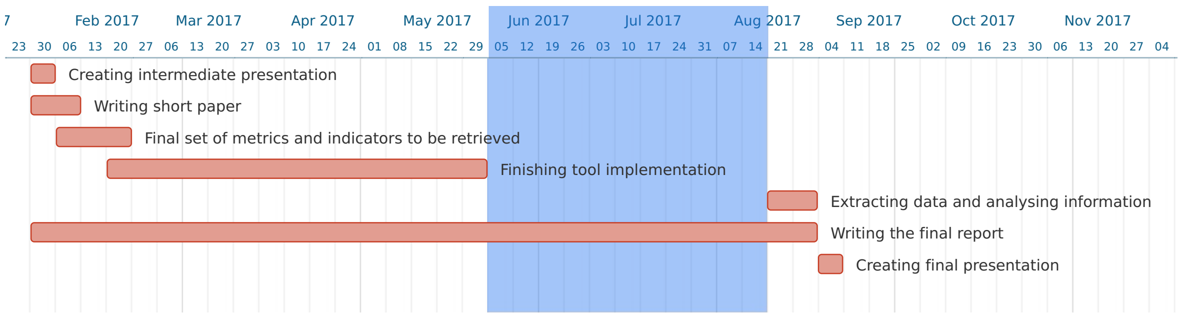

Figure 7.4: Actual work performed on the second semester.

8. Future work

Naturally, every project's conclusion brings with it a few ideas that seem worthy of expanding upon in the future, ways for it to be improved as well as, in hindsight, different ways of accomplishing the same goals. This is particularly apparent in projects that did not meet all of its originally intended goals.

A few options seem obvious to develop upon in future work. Naturally, the intended large-scale study focusing on the quality and state of Hypertext Markup Language (HTML) currently available online is the most evident. This is backed by the lack of recent data regarding this subject, particularly concerning HTML5, and can surely be of use by researchers and developers alike. The subject of HTML complexity may also require a more extended look into it. Making use of additional techniques in order to closer resemble the average online user experience seems a possibility. As shown by previous studies [68], following the directions contained in the Robots Exclusion Standard can askew one's results. This happens because many web pages favour only the most popular search engines, white-listing Google, while blocking access to any other crawlers. Evidently, this leads to an ethical problem, should one ignore the Robots Exclusion Standard in an attempt to obtain the best data possible while disregarding the wishes of the web page creators? Or adhere to it and try to work around data that does not accurately represent the subject being studied. Further thought is needed regarding this matter. This, however, is just one example of the way the gathering of data can lead to a biased set of results.

In regards to improvements for the tool itself, as may be expected, a fix for the occasional problem that prevents generating the files containing the totals, for attributes and characteristics gathered, should be the top priority. Beyond this, cleaning up the output for certain attributes would improve ease of use by the tool's users. Assuredly, the amount of information gathered by the program can also be increased. Integrating a Data Base Management System (DBMS) would most likely enhance the process of analysing all the information collected, again, improving its ease of use. Finally, an increase in performance can certainly be achieved if further optimisation is attempted.

References

[1] Ejike Ofuonye et al. “Prevalence and classification of web page defects”. In: *Online Information Review* 34.1 (2010), pp. 160–174.

[2] Shan Chen, Dan Hong, and Vincent Y Shen. “An experimental study on validation problems with existing html webpages”. In: *Proceedings of the 2005 International Conference on Internet Computing, ICOMP’05*. 2005, p. 373.

[3] Tim J Berners-Lee. “The world-wide web”. In: *Computer networks and ISDN systems* 25.4 (1992), pp. 454–459.

[4] World Wide Web Consortium. *The web standards model - HTML CSS and JavaScript*. URL: `https://www.w3.org/wiki/The_web_standards_model_-_HTML_CSS_and_JavaScript`.

[5] World Wide Web Consortium. *HTML doctype declaration*. URL: `http://www.w3schools.com/tags/tag_doctype.asp`.

[6] World Wide Web Consortium. *HTML & CSS*. URL: `https://www.w3.org/standards/webdesign/htmlcss`.

[7] International Organization for Standardization. *Standard Generalized Markup Language (SGML)*. URL: `https://www.iso.org/obp/ui/#iso:std:iso:8879:ed-1:v1:en`.

[8] W3C. *Web Content Accessibility Guidelines (WCAG) 2.0*. URL: `https://www.w3.org/TR/WCAG20/`.

[9] Alexa.com. *Alexa Top 500 Global Sites*. URL: `http://www.alexa.com/topsites`.

[10] Fabrizio Giustina. *JTidy*. URL: `http://jtidy.sourceforge.net/`.

[11] World Wide Web Consortium. *The W3C Markup Validation Service*. URL: `https://validator.w3.org/`.

[12] The World Bank. *GDP per capita, PPP (current international $)*. URL: `http://data.worldbank.org/indicator/NY.GDP.PCAP.PP.CD`.

[13] Economist Intelligence Unit. “E-readiness rankings 2009: The usage imperative”. In: *The Economist. A report from the Economist Intelligence Unit written in cooperation with the IBM Institute for Business Value* (2009).

[14] International Monetary Fund. *World Economic Outlook database for 2005*. URL: `http://www.imf.org/external/pubs/ft/weo/2005/01/data/`.

[15] Economist Intelligence Unit. *The 2005 e-readiness rankings*. URL: `http://graphics.eiu.com/files/ad_pdfs/2005Ereadiness_Ranking_WP.pdf`.

[16] Bryan Wilson. *MAMA - Metadata Analysis and Mining Application.* URL: http://maqentaer.github.io/devopera-static-backup/http/dev.opera.com/articles/view/mama/index.html.

[17] DMOZ. *The Directory of the Web.* URL: http://www.dmoz.org/.

[18] Dagfinn Parnas. "How to cope with incorrect HTML". In: (2001).

[19] Liam Quinn. *WDG HTML Validator.* URL: http://www.htmlhelp.com/tools/validator/.

[20] Rene Saarsoo. *Coding practices of web pages.* URL: http://triin.net/2006/06/12/Coding_practices_of_web_pages.

[21] W3C. *The W3C CSS Validation Service.* URL: https://jigsaw.w3.org/css-validator/.

[22] Tomislav Jakopec, Anita Papić, and Josipa Selthofer. "Inside Croatian national top-level domain: Analysis of technical quality according to W3C standards". In: *Information Technology Interfaces (ITI), Proceedings of the ITI 2011 33rd International Conference on.* IEEE. 2011, pp. 471–476.

[23] Allison Woodruff et al. "An investigation of documents from the World Wide Web". In: *Computer Networks and ISDN Systems* 28.7 (1996), pp. 963–980.

[24] N. Bowers. *Weblint Home Page (version 1.013).* URL: http://www.khoral.com/staff/neilb/weblint.html.

[25] Patricia Beatty, Scott Dick, and James Miller. "Is HTML in a race to the bottom? A large-scale survey and analysis of conformance to W3C standards". In: *IEEE Internet Computing* 12.2 (2008), pp. 76–80.

[26] Andreas Pinterits, Horst Treiblmaier, and Irene Pollach. "Environmental websites: an empirical investigation of functionality and accessibility". In: *International Journal of Technology, Policy and Management* 6.1 (2006), pp. 103–119.

[27] Audit IT. *Web site evaluation.* URL: http://www.audit-it.com/web_site_evaluation.php.

[28] Watchfire. *Watchfire WebXACT.* URL: http://www.w3c.hu/talks/2006/wai_de/mate/watchfire.html.

[29] Heitor de Souza Ganzeli, Graça Bressan, and Antônio Marcos Moreiras. "ICT web: analysis of the Brazilian governmental web". In: *Proceedings of the 18th Brazilian symposium on Multimedia and the web.* ACM. 2012, pp. 383–386.

[30] Governo Federal. *ASES Avaliador e Simulador de Acessibilidade em Sítios.* URL: http://asesweb.governoeletronico.gov.br/ases/.

[31] Dave J Beckett. "30% accessible – a survey of the UK Wide Web". In: *Computer Networks and ISDN Systems* 29.8 (1997), pp. 1367–1375.

[32] Kelly Waters. "Prioritization using moscow". In: *Agile Planning* 12 (2009), p. 31.

[33] SimilarWeb. *SimilarWeb.* URL: http://www.similarweb.com/.

[34] Quantcast. *Quantcast.* URL: http://www.quantcast.com/.

[35] Majestic. *Majestic.* URL: https://majestic.com/.

[36] Common Crawl. *Common Crawl.* URL: http://http://commoncrawl.org/.

[37] Martijn Koster. *Robots in the Web: threat or treat?* URL: http://www.robotstxt.org/threat-or-treat.html.

[38] Gautam Pant, Padmini Srinivasan, and Filippo Menczer. "Crawling the web". In: *Web Dynamics*. Springer, 2004, pp. 153–177.

[39] Vladislav Shkapenyuk and Torsten Suel. "Design and implementation of a high-performance distributed web crawler". In: *Data Engineering, 2002. Proceedings. 18th International Conference on.* IEEE. 2002, pp. 357–368.

[40] Ziv Bar-YossefÝ et al. "Approximating aggregate queries about web pages via random walks". In: (2000).

[41] Martijn Koster. *A Standard for Robot Exclusion.* URL: http://www.robotstxt.org/orig.html.

[42] Yang Sun, Ziming Zhuang, and C Lee Giles. "A large-scale study of robots.txt". In: *Proceedings of the 16th international conference on World Wide Web.* ACM. 2007, pp. 1123–1124.

[43] Monika Yadav and Neha Goyal. *Comparison of Open Source Crawlers - A Review.* URL: http://www.ijser.org/researchpaper%5CComparison-of-Open-Source-Crawlers--A-Review.pdf.

[44] C Serrão, A Ricardo, et al. "Comparison of existing open-source tools for Web crawling and indexing of free music". In: *Journal of Telecommunications* 18.1 (2013).

[45] Carmen Klaussner. *NUTCH FIGHT! 1.7 vs 2.2.1.* URL: http://digitalpebble.blogspot.pt/2013/09/nutch-fight-17-vs-221.html.

[46] Validome. *Validome.* URL: http://www.validome.org.

[47] Ed Kubaitis. *Anybrowser Weblint Gateway.* URL: http://www.anybrowser.com/cgi-bin/lint/weblint.cgi.

[48] DJ Delorie. *Web Page Purifier.* URL: http://www.delorie.com/web/purify.html.

[49] Dr. Watson. *Dr. Watson.* URL: http://watson.addy.com/.

[50] Online Web Check. *CSE HTML Validator Pro.* URL: http://watson.addy.com/.

[51] Jonathan Hedley. *HTML Tidy Online.* URL: https://infohound.net/tidy/.

[52] Marc Gueury. *Html Validator.* URL: https://addons.mozilla.org/en/firefox/addon/html-validator/.

[53] Total Validator. *Total Validator.* URL: https://www.totalvalidator.com/.

[54] HTML Validator Pro. *HTML Validator Pro.* URL: http://app.validator.pro/.

[55] Mathias Bynens. *Ambiguous ampersands.* URL: https://mathiasbynens.be/notes/ambiguous-ampersands.

[56] Squareup. *Retrofit - A type-safe HTTP client for Android and Java.* URL: https://square.github.io/retrofit/.

[57] Jonathan Hedley. *jsoup Java HTML Parser, with best DOM, CSS and jquery.* URL: https://jsoup.org/.

[58] Apache Software Foundation. *Commons CSV.* URL: https://commons.apache.org/proper/commons-csv/.

[59] Google. *Gson: A Java serialization/deserialization library that can convert Java objects into JSON and back.* URL: `https://github.com/google/gson`.

[60] Oracle. *Oracle VM VirtualBox.* URL: `https://www.virtualbox.org/`.

[61] Linux Foundation. *The Xen Project, the powerful open source industry standard for virtualization.* URL: `https://www.xenproject.org/`.

[62] W3C. *World Wide Web Consortium.* URL: `https://www.w3.org/`.

[63] W3C. *HTML5.* URL: `https://www.w3.org/TR/html5/`.

[64] WhatsApp Inc. *WhatsApp.* URL: `https://www.whatsapp.com/`.

[65] Tencent. *QQ.* URL: `https://im.qq.com/index.shtml`.

[66] Microsoft. *Skype.* URL: `https://www.skype.com/en/`.

[67] W3C. *Error Explanations for The W3C Markup Validation Service.* URL: `https://validator.w3.org/docs/errors.html`.

[68] Yang Sun. "A comprehensive study of the regulation and behavior of web crawlers". PhD thesis. The Pennsylvania State University, 2008.

[69] John Wiley & Sons Ltd. *Software: Practice and Experience.* URL: `http://onlinelibrary.wiley.com/journal/10.1002/(ISSN)1097-024X`.

Appendices

A. Requirements

A.1 Functional requirements

Table A.1: FR01 - WARC file manipulation

Identifier	FR01
Category	Data handling
Name	WARC file manipulation
Prioritisation	Must have
Description	The tool must be able to successfully parse WARC files generated by the crawler and correctly process their information.

Table A.2: FR02 - HTML file manipulation

Identifier	FR02
Category	Data handling
Name	HTML file manipulation
Prioritisation	Must have
Description	The tool must be able to successfully parse web pages created with HTML and correctly process their information.

Table A.3: FR03 - Validator integration

Identifier	FR03
Category	Data handling
Name	Validator integration
Prioritisation	Must have
Description	Must be able to send requests to the HTML validator, receive its answer and correctly process their information.

Table A.4: FR04 - Valid results

Identifier	FR04
Category	Data handling
Name	Valid results
Prioritisation	Must have
Description	Information produced by the tool must accurately represent the original data sources.

Table A.5: FR05 - DBMS integration

Identifier	FR05
Category	Data handling
Name	DBMS integration
Prioritisation	Could have
Description	The tool can be integrated with a Database Management System for easier data manipulation.

A.2 Quality requirements

Table A.6: NFR01 - Resume functionality.

Identifier	NFR01
Category	Recoverability
Name	Resume functionality
Prioritisation	Should have
Description	Due to the large amounts of data expected to be gathered and their anticipated processing times, the tool should have a way for users to resume their work from the point they stopped. This interruption can be caused by many different factors such as, power failure, hardware malfunction, user or software error.

Table A.7: NFR02 - Processing data times.

Identifier	NFR02
Category	Performance
Name	Processing data times
Prioritisation	Must have
Description	The tool must be able to process the desired information within an acceptable time frame.

Table A.8: NFR03 - Hardware dependent performance

Identifier	NFR03
Category	Scalability
Name	Hardware dependent performance
Prioritisation	Should have
Description	Time spent by the tool processing data should be able to be improved if it is run on more powerful hardware.

Table A.9: NFR04 - Ease of use

Identifier	NFR04
Category	Usability
Name	Ease of use
Prioritisation	Could have
Description	Installing and running the tool should not require too many steps or constant user supervision.

B. Short paper

As of the 28th of August, 2017, a version of the following short paper has been accepted for publication by the Software: Practice and Experience journal [69].

Towards Characterizing HTML Defects on the Web

Short Paper

Joaquim Mendes, Nuno Laranjeiro, Marco Vieira
CISUC, Department of Informatics Engineering,
University of Coimbra, Portugal

Abstract

HTML is being massively used as an interface to provide services to users. Web developers are producing and changing sites at high pace, while trying to support the latest HTML standards. In this context, it is common to find websites that do not comply with the standards and fail to be correctly processed by browsers. Considering this dynamic environment and the increasingly large diversity of browsers, with frequent updates, the appearance of problems in web pages is a common, sometimes severe, hard-to-track problem. In this short paper, we describe the initial design of an approach that will be used to obtain large-scale information regarding the characteristics of HTML documents on the Web and to extract indicators of representative errors made by their developers. Preliminary results show nearly 90% of the pages analyzed having at least one type of error and the prevalence of a small number of error types. The information that the tool will provide can be used by developers to build more reliable websites.

1. Introduction

HTML is nowadays being used to not only hold information but also to support businesses, that rely on the correctness of the HTML documents to reach clients. Thus, this language is being used as an endpoint interface with clients, through which they access operations of all kinds, including entertainment or business. This became particularly true with the advent of HTML5, which adds complex features to the standard, including multimedia, device access, or even semantics [1].

The typical time-to-market constraints of web application development, bring in the need for fast development of new websites and for fast application of changes to existing sites, so that new or different functionality can be accommodated. The creation of new HTML standards also leads developers to change their web applications, to benefit from the latest features and to be up to date with the latest trends. In any case, developers mostly concentrate on assuring correctness of existing functionality and overlook the necessary time for verification and validation activities [2].

Nowadays it is common to find websites that, due to small details, do not comply with the standards and fail to be correctly processed by browsers [3]. Sometimes developers are just trying to take advantage of browser-specific features and overlook standards compliance in return of some benefit, while some other times they just make mistakes, leaving residual defects on the pages.

Browsers gradually became tolerant to small mistakes, which actually leads developers to further disregard compliance with standards, from the moment that web pages are apparently rendered correctly [3]. The problem is that the Web environment is hugely dynamic and browsers are increasingly diverse, being very frequently updated. In this environment, there is no guarantee that a page holding (or prone to hold) residual defects will still work in the next version of a given browser. This is a hard-to-track problem that can have serious consequences on the service being delivered [2].

In this paper, we present the initial design of an approach for: i) analyzing HTML document characteristics (e.g., size, complexity, number of outgoing links), ii) validating their conformance to the standards, including obtaining detailed information regarding errors or bad practices in each document, and iii) analyzing the data to extract meaningful problem indicators (e.g., frequent errors, errors occurring in documents holding specific characteristics).

We used our approach to carry out an initial assessment of a small set of 1344 web pages, collected their characteristics and identified an initial set of frequent problems. Despite the reduced size of the experiments, results tend to agree with previous research in this area, with nearly 90% of the analyzed pages showing at least one type of error. We also observed the prevalence of a reduced number of error types, with just 14 different types accounting for 76.91% of all errors found.

Our end goal is to extend the characterization and analytics steps of our approach and carry out a large-scale experiment to characterize the HTML contents of a large set (at the million scale) of popular websites. We will also gather information regarding representative mistakes made by developers and will use it to build a model of the typical faults (i.e., the defects) introduced by developers in HTML documents.

Although this kind of work is quite well-known in the dependability community [4], it is far less common in the Web research community. In fact, the work that is closer to ours (although with different goals) has been carried out about a decade ago, according standards that are in disuse now. Thus, there is currently no up-to-date characterization information or a reusable HTML fault model based on large-scale data that can be used by developers or researchers. This kind of information is vital for web developers to understand the reliability of their websites and also to help building more reliable sites, that can accommodate the presence of small mistakes (e.g., introduced by some change to the site).

This paper is organized as follows. Section 2 presents related work and Section 3 overviews our approach to analyze HTML. Section 4 discusses preliminary results and Section 5 concludes this paper.

2. Related Work

More than a decade ago, the authors in [2] used the W3C validator [5] against 44,340 web sites composed of random sites, popular sites, and sites found using three search engines. Only 5% of the web pages analyzed were standards compliant. Top problems included Missing DOCTYPE, the omission of the *alt* attribute, and the lack of *type* attribute. Apart from the use of home and inner pages, no filtering (e.g., removing 404 error pages) or removing bias (e.g., same-source pages) from the URL list appears to have been made.

The work in [3] used *Alexa.com* as source of 100,000 URLs to find only 3% of valid pages, with the analysis per geographical origin showing significant differences. The top problems found were, the lack of a DOCTYPE declaration, missing character set and a conflict between the character set declaration and the *meta* attribute. The URL list was built by Internet Explorer users with Alexa's toolbar and could be also biased to regional variants of some sites (e.g., Google). There is no description of filtering or removing bias from the list.

The Metadata Analysis and Mining Application (MAMA) [6] analysed ~3.5 million URLs from DMOZ, Alexa.com top 500, and a set of W3C member companies. Domain parking sites, error pages, and non-http(s) links were filtered out. Characterization included the most and least used attributes, types of encoding, the total number of hyperlinks and images, among others.

Only 4.13% of the URLs were found to be W3C valid in [6]. 51% of the documents included the DOCTYPE declaration, with *Strict* flavors displaying higher validation rates when compared to *Transitional* and *Frameset*. XHTML, with 13.4% validation, also fared better than HTML, with 6.6%. Of the 27 possible types of warnings, 14 were signaled. The largest amount of warnings per web page was 5, with 2 being the most common. The 17 most common errors occurred on more than 10% of the total web pages and included: a wrong attribute; the absence of a required attribute; missing document type; or a close tag for a not open element. There is much space for improvement at the level of techniques such as bias removal and filtering, but especially on the richness of the results, including the analysis of the HTML documents defects.

The authors in [7] used the WDG HTML validator [8] over 2.5 million URLs gathered from the Open Web Directory. Only 0.71% of pages were found to be valid. The most frequent errors were *No DTD declared*, followed by *Non-standard attribute specified* and *Required attribute not specified*. The most common number of errors per page was 4, with an average of 5.2.

The work in [9] analysed 1 million pages from the Open Web Directory. Filtering techniques included removing duplicates, empty pages, and URLs not returning 200 OK status codes. The WDG HTML Validator and the W3C CSS Validator [10] revealed only 2.6\% of valid web pages (with an average of 6 errors per page, with 3 being the most frequent). The top errors included non-existent attributes, missing required attributes, missing document type declaration, closing not open elements, or elements in a wrong location. In [11] Transitional flavors of HTML were found to be more frequent than their Strict versions. [12] also confirmed the low percentage of successful validation. The authors in [13] analyzed *gov.br* websites and obtained slightly higher values regarding standards compliance.

The above works have the large problem of being at many years of distance, with none focusing on HTML5, leaving researchers with no updated information of the current state of the Web, regarding the characteristics, validity of modern HTML documents, and typical mistakes done by developers.

3. Approach for Characterizing HTML Defects

In this section, we describe the main steps of our approach, which we summarize as follows:

1) Define an initial set of URLs that point to HTML documents that will be the target of analysis;
2) Perform a crawl of the HTML files specified by the initial URL set and store the files;
3) Analyze the HTML document characteristics (e.g., size, complexity, number of outgoing links);
4) Use state of the art HTML validators to check the HTML code compliance against the standards, including detailed information regarding errors or warnings (i.e., bad practices) in each document;
5) Analyze the data to extract meaningful problem indicators (e.g., frequent errors, errors occurring in documents holding specific characteristics).

The approach steps are currently supported by a prototype tool, which we depict in Figure 1, and that is based on the combined use of free existent tools and custom code. The whole solution is open-source and will be freely available, for use and adaptation by other researchers once completed. The next paragraphs explain our approach, mapping its 5 steps to the tool components.

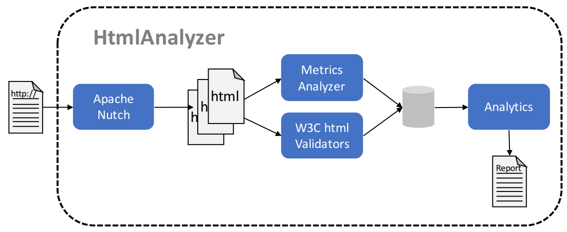

Figure 1 – Tool overview.

The input for our tool is a set of URLs (*step 1*), provided by the user, which are crawled by **Apache Nutch** [14] in our prototype, the HTML code is extracted and stored locally (*step 2*). The tool then triggers a metrics analysis (*step 3*) and a validity analysis (*step 4*) over the stored HTML files. The **metrics analysis** involves analyzing the HTML documents for characteristics such as document size, number of HTML tags present, text/ HTML ratio, among others. To define this set of metrics, we collected the HTML metrics used by several authors in this domain [2], [6], [9]. The idea of defining a set of HTML metrics is that, by the end of the process, we have the ability of correlating HTML problems with some of the characteristics of the document (e.g., document complexity with a wrong tag structure). This is actually ongoing work, and we intend to extend and refine this set in the future. Table I presents the current set of metrics being used.

The second type of analysis is directed to the **validation of the HTML code**. With the purpose of selecting one or more HTML validators for integration in our approach and tool, we analyzed 11 different validators, including the most well-known CSE HTML validator, Total Validator, WGD HTML, and the W3C validator,

Table I – Preliminary HTML metrics for individual pages.

Metric	Description
fileSize	HTML file size
textSize	Size of textual content
elemSize	Size of HTML elements
textHtmlR	Ratio of text to HTML elements
version	Version of HTML in use
elemCount	Total number of HTML elements
elemFreq	List of HTML elements and the number of times each of them occurs
linkCount	Total number of hyperlinks
protLFreq	Types of protocols used in hyperlinks and the number of times each of them occurs

against key features such as the ability to analyze HTML5, validating pages served through HTTPS, maximum file size and error limits. We opted for using the W3C validators [5] (which consist of essentially two tools, one for HTML5 and another one for other versions of the language) due to compliance with the key features and their overall recognition in the industry.

The first task at this validation step is to understand the type of document being handled (e.g., an HTML5 document, a XHTML 1.0 document), which should be present at the beginning of each file. The next step is to understand if the document is syntactically valid (i.e., no error is found by the validator) or not. If HTML errors are detected (i.e., if syntactic rules are broken, such as not closing a <*div*> tag, or having a wrong name in a tag), we collect detailed information regarding the errors. The W3C validators also produce warnings for less serious mistakes found in the web page (e.g., using attributes that are not supported by all browsers, or using attributes that can be safely omitted for particular elements).

As the validators produce errors and warnings that are specific for the code being analyzed, our tool is then responsible for identifying the generic type of issue being identified (i.e., not closing a <div> tag or a <table> tag should be reported as the same type of error). The identification of generic errors has been manually verified in our prototype tool (for the errors obtained during the experiments described in the next section).

Finally, we analyze the data collected to extract relevant indicators regarding the defects introduced by developers (*step 5*). Currently we are considering the preliminary indicators displayed in Table II, mostly based on empirical evaluation and on what was used in previous work [3], [6].

Table II – Defect indicators for individual pages.

Defect Indicator	Description
errorCount	Nr. of HTML errors according to the standard
errorTypes	Nr. of occurrences per type of HTML error
warningCount	Nr. of HTML warnings according to the stand-
warningTypes	Nr. of occurrences per type of HTML warning

Again, our intention to enrich this set of indicators, and further use the data to obtain a fault model for HTML that holds representative web page defects. Such model can be helpful for verification and validation activities, especially considering that nowadays HTML is being generated at runtime, which makes it more difficult for developers to assure that it will be correct.

4. Preliminary results

In this section, we discuss the results obtained during a preliminary experimental evaluation carried out using our tool and using a total of 1344 URLs as input. The URLs were randomly selected from the Directory of the Web (as in [6]). In the future, we intend to complement this information with data gathered from known popular web sites. We first analyze the global results, and then discuss the results for HTML5, the latest standard.

4.1 Global Dataset Analysis

Table III highlights the **overall characteristics** of the dataset, as reported by our tool.

Table III – Overview of the HTML dataset characteristics.

Characteristic	Value
Average file size	33.73 KB
Average size of text content	3.71 KB
Average size of HTML elements	30.02 KB
Average ratio of text to HTML elements	12.36%
Most common HTML version	HTML5 (30.43%)
Average number of HTML elements	342
Most frequent HTML element	<a> (17.56%)
Average number of hyperlinks	60
Most common protocol used in hyperlinks	HTTP (72.35%)

As we can see in Table III, the average size of text content in a page is about ten times lower than the size of used by the HTML elements themselves, which suggests the presence of other types of elements (e.g., multimedia content). HTML5 is the most popular version, with the average number of elements used per page being quite high (342). The *<a>* tag defines hyperlinks and is the most frequent element in the dataset. The next paragraphs detail these latter three characteristics.

Figure 2 shows the frequency of each version and flavor of HTML found in the global dataset.

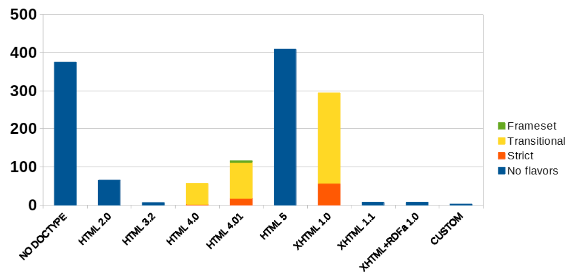

Figure 2 – Frequency of HTML versions and flavors

As we can see in Figure 2, 28% of web pages omit the version information, leaving this task up to the browsers. HTML 5 is the most common version present in our dataset (30% of the pages), followed by XHTML 1.0 (22% of the pages). We can also see (also as in [11]) that the Transitional flavors of HTML are more popular than their strict counterparts, which is understandable from an ease of development perspective.

It is interesting to observe that the average number of elements per page generally increases with the HTML version, this trend is shown in Figure 3. The low number observed for XHTML 1.1 is likely due to the small number (8) of XHTML web pages in the dataset.

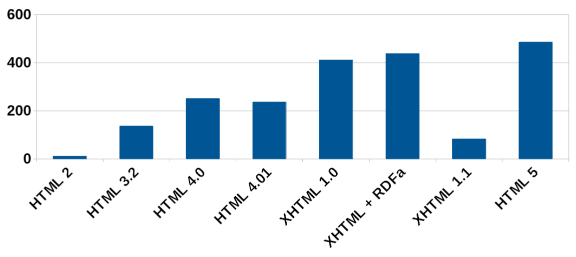

Figure 3 – Average number of HTML element per version

Finally, regarding the most common elements, it comes with little surprise that the *<a>* tag is the most used (17.56%), as a web page with no hyperlinks defeats the purpose of the interconnected information of the web. It was followed by the *<div>* tag (17.12%) which is a generic container for data or other elements, many times used to group elements and apply the same Cascading Style Sheet (CSS) styles to the whole group. The remainder of the most frequent tags were *<li>* (8.99%), which represents a list item, *<span>* (6.70%), which is a generic inline container, and *<img>* (5.30%) the tag for representing an image.

Table IV summarizes the overall results for **HTML validation** of the documents in the dataset, in particular in what concerns detected errors. In accordance with previous studies, the rate of compliance with the standards is very low, with only 11.09% of pages managing to pass validation with no errors. The common web page seems to be plagued with errors, an average of 41 per page. If we consider only the pages with errors, this number rises to 47. A reduced number of error types (14 different error types) accounts for about 77% of all 56257 validation errors found.

Table IV – Common errors present in the global dataset.

Error	Frequency
Incorrectly encoded character	13.14%
Absence of a required attribute	9.88%
Omission of a close tag	7.89%
Attribute that does not belong to the standard in use	6.58%
An attribute value not contained in double quotes	6.45%

Regarding validation warnings, the five most frequent validation warnings represent almost 88% of the total 19569 detected. The use of self-closing tags on document types that do not allow it was the most frequent case. The remaining top warnings were related with the use of incorrectly encoded characters in a variety of ways. Table V summarizes these results.

Table V – Top validation warnings in the global dataset.

Warnings	Frequency
Using a self-closing tag on a non HTML5 /XHTML file	31.97%
Entity name with a badly encoded special character	18.25%
Attribute name with a badly encoded special character	15.24%
Attribute value with a badly encoded special character	13.56%
Incorrectly encoded character on the page content	8.67%

4.2 HTML5 results

Regarding the **HTML5 documents characteristics**, it is interesting to observe that all the metrics show higher values, with the exception of the text / HTML elements ratio. Table VI summarizes this information.

Table VI – HTML5 documents characteristics.

Characteristic	Value
Average file size	53.31 KB
Average size of text content	4.50 KB
Average size of HTML elements	48.80 KB
Average ratio of text to HTML elements	9.22%
Average number of HTML elements	487
Most frequent HTML element	<div> (22.38%)
Average number of hyperlinks	91
Most common protocol used in hyperlinks	HTTP (76.94%)

We observed that the top HTML5 elements used are the same as the ones previously reported for the overall dataset. Thus, we again find *<div>* (22,38%), *<a>* (18.55%), *<li>* (11.56%), *<span>* (8.75%), and *<img>* (4.16%).

Regarding the **validation of the HTML5 documents**, we found that 93.6% of the HTML5 pages had at least one error. The average number of errors that we can expect to find in an HTML5 page in our dataset is 22, raising to 24 if we consider only the pages with errors. The top five errors represent more than 60% of the total number of errors (9250). The top ten errors would allow us to reach the same value observed for the global dataset. Table VII summarizes the top problems observed.

Table VII – HTML5 documents top errors.

Error	Frequency
Inclusion of an attribute not allowed for an element	17.41%
Deprecated attributes related to style	16.86%
Absence of the *alt* attribute for *<img>* elements	12.26%
Element nested in a wrong element type	8.86%
Use of an *itemprop* attribute with no parent	5.23%

We can see that 4 out of the top 5 problems involve the improper usage of attributes. It is important to note that the errors shown in Table VII are specific for this version of the standard and are not shared by the previous versions. However, it is our intention to further understand the similarities, causes, and effects of the errors that are currently reported as different by the W3C validators.

Regarding warnings, a total of 1978 were uncovered. The following five, listed in Table VIII, refer to about 69% of all occurrences.

Table VIII – HTML5 documents top warnings.

Warning	Frequency
Use of the deprecated *border* attribute	20.98%
Duplicate IDs	17.64%
Absence of headers	17.59%
Incorrectly described attribute that is not serializable	6.83%
Use of an unnecessary attribute, which can be omitted	5.92%

Almost 21% of the warnings were related to the use of the deprecated *border* attribute, followed by duplicate IDs (17.64%), the absence of headers (17.59%), an incorrectly described attribute that, thus, cannot be serializable (6.83%) and the use of an unnecessary attribute, which can be omitted (5.92%).

HTML5 further reduced (when compared to the previous versions) the number of style and presentation elements, delegating that type of task for Cascading Style Sheets. It is interesting to observe that the second most common error and the top warning both refer to the use of deprecated elements and attributes, which means their use is still quite high among developers. This can attributed to many factors, including lack of knowledge regarding the new standard, or indifference regarding HTML validation, among other causes [3].

Regarding the overall results and the HTML5 results, it is interesting to note that most of the problems are related with attributes, suggesting that the use of HTML elements is less error prone. It is our intention to further analyze the gathered data and also run specialized data analytics tools, so that we can get further insights over the data (which are extremely difficult to obtain) as they involve complex analysis and correlations.

5. Conclusion

In this paper, we present the initial design of an approach and tool for characterizing and analyzing the syntactic validity of HTML documents. Preliminary results show interesting insights regarding the characteristics of HTML documents and their overall low compliance with standards. It was interesting to observe that this low compliance trend continues with HTML5 documents, which also appear to be more complex.

As the first step of a larger experiment, besides enlarging the dataset to the million scale, we intend to enrich the HTML documents characteristics and run a wider set of analysis over the data. Besides providing an up-to-date vision of the use of HTML on top websites, we intend to further understand the root causes of the errors, and actually use errors for understanding how reliable, in the presence of changes, a particular website is. Such information is relevant for website developers (for training, or applying website changes), but might also be helpful for browser development and optimization.

References

[1] World Wide Web Consortium, “HTML5.” [Online]. Available: https://www.w3.org/TR/html5/.

[2] S. Chen, D. Hong, and V. Y. Shen, “An experimental study on validation problems with existing html webpages,” in *Proceedings of the 2005 International Conference on Internet Computing, ICOMP’05*, 2005, p. 373.

[3] E. Ofuonye, P. Beatty, S. Dick, and J. Miller, “Prevalence and classification of web page defects,” *Online Information Review*, vol. 34, no. 1, pp. 160–174, 2010.

[4] J. Durães and H. Madeira, “Emulation of Software Faults: A Field Data Study and a Practical Approach,” *IEEE Transactions on Software Engineering*, vol. 32, no. 11, pp. 849–867, Nov. 2006.

[5] World Wide Web Consortium, “The W3C Markup Validation Service,” 2017. [Online]. Available: https://validator.w3.org/.

[6] B. Wilson, “MAMA - Metadata Analysis and Mining Application,” 2008. [Online]. Available: http://maqentaer.github.io/devopera-static-back-up/http/dev.opera.com/articles/view/mama/index.html.

[7] D. Parnas, “How to cope with incorrect HTML,” 2001.

[8] L. Quinn, “WDG HTML Validator,” 2007. [Online]. Available: http://www.htmlhelp.com/tools/validator/.

[9] R. Saarsoo, “Coding practices of web pages,” 2006. [Online]. Available: http://triin.net/2006/06/12/Coding_practices_of_web_pages.

[10] World Wide Web Consortium, “The W3C CSS Validation Service,” 2009. [Online]. Available: https://jigsaw.w3.org/css-validator/.

[11] P. Beatty, S. Dick, and J. Miller, “Is HTML in a race to the bottom? A large-scale survey and analysis of conformance to W3C standards,” *IEEE Internet Computing*, vol. 12, no. 2, pp. 76–80, 2008.

[12] A. Pinterits, H. Treiblmaier, and I. Pollach, “Environmental websites: an empirical investigation of functionality and accessibility,” *International Journal of Technology, Policy and Management*, vol. 6, no. 1, pp. 103–119, 2006.

[13] H. de S. Ganzeli, G. Bressan, and A. M. Moreiras, “ICT web: analysis of the Brazilian governmental web,” in *Proceedings of the 18th Brazilian symposium on Multimedia and the web*, 2012, pp. 383–386.

[14] Apache, “Apache Nutch.” [Online]. Available: http://nutch.apache.org/.