Is Common Crawl a reliable source of Web statistics?

Lukasz Domanski

MInf Project (Part 1) Report

Master of Informatics

School of Informatics

University of Edinburgh

2019

Abstract

Common Crawl is a corpus containing petabytes of data covering over 9 billion webpages collected since 2008. This dataset is widely used in research, however, the issue of whether it is representative of the Web as a whole has not yet been adequately addressed. Existing Web analytics literature focuses on small subsets of the Web, or only examines the distribution of Top-Level Domains. I address the shortcomings of existing literature by investigating if Common Crawl can be used as a reliable source of Web statistics through a comparative analysis including a variety of statistics and multiple datasets. I show inconsistencies between datasets and discover the existence of crawling artefacts in Common Crawl. I attempt to explore what kinds of statistics can be reliably computed using Common Crawl, however, in light of my results, I cannot provide conclusive answers and further research is needed.

In this report, I also address theoretical issues such as the desired properties of a representative sample, whether it is possible to obtain a representative crawl of the Web at all, and how dynamically generated content affects web crawls.

Acknowledgements

I would like to express my special thanks to my supervisor Henry Thompson for all the patient guidance and sharing of his knowledge. I appreciate all the feedback he provided, his support and responsiveness. I was very lucky to have a supervisor who was invested in my topic and was always happy to answer my questions.

Thanks to everyone who contributes to Common Crawl and makes the existence of this incredible resource possible. In particular, I'd like to thank Sebastian Nagel for his prompt and extensive answers to all of my questions on the Common Crawl mailing group.

Thanks to Mucahid Kutlu who graciously agreed to share additional, unpublished data related to ArabicWeb16.

Use of Cloud computing facilities for the work reported here was supported by Microsoft's donation of Azure credits to The Alan Turing Institute. This work was supported by The Alan Turing Institute under EPSRC grant EP/N510129/1.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Lukasz Domanski*)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The World Wide Web, usually shortened as Web, is ”the universe of network-accessible information, the embodiment of human knowledge” [71]. It is an information space where interconnected documents and a variety of other kinds of resources can be shared and accessed via the Internet - a global network that enables communication between any two devices that are connected to it. The Web and the Internet enable communication, sharing resources, and exchanging knowledge on an unprecedented scale. For most people in developed countries, the Web became an essential part of their everyday life.

Because of how important and complex the Web has become, investigating its properties is a significant and popular area of research. However, due to Web’s heterogeneity and its decentralised nature, such research presents many challenges, especially when it comes to gathering data [2].

For reasons explained in Section 2.5, it is impossible to say exactly how many pages the Web contains, but Gulli and Signorini [29] estimated that the Web consists of at least 11.5 billion webpages. Processing this magnitude of information is unfeasible for any individual or organisation, other than a handful of the biggest companies in the world.

Moreover, obtaining a sample of the Web, a process called web crawling, possesses a multitude of ethical and legal implications and presents significant technical challenges

[2, 3, 30]. Those challenges, described in more detail in Appendix A, include but are not limited to:

- **Legal implications** - Crawling might violate copyright laws or website‘s Terms of Service. In the past, website owners decided to pursue legal action against those responsible [15, 36].
- **Crawlability** - Many sites limit or forbid web-crawler access, which might make obtaining necessary data more difficult or impossible [24].
- **Computational cost** - Extracting data from webpages involves parsing (often syntactically incorrect) HTML and significant amount of computation.
- **Long download time** - Aggressive crawling can lead to accidental Denial of Service (DoS) attack. The solution is to increase the interval between requests, but it drastically increases the time it takes to complete a crawl.
- **Need for filtering** - Duplicate elimination and other types of necessary filtering increase the computational cost of web crawling.

I have described those challenges as I believe they illustrate why the existence of a sizable, representative, freely available sample of the Web is so important. Dealing with those issues would require significant resources and could make conducting research unfeasible for most individuals and small organizations.

This is especially true for longitudinal studies that investigate trends and require data collected over many years. However, internet research should not be restricted to the biggest technological giants such as Google or Amazon, and that is the motivation behind Common Crawl.

Common Crawl is a corpus containing petabytes of data covering over 9 billion web pages collected since 2008. It is a publicly available dataset hosted on Amazon Elastic Cloud and is available for download free of charge. It is meant to be a "a copy of the internet to internet researchers, companies and individuals at no cost for the purpose of research and analysis" [18].

Common Crawl allows its users to avoid having to deal with after-mentioned issues and allows researchers and organizations to conduct research that would otherwise be impossible due to prohibitive cost of gathering necessary data.

Common Crawl is widely used for different purposes: to extract data from webpages

[11, 68], create new, more specialized datasets [64], investigate security vulnerabilities [1, 70] and so on.

Because of how widely it is used, it is worth questioning whether Common Crawl is a reliable source of Internet statistics and that is the main goal of this project.

Multiple authors investigated the issue of Common Crawl's representativeness and I summarize their findings in Chapter 3. Existing literature does not address the question of whether Common Crawl is representative of the Web **as a whole** adequately, as authors either focus on a subset of the Web or limit their analysis to the distribution of Top Level Domains.

This report aims to fill that void by attempting to establish what kinds of statistics can be sourced from the dataset by performing a multi-level, in-depth, comparative analysis of Common Crawl and alternative web crawls.

1.2 Objectives

The main objectives of the project are as follows:

1. Define the desired properties of a representative sample of the Web that would be a good source of Web statistics.
2. Empirically establish if Common Crawl is a good source of Web statistics (and if so, to what extent). For that, it is necessary to establish what kinds of statistics can be collected from the dataset if the aim is to make inferences about the Web as a whole.
3. If Common Crawl is not a reliable source of internet statistics, then to describe its shortcomings and investigate what measures can be taken to address them.
4. Establish what kinds of Web properties and statistics can be investigated using a web crawl and which ones are crawl dependant.

1.3 Contributions

I performed a thorough review of the existing literature to establish what questions about Common Crawl's representativeness remain unexplored. As detailed in Chapter 3, many authors examined the topic, however, they also restricted their analysis to a small subset of the Web, or examined just one statistic.

To address that issue, as suggested by my supervisor, I developed a methodology based on comparing different datasets and searching for anomalies. This approach allowed me to discover many inconsistencies between Common Crawl and other datasets, which shows that a surprising number of statistics are crawl-dependant. I showed that every statistic that I analysed was in some way affected by the crawling process, which is an unexpected result. I discussed how this impacts our ability to make inferences about the Web as a whole based on Common Crawl.

I discovered the existence of ”anomalies” in Common Crawl, which are most likely the artefacts of the crawling process, such as the unusual distribution of the days of the month (see Section 5.4.3). If not taken into consideration, those ”anomalies” can lead to erroneous conclusions.

Common Crawl's data gathering process was tuned to make the dataset as useful for researchers as possible, rather than being a random sample of the Web. However, there is no complete, comprehensive documentation of how the data is collected - the details of the crawling process are scattered across dozens of blog posts. I enquired Sebastian Nagel of Common Crawl about the missing details and created an in-depth description of the crawling process in Section 2.7.1.

A single month of Common Crawl data can contain more than 260 Terabytes of data. For January 2019 crawl, even the compressed metadata alone occupies over 20 Terabytes of disk space. I developed, tested, and debugged jobs running on a distributed Hadoop cluster that processed this magnitude of data efficiently.

Last but not least, I discussed theoretical questions such as the desired properties of a representative sample, the issue of content generated based on URLs, the inclusion of spam pages, and how those issues affect web crawls.

1.4 Structure of the report

Chapter 2 starts by clarifying important terminology that is often misused. It also discusses a few closely related theoretical issues: how we define pages to be ”the same”, the issue of spam pages and pages generated based on URL query parameters. I then define the desired properties of a good sample and describe data collection methods used by the crawls utilised in this project.

Chapter 3 provides an overview of existing literature on the topic of representativeness of Common Crawl.

Chapter 4 briefly talks about issues related to processing large-scale data and outlines problems that I have encountered when working with Hadoop clusters.

Chapter 5 contains a description of my methodology, description of each experiment that I performed, and the results.

Chapter 6 includes a description of the work that should be performed in the future to investigate the topic further and attempt to give a more convincing answer to my research question.

The report concludes with a discussion of my empirical results and what can and cannot be concluded based on those results in Chapter 7.

Chapter 2

Background

I start this chapter by explaining commonly misused terminology related to URLs. I then discuss a few closely related theoretical issues: how we define pages to be the same, the issue of spam pages and pages generated based on URL query parameters. Next, I define the desired properties of a good sample and discuss whether Common Crawl has those properties. Last but not least, I describe data collection methods used by crawls utilised in this project.

2.1 Terminology

2.1.1 Uniform Resource Locator (URL)

Uniform Resource Identifier (URI) is a string of characters that ”provides a simple and extensible means for identifying a resource” [5] on the Web.

Uniform Resource Locator, usually shortened as URL, is a type of a URI that in addition to identifying a resource (e.g. a webpage) also provides a way to identify the location of that resource on the Web, and how it can be accessed [5]. There are a few essential terms related to URLs that I need to clarify as they are often misused. Many articles use terms like ”domain” or ”website” without giving a formal definition, which can lead to confusion, as was the case with the ArabicWeb16 paper [67] (see Section 5.2.1).

Each URL, assuming it does not contain credentials, has the following structure (items enclosed in brackets `[]` are optional):

```
scheme://host[:port]/path[?query][#fragment]
```

A (very simplified) explanation of each part [5]:

- `scheme` - tells us how to interpret the remaining part of the URL
- `host` - tells us which server to contact...
- `port` - ...and on which port
- `path` - identifies the exact location of the resource within the scope of the host
- `query` - contains additional data and is often used to store key-value pairs
- `fragment` - is typically used to identify a location within the resource (e.g. paragraph of a document)

The part we are most interested in is `host` - an IP address or a registered name (**hostname**). Let's look at some typical examples of **hostnames**:

(1) `support.wordpress.com`
(2) `blog.example.co.uk`

The red part of the URL is called a **subdomain** and the green part of the URL is called **Top-Level Domain** (TLD). In this report I will refer to the blue part of the URL as **Pay-Level Domain** (PLD). This is the part of the URL that can be ”rented” by registering it with a domain registrar - an organization that manages registration of internet **domains**. The black part of the second URL is formally called **Country Code Second Level Domain** (ccSLD).

I will use the term **domain** to denote the combination of PLD, ccSLD (if applicable), and a TLD (for example `ed.ac.uk`). The term **public suffix** stands for the combination of ccSLD (if applicable) and TLD (for example: `.com` or `.ac.uk`). The term **hostname** is sometimes used interchangeably with the term **website** [2]. For example, the following URLs all share the same **domain**, however these are three different **hostnames** (or websites):

(1) `http://rss.example.co.uk`
(2) `http://blog.example.co.uk`
(3) `http://example.co.uk/blog`

2.1.1.1 Absolute and relative URLs, suffix reference

URLs can be written in multiple forms: an **absolute URL** must contain the scheme, **hostname**, and the entire path. However, if a URL is used on a webpage `w`, then we already know the location of `w`, and therefore we can use a **relative URL** which contains the location of some resource relative to `w` [5]. For example `http://example.com` might contain 2 links, both pointing to the same resource: `http://example.com/article/12` (absolute URL) or `article/12` (relative URL).

URLs can also be written as a **suffix reference** that only contains the `host` and `path` components [5]. For example:

`www.w3.org/Addressing/`

This form does not identify a resource uniquely and requires disambiguating using some heuristics (for example, by adding `http://` at the beginning).

2.2 Static and dynamic webpages

We can distinguish two main types of webpages: **static** and **dynamic**. **Static** webpages ”only change when manually changed by their creators, not on a regular and automated basis” [38]. **Static** pages are often created manually and stored on the server as an HTML file, which is served to visitors when requested. In contrast, **dynamic** webpages are generated automatically by the server, for example, by fetching a list of forum posts from a database and then generating appropriate HTML.

2.3 Archetypal ”real-world” problem

Let’s consider a ”real-world” problem: suppose that we have to compare, based on Common Crawl data, the popularity of HTTP and HTTPS protocols. In the following sections, I discuss a few issues that might affect web crawls. However, the reader might wonder if those issues have any practical significance. In each of the following sections, I include an explanation of how the particular issue might affect Common Crawl and make it more difficult to use it as a source of that statistic.

2.4 Hash-based equality testing

A web crawler might come across the same page multiple times, which makes duplicate elimination necessary. However, to remove duplicate pages, we first need to decide how we define two pages to be the same.

Typically, crawlers employ one of two methods of equality testing - hash-based or URL-based. The former method is based on computing a hash of page's contents - for example, a SHA1 hash of unencrypted and uncompressed body of an HTTP response might be used - we consider two pages to be the same if their hashes are the same. In the latter method, we consider two pages identical if their URLs are the same.

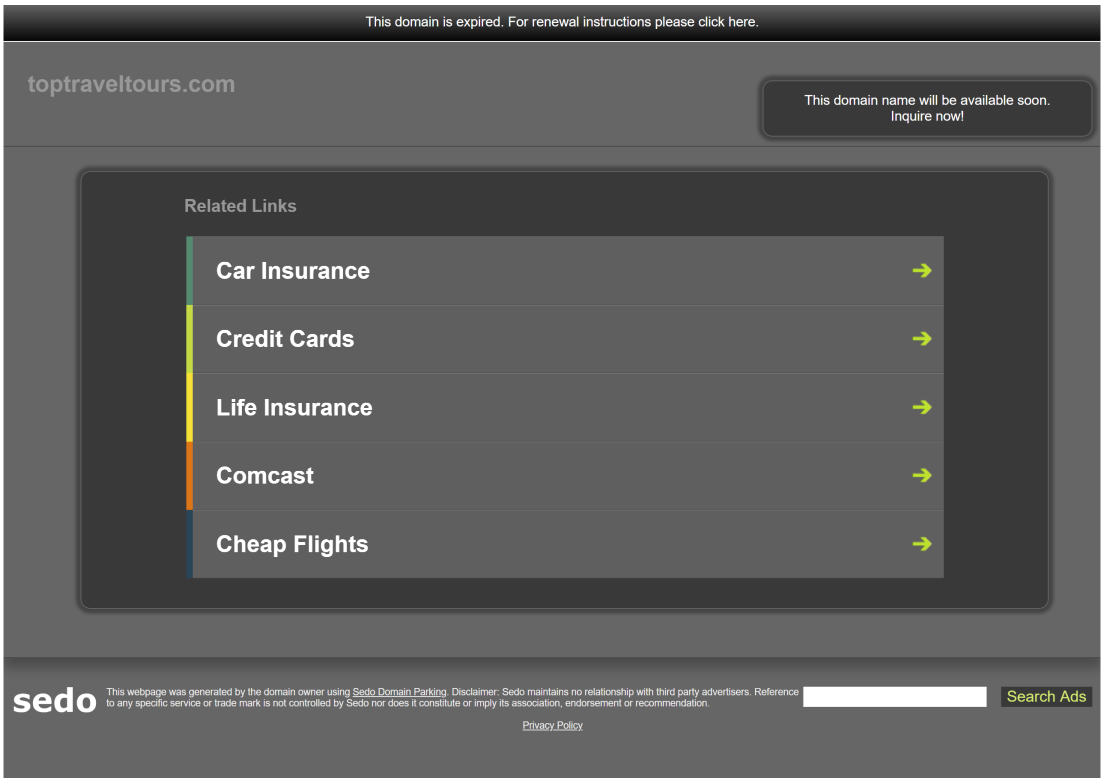
Using URLs for deduplication is not an effective strategy as often URLs that only differ in query parameter values might lead to the same page [68]. Moreover, this strategy is insensitive to how pages change over time.

However, using hash functions for deduplication has disadvantