

Generating Website Analyses with Automatic Crawling Processes

Jennifer Swoboda



MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im Juni 2017

© Copyright 2017 Jennifer Swoboda

This work is published under the conditions of the Creative Commons License *Attribution-NonCommercial-NoDerivatives 4.0 International* (CC BY-NC-ND 4.0)—see <https://creativecommons.org/licenses/by-nc-nd/4.0/>.

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Hagenberg, June 26, 2017

Jennifer Swoboda

Contents

Declaration	iii
Abstract	vii
Kurzfassung	viii
1 Introduction	1
1.1 Problem Definition and Relevance	1
1.2 Goals	2
1.3 Structure	2
2 Technical Basics	4
2.1 Web Trends	4
2.2 Web Analysis	4
2.3 Webarchive	5
2.4 Web Crawler	5
2.4.1 User Agent	5
2.4.2 Crawling Methods	5
2.4.3 Challenges	7
2.5 Web Scraping	7
2.6 Data sets	7
3 State of the Art	9
3.1 Manually Written Trend Analyses	9
3.1.1 Advantages	9
3.1.2 Disadvantages	10
3.2 Automated Web Analyses	10
3.2.1 Advantages	10
3.2.2 Disadvantages	11
3.3 Web Crawlers	11
3.3.1 GoogleBot	11
3.4 Web Scraping	11
3.4.1 Scrapy	12
3.4.2 BeautifulSoup	12

4	Outline and Implementation	13
4.1	Requirements And Architecture	13
4.2	Python	14
4.3	Crawling	14
4.3.1	Downloading A Web Page	14
4.3.2	Crawling Selection	15
4.3.3	Scraping A Web Page	15
4.3.4	General Information	16
4.3.5	Technical Aspects	16
4.3.6	Design Aspects	19
4.3.7	Crawling Process Improvements	20
4.3.8	Crawling Problems	20
4.4	Data Sets	21
4.4.1	Top 1000 Global	21
4.4.2	Top 1000 Austria	22
4.4.3	Top 1000 Design	22
4.4.4	1000 Randomly Selected URLs	23
4.5	Data	24
4.5.1	Data Exploitation	24
4.6	Data Representation	26
4.6.1	Flask	27
4.6.2	Representation Structure	27
4.6.3	Information Visualization	28
5	Implementation Results	31
5.1	Outcome	31
5.1.1	Fonts	32
5.1.2	Font Scripts	35
5.1.3	Colors	36
5.1.4	Image Formats	37
5.1.5	Programming Language	40
5.1.6	Markup Languages	41
5.1.7	JavaScript Frameworks	42
5.1.8	Character Encoding	44
5.1.9	Site Elements	45
5.1.10	Web Servers	46
5.1.11	Mobile Optimization	48
5.1.12	Analysis Tools	49
5.1.13	Web Frameworks	51
5.1.14	Content Management Systems	52
5.2	Summary	53
6	Evaluation	56
6.1	Manually Written Trend Analyses	56

Contents	vi
6.1.1 Awwwards Trends 2017	56
6.1.2 Webflow Image Formats Trends 2017	57
6.1.3 Hackernoon’s Best JavaScript Frameworks In 2017 . .	57
6.1.4 1&1 Website Performance Trends	58
7 Conclusion	59
A Installation Guide	61
B CD-ROM/DVD Contents	62
B.1 Masterthesis	62
B.2 Literature	62
B.3 Images	62
B.4 Projectfiles	62
References	63
Literature	63
Online sources	64

Abstract

As the growth of the web outpaced all expectations, also the work of web developers and web designers has become essential. To accommodate the variety of requirements for web enthusiasts, it is necessary to acquire knowledge of the newest technologies and design trends. This work takes a look at the generation of different web and web trend analyses and provides an approach for creating analyses without the issues of personal expertise opinions. The implementation uses crawling processes to collect and build different website data sets. Design and technological aspects of each website are extracted and representations for the data collections are built. To evaluate the implementation, comparisons with various manual written trend analyses are conducted, similarities and deviations examined.

Kurzfassung

Als das Wachstum des Internets alle Erwartungen überschritt, wurde auch die Arbeit von Webentwicklern und -designern und deren kontinuierliche Weiterbildungsprozesse über aktuelle technische- und design Webtrends unerlässlich. Die Arbeit befasst sich mit der Generierung genau dieser verschiedenen Webanalysen und Webtrendanalysen. Darüberhinaus stellt das Projekt einen Ansatz vor, mit dem objektive Webstatistiken ohne die Inkludierung von persönlichen Autorenmeinungen erstellt werden können. Die Implementation verwendet Crawling Prozesse um Websites zu sammeln und bildet damit verschiedene Datensammlungen. Um die durchgeführte Realisierung beurteilen zu können, werden Vergleiche mit verschiedenen manuell geschriebenen Trendanalysen durchgeführt, sowie Ähnlichkeiten und Abweichungen untersucht.

Chapter 1

Introduction

Nowadays, the web contains over a billion websites and every day new pages join it. As the World Wide Web continues to grow at a rapid pace, a clear network structure is necessary for search engines to retrieve valuable information. Due to this increasingly growing web space, the service of automated tools for finding, extracting and filtering data has become irreplaceable [9]. For this process, so-called *web crawlers* are responsible.

Web crawlers, also known as robots or spiders, are programs which download web pages from several websites. Typical usages for crawlers are pulling data for search engines, web archiving or data mining. The term *web archiving* describes a service, which collects and archives significant sets of web pages, for example, provided by *The Internet Archive* [30]. In comparison, *data mining* is the process of collecting and analyzing large amounts of data with a focus on an automatic discovery of patterns, classification of the data and analyzing patterns that can lead to reasonable predictions.

As the growth of the Web surpassed all beliefs, also the work of web developers and the continuous process of expanding and optimizing their professional knowledge about current technology and design web trends has become essential. Most of the time these web trends or web predictions are created by experts. This method for creating web trends leads to the disadvantage of carrying the personal opinion of the author as a more or less established component of the work.

With the process of generating website analyses and web trends with automatic crawling processes, this particular issue can be remedied. For many businesses the usage of web crawlers has become a valuable part anyway, thus, why not try to combine these parts?

1.1 Problem Definition and Relevance

The World Wide Web and consequentially also web design and web technologies are subject to permanent and rapid changes in web development.

Yesterday's trends are already outdated today, and new trends continue to arise. Therefore as an ambitious web developer or web designer, it is even more important to be permanently informed about the newest state of the art development. To be able to ensure the relevance and accuracy of these trends and analyses, the selection must be considered carefully.

There are uncountable websites, which provides such trends. The issue with these trends are often the personal opinions of the author and their related error susceptibility of the result. To avoid this inaccuracy and the connected trend distortion, websites should be indexed and analyzed by crawlers. The website *W3Techs* [32] already is using this method of automatically retrieving specific technical properties from websites. Additionally, it would be desirable to collect also design aspects from the websites with the help of crawling processes. In order to do this, an extensive web analysis of technical and design web aspects will take place.

1.2 Goals

To circumvent the issues related to personal opinions in manually written web analyses, the approach of automatically indexing and analyzing websites by crawling is the solution. The goal is to retrieve an extensive web analysis for predefined data sets. These data sets should be crawled and additionally different technical and design properties extracted from their websites and stored in the database. For trend analyses, the crawling process for the various data sets should be retried and stored again within a specific time range. To retrieve a deeper understanding for the crawled data, the detection of correlations between different aspects is covered. Finally, a comparison with manually written trend analyses should be performed, to evaluate the result.

1.3 Structure

For a better understanding, the thesis is subdivided into several main chapters. Recommended is reading the thesis from the beginning to the end although it is still possible to read only certain chapters if desired. Chapter 2 describes some technical basics, which should serve as a preparation for reading the whole thesis. However, for more experienced people, this chapter can be skipped if no further explanations are needed. Chapter 3 lists some current web trend analyses, with advantages, disadvantages, and examples for web crawler and web scraping methods. Chapter 4 presents an outline of the actual new approach and describes the process of the implementation. The crawling, trend, and correlation results of the implementation can be found in Chapter 5. Chapter 6 evaluates the result of the implemented approach by comparing its outcome with other web trend analyses. Finally,

Chapter 7 sums up the different problems and solutions of the project and tries to give a conclusion about the result of the implementation.

Chapter 2

Technical Basics

The web is changing at a rapid pace, therefore it is essential to clarify a few technical basics before delving deeper into the subject. The following chapter will ensure some fundamental knowledge, which is helpful to understand the implemented approach. A good foundation of basic IT knowledge and general understanding of web development is still needed.

2.1 Web Trends

As already mentioned, web trend analyses are constantly gaining on relevance. However, to be aware of recent web changes might be urgent for business owners as well. With some additional knowledge, they might be capable of improving their competitive situation and furthermore, be able to develop new business models. The quintessence for web enthusiasts lies in detecting high-quality sources for web trend analyses, filtering relevant from irrelevant trends within a short period and sensing the right balance for implementing these trends.

One extensive mechanism for web trend analyses is the prediction using expert advice. Such specialists deal intensively with the World Wide Web and all its changes. Using surveys is another method, however, it might suffer from the disadvantage that all participants must build previous knowledge about the web to achieve a useful result. Additionally, the possibility exists to generate automatic web trends with the help of web crawlers. In this process, web crawlers are fulfilling the task of indexing websites and comparing the aspects within crawling intervals to detect value increases and decreases.

2.2 Web Analysis

The term *web analysis* describes a continuous process of establishing, analyzing and optimizing websites and is particularly used in online marketing. Key features of web analyses are the identification of possible vulnerabilities

and potential improvements, as well as quality assurance [5]. In this thesis, when speaking of web analysis, it describes a complex analysis of multiple websites with a focus on different website aspects or properties. Such aspects can have various forms, including general info about the website, as well as technical or design properties.

2.3 Webarchive

Webarchive is a service, where large sets of websites are collected over time and subsequently archived. *The Internet Archive* [30] is a non-profit library, which provides millions of free books, software, and movies. One service of this online library is especially interesting for this thesis, the so-called *Wayback Machine*. With this Wayback Machine, the user can enter a domain name and browse through its time changes. When entering a specific date, an HTML preview of the website from this timestamp is shown. Overall, more than 283 billion websites were saved over time. For example, *derstandard.at* was stored 6422 times until now, with its first capture in 1997. Compared to that, *google.com* was stored remarkable 515219 times starting one year later, in 1998.

2.4 Web Crawler

A *web crawler*, or *spider*, is a program which downloads web pages from several websites, mostly to pull data for search engines. Other usages are web archiving and data mining. Despite the fact that crawlers are becoming increasingly quicker and more complex, even the biggest and fastest web crawler is not able to crawl all websites on the web. However, with today's state of the art development, it is possible to crawl enormous website data sets with millions of pages [7].

2.4.1 User Agent

The term *user agent* refers a software that is reacting for a user. Commonly a user agent identifies a web browser like *Safari*, *Google Chrome*, *Opera*, *Microsoft Edge*, *Mozilla Firefox* or *Opera* and operating systems. When sending a request to a website the user agent is transmitted. However, if no user agent is indicated, the default agent will be taken.

2.4.2 Crawling Methods

The following paragraphs will cover different approaches for extracting data with web crawlers. Most of the time, the creation of a crawler is a combination of several methods to ensure a complex and efficient operating principle.

Selective Crawling

Selective Crawling is a method to retrieve web pages based on some selected criteria. Furthermore, a scoring function is used to filter relevant contents from a page. The fetched URLs are sorted according to the scoring system, in which the pages with a high ranking are ordered first. Scoring aspects can, for instance, be Depth and Popularity [2].

Focused Crawling

Most of the time, a web crawler does not need to cover every possible web page on the Internet, instead, it is often useful to focus on a particular topic. Similar to the URL ordering technique used for selective crawling, the scoring function can be extended for focused crawlers. The main task for focused crawlers is collecting documents to a specific topic and classifying the crawled pages into categories [2].

Distributed Crawling

For large-scale search engines, single crawling processes would be insufficient, since these engines have to fetch large amounts of data very fast. Distributed crawling is a method for partitioning crawling tasks which uses PageRank algorithms for its efficiency and quality search. Advantages of distributed crawlers are the scalability and the memory efficiency in comparison to other crawling methods [2].

Collaborative Crawling

A collaborative crawler is a group of crawling nodes, in which each node is responsible for a specific part of the Web. It uses multiple crawlers to explore a web-space, to generate summaries and to store these for future reference [1].

Incremental Crawling

An incremental crawler indexes the web regularly and revisits URLs periodically to ensure freshness. A big benefit of the incremental crawler is that it can be scheduled, paused and stopped [1].

Web Dynamics

The last crawling method deals with the question how the web changes over time. This knowledge would allow crawlers to define how often a search engine should index a website. The goal is to have an up-to-date version of the website on the local storage, to retrieve freshness and moreover to determine the website age, specified by the time the website has not changed [2].

2.4.3 Challenges

This paragraph will give a short overview of the challenges and issues, web crawling methods and web crawlers have to deal with.

Caused by the fact that the World Wide Web has extremely large sets of data, a crawler that visits an extensive amount of pages has to use the available network resources efficiently to maximize the throughput of the crawler and minimize the amount of wasted bandwidth [3].

The quality of the crawled content retrieving from several websites is another problem web crawlers have to cope with. Additionally, duplicated and similar content on the web influences the quality of the result too. Duplicated content mostly arises from the circumstance that many websites allow multiple URLs to refer to the same content, for example, a URL with *www*, and without this prefix, deliver the same content.

A possible way to exclude web pages from crawling processes is a Robots-File¹, which allows website owners to declare pages of their sites restricted. Unfortunately, these guidelines are not respected from all crawlers [4].

Besides these issues, a crawler also has to frequently avoid crawler traps. The term crawler trap describes a URL or a set of URLs, which might cause an infinite crawling loop [4]. Other setups can be crawling pattern checks or honeypot traps. A honeypot is a defense system for network security and can trap attacks and record information about tools and activities of the hacking process [10].

Furthermore, crawlers should not cause increased load on the web pages they are indexing. Otherwise, high-performance crawlers can create a denial-of-service (DoS) attack without the right security measurements [3].

2.5 Web Scraping

Web scraping, *web harvesting* or *web data extraction* basically defines the process of pulling needed information from crawled web pages and storing the data into a local file or database. Before this step, actual thoughts about the data, which should be filtered are essential. To be able to target specific parts of the crawled data, CSS selectors and regular expressions are used[6].

2.6 Data sets

The term *data set*, or *data collection* describes a scalable list of data, in this specific case, containing various URLs of websites. The purpose of this list is to represent the web pages for the crawler to process. Before the generation of such a data set can take place, it is necessary to put thoughts into its meaning. Categorizations by topics, countries, popularity or other common

¹<http://www.robotstxt.org>

aspects of the data set are useful. For the creation of a data set, different methods can be found. One service is provided by link crawlers and serves the purpose of following links on websites and adding them to the data set.

Chapter 3

State of the Art

To compare the new approach with existing solutions, this section will sum up different web trend analyses and web crawlers already dominating on the market. With the growth of the web, also the service of web trend analyses increased. Fundamentally, there are two different types of these services:

- manual written trend analyses by experts
- and automatically generated web trends by crawling processes.

This section will cover a short description, examples and pros and cons of different methods for creating analyses.

3.1 Manually Written Trend Analyses

An uncountable number of manually written web trend analyses exist until now. These trend analyses are commonly written by web experts or web agencies. Websites which provide these kind of services are *awwwards.com*, *forbes.com*, *creativebloq.com*, *entwickler.de* and numerous others.

3.1.1 Advantages

A common way of writing trend analyses is to occupy oneself intensively with the subject area and all its subtopics. Thus, most of the time when web trend predictions are reported by experts or web agencies, the manually written trend analyses produce a fairly accurate result. Another benefit is the fact that a few specific aspects can be reported by humans, but can not be detected easily by automatic crawling processes. For example, identifying connections between different aspects of websites are simpler revealed, analyzed and evaluated by human beings.

3.1.2 Disadvantages

Website design and technology forecasts can create the impression of appearing rather objective. However, web trend predictions that require expert judgments often suffer from carrying personal opinions and biases. Furthermore, no expert knows everything. Thus, experts also gather information from other specialists and copied content and trends may no longer be avoided.

3.2 Automated Web Analyses

In comparison to expertise trend analyses, automated web analyses are created by crawling processes. Well-known service providers for this method are *builtWith.com* and *w3techs.com*.

BuiltWith

BuiltWith [18] is a tool for gathering technical information from websites. Information about web servers, SSL certificates, nameserver providers, email services, hosting providers, content management systems, frameworks, advertising, analytics and tracking, JavaScript libraries, media types, payment providers, mobile trends, encoding, CSS media queries, document information, aggregation functionalities, widgets and content delivery networks are detected from BuiltWith crawling processes. By reviewing these aspects *Web Technology Usage Trends* are created.

W3Techs

W3Techs [32] provides knowledge about the usage of various technology types on the web. The technological properties are separated into content management, server-side languages, client-side languages, JavaScript libraries, markup languages, character encodings, image file formats, site elements, SSL certificate authorities, social widgets, content delivery, traffic analysis tools, advertising networks, tag managers, web servers, operating systems, web hosting, reverse proxies, top level domains, server locations and content languages [32]. For statistics, W3Techs crawls the top ten million websites from Alexa [12] in a specific interval timing. Through these intervals, technological changes and modifications can be detected and W3Techs can establish web technology trends.

3.2.1 Advantages

In correlation to expertise web trend analyses, one main advantage of automatic trend analyses through crawling is the fact that no personal opinion or bias are included in the trend result. Furthermore, with an accurate data

set, crawlers can index various websites in a short amount of time. Experts would need an infinite period for gathering this load of information. In consequence, when crawling these quantities of data over some time, web crawlers are also faster in detecting changes and modifications.

3.2.2 Disadvantages

Essential for an accurate crawling result is the data set with the websites to crawl. The data set must be selected with careful consideration or otherwise, it might lead to issues. To deliver useful analysis results and to be able to detect web modifications, the crawling process needs to be repeated after some time. This crawling interval could lead to timing issues because the results are more accurate if the crawling takes place over longer time. Despite these disadvantages, crawler traps, limits in bandwidth, processing or storage can cause further problems for the trend analyses.

3.3 Web Crawlers

As mentioned in the previous chapter, different approaches and methods exist for fetching website data with web crawlers. Furthermore, several functioning web crawlers are already available on the market. This section will cover one of the most popular crawlers, which was developed by Google.

3.3.1 GoogleBot

The *GoogleBot* is a web crawling bot from Google. By crawling the GoogleBot discovers new and updated pages which need to be added to the Google Index. Google provides an enormous set of computers to fetch billions of web pages on the web. By using a specific algorithm, the bot knows exactly which sites to crawl, how often and how many pages to fetch from each web page. Data sets of URLs from previous crawling processes and additionally sitemap data are building the list for the crawler to index [21].

Google Index

Websites, which should be included in the result of a search engine, need to be inserted into the Google Index. The Google Index lists all of the websites Google knows about and is updated by new or modified pages [22].

3.4 Web Scraping

As mentioned before, web scraping tools are gaining on importance lately. Several web scraping tools and libraries are available on the market and dur-

ing the preparation phase of this project the two tools *Scrapy* and *Beautiful Soup* were examined precisely.

3.4.1 Scrapy

Scrapy [28] is an application framework built with Python for crawling and extracting data from websites. This framework can be useful for data mining, information processing and historical archival. Other usages despite web scraping are extracting data using APIs or the general purpose of a web crawler. The functionality of the crawler is basically making requests to the defined URLs from a data set and passing a response object. This object is searched through for new links to other pages to retry the process. With this method, Scrapy has gained the benefit to be able to schedule and process the requests asynchronously, which ensures faster crawling pace. For the selection and extraction of data from the HTML source, Scrapy is pulling CSS Selectors and XPath expressions to extract with regular expressions [28].

3.4.2 Beautiful Soup

Beautiful Soup is a library for fetching data out of HTML and XML files based on Python. It works with different parsers and provides simple ways of navigating, searching and modifying the parse tree. One benefit of Beautiful Soup is the compactness of code, since it does not need much to write an application. Furthermore, Beautiful Soup handles encoding automatically and allows the use of different Python parsers, for example, lxml or html5lib [16].

Chapter 4

Outline and Implementation

The following chapter describes own reflections about automatic web trend generation through multithreaded crawling processes and the consequential project implementation. For this mechanism, a combination of different crawling methods and data sets will be applied (see Chapter 2). The goal of this approach is to support web developers, web designers as well as business owners to extend their knowledge of the newest web trends. Through automatic creations of categorized web analyses, the issues with personal opinions and biases of experts and the resulting error-susceptibility can be reduced.

4.1 Requirements And Architecture

In consideration to the fast moving nature of the web, also the demands on web developers and web designers grow continuously. Thus, they always have to be up to date about new changes and trends in the World Wide Web. The challenge of information retrieval nowadays consists of finding valuable trends with the possible highest degree of truth. Furthermore, implementing these trends before they are obsoleted again is another challenge. Therefore the desired approach has to recognize changes and trends in a fast pace, cover a wider range of websites for the analyses and represent the data in a useful way.

The main focus of this approach will be on these three essential aspects:

- the selection and procurement of the data sets,
- the crawling processes,
- and the representation and exploitation of the data.

4.2 Python

The approach is implemented in Python¹, an interpreted, object-oriented, high-level programming language with dynamic semantics. The simpleness of Python ensures developers to learn the syntax rather easy and furthermore helps to reduce the programming effort. Moreover, many supported modules and packages encourage program modularity and code reuse [27].

The latest version of Python until now is 3.6.1, which was released on March 21, 2017. However, for this approach, the Python version 2.7.8 is used because at the moment of the implementation, some of the needed libraries were only available for the older version.

4.3 Crawling

With completing the preparation phase of the implementation within all its thoughts about the project structure and desired accomplishments, the actual crawler can be built. This section will cover the crawler architecture and the structural process from downloading the web page to extracting its data.

4.3.1 Downloading A Web Page

To download the web page for further processing, the Python module `urllib2` [31] is used, which is an extensible library for opening URLs. With this module, the actual downloading process is built.

The response of the URL opening request is the HTML output of the requested web page. The default user agent, for the `urllib2` module, is declared as *Python-urllib/2.6*. To prevent problems, caused by using the default user agent and consequentially to circumvent the issue of being blocked from websites, for the approach an identifiable user agent is added to the request.

The retrieved request data, however, needs to be checked for errors. When getting a *503 Service Unavailable Error* the downloading process will be retried once, but in the case of retrieving a *404 Not Found Error* this step is not required, because the listed website URL is either incorrect or not online anymore. To avoid further issues, the downloading process of the project is throttled. One simple modification for decelerating the downloading process is to add a time delay. This avoids the issue of downloading web pages too fast and the arising disadvantage of being blocked or overloading the server of a website.

¹<https://www.python.org>

Program 4.1: Extraction of the <title> tag from a website. This process is done with lxml and CSS selectors.

```
1 @staticmethod
2 def extract_data_by_css(html, sel, base_url):
3     """
4     extracting data using CSS selectors
5     :param html: html
6     :param sel: selector, for example "title"
7     :param base_url: base_url of the website to scrape
8     :return: content of the selector
9     """
10    try:
11        tree = lxml.html.fromstring(html)
12        selector = tree.cssselect(sel)[0]
13        sel_content = selector.text_content()
14        return sel_content
15    except exceptions.Exception as e:
16        logfile.write("!ERROR! FUNCTION: extract_data_by_css " + " URL:
    " + base_url + ", " + "\n")
```

4.3.2 Crawling Selection

Before the actual crawling process can happen, a selection of properties the crawler should pull, have to be defined. These properties can be separated into three groups:

- general information,
- design aspects,
- and technical aspects.

4.3.3 Scraping A Web Page

With the downloaded HTML page, the extraction of data from the website can be processed. As mentioned in Chapter 3, BeautifulSoup is a popular web scraping module that helps to parse a web page and to navigate through its content. After parsing the downloaded HTML into a soup document, BeautifulSoup can navigate through elements using the *find()* method.

Another library for scraping websites is lxml [24], which is a tool for processing XML and HTML with Python. Lxml has the advantage of extracting data at a swift pace. To scrape the title <title> of a website with lxml, this approach uses the CSS selector (see Program 4.1). With CSS selectors, data from the websites can be found and extracted. In addition to the title of the website, the general aspect, as well as the design and technical aspects can be obtained from the web page too.

4.3.4 General Information

To retrieve basic facts about the websites and be able to further classify them, one functionality of the crawler can be the pulling general information about the websites. Such basic, but valuable information is, for example, the title, the popularity and the category of each site. Before crawling general information, it is necessary to get a uniform convention of all website URLs. Therefore the URLs need to be simplified. This is done in a process, where the URL protocol and anchors are removed and what remains is the base URL of the website (for example *orf.at*). Nevertheless, for the sake of completeness, the full URL of the website is also stored.

As mentioned before, titles can be extracted easily by searching through the HTML Dom with lxml. To get the categorization and the popularity of the website, some further steps are necessary. The service of categorizing websites for example into *News*, *Entertainment* or *Shopping* is provided by the *bluecoat sitereview* [17]. Bluecoat is a website, which provides the service of assigning website categories by entering the required URL. The approach automatically sends requests to the Bluecoat site, retrieves the categorization of it and stores it in the database (see Program 4.2). *Orf.at* for example got the categories *News and Media* assigned.

Popularity and traffic ranking of websites can be fetched from Alexa, which is a common way to measure the approval rate. The algorithm for this specific ranking is based on reach and page views and is recorded from users, which have installed the Alexa toolbar [12]. By entering a URL, Alexa provides an online XML, where the popularity rank can be read out (see Program 4.3).

Classifying the sites with categories and popularity is fundamental to be able to sort the websites by groups.

4.3.5 Technical Aspects

Most of the technical properties cannot be entirely obtained with the crawling process itself. Although, with the right procedure, aspects such as web servers, programming language, analysis tools, markup language, image formats, character encoding, JavaScript frameworks, web frameworks, font scripts, site elements and content management systems can be detected. BuiltWith [18] and W3Techs [32] are providing technology lookups for websites, where several technical aspects can be extracted and stored in the database. Unfortunately, these technologies do not give a lot of insights into the actual implementation and collecting process. The following list gives a short overview of the crawled properties containing a short description:

- **Web Servers:** Web servers are services which ensure the delivery of documents to a client. Some services ensure the encryption of the used web server.

Program 4.2: Extraction of the website categories from Bluecoat.

```

1 @staticmethod
2 def get_category(base_url):
3     """
4     get the category of the website
5     :param base_url:
6     :return: category array
7     """
8     url = 'http://sitereview.bluecoat.com/rest/categorization'
9     useragent = {
10         'User-Agent': 'Mozilla/5.0',
11     }
12
13     payload = {"url": base_url}
14     print payload
15     req = requests.post(
16         url,
17         headers=useragent,
18         data=payload
19     )
20     response = json.loads(req.content)
21     if req.status_code == 200:
22         try:
23             category = bs4.BeautifulSoup(response["categorization"], "
24             lxml").get_text()
25             categories = category.split("/")
26             return categories
27         except exceptions.Exception as e:
28             logfile.write("!ERROR! FUNCTION: get_category " + " URL: " +
29                 base_url + ", " + "\n")

```

- **Programming Languages:** The aspect *programming languages* contains client-side programming languages as well as server-side web programming languages. A complete list of all programming languages containing a short description can be found on *medium.com*².
- **Analysis Tools:** Analysis tools describes the usage of traffic analysis tools for websites. The term traffic analysis tools characterize a variety of analytical methods and procedures, which support different traffic analyses. Further information and a complete list of all tools can be found on *tools.seobook.com*³.
- **Markup Languages:** The two most important markup languages for structuring data on websites are HTML and XHTML. After XHTML the last version of HTML, HTML5 was developed. In the latest ver-

²<https://medium.com/web-development-zone/a-complete-list-of-computer-programming-languages-1d8bc5a891f>

³<http://tools.seobook.com/analytics-tools>

Program 4.3: Alexa Popularity and Traffic Ranking [13].

```

1 <ALEXA VER="0.9" URL="orf.at/" HOME="0" AID="" IDN="orf.at/">
2 <RLS PREFIX="http://" more="0">
3 <RL HREF="www.zdf.de/" TITLE="Zweites Deutsches Fernsehen"/>
4 <RL HREF="www.rtl.de/" TITLE="RTL"/>
5 <RL HREF="www.news.at/" TITLE="NEWS Online"/>
6 <RL HREF="www.dsf.de/" TITLE="DSF"/>
7 <RL HREF="www.3sat.de/" TITLE="3sat"/>
8 <RL HREF="www.wirtschaftsblatt.at/" TITLE="Wirtschaftsblatt:Online"
9 />
9 <RL HREF="www.wienerzeitung.at/" TITLE="Wiener Zeitung - Die
10 Offizielle österreichische Tageszeitung"/>
10 <RL HREF="www.wdr.de/" TITLE="Westdeutscher Rundfunk"/>
11 <RL HREF="www.vox.de/" TITLE="Vox"/>
12 <RL HREF="www.zamg.ac.at/" TITLE="Austrian Central Institute for
13 Meteorology and Geodynamics"/>
13 </RLS>
14 <SD TITLE="A" FLAGS="" HOST="orf.at">
15 <TITLE TEXT="ORF ON"/>
16 </SD>
17 <SD>
18 <POPULARITY URL="orf.at/" TEXT="1410" SOURCE="panel"/>
19 <REACH RANK="1696"/>
20 <RANK DELTA="-26"/>
21 <COUNTRY CODE="AT" NAME="Austria" RANK="7"/>
22 </SD>
23 </ALEXA>

```

sion, many extensions and features were added and the problems with further versions were solved. Further information about the different markup languages can be found on *w3.org*⁴.

- **Image Formats:** The most popular web image formats for websites are PNG, JPEG, GIF and SVG. The assets and drawbacks of each format are dependent on its usage.
- **Character Encodings:** The most common character encoding type is UTF-8. Other character encodings can be found on *w3schools.com*⁵.
- **JavaScript Frameworks:** There are a lot of Open Source and commercial JavaScript frameworks available. A listing of all existing frameworks can be looked up on *javascripting.com*⁶.
- **Web Frameworks:** The most used web framework is *Twitter Bootstrap*, which is popular for its mobile-friendliness. For further informa-

⁴<http://www.w3.org/TR/>

⁵<https://www.w3schools.com/charsets/default.asp>

⁶<https://www.javascripting.com>

tion, the website *bestwebframeworks.com*⁷ provides knowledge about the different web frameworks.

- **Font Scripts:** Font scripts are scripts, which provide fonts without font-face embedding and without the effort of loading the fonts to the server.
- **Site Elements:** For this approach, the term site elements includes inline CSS, external CSS, compressions, cookies, ETags, HTTP/2, IPv6, HTTP Strict Transport Security, framesets and some other elements, which occur on the site.
- **Content Management Systems:** content management systems are software, which support the creation of online content. The different types of content management systems can be read on *cmssritic.com*⁸.
- **Mobile Optimization:** The aspect mobile optimization describes the distribution of adaptive and responsive websites. The actual process of fetching this website aspect will be described in the following paragraph.

Mobile Optimization

The technical aspect *mobile optimization* was not pulled from an external site but was implemented through different processes. From pulling the HTML Dom and the CSS stylesheets, it is possible to identify the mobile optimization of the websites. More specifically, it can identify if the website is responsive, adaptive or not mobile optimized. One of the processes was to index the website with a mobile user agent and to wait for the response to extract. If the response status is not failing and the base URL of the website changed, for example, starts now with *mobile.* or *m.*, then the website is most probable an adaptive one. The viewport of the website can also be examined in this way. If the viewport meta tag includes *width=device-width* the website is most likely responsive. Another process is to pass the stylesheets and search for *@media* tags or search the external embedded stylesheets for bootstrap or other mobile libraries.

4.3.6 Design Aspects

For this approach the crawled design aspect should be held to the minimum. Due to crawling restrictions, the included aspects are fonts, text, and background colors. CSS tags from stylesheets and HTML files should be filtered, analyzed and stored. When focused on the design aspects *fonts* and *colors*, it is necessary to pull inline CSS-styles and links to stylesheets from the HTML files. To retrieve the stylesheets, the links have to be converted first,

⁷<http://www.bestwebframeworks.com>

⁸<https://www.cmscritic.com/dir/cms>

due to the fact that the links can be absolute, relative or in reference to other URLs.

Potentially emerging conflicts of this crawling process could be that stylesheets contain multiple CSS tags, although they are not used in the HTML at all. The consequence of this problem would be the distortion of the result. To minimize the risk, the CSS tags will be parsed and the unused tags removed. To fulfill this task the tool *mincss* [26] is used, which ensures the above process does not happen. The next step is to actually scrape the wanted CSS properties from the cleared CSS file. This process is done with the Python library *cssutils* [19]. *Cssutils* defines a rule set, which enables a developer to choose a CSS selector and extract its value. All stylesheets and inline styles are scanned for *font-families*, *background-colors* and *colors*.

After extracting these aspects, a uniform convention needs to be applied for later comparisons. Most of the time font-families consist of more than one font and therefore have to be separated. Thus, only the first font is counted. Also, all fonts are transformed lowercase and all quotes are removed. For colors, it is essential to bring all into the same format. Hexadecimal, RGB, RGBA and browser color names are extracted and need to be changed into hexadecimal.

4.3.7 Crawling Process Improvements

To document any improvements, a timestamp will be appended to every aspect that is likely to change over time. Such properties can be found in design as well as in technical aspects. The advantage of this is to specify the crawling time of each aspect, and to detect if its value increased or decreased over a specific time range.

The process in this project was to select a data set from the categories: global top 1000, Austrian top 1000, design or random, and then search for the correct timestamp in the database. Every entry with the actual timestamp, which is not crawled yet, will be added to the crawling list. Following, general information, technical aspects and design aspects are crawled for each website URL on the crawling list. To do this faster, the whole mechanism is multithreaded. Up to four websites are crawled simultaneously, which increases the speed tremendously.

4.3.8 Crawling Problems

One of the main issues which occurred during the project was the problem of being blocked due to crawled timing issues. Especially, when the process is multithreaded, the crawler is indexing websites in a fast pace and this has the effect of being frequently blocked for some time. To solve this problem, it is necessary to try and test different timing intervals for the crawler and to implement timeouts between each different process. This solution is causing

a deceleration of the actual crawler but is one of the simplest and fastest ways to eliminate the issue.

4.4 Data Sets

The essential key for obtaining a correct web analysis is the data set. A data set defines a set of URLs, which forms a list of websites for the crawler to index. Especially for the evaluation of the trend results, it is necessary to prepare useful selections of websites. Just taking random URLs is not enough, there needs to be different categorized data sets. Before creating data sets, there are several issues to deal with. The question, what each data set should represent is crucial. Data sets with international, national or randomly picked URLs will retrieve different results. Thus, an order by area, category or traffic rank should be achieved. Focused on this structure, the data sets for the approach are:

- top 1000 global websites,
- top 1000 Austrian websites,
- 1000 websites of the category design,
- and 1000 randomly crawled websites.

For collecting the different data sets, the crawler serves the purpose of gathering website URLs from web ranking websites. Such web ranking pages, for example, provide the top websites with a categorization by topic. To implement the so-called *link crawler*, Beautiful Soup was used again. The link crawler searches for website URLs on the ranking web pages and following, is adding the links to the appropriate categorized data set. The ranking websites are crawled for each website and additionally, each URL is added to the categorized data set. Link crawlers often have to deal with the issue of spider traps. To minimize this risk, the crawler checks the link depth to avoid infinite loops.

4.4.1 Top 1000 Global

The first idea was to use the Alexa topsites [12] to retrieve the top 500 global websites. When the implementation started (October 2016), Alexa's service of providing the top 500 international websites was still free and its information of the top websites was gathered through the link crawler. Unfortunately, Alexa decided in February 2017 to change its service into a paid one and to provide only the top 50 websites for free. This caused the necessity of searching for another service which is provided for free. After intensive research, Majestic Million [25] was found as a plausible compensation (see Figure 4.1). Overall, *Majestic Million* has over 1.1 billion websites crawled and calculates its ranking from referring subnetworks. From this website, the first 1000 website URLs are indexed and stored.



















Position	+/-	Domain	TLD	TLD-Ranking	Verweisende Subnetze	Verweisende IPs
1		google.com	✳ 	1	409.208 57 	2.613.943 336 
2		facebook.com	✳ 	2	401.996 173 	2.762.769 878 
3		youtube.com	✳ 	3	367.747 89 	2.227.596 824 
4		twitter.com	✳ 	4	362.880 159 	2.323.228 854 
5		microsoft.com	✳ 	5	264.128 51 	847.415 582 
6		wikipedia.org	✳ 	1	256.601 157 	1.134.328 276 

Figure 4.1: Majestic Million Global [25]

4.4.2 Top 1000 Austria

Similar to the crawling process of the top international websites, the top 500 websites of Austria were crawled from Alexa first. Alexa provides also subcategories, which enables users to look into the most popular websites from each country. When switching to Majestic Million, there was a small change, because Majestic provides not the most popular websites for Austrian people, but shows the top Austrian websites. This selection derives the benefit of retrieving a better comparison between international and Austrian website statistics. Furthermore, the amount of websites was also changed. While from Alexa only the top 500 websites were crawled, from Majestic Million the top 1000 Austrian web pages are fetched.

4.4.3 Top 1000 Design

To retrieve the top 1000 design websites, Alexa was used again to find the top sites. Besides the countries, Alexa also divides its top sites by categories as *Computers*, *Health*, *News* and many more. These categories then again have subcategories. One of these categories is about *Arts* and a subcategory of it is *Design*. Before the service was monetized, the websites of the subcategory *Design* were added to the data set. Afterwards, the search for design websites, began once again.

The website *Awwwards.com* [14] offers a useful replacement for the Alexa sites. Awwwards honors web designers, web developers and web agencies for their websites. Its goal is to promote and recognize the best web designs.

For collecting the design data set, 1000 Awwwards Nominees are crawled frequently. Due to the structure of the Awwwards website, this crawling process was not possible with the implemented crawler. The Awwwards website consists of an infinite scrolling page, where the Nominees are reloaded if a scrolling event is triggered. Therefore an ordinary crawler is only able to index the first elements that are visible without scrolling. To solve this prob-

Program 4.4: The process of fetching the randomly crawled websites from Google.

```

1 mongo = mongocache.MongoCache()
2
3 # dictionary which returns list of words
4 word_site = "http://svnweb.freebsd.org/csrg/share/dict/words?view=co&
   content-type=text/plain"
5 delimiter= ' '
6
7 response = requests.get(word_site)
8 WORDS = response.content.splitlines()
9 for i in range(50):
10     # randomly select one word
11     word = random.choice(WORDS)
12     # store first 20 google entries
13     for url in search(word, stop=20):
14         # save it into db, with base URL and alexa rank
15         base_url = crawler.Crawler.get_base_URL(url)
16         alexa = crawler.Crawler.popularity_rank(base_url)
17         mongo.__setitem__('', base_url, alexa, '')
18         print('%s%s%s' % (alexa, delimiter, base_url))

```

lem the *Selenium WebDriver*⁹ was implemented, which enables the crawler to open a URL in a browser and imitate real user scrolling behavior. With this process, the websites from Awwwards can be reloaded and crawled.

4.4.4 1000 Randomly Selected URLs

The idea behind this randomly selected data set, was to have a comparison between the four different data sets. Moreover, this project wanted to see if the collections actually have different results and meanings. To collect the URLs for the random data set, words are selected from a dictionary by chance and inserted into the Google search. The first 20 URLs of the search result are picked and stored in the database. For each timestamp, this process is repeated until 1000 randomly chosen websites are collected (see Program 4.4).

Crawling Interval

To detect value changes and to be able to further expect web trends, the data sets have to be updated regularly. The majestic website rank is refreshed frequently and in addition, the data sets need to be updated and extended. New websites might have been added or the rank of some others may drop below the top 1000. Considering the fact that every day new websites join

⁹<http://selenium-python.readthedocs.io>

the nominees of the Awwwards website, the data set for the category design needs to be updated too. For better comparisons, the random websites are newly crawled from Google at every interval. The update interval for the data sets is set to one month, which means every 30 days new URLs are added to the data set. This process might be confusing, due to the fact that the data set also consists of URLs that are not accurate anymore, but were crawled months before. Therefore a timestamp, which represents the date of the crawling process, is stored with every website in the database. As time goes by, it is possible that a website was assigned more than one timestamp. To circumvent wrong timestamp issues, not every website, but all technical and design aspects are retrieving the correct timestamp.

4.5 Data

To store the data from the crawling processes, for the implemented approach *PyMongo*¹⁰ is used. PyMongo is a Python distribution containing tools for working with MongoDB¹¹. MongoDB is a schemaless, NoSQL database, which ensures the possibility of storing data with changing data types. One advantage over a relational database is that the schema must not be defined before adding data and therefore agile development is ensured. The project was a work in progress, thus, also the schema of the database sometimes needed to be changed, which is no problem for a noSQL database like MongoDB. Another convincing benefit of MongoDB is the simplicity to scale the database and to work with big data. During the project, the usage of PyMongo was never an issue, as it was able to simplify the storage process in many ways.

4.5.1 Data Exploitation

The next structural step is to crawl all aspects from the websites of each data set and additionally store the data with timestamps in a database. Most data, however, have to be processed first. For instance, such processes are preparing the design aspects, which include fonts and colors, in a uniform format. The retrieved fonts can be in uppercase, lowercase or with space characters in between and have to be standardized. The same issue arises with colors, which can occur in a hexadecimal, RGB, RGBA, HSL, HSLA format or in cross-browser color names (see Table 4.1).

Before being able to visualize the data some basic methods have to be implemented. The visualization should represent

- the crawled aspects with their top elements,
- of each data set,

¹⁰<https://api.mongodb.com/python/current/>

¹¹<https://www.mongodb.com>

Table 4.1: Color conversion of the fetched CSS stylesheet data.

darkgrey	→	#a9a9a9
rgb(169,169,169)	→	#a9a9a9
rgba(169,169,169,1)	→	#a9a9a9
hsl(0, 0%, 66%)	→	#a9a9a9
hsla(0, 0%, 66%,1)	→	#a9a9a9

- from different timestamps.

Thus, it is necessary to create a process which pulls the aspect to represent, with its elements from a data set, with a specific timestamp. For the calculation of the percentages, all elements need to be counted. Furthermore, to retrieve the top values of each aspect, the elements are sorted in descending order. For example, the aspect to visualize is *web servers*, the data set is *global* and the for the timestamp *2017-05-01* is selected. The process will return the top elements of the *web servers*, with its usage share in a descending order (see Figure 4.2).

To be able to show the actual trending elements of each aspect, a method for comparing the values in a specific time range, with two timestamps, is created. This method returns the developments of each aspect element. Other helper methods had to be implemented, for example, a method for retrieving the number of all websites with a specific timestamp, or all websites from one data set with a specific timestamp.

Correlations

The last important data exploitation factor is calculating the correlations between different aspects and its values. For this method, the comparison of two database tables is performed. It is extremely interesting to observe, which interdependence different aspects perform. Some correlations might be predictable, for example the *Content Management System TYPO3* and the *programming language PHP* with a correlation of 100%. But the unexpected ones might be even more interesting. For example, when a specific *color* can be detected, a special *web framework* is used, too.

The problem with missing data of the crawled aspects need to be considered and accordingly, only data, where the aspect for the first table is not empty, is taken into account. Nevertheless, for this process, it is necessary to be aware of the fact that some aspects can not be detected for all websites, and therefore the result sometimes might be vulnerable to errors.

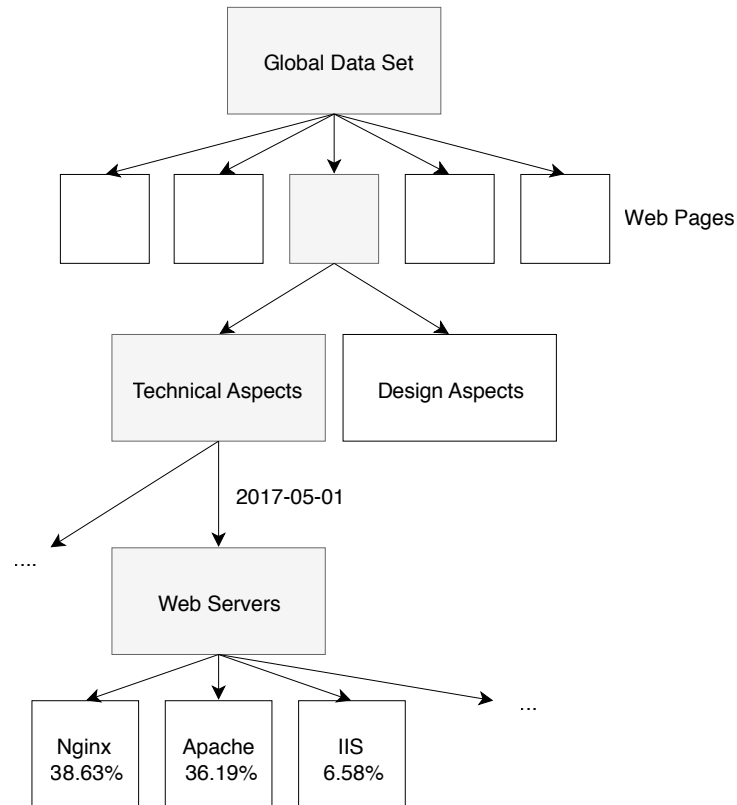


Figure 4.2: The Structure of the Data Representation.

4.6 Data Representation

In this solution, the representation of the data should primarily contain the different data sets with all technical and design aspects. Time intervals are necessary for the comparison and the visibility of trend changes and thus, have to be apparent on the representation graphs. Increased and decreased aspect values can be distinguished through the crawling time intervals and will help measure trends. Additionally, the correlations between two different aspects should also be represented. To represent the data a web application has to be built. For creating a Python-driven web application a fairly common way is to use a Python Web Framework. Frequent representatives of such Web Frameworks are *Django*, *Flask*, *web2py* and *Pyramid*. Each of these frameworks provides the usage of templates, for easily injecting dynamic content into HTML pages. For representing this approach, Flask is used.

4.6.1 Flask

*Flask*¹² is a Python-driven microframework based on *Werkzeug*¹³, a WSGI (Web Server Gateway Interface) utility library, and *Jinja2*¹⁴, a template engine. In comparison to other Python web frameworks, Flask does not provide existing components but supports the opportunity to embed libraries. Thus, the core functionalities of Flask are kept simple and compact, but can, however, be extended easily [20].

Flask provides a simple built-in server, and also a debug mode, which enables a real-time reload if code changes. Additionally, it uses routing to bind functions to URLs and to make certain parts of the URL dynamic and moreover attach rules [20].

The static files for the web application, usually CSS and JavaScript files, can be implemented through URL embedment too. With the render function of Flask, the cumbersome task of generating HTML from within Python will be improved immensely. Flask is using the Jinja2 template engine to do that automatically. For passing data to the HTML documents, developers can deliver data as parameters to these render functions, which can, in turn, be accessed from the HTML through variables, extended with filters and processed through methods [20].

4.6.2 Representation Structure

The idea of representing all the data was to structure the pages into the different data sets *top 1000 global*, *top 1000 Austrian*, *1000 of category design* and *1000 randomly crawled*.

For each data set, all the different aspects and their top ten elements are listed and visualized. Additionally, these elements get two timestamps and show how the values of each element changed despite these dates. Instead of putting all data in a scrollable website format, the data set, the desired aspect to show and the crawling time range is selectable via buttons. This enables the user better usability and faster finding. Moreover, it is, of course, possible to scroll down the page without selecting these items and just obtaining an overview.

To visualize trends a separate page is provided, to look through the top ten value increasing elements, of each aspect, from each data set. Similar to the web page with the top elements, on the trend page also the data set, the aspect and the timestamps are able to filter.

The correlations of two database tables are represented onto an own page. All data sets are provided at once and the user is able to select one aspect, one element of this aspect and additionally one other aspect, for the

¹²<http://flask.pocoo.org>

¹³<http://werkzeug.pocoo.org>

¹⁴<http://jinja.pocoo.org>

Correlations

Select the two categories and properties you want to compare

CHARACTER_ENCODING UTF-8 WEB_SERVERS SEND

CAT1	PROP1	CAT2	PROP2	%
character_encoding	UTF-8	web_servers	Apache	38.57 %
character_encoding	UTF-8	web_servers	Nginx	34.69 %
character_encoding	UTF-8	web_servers	Microsoft-IIS	7.17 %
character_encoding	UTF-8	web_servers	IIS	6.40 %
character_encoding	UTF-8	web_servers	Tomcat	1.16 %
character_encoding	UTF-8	web_servers	Google Servers	1.16 %
character_encoding	UTF-8	web_servers	OpenResty	0.87 %
character_encoding	UTF-8	web_servers	LiteSpeed	0.78 %
character_encoding	UTF-8	web_servers	Erlang	0.78 %
character_encoding	UTF-8	web_servers	Cowboy	0.48 %

Figure 4.3: The approach implementation of the correlation between two website aspects. The first line shows the selection of the different website aspects.

elements to compare. The output of this function is a table with correlations between the element of the first selected aspect and all the other elements from the second aspect, with a maximum of 10 entries (see Figure 4.3).

4.6.3 Information Visualization

The most important visualization goal of the project is to achieve a high informational despite with this amount of data. Primarily the design is not as important as the information and the data itself.

A common way to classify an information visualization is to analyze its data dimensions. The data dimensions are describing the number of information types, which should be visualized in the diagram [8]. Due to the structural set-up of the representation website, four different dimensions can be found:

- one dimension for the data sets (e.g. global data set),
- one dimension for the aspects (e.g. web servers),
- one for the elements of the aspects with its values (e.g. Nginx: 38.63%, Apache: 36.19%,...),
- and one dimension for the time (e.g. 2017-05-01).

Additionally, the user intent of the information visualization can be

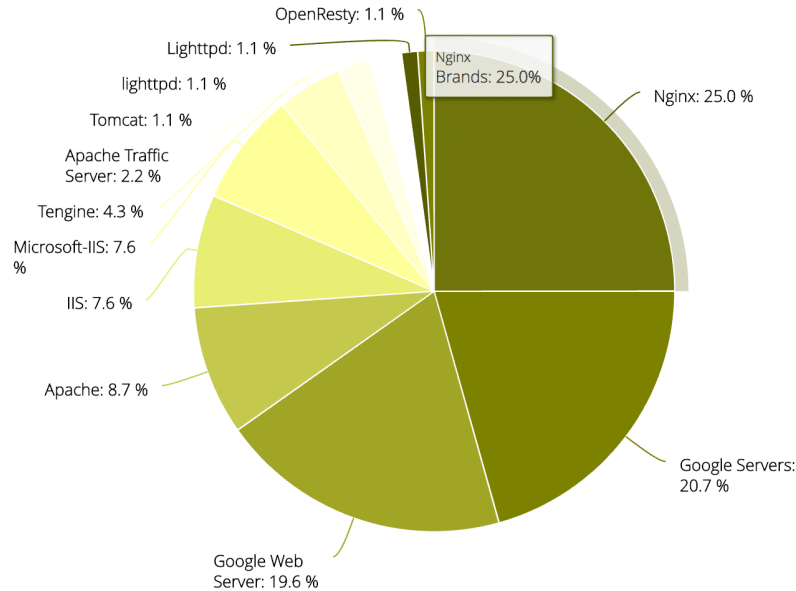


Figure 4.4: The first data representation with a pie chart.

stated. For this approach, the user is able to *select and focus* different types of data sets, aspects and timestamps, *reconfigure filters*, *explore* the data and the outcome of different filter configurations.

Through its presentational content, despite its possibility to select different filters (data sets, aspects, timestamps), the visualization can be declared as a Hybrid of Explanatory and Exploratory [8]. This means that the visualization includes not only reading but also exploring parts.

Pie Chart

Finding the best visualization was a laborious process. The first attempt was to try a pie chart for representing the different elements and values of each aspect. The clearness of the data was ensured and the information was easy to understand. The problem with this type of visualization was the fact, that different timing values are hard to show in pie charts (see Figure 4.4). After trying different variations of the pie charts to include the time range, a bar chart was found to be a better way to visualize this data.

Bar Chart

Upon implementation, it turned out the selection of the bar chart was a good choice for this data. Bar charts are useful in representing data with

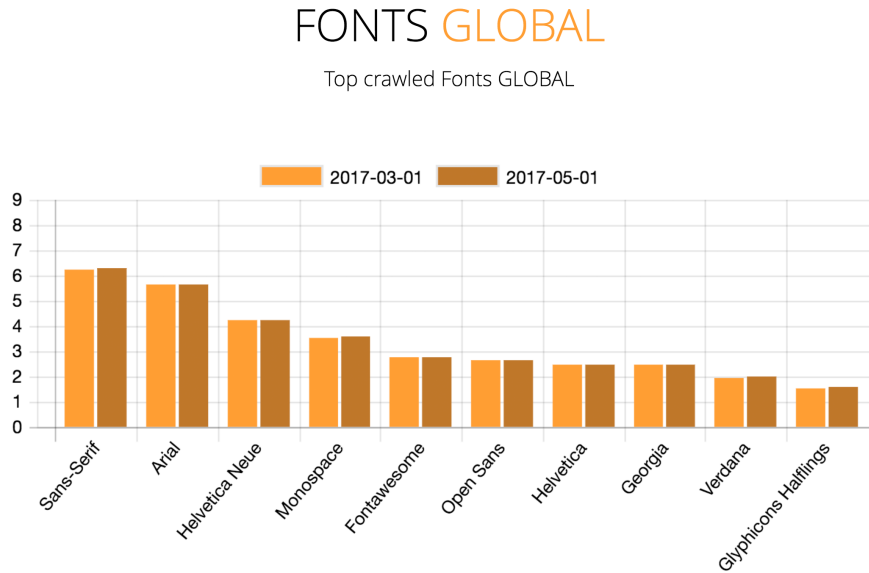


Figure 4.5: The current data representation with a bar chart.

the same categories and moreover are able to show significant amounts of data in a compact and manageable way. In particular, the timing issue is no problem anymore. Two different graphs side by side serve to show the various crawling times and with that the value changes of each element (see Figure 4.5). To be able to compare one aspect of multiple data sets, the maximum value of the bar graph is adjusted for all data sets with the same aspect.

Chart.js

The actual implementation of the bar graph is programmed with Chart.js¹⁵, an open source library for creating various charts and graphs. With its extensible configuration and its different graph types, it is easy to use and simple to embed.

¹⁵<http://www.chartjs.org>

Chapter 5

Implementation Results

To evaluate the approach, it is important to recap the goals of the project. Web developers, web designers as well as business owners should be supported in extending their knowledge about web changes. With only a few clicks the interested parties should retrieve a web data analysis of different website aspects, separated into different data sets. Additionally, with the automatic creation of web analyses, categorized by different data sets, the issues with personal opinions in web analyses should be reduced. In this chapter, the crawling outcome of the different datasets will be analyzed and compared.

5.1 Outcome

An important role of the this project's result is the selection of the different data sets, the websites taken into account, and the change they will have. Therefore, the decision of which data set category to use was carefully considered. This section will compare the different crawled aspects:

- fonts,
- font scripts,
- colors,
- image formats,
- programming languages,
- markup languages,
- JavaScript frameworks,
- character encoding,
- site elements,
- web servers,
- mobile optimization,
- analysis tools,

- web frameworks,
- and content management systems.

These aspects will be separated into the four different data sets. As a short recap, these websites collections are:

- top global websites from Majestic Million,
- top Austrian websites from Majestic Million,
- design websites from Awwwards,
- and randomly crawled websites from Google.

For the comparison between the data sets the crawling timestamp 2017-05-01 is used. For the trend results, the timestamps

- 2017-03-10,
- and 2017-05-01

are analyzed. Moreover, correlations of different aspects are detected and described. Due to the enormous amount of data, only a few correlations can be evaluated more precisely. These correlations do not include missing data of the first argument, which means that only aspects where this value is represented are included in the calculation.

The amount of data is constantly increasing, because of the remaining crawling processes taking place, which means the correlations are also changing. Therefore, this correlation analyses includes the crawled data up until May 14, 2017.

5.1.1 Fonts

When considering all of the top ten fonts from the different data sets, a short overview is enough to see the first similarities. As mentioned before and as a short reminder, from the crawled font-families, only the first of the specified font is stored. Thus, for the following example just *Arial* will be stored into the database:

```
1 body {  
2   font-family: Arial, Helvetica, Sans-Serif;  
3 }
```

The top two fonts, *Arial* and *Sans Serif*, are represented in equal positions at the data sets global, Austrian and random. The largest similarities show the top global websites from Majestic Million and the randomly crawled websites from Google, with the fonts *Helvetica Neue*, *Monospace*, *Verdana*, *Open Sans*, *Helvetica*, *Fontawesome* and *Georgia* on position three to nine, with few deviations. This may be caused by the fact that for the crawling process of the random websites, words from a dictionary are inserted into the Google search, and the first twenty Google results are crawled. Therefore both data sets, the random as well as the global, mainly consist

of high ranked and popular websites and this could be the reason for the similar results.

The data set of the top websites in Austria also presents some affinities, but with some interesting outliers. It seems that Austrian web developers like to embed *Glyphicons Halflings*, which is in the eighth position in the top ten. Just one rank before is the rather unknown font *Cuprum* located. This font is a remarkably unique one and found 96 times in the top 1000 Austrian websites. On the other hand, this font was found just once in the data set design, and never found on the top 1000 global or random websites. Additionally, the design data from Awwwards represents the websites, which differ the most in regards to the aspect fonts. The top font of all other sets, *Arial*, is only in position six, with a rather low percentage of 2.5%. In addition to *Arial*, among the top ten are also some new fonts that are not represented in any other set. These includes *Roboto*, *Lato* and *Montserrat*. The reason for this deviant behavior of the design websites in relation to the usage of fonts might occur from the circumstance that the Awwwards nominees are likely representing the newest design trends. The design data sets might give valuable signposts for current font trends. A detailed representation of the different used fonts with its typesets can be seen in Figure 5.1.

Font Correlations

Concerning all fonts, one especially stuck out, the rather unknown *Cuprum*. Cuprum is highly represented in the Austrian data set, and therefore, a closer examination of other correlations to this font is performed. Especially noteworthy is the fact that every time the Cuprum font is embedded, which is exactly 96 times for the selected timestamp, the web server Jetty is detected, too. Additionally, the JavaScript frameworks Modernizr and jQuery also occurred every time. With further investigations, the reason for this conspicuous behavior was found. All these websites have one thing in common: they are all websites, which do not provide any real content, but the domains for this websites are for sale. Additionally, all these websites have the same seller, same layout, same font style, web server, and the same JavaScript frameworks. The question, why these websites are having so much referring subnetworks that they are listed above the top 1000 Austrian websites, might have the reason that this web pages also provides links to other domains-on-sale.

Font Trends

In this paragraph, the top ten value increase of fonts over the time of two months are observed and described. For the global data set, the biggest increase in usage represents *Monospace* with 0.1%. As the name suggests, *Monospace* is a fixed-width font, where all letters and characters have the

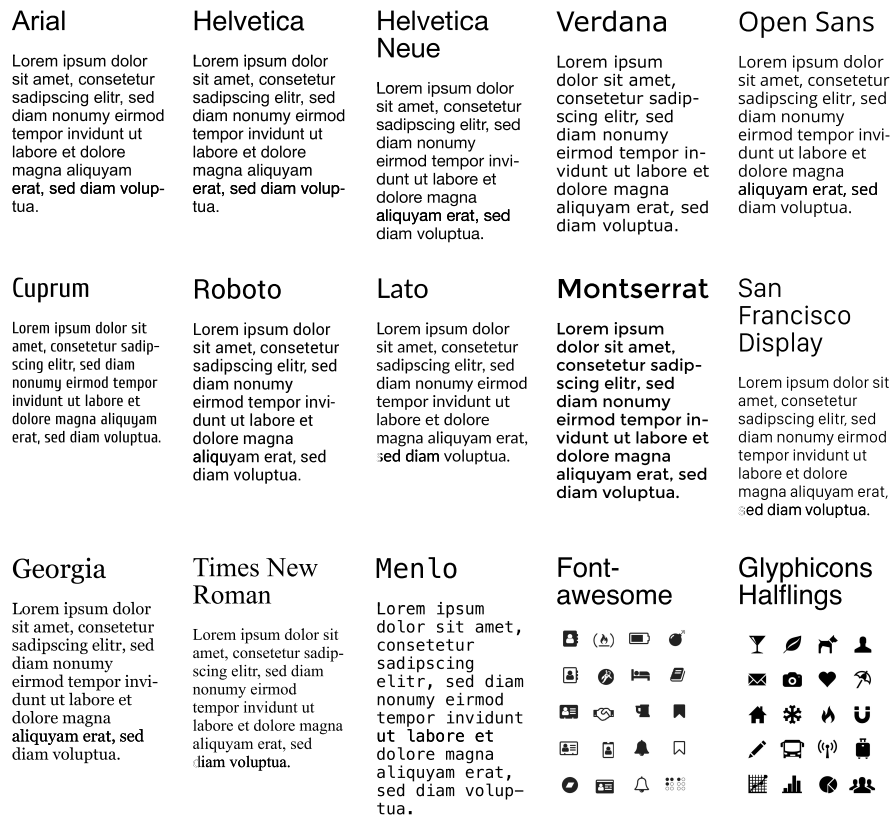


Figure 5.1: Shown are the most used and trending fonts with its typesets.

same width. If this font is set in the stylesheet, the used browser will choose its own predefined *Monospace* font. The font with the second highest embedment increase for the global data set is *Serif* with 0.09%, which uses the predefined serif font of the browser again. In the third place the *Dashicons* with 0.06% are found, which are WordPress icons. Other notable increases in the global data set are: *Sans-Serif*, *Open Sans*, *Verdana* and *San Francisco Pro Display*. To note, *San Francisco Pro Display* is the system font for macOS and iOS. For the Austrian data set the font with the most growth is *Fontaweseome* with a rather high increase score of 0.25%. After that, *Helvetica Neue* (0.16%) and *Monospace* (0.15%) are following. *Icomoon*, *Menlo*, *Raleway*, *Helvetica* and *Consolas* are also gaining importance across Austrian websites. Significant increases for the design data set shows the fonts *Sans-Serif* with 0.21%, *Monospace* with 0.19% and in the third place *Helvetica* with 0.11%. Other fonts which also recorded a slight increase are *Open Sans*, *Helvetica Neue* and *Tahoma*. The data set, which demonstrates the highest font value increases is the random data set. This might be the cause

of the selection process, where for each timestamp entirely new websites are crawled. The highest gain represents *Open Sans* with a significant 1.18% increase. In the second position is *Fontawesome*, which increased 0.85%, followed by *Montserrat* with 0.44%. Other fonts, which extended their usage for this collection are *Times New Roman*, *Menlo*, *Raleway*, *Proxima-Nova* and *Roboto*.

To summarize, although there was a variety of different font increasing in usage, some fonts even increased throughout each data set. *Open Sans* retrieved fairly good growth results overall website categories, similar to *Monospace* and *Helvetica*.

5.1.2 Font Scripts

The differences between the data sets concerning font scripts are by far not as significant as for the fonts. All four data sets represent the font script Google Fonts as their number one with a great percentage between 63 and 70. Google Fonts is followed by FontAwesome, which is in the second rank of all website categories with a proportion of 28 to 34. In third place, with only two percent, are Typescript in the global and the random data set, Ionicons in the design data set and Cufon in the Austrian data set. Furthermore, also the font script Glyphicons occurs. Noticeable about the font scripts is the widely spread usage of Google Fonts, which has positioned itself as a clear leader in this area.

Font Script Correlations

Because of the wide usage of the number one crawled font script, the correlation comparison is focused on *Google Fonts*. The correlation between Google Fonts and the web server Apache is 42.62%. In contrary, the correlation with the web server Nginx is a little less, with a percentage of 30.92. These percentages mean, that every time Google Fonts was used on websites, 42.62% also used the web server Apache and 30.92% used Nginx. The thought of comparing the correlation of the two Google services, Google Fonts and Google Analytics was fascinating. Surprisingly, the correlation result was low at just 6.63%. Reversed, when analyzing how often Google Fonts are embedded if Google Analytics is used, the percentage is significantly higher with 30.04. From this research, there were some interesting facts about the correlations: The most popular content management system in combination with Google Fonts is Wordpress and the most used programming language is PHP. The web framework Twitter Bootstrap was also implemented occasionally with Google Fonts with a correlation percentage of 21.84.

Font Script Trends

When looking at the font script value increases, for the Austrian, the design and the random data set some similarities can be found. In comparison to that, the global data set retrieves rather different results. For the global data set the Google Fonts are the only font script, which can record an increase at just 0.89%. For the Austrian data set, however, FontAwesome retrieved pretty good results and could record an increase of 2.52%. The design data set, also represent FontAwesome's development with an increase of 0.45%. Moreover, Typekit, Glyphicons and Ionicons retrieved also slight value improvements for the design set. Quite similar is the behavior for the random collection, where Typekit, Cufon, FontAwesome and Glyphicons extend their usage. In a nutshell, it can be said that especially FontAwesome is expanding its importance for all collections, Typekit and Glyphicons could rise their values for the design and the random data sets.

5.1.3 Colors

The first look at the colors of the data sets, reveals the anomalies that nearly all colors are achromatic, which means the colors are either black, white or grayish. Just the design data set shows two non-achromatic colors above the top ten (see Figure 5.2). The top three colors are identical for each data set:

white: #ffffff,
black: #000000,
gray-tone: #333333.

After these three front-runners, other grayish colors are following. Only when considering the top 20 of each set, also for the global, Austrian and random data set, colors, which are not achromatic are appearing. For the global data set, the two non-achromatic color above the top 20 are #337ab7 (blue-tone) and #ff0000 (red). For the Austrian data set the color #337ab7 (blue-tone) can be found again and furthermore #23527c (blue-tone) can be seen. In the design data set three non-achromatic colors are represented, which are #337ab7 (blue-tone), #23527c (blue-tone) and #ff0000 (red). Despite all other collections, the design websites show most non-achromatic colors, two of these are even above the top ten. Furthermore in the random data set the color #355998 (blue-tone) and #ff0000 (red) can be found.

Color Correlations

For inspecting the correlations of colors, two non-achromatic colors were picked for further examinations, #337ab7 and #23527c. Comparing the correlations with other website aspects, revealed the fact that when these colors occurred, also the percentage of jQuery and furthermore Twitter Bootstrap was pretty high. With further evaluations and searches through the Twitter



Figure 5.2: Most used colors above the top ten of each data set.

Bootstrap stylesheet, exactly these colors were found. As default, the Twitter Bootstrap default link color, text-primary color, background-primary color and the btn-primary color are precisely the blue-tone `#337ab7`. Altogether the color appeared 25 times on the Twitter Bootstrap default CSS. A similar behavior could be noticed for `#23527c`, which was found as default link hover color.

Color Trends

A noticeable trend regarding the colors is that the increasing values are primarily represented by achromatic colors. This color trend evaluation will therefore mainly focus on the few non-achromatic color increases.

The top three color increases of the global data set are all represented by grayish colors. The fourth place, however, is represented by the red-tone `#e3000f`, which record a slight increase of 0.07%. The blue-tone `#005c9e` is also above the top ten increases. For the Austrian data set, only one of these increases is a non-achromatic color, which is the blue-tone `#1a3058` with 0.06%. Most non-achromatic colors above the increases are found on the design and the random data set, with three for each. For the design data set the colors are `#4fb3c9` (light-blue-tone), `#ed496f` (pink-tone) and `#65c583` (green-tone). In contrary, for the random data set the non-achromatic color increases are all blue tones, with the hexadecimal codes `#656e79`, `#00647a` and `#38424d`. In Figure 5.3 all non-achromatic colors of the different data collections are represented.

5.1.4 Image Formats

The occurring image formats are PNG, JPEG (JPG), GIF, SVG, WebP, BMP and ICO. While the most common image format is PNG, with a percentage between 36 and 39, JPEG, follows closely behind with a share between 31 to 36. Especially noteworthy for the third place is the difference between the data sets of global, Austrian and random websites compared



Figure 5.3: This figure shows the non-achromatic color trends of the different data sets.

to the design websites. While the image format GIF achieves between 20 to 24 percent for the global, Austrian and random websites. For the design websites, it only retrieves a share of 11% and is displaced to position four. For this data set, in the third position is the format SVG with 17%. For all other data sets, the share of the SVG format accomplished significantly less. This image format only achieves 9.07% for the global and 7.91% for the random data set. However, SVG retrieves its worst result for the Austrian websites, with a usage of only 2.63%. The reason for this unequal distribution of GIF and SVG, might be that SVG is a file format, which is more modern and it is just recently gaining in importance.

In Figure 5.4 the distribution of the data sets for the top four image formats can be examined even more precisely. One again, the pattern that the design websites show higher values for more modern aspects can be observed. Perhaps the most plausible explanation for this behavior is that these websites are implemented just lately and might use newer technologies and design trends. Moreover, the rather new image format WebP can be found on the global and on the random data set. Additionally, on the random data set the image format BMP is located with a percentage of 0.21. ICO, an image file format used for icons in Microsoft Windows, only occurs in the design data set with a percentage of 0.16.

Image Format Correlations

For evaluating the correlations of the image formats SVG is picked and will be examined carefully. One conspicuous point is that the most used web server in combination with SVG is Nginx with 40.17%. In comparison to that, Apache only reaches 17.09%. This difference might arise from the fact

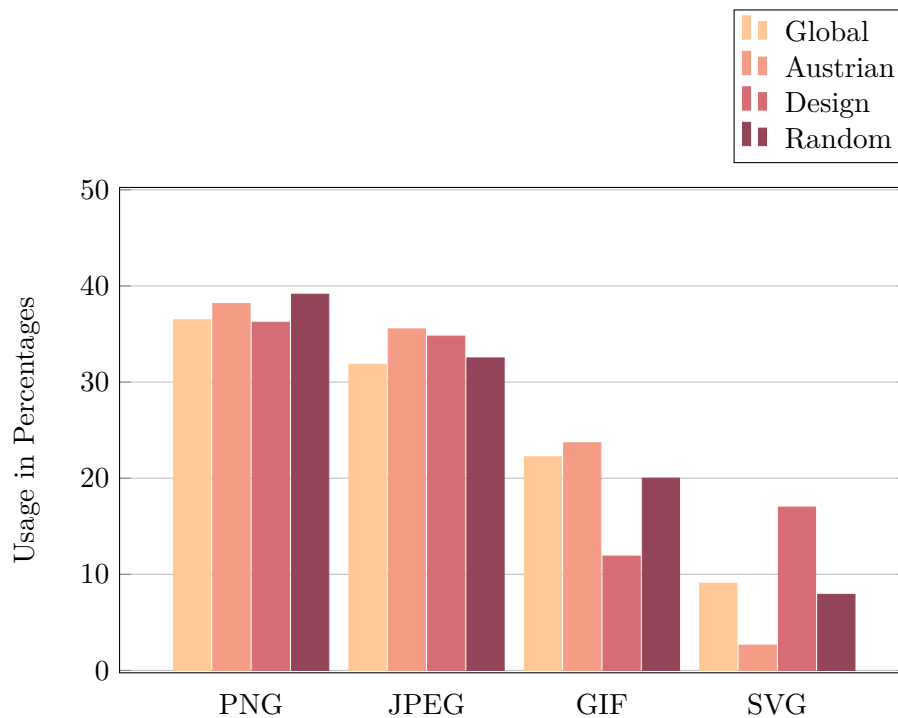


Figure 5.4: Most used image formats regarding the four different website data sets.

that SVG is likely used for modern websites and is gaining in importance lately. The behavior for the usage of Nginx is obtained similarly. Therefore it is not unexpected that these two aspects show correlations. Furthermore, the programming language with the highest correlation with SVG is JavaScript, additionally the JavaScript framework jQuery and the web framework Twitter Bootstrap also retrieved rather high correlation percentages .

Image Format Trends

In relation to the image format trends, the global data set is able to record an increase for the SVG format with slight 0.26% and also for the image format WebP with 0.11%. The common image formats JPEG, PNG and GIF, however, are subject to losses. The Austrian data set only retrieved an increase for the image format GIF, with 6.12%. The image format values for the design data set increased for JPEG with 1.13%, for SVG with 0.64% and for ICO with 0.16%.

Especially noteworthy are the high-value improvements for the random data set. SVG retrieved a significant growth of 5.47%, followed by the rise of GIF with 2.93%. JPEG, WebP and BMP also record slight increases for

this collections.

5.1.5 Programming Language

For the programming language aspect, the *server-side* and *client-side* programming languages are fetched from W3Techs.

Except for the global websites, all data collections are representing PHP as the leader of this category, while JavaScript and Java also retrieved rather good results (see Figure 5.5). The global data set, however, shows JavaScript in the first position with a percentage of 37.75, followed by PHP with 31.29%. The third position of this data set is assigned to Java with a share of 16.89. In the Austrian, design and random data set, PHP was used between 43 and 71% of the time. JavaScript is located in the second position for the design and random data set. Especially significant is in contrary the assignment of the second most used programming language in Austria. For the Austrian crawled websites, Java is even located in the second place and JavaScript is only following. The distribution of the random data set shows similar results, but with Java in the third position. In contrary, when looking at the design data set, Java is only found in position seven, while nodeJS in third place. All other top programming languages above the different data sets are Ruby, Lua, ASP.NET, Perl, Python, ColdFusion, Erlang and CFML.

The distribution of these languages is rather different on all data sets. However, it is noteworthy that Lua and Erlang score pretty well on the design, random and global websites, while they are not occurring above the Austrian top ten at all.

Programming Language Correlations

In consideration of the programming languages, many correlations were found noteworthy. In this section, the correlations between the different programming languages and web servers will be evaluated.

PHP has the highest correlation with the web server Apache (36.5%), followed by HHVM with 29.19%. The correlation with Nginx only retrieves 25.42%. JavaScript obtains nearly the same correlation percentages for Apache and Nginx. In contrast, the correlations for Python and Ruby to Nginx are predominating clearly. Furthermore, for the programming language ASP.NET, IIS is used and for Java the server Jetty is dominating the correlations.

Programming Language Trends

In consideration of all data sets, the programming languages ASP.NET and Python gained on importance:

ASP.NET: 0.04 – 5.33% increase,

Python: 0.12 – 1.64% increase.

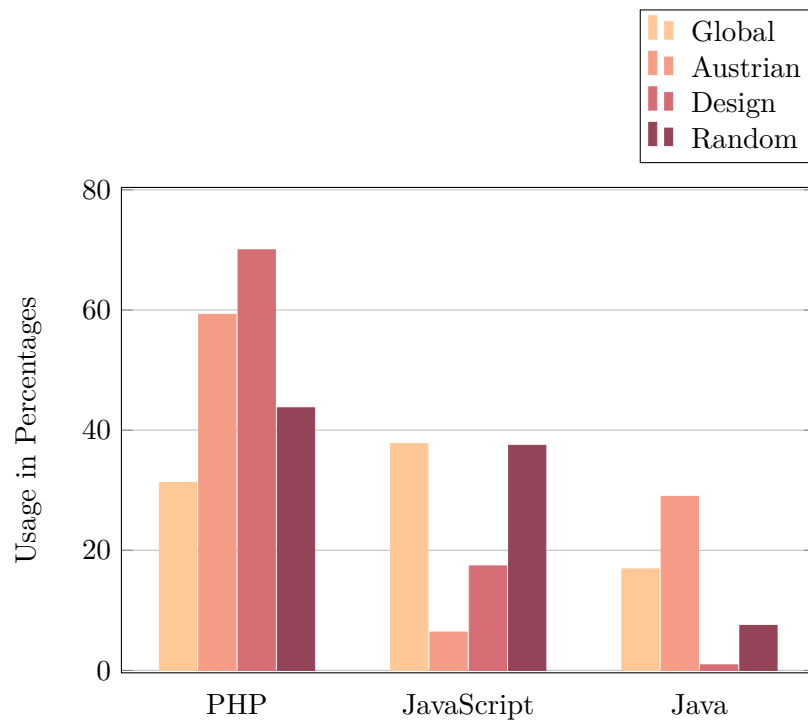


Figure 5.5: Most used programming languages regarding the four different website data sets.

The programming language Perl retrieved an improvement for all but the design data set. Moreover, PHP gained on usage for the global, Austrian and the design websites. JavaScript, however, did obtain an increase for the global, design and random data set. Other programming languages which recorded a rise are Erlang for the global data set, nodeJS for the Austrian and random set, Java for the design and random set and furthermore Ruby and Lua for the randomly selected websites.

5.1.6 Markup Languages

The crawled markup languages can be separated into

- HTML5,
- HTML Transitional,
- XHTML Strict,
- XHTML Transitional,
- XHTML Basic,
- HTML Strict,
- and HTML.

The number one markup language of all data sets is HTML5, with percentages between 54 and 95. HTML5 retrieves the best result for the design websites, all other markup languages are rarely represented with none reaching above 2%. In regard of the global data set, HTML5 is used 83.6%, HTML Transitional reaches 5.46%, XHTML Strict 5.02% and XHTML Transitional 3.36%. Compared to that, XHTML Transitional and HTML Transitional accomplish 8.01 and 3.64 percent for the randomly crawled websites.

In Austria, the markup language XHTML is still strongly represented, with 19.4% for XHTML Transitional and 9.7% for XHTML Strict. HTML5 lists for this data collection its worst result with only 53.8%.

Markup Language Correlations

To observe the correlation of a markup language with other website aspects, the markup language XHTML Transitional is selected. Nowadays, XHTML has faded into the background and HTML5 has received the prominent status as far as markup languages go. Therefore, most of the websites which are still using XHTML Transitional are older and not updated web pages. The highest correlation retrieves XHTML Transitional with the web server Apache with 59.25%. The correlation with other web servers is significantly less, with above 14% for IIS and Nginx.

Regarding the web frameworks, the highest correlation happens between XHTML Transitional and ASP.NET, but the percentage is nevertheless rather low with 7.41%. Twitter Bootstrap and Ruby on Rails, however, retrieve even lower results with above three percent.

The most popular content management system in combination with XHTML Transitional is TYPO3, followed by Joomla and WordPress.

Markup Language Trends

Between the two timestamps, the markup languages of the global and the design data set do not show considerable changes. However, the Austrian data set is recording significant increases regarding to the XHTML Strict, XHTML Transitional and HTML Transitional. In contrary, the random data set retrieved prominent growth for the markup language HTML5.

5.1.7 JavaScript Frameworks

Related to all data sets, the top six JavaScript frameworks are

- **jQuery**: 40 – 50% in usage,
- **Modernizr**: 10 – 16% in usage,
- **RequireJS**: 3 – 15% in usage,
- **Prototype**: 3 – 10% in usage,

Table 5.1: Correlations of JavaScript frameworks with web servers.

JavaScript Framework	Web Server	Correlation
jQuery	Apache	43.78 %
RequireJS	Apache	38.04 %
Modernizr	Apache	40.05 %
jQuery	Nginx	30.98 %
RequireJS	Nginx	29.05 %
Modernizr	Nginx	31.08 %

- **jQuery UI:** 5 – 8% in usage.

The undisputed market leader of the JavaScript frameworks for the crawled websites is jQuery with percentages between 40 and 50. For the global data collection the frameworks RequireJS (15.05%) and Prototype (10.59%) are following, quite equally is the behavior for the random data set. The design data sets shows Modernizr in the second position with 14.4 percent, and RequireJS on the third with 7.6%. All other frameworks are distributed on the different data sets rather similar, with one exception, which is the Austrian collection. For the Austrian data set three JavaScript frameworks occur above the top ten, which are not detected in any other data set. These frameworks are MooTools in the fifth position, Scriptaculous in the eighth position and on position ten PrettyPhoto.

JavaScript Framework Correlations

To observe the correlations of JavaScript frameworks with other website aspects, the popular JavaScript frameworks jQuery, RequireJS, Modernizr, React and AngularJS will be evaluated.

When observing the correlation between JavaScript frameworks and web servers, the results show that jQuery, RequireJS and Modernizr retrieve rather similar results. The descending correlation ranking for these three frameworks represents Apache in the first position, followed by Nginx. Significant noteworthy is that these proved and persistent JavaScript frameworks obtain the highest correlation with the as well persistent web server Apache (see Table 5.1).

JavaScript Framework Trends

For the global data set the JavaScript frameworks jQuery, Lightbox, Underscore and Backbone recorded slight, but constant improvements. In con-

trary, the usage of the JavaScript framework Prototype dropped, while React, jQuery UI, Require and Modernizr stayed unchanged. For the Austrian data collection, quite similar behavior was observed. Lightbox, MooTools, Mustache and React constantly grew, while the Frameworks Prototype and RequireJS had to cope reduction in usage. For the design data set React received a significant jump, which is similar to the results for jQuery, TweenMax and Handlebars. The usage of Prototype and RequireJS however steadily declined. For the randomly crawled websites, jQuery, jQuery UI, PrettyPhoto and spin.js gained in importance, while Require and Prototype recorded significant decreases.

Especially noteworthy is also the usage drop of Require and Prototype, which could be observed for all four data sets.

5.1.8 Character Encoding

When having a look at the character encoding usage of the different data sets, the main encoding types, which occurred on the data sets are UTF-8, ISO-8859-1, ISO-8859-2, GBK, GBK2312 and Windows-1252.

The most commonly used type is UTF-8, which is represented in the first position of every data set with a percentage range between 85 and 100. In regards to the global data set, the encodings GBK, GBK2312 and ISO-8859-1 appear but could reach only among two percent. In Austria, UTF-8 retrieves its worst result. Austrian websites, however, present the encoding ISO-8859-1 on the second position with considerable 8.8% and furthermore Windows-1252 and ISO-8859-2 among three percent. Rather remarkable are also the results for the design collection, with an UTF-8 distribution of 100%. For the random data set, UTF-8 obtains also relatively high results with 96.4% and also ISO-8859-1 shows an 3.12% usage.

Character Encoding Correlations

Particularly prominent is the appearance of the encoding for the design data collection. For the design data set, the character encoding UTF-8 retrieves significant 100%. By looking deeper at other timestamps, the same behavior occurred. A possible reason might be that the guidelines for submitting a website for the Awwwards nominees demand this character encoding format. Despite all research, no such requirements could be found. Nevertheless, it is probably no coincidence that all the crawled Awwwards nominees, for all timestamps can record the same encoding type.

Character Encoding Trends

Looking at the character encoding changes, the format UTF-8 increased slightly for the global, as well as for the random data set. In contrary, the Austrian data set can not record any encoding format increases. However,

as mentioned before, for the design data collection, the same conspicuous behavior can be observed again. The website encoding format UTF-8 maintains 100%. For the top global websites the format ISO-8859-1 and GB2312 can record a slight rise. For the random data set, UTF-8 gained a significant 4.72%, while ISO-8859-1 decreased its usage value.

5.1.9 Site Elements

The most common crawled site elements which were found on the websites of the four different data collections are

- Gzip compression,
- inline CSS,
- external CSS,
- embedded CSS,
- SPDY,
- Non-Secure cookies,
- Non-HttpOnly cookies,
- HTTP/2,
- cookies expiring in months,
- cookies expiring in hours,
- session cookies,
- HttpOnly cookies,
- HTTP Strict Transport Security,
- Strong ETag
- and IPv6.

The distribution of these site elements is rather equal and no further anomalies between the different data sets can be detected.

Site Elements Correlations

For observing the correlations of site elements with other website aspects, the site element SPDY is picked. Overall, SPDY occurs 229 times within the data collections. SPDY is a new experimental protocol from Google, which should help to improve the pace of the web. More precisely, it is an application-layer protocol for transporting content over the world wide web, designed for minimal latency [29]. Therefore, it is not surprising that SPDY retrieves high correlations with the web server Nginx, with 70.71%. The Google Server, which did not appear as prominent before, can obtain a correlation share of 13.08. In the third position Lite Speed is located. In contrary, Apache can only record a correlation of three percent.

The correlation of the Google protocol SPDY and the traffic analysis tool Google Analytics gains 32%, all other analysis tools, however, obtain only below two percent. In regard to content management systems, the most popular combination is SPDY with Wordpress, which retrieved a correlation of 20.2%. In contrary, Drupal obtains significant less with 4.04%.

Site Elements Trends

The usage of the site elements HTTP Strict Transport Security, Secure Cookies, HTTP/2, SPDY and IPv6 improved their importance for the global data set. Although the values of the Austrian data set differs quite often, the site elements IPv6 and SPDY increased its usage for this data set as well as for the random data set. Moreover, the implementation of embedded CSS, Non-HTTPOnly cookies and session cookies rose constantly. For the design data set, the elements non secure cookies as well as secure cookies, Non-HTTPOnly cookies and session cookies retrieved a value growth.

5.1.10 Web Servers

Among the top ten web servers of all data sets, the following are appearing: Nginx, Apache, Google Web Server, IIS, Tengine, OpenResty, Apache Traffic Server, Jetty, JBoss, HHVM, Phusion Passenger, Apache Tomcat, Open GSE, LiteSpeed, Erlang and Goolge App Engine.

For the Austrian and the design collection, Apache has still the highest percentage between 51 and 69 %. In comparison, Apache retrieves 36.19% in the top global websites and 36.64% in the random collection, which puts it one position behind the top web server, Nginx.

For the global data set the server IIS (6.58%) is located in the third position, followed closely by the Google Web Server with 3%. In Figure 5.6, the distribution of the top three web servers for the different data sets can be seen. Clearly identifiable is the fact that Apache is strongly established in Austria. It could even reach 68.8%, in contrary, Nginx can only exhibit 13.8%. Even more surprising is the third position, Jetty with 10.3%. The web server Jetty occurs only in one data set within the top ten again, the global data set, with sparsely 0.4%. Strong anomalies in the Austrian data set could be detected before. Further investigations revealed that the domain-for-sale web pages, which are occurring in the Austrian data set, are producing an incorrect outcome. Consequently, Jetty, which is the web server for all the domain-for-sale web pages, can retrieve such a high ranking for the Austrian data set.

The reason for the unequal distribution in consideration to the other web servers, especially considering the global and the Austrian data set, might be that for the top global websites the performance is more essential as for Austrian websites. Therefore, the most common Austrian web server is the

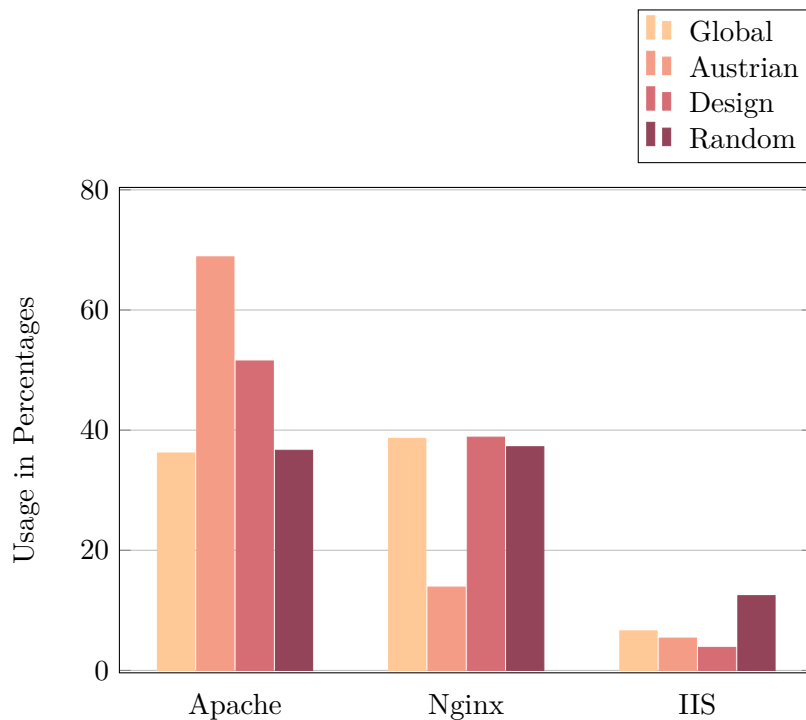


Figure 5.6: Three most used web servers regarding to the four different website data sets.

proven Apache and the highly performant web server Nginx is not as spread as in the global website collection.

Web Server Correlations

As previously mentioned, the two most common used web servers are Apache and Nginx. For this reason, the correlations of these two web servers with other website aspects are particularly interesting.

Concerning the interaction with the different programming languages, Apache has the highest correlation share with PHP (38.6%), followed by JavaScript (16.22%) and in the third position is Java with 4.37%. Within one percent correlation are Perl, Ruby and Python are shown.

In comparison, the correlation for Nginx with PHP is rather similar with 38.62%, in the second position is also JavaScript with 20.78%, but on the third rank Lua is located with 2.69%. The top three programming language correlations are followed by Ruby, Java, nodeJS and Python (see Table 5.2).

Table 5.2: Correlations of the two most popular web servers with programming languages.

Web Server	Programming Language	Correlation
Apache	PHP	38.60 %
Apache	JavaScript	16.22 %
Apache	Java	4.37 %
Apache	Perl	0.82 %
Apache	Ruby	0.53 %
Nginx	PHP	38.62 %
Nginx	JavaScript	16.22 %
Nginx	Java	4.37 %
Nginx	Perl	0.82 %
Nginx	Ruby	0.53 %

Web Server Trends

The recorded data from the different data sets with the web servers, Nginx and Apache, are described in the following paragraph.

For the global data set, Nginx gained slight 0.3%, while Apache lost usage for this data collection. Moreover the web servers IIS, Google Servers, Open GSE and Erlang also increased its values for the global data collection. In relation to the Austrian data set, Apache as well as Nginx retrieved a slight increase of 0.3%. Although, it must be said that in Austria Apache is the clear leader. For the design data set Nginx and IIS increased its value, while Apache incurs significant losses again. The websites, which were randomly selected obtained an increase for all three web servers Nginx, Apache and IIS.

5.1.11 Mobile Optimization

The crawled aspect mobile optimization can be separated into the two parts, *adaptive* and *responsive* websites. For this aspect, the non-mobile optimized websites are not taken into account. The distribution of the two parts is for all four data sets rather similar, with a few deviations.

From the global data set 89.19% are responsive and the remainder are the adaptive websites. Consequentially, the top global websites record the most share of any data set for the adaptive websites. In contrary, only 6.4% of the mobile optimized website of the Austrian websites are adaptive. The random data sets obtain a similar result with a percentage of 6.5% of adap-

tive websites and for the design data collection, even less adaptive websites are detected.

Mobile Optimization Correlations

To retrieve valuable information, different correlation configurations for the mobile optimization aspect have to be analyzed.

The first consideration is to compare mobile optimized websites and their correlation with web servers. For responsive websites the highest correlation rank is with Apache with 35.94%, but Nginx follows closely with just 5% less. However, adaptive websites obtain higher correlation shares with Nginx (17.11%), but the result for Apache is also just a few percent less.

Considering programming languages, the two mobile optimized website aspects responsive and adaptive fetch fairly different results. Responsive websites accomplish better correlations with PHP, followed by JavaScript and Java. In contrast, adaptive websites have higher correlations with JavaScript, which is in the first position, Java in the second and PHP in third.

The top four correlating JavaScript frameworks for responsive websites are jQuery, RequireJS, Modernizr and Prototype. In comparison to that, for adaptive websites the last two JavaScript frameworks are different. In the third position Prototype followed by React.

Mobile Optimization Trends

When observing mobile optimization developments, the global and the Austrian data set record similar results, which is a slight increase for adaptive websites. In contrast, the random and design data set see the responsive websites gaining more importance.

5.1.12 Analysis Tools

Within the top ten analysis tools of every data set, the usage of the following tools was discovered: Google Analytics, New Relic, Alexa, Lotame, Full Circle Studies, Adobe Analytics, Monetate, Piwik, WordPress Jetpack, CrazyEgg, Urchin, WebTrends, Clicktale, StatCounter and Chartbeat.

The analysis tools market leader is Google Analytics, with 61.43% for the global data set, 95.8% for the Austrian, 96% for the design and 79.8% for the random data collection. All other analysis tools are differently distributed on each set. The usage of the most various types of analysis tools is found on the global and on the random data set. Compared to that, other collection set do not show the variety of different tools. The distribution of the top three analysis tools Google Analytics, New Relic and Piwik can be seen in Figure 5.7.

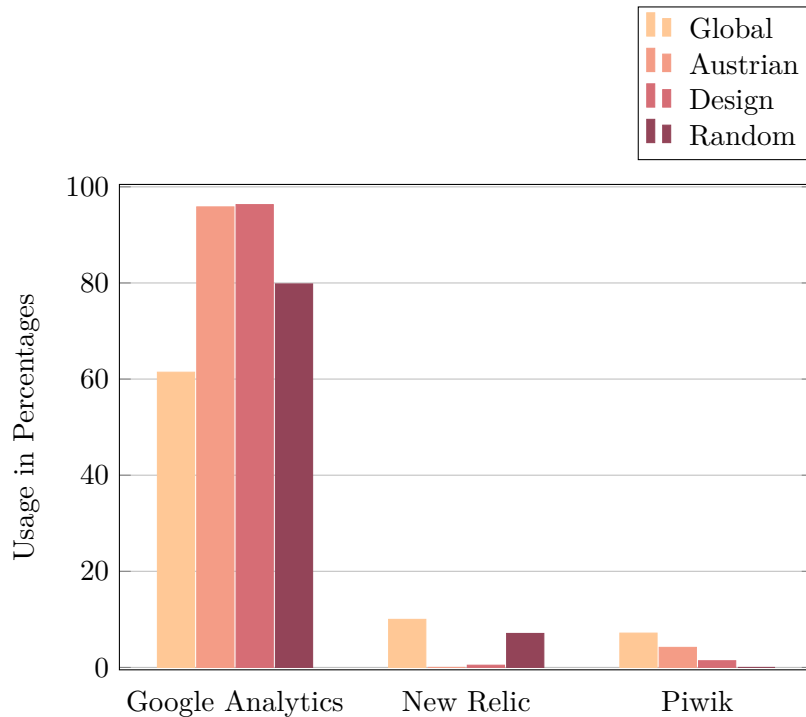


Figure 5.7: Google Analytics, New Relic and Piwik and their distribution of the four different website data sets.

Analysis Tool Correlations

To evaluate the correlations of the analysis tools with other web page aspects, the aspect *site elements* was picked. The reason for this selection is that this aspect has not been described much until now.

The most popular traffic analysis tool Google Analytics will be compared. Of all websites which embed Google Analytics, a significant 98% include external CSS, 90% are showing inline CSS and 84.06% of these websites are optimized by Gzip compression. Moreover, 53.9% include embedded CSS. In relation to cookies, 45.42% of these websites are using non-secure cookies, 38.65% session cookies, 36.65% non-HTTP-only cookies and 26.29% HTTP-only cookies. Another fact of the interaction between Google Analytics and site elements is that 19.52% of the websites, which are embedding Google Analytics are also supporting HTTP/2.

Analysis Tool Trends

Especially interesting is also the development for the analysis tools. For the global data set the analysis tools Adobe Analytics gained over 5% between the two crawling intervals. Webtrends, Quantcast, Piwik and CrazyEgg also

increased their value. For the Austrian collection the usage of Google Analytics constantly grew, while the usage of Piwik decreased. In regard to the design websites, however, the analysis tools Piwik, Google Analytics, New Relic and Wordpress Jetpack gained on importance. Especially noteworthy is also the increase of New Relic with over seven percent for the random data set. Further tools which could improve their usage value for this data set are Clicktale, Full Circle Studies, WordPress Jetpack, Webtrends, StatCounter, Monetate, Chartbeat and Lotame.

5.1.13 Web Frameworks

The most common web frameworks, which are detected in the different data collection are Twitter Bootstrap, ASP.NET, Ruby on Rails, ZURB Foundation, Express, Java Servlet, UIKit, JavaServer Pages, Laravel, CodeIgniter, TYPO3 Flow, Cowboy, Nette Framework, Materialize CSS and Django. These web frameworks are differently based on PHP, Java, Python, JavaScript and other programming languages. The most popular framework of all four data sets is by far Twitter Bootstrap, with a remarkable percentage range of 54 to 74.

ASP.NET retrieves a rather high ranking too and positions itself in the second place for the Austrian and the random data set. However, for the design and the global data sets the framework ZURB holds the second position, which is one rank above its position for the Austrian and random data set (see Figure 5.8).

Ruby on Rails is located in the fourth position for the global data set. The framework Express also received high ranks, except for the Austrian data set with just 0.43%. Remarkable for the design websites are the appearance of frameworks, which do not occur in other data sets above the top ten. In particular, these frameworks are Nette Framework, which is a PHP framework, Materialize CSS, which is a front-end framework and Cowboy, an Erlang framework. Other deviations are found in the Austrian data set with TYPO3 Flow.

Web Framework Correlations

To compare the correlations of a web framework with other web page aspects, due to former value increases, the framework Ruby on Rails is selected.

The correlation with the web server Nginx is fairly high, with 62.2%. In contrast Apache only reaches 11.1%. The programming language for the framework Ruby on Rails is as expected Ruby, additionally other languages like JavaScript, PHP, Erlang or Lua occurred. In regard to all image formats, Ruby on Rails retrieved a pretty high rank for SVG, in third place after JPEG and PNG with both capturing 8.9%. Most common JavaScript Frameworks in combination with Ruby on Rails are RequireJS, jQuery, Pro-

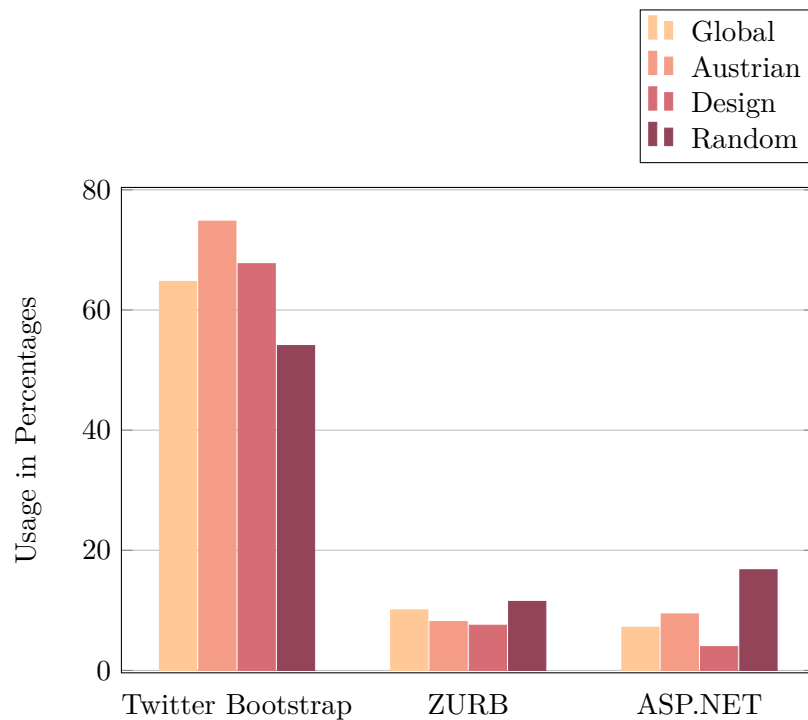


Figure 5.8: Three most used web frameworks regarding the four different website data sets.

totype, Modernizr and React.

Web Framework Trends

A characterization for the web framework trends regarding the development is the diversity of the data collections. For the global data set, ZURB, ASP.NET and Cowboy record increases, while the usage of Twitter Bootstrap dropped. However, for the Austrian data set the usage of Twitter Bootstrap grew, as well as the usage of ZURB and UIKit. The values for ASP.NET, in contrary, slightly declined. For the design websites, the web frameworks Laravel, Twitter Bootstrap, Django and Pure CSS gained on importance.

Particularly prominent, however, was the increased value of Twitter Bootstrap for the random data set, with over ten percent.

5.1.14 Content Management Systems

An integral part of the content management systems is Wordpress, which positions itself as the most popular CMS for the design and random data sets. For these collections, Drupal is following and is represented in second

place. However, for the global data set, the order is reversed and Drupal is shown in the first position. Especially noteworthy for the Austrian data set is that TYPO3 plays an important role. With a percentage of 33.2, it represents the most popular CMS in Austria and also topples WordPress from its first place. The global and the random data sets retrieved a rather high percentage for the Adobe CQ5 in the third position. In contrast, Joomla is listed in the third position for the Austrian data set and Craft CMS in third for the design data set.

Additional content management systems, which were found within the top ten of the different data sets are Tiki Wiki CMS, Wix, DataLife Engine, OpenCMS, Concrete5, ExpressionEngine, DNN, Contao, OpenText Web Solutions, Plone, Craft CMS, October CMS, Sitecore, 1C-Bitrix, Hippo, SDL Tridion, Concrete5, ez Publish and SilverStripe.

Content Management System Correlations

In this paragraph, correlations between content management systems and web servers are analyzed. WordPress retrieves nearly equal results for the correlation with Nginx and Apache. Additionally, Drupal obtains higher correlations with Nginx (48.83%) as with Apache (41.3%). Especially noteworthy is the clear correlation ranking for TYPO3, Joomla and Adobe CQ5. All three content management systems gained a high correlation share with Apache. More precisely, 68.25% of websites which are using Adobe CQ5 are on the Apache server. TYPO3 and Joomla get even higher correlations with Apache (78%).

Content Management System Trends

Particularly remarkable in the development of the content management systems is the fact that for all data set, except for the random, Wordpress could increase its already high value. Moreover, for the global data set, Adobe CQ5 usage also grew. Considering the Austrian websites, the top five content management systems which retrieved an increase are WordPress, Contao, Liferay, Plone, TYPO3, October CMS, Wix and Joomla could expand their distribution for design websites. Rather more significant is the high increase for the Sitecore content management system for the random selected websites. Moreover, also Drupal, Adobe CQ5, Weebly, SilverStripe and Typo3 increased for this data set.

5.2 Summary

To evaluate the results, the different data collections with their differences and similarities need to be observed. Typically for the design data set are the deviations, regarding the design aspects *fonts, colors, image formats*

and *font scripts* compared to all other data sets. When speaking of design trends, the design data set, which websites were pulled from the Awwwards nominees represents itself as the clear leader. In this data set more modern *fonts* are used, more non-achromatic *colors* are shown and the *image format SVG* is constantly gaining in importance. Therefore, for the calculation of design trends, this specific data set should retrieve more weighting.

On the other hand, the global data set shows more diversities for the technical aspects. Web servers for the top 1000 global websites need to be fast and the technologies flexible. That is why this data set includes accurate technical trends.

A rather good comparison is possible with the Austrian data set. The top 1000 Austrian websites are sometimes compared to the top global out-of-date. The used technologies, web servers, markup languages and content management systems are quite different. Of course, it should not be forgotten that the number of websites in Austria can not be compared with the number of websites which occurs all around the world. This is the logical reason, why the Austrian top 1000 websites also include web pages, which are not competitive to top global websites and which might not be updated regularly.

The last data set is consisting of randomly crawled Google search websites. The results for this data set are rather similar to the global data set. This outcome might retrieve from the fact that these websites are ranked pretty well from Google, otherwise they would not show in the top twenty search results. However, this data set sometimes exhibits outliers with technologies which are not seen in any other data set.

Speaking of outliers, inspecting them showed the result that some websites of the Austrian data set, included web pages which are not showing any real content. More precisely, these websites are *domain-for-sale* web pages, which occur 96 times in the Austrian data set. This had the unfortunate effect of distorting the result. Fonts, web servers, and JavaScript Frameworks, which were not seen before on other data sets in these high positions, showed anomalies and further observations demonstrated that these conspicuous aspects were only found on these *domain-for-sale* web pages. This problem might be solved by detecting similar content after the downloading process has taken place. However, the automatic detection of websites with real content and websites with copied content might not be rather easy because all these websites nevertheless include differences. This special problem might also occur on other data sets. Especially the global data set could be affected. The probability for the design and the random data set, however, is rather low.

When evaluating the trend results, especially one data set is conspicuous, which is the collection of randomly crawled websites. This data set shows the highest change of the different aspects. Whether the properties could register an increase or decrease in usage, the percentages of the random data set are rather high compared to all others. This issue might occur from

the fact that the websites of this data set are completely changing for every crawling timestamp. Every new crawling interval, the websites are fetched again from the Google search, and therefore there are mostly not consisting websites over the time. This implicates that the websites are built with different technologies and designs. In comparison to the global or Austrian data collections, this issue does not occur, since the ranking of the Majestic Million does not change as rapid and only a few modifications are visible in the top 1000 global or Austrian websites. One might wonder if this issue does appear for the design websites as well. Of course, these websites are also changed for every crawling interval. But the difference between the Awwwards nominees and the Google results is that the websites, which are shown on the Awwwards nominees, are all from the same category *design*, while on the other hand, the websites from the Google search are from completely different categories and also countries.

To conclude, the research points out that the trend result would be more accurate if the data sets would involve even more websites and the crawling intervals would be processed over longer time. For example, if the crawling processes would be over a year, the changes of different properties would be more visible. Nevertheless, especially the design collection retrieved rather good results for the design aspects and the global data set for the technology aspects.

Chapter 6

Evaluation

To evaluate the accuracy of the approach and if this prototype actually could help web developers and web designers, it is important to compare the approach outcome with other existing web trend analyses. The method used is to compare the trend analyses of the approach with manually written web trends and to have a look at the differences and similarities.

6.1 Manually Written Trend Analyses

A lot of manually written trend analyses involve trends that are not observed in this approach. Therefore trend analyses which include the crawled website aspects are picked and compared. One important thing to keep in mind is that these trend analyses, written by experts, are predictions and it is not accurate to say that they have to be fulfilled a hundred percent, but they provide a solid guideline.

6.1.1 Awwwards Trends 2017

The Awwwards websites present an article, which predicts some major trends which appearance should increase in 2017 [15]. As mentioned before, not all of these predictions can be compared to the prototype, because some of these aspects are not observed. Thus, this evaluation will focus on the crawled website aspect *fonts*. One of the predictions of the Awwward trends is that typography in 2017 will get richer and more decorative. Designers are using different kinds of typefaces on one website and are also willing to work with contrasts. For example, this would include mixing serif typefaces with non-serif ones. Thus, they are forecasting, that *Sans-Serif*, as well as *Serif* and *Monospace* fonts are significantly increasing in 2017 [15].

This project's approach can fully support this prediction. For all data sets, these three typefaces gained in importance between the crawling intervals. As a short recap, for the global data set *Monospace*, *Serif* and *Sans-*

Serif are located in position one, two and four of the fonts with the most increase in usage. Also for all other data sets these types retrieve rather high percentages in usage, and the trend of improved occurrence can be observed as well.

6.1.2 Webflow Image Formats Trends 2017

On the list of webflow's web predictions for 2017, in the sixth position, the image format SVG is located. According to webflow, SVGs prevent web designers and web developers with lots of advantages over other image formats as JPG, PNG and GIF. The main advantage of SVG is the scalability and the fact that it is composed of vectors. This provides its resolution independence and gives the developer lots of possibilities. Furthermore, SVGs also fasten up the web page and do not require any HTTP requests [33].

Except for the Austrian data collection, this trend of using the file format SVG can be monitored as well for this approach. However, it is to say that one other image format reported an increase in usage and that was WebP. In particular, this image format rose for the global and random data sets.

6.1.3 Hackernoon's Best JavaScript Frameworks In 2017

Regarding to hackernoon's best JavaScript frameworks in 2017, Angular, ReactJS and VueJS are gaining on importance. With the new release of Angular2, new features that enable building everything, ranging from web to desktop and mobile were added. With these features AngularJS should grow in popularity in 2017. However, React is considered the fastest growing JavaScript framework and it makes app development straightforward and easy to understand, but is also perfect for complex and high-load software solutions. Compared to React and Angular, VueJS took best properties from both and proved to be faster and leaner. Therefore VueJS should also be able to improve its usage [23].

For observing this theory, all data sets must be checked first. A fact which was quickly noticeable is that all these frameworks are included in the top twenty crawled JavaScript frameworks. The share for the usage of AngularJS lies between 0.68 and 1.29 percentage, with its best result for the design websites. React's usage is a little bit higher, with percentages between 0.68 and 3.85. Its worst result obtains React for the Austrian data set. In contrary, the best result retrieves React for the global data set, where it is even located in position six of the most used JavaScript frameworks. The framework VueJs obtained the lowest usage share of these three frameworks, with percentages between 0.09 and 0.52%, however, it needs to be mentioned that the increase in usage was the highest. To sum it up, it can be said that these three JavaScript frameworks are likely to increase their usage throughout 2017. The extent to which they will increase is the only

uncertainty.

6.1.4 1&1 Website Performance Trends

According to the 1&1 website performance trends, the image format WebP, which was created by Google, will increase its usage in 2017. The image format combines all advantages of JPEG, PNG and GIF and the performance is significantly enhanced. There is, however, the problem that the browser adaptation of WebP is suboptimal and it is only supported by Chrome, Opera and the Android browser until now [11].

Regarding the project, the fact that WebP is gaining on importance could be observed as well. In particular, for the global and the random data set, the image format is represented and also increased its value.

Another trend which is predicted by 1&1 is the further growth of the protocol update HTTP/2. The protocol is also contributing to the performance of websites and is, therefore, an essential part of optimizing the web [11].

When observing the crawled data, this trend can be confirmed, as well. The protocol is a major component of the top ten site elements of each data set. Only for the Austrian data collection, it retrieved rather low usage. For the global data set, however, it even reached the position four of the most increased site elements.

Chapter 7

Conclusion

For web enthusiasts, it is essential to extend their knowledge in web design and web technology permanently. Since the World Wide Web changes at such a quick pace, staying on track with newest web trends is not always easy. The main advantage of web analyses and statistics generated by crawling processes is the objectiveness of this method. Therefore the error probability due to personal opinions is extinguished. For the crawling part, website aspects which can not be included in expert analyses without extensive research can be fetched immediately with this method. Another benefit is the classification of analyses in categories. This gives the opportunity to prevent web analyses for specific website groups and collections.

The implemented prototype represents web statistics with the classification *global*, *Austrian*, *design* and *randomly crawled*. For these categories, different technical and design aspects are fetched and analyzed. After a specific crawling interval, the same process of extending the data sets and pulling the aspects is repeated to measure aspect value changes over time. Furthermore, the prototype can create several correlations between the crawled website aspects, which are evaluated afterward.

The review of the fetched data and the comparison to manually written trends show the individual results of each data set and how these collections differ. A forerunner for design trends is the data set *design*, which includes the nominees of the Awwwards website. Especially this data set speaks for its modernity and creative audacity. In contrary, for technical analyses, the *global* data set, which are the top 1000 websites of the Majestic Million Global and the *randomly crawled* data set from the Google results show similarities to the predicted web trends in technology for 2017. The evaluation of the *Austrian* data collection, which includes the top 1000 websites of the Majestic Million Austria, shows the differences concerning all other data sets. The Austrian websites often show trends of technologies and fonts which are already decreasing its value for the other data sets. Thus, it can be said that the top Austrian websites are not changing as swiftly as the global web-

sites in technology and the Awwwards websites in design. However, it needs to be mentioned that the obtained manual web analyses retrieved rather similar results to the prototype, with little deviations. The main difference between these two methods are the aspects, which were taken into account. For manually written trend analyses mostly website properties which can be seen visually are analyzed, in contrary for the crawling process most aspects are extracted from the program, code or stylesheet.

One of the issues, which occurred during the implementation was the problem of duplicated websites containing similar content above the Majestic Million. This issue had the effect of distorting the crawled result lightly. Future work could improve the process and check the crawled websites from Majestic Million for duplicated or similar content to exclude it from further processing. Additionally, the method of ranking colors of websites could be improved by detecting how much space this color takes on the website and recalculate the ranking.

Appendix A

Installation Guide

1. Installation of Python 2.7:
<https://www.python.org/downloads/release/python-2713/>
2. Installation of PyCharm:
<https://www.jetbrains.com/pycharm/>
3. Open project in PyCharm
4. Installation of PyMongo
5. Type in PyShell: *from pymongo import MongoClient*
6. Installation of the libraries: pymongo, flask, urllib2, whois, builtwith, lxml.html, urllib, urlparse, exceptions, bs4, requests, cssutils
Type in PyCharm: PyCharm → Preferences → Project → Project Interpreter → Add Libraries
7. Import all databases (*databases/*)
8. Start PyMongo: *mongod -dbpath .*
9. Run *web.py (src/Web/web.py)* for the web visualization
10. Go to <http://127.0.0.1:5000/>

Appendix B

CD-ROM/DVD Contents

Format: CD-ROM, Single Layer, ISO9660-Format

B.1 Masterthesis

Pfad: /

Swoboda_Jennifer_2017.pdf Master thesis with instructions (entire document)

B.2 Literature

Pfad: /OnlineSources

*.pdf Copies of the online sources

B.3 Images

Pfad: /images

*.png, *.jpg Rendered images

B.4 Projectfiles

Pfad: /SourceCode

project.zip Source code of the developed prototype

Pfad: /Database

*.json Database of the developed prototype

References

Literature

- [1] Satinder Bal Gupta. “The Issues and Challenges with the Web Crawlers”. *International Journal of Information Technology & Systems* 1.1 (2012), pp. 1–9 (cit. on p. 6).
- [2] P. Baldi, P. Frasconi, and P. Smyth. *Modeling the Internet and the Web: Probabilistic Methods and Algorithms*. Wiley Series in Probability and Statistics. Wiley, 2003 (cit. on p. 6).
- [3] Namrata H .S. Bamrah, B .S. Satpute, and Pramod Patil. “Web Forum Crawling Techniques”. *International Journal of Computer Applications* 85.17 (Jan. 2014), pp. 36–41 (cit. on p. 7).
- [4] Christos Bouras, Vassilis Pouloupoulos, and Athena Thanou. “Creating a polite adaptive and selective incremental crawler”. In: *Proceedings of the International Conference WWW/INTERNET*. (Lisbone). 2005, pp. 307–314 (cit. on p. 7).
- [5] Heiko Haller, Markus Hartwig, and Arne Liedtke. *Google Analytics und Co. Methoden der Webanalyse professionell anwenden*. Addison-Wesley, 2010 (cit. on p. 5).
- [6] D. K. Mahto and L. Singh. “A dive into Web Scraper world”. In: *Proceedings of the 3rd International Conference on Computing for Sustainable Global Development*. (New Delhi). Mar. 2016, pp. 689–693 (cit. on p. 7).
- [7] Christopher Olston and Marc Najork. “Web Crawling”. *Foundations and Trends in Information Retrieval* 4.3 (2010), pp. 175–246 (cit. on p. 5).
- [8] Julie Steele and Noah Iliinsky. *Designing Data Visualizations*. O’Reilly Media, Inc., 2011 (cit. on pp. 28, 29).
- [9] Guandong Xu, Yanchun Zhang, and Lin Li. *Web Mining and Social Networking: Techniques and Applications*. New York: Springer-Verlag, 2010 (cit. on p. 1).

- [10] Feng Zhang et al. “Honeypot: a supplemented active defense system for network security”. In: *Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies*. (Chengdu). IEEE, Aug. 2003, pp. 231–235 (cit. on p. 7).

Online sources

- [11] *1&1 Web Performance Trends*. URL: <https://hosting.1und1.de/digitalguide/websites/web-entwicklung/tipps-zur-website-performance-optimierung-das-sind-die-trends/> (visited on 05/23/2017) (cit. on p. 58).
- [12] *Alexa*. URL: <http://www.alexa.com> (visited on 04/05/2017) (cit. on pp. 10, 16, 21).
- [13] *Alexa Popularity*. URL: <http://data.alexa.com/data?cli=10&dat=snbamz&url=orf.at> (visited on 05/09/2017) (cit. on p. 18).
- [14] *Awwwards*. URL: <https://www.awwwards.com> (visited on 04/24/2017) (cit. on p. 22).
- [15] *Awwwards Web Design Trends 2017*. URL: <https://www.awwwards.com/web-design-trends-for-2017.html> (visited on 05/19/2017) (cit. on p. 56).
- [16] *Beautiful Soup Documentation*. URL: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/> (visited on 04/06/2017) (cit. on p. 12).
- [17] *Bluecoat*. URL: <https://sitereview.bluecoat.com> (visited on 04/05/2017) (cit. on p. 16).
- [18] *Built With*. URL: <https://builtwith.com> (visited on 04/05/2017) (cit. on pp. 10, 16).
- [19] *CSSUtils Library*. URL: <https://pypi.python.org/pypi/cssutils/> (visited on 04/20/2017) (cit. on p. 20).
- [20] *Flask*. URL: <http://flask.pocoo.org> (visited on 04/05/2017) (cit. on p. 27).
- [21] *Google Bot*. URL: <https://support.google.com/webmasters/answer/182072?hl=en> (visited on 04/13/2017) (cit. on p. 11).
- [22] *Google Index*. URL: <https://support.google.com/customsearch/answer/4513925?hl=en> (visited on 04/13/2017) (cit. on p. 11).
- [23] *Hackernoon’s Best JavaScript Frameworks 2017*. URL: <https://hackernoon.com/5-best-javascript-frameworks-in-2017-7a63b3870282> (visited on 05/19/2017) (cit. on p. 57).
- [24] *Lxml*. URL: <http://lxml.de> (visited on 04/19/2017) (cit. on p. 15).
- [25] *Majestic Million*. URL: <https://de.majestic.com/reports/majestic-million> (visited on 05/09/2017) (cit. on pp. 21, 22).

- [26] *Mincss Library*. URL: <https://github.com/peterbe/mincss> (visited on 04/20/2017) (cit. on p. 20).
- [27] *Python Description*. URL: <https://www.python.org/doc/essays/blurb/> (visited on 04/18/2017) (cit. on p. 14).
- [28] *Scrapy*. URL: <https://scrapy.org> (visited on 04/13/2017) (cit. on p. 12).
- [29] *SPDY Protocol Whitepaper*. URL: <https://www.chromium.org/spdy/spdy-whitepaper> (visited on 05/19/2017) (cit. on p. 45).
- [30] *The Internet Archive*. URL: <https://archive.org> (visited on 04/18/2017) (cit. on pp. 1, 5).
- [31] *Urllib2 Library*. URL: <https://docs.python.org/2/library/urllib2.html> (visited on 04/18/2017) (cit. on p. 14).
- [32] *W3Techs*. URL: <https://w3techs.com> (visited on 04/05/2017) (cit. on pp. 2, 10, 16).
- [33] *Webflow Web Design Trends 2017*. URL: <https://webflow.com/blog/18-web-design-trends-for-2017> (visited on 05/19/2017) (cit. on p. 57).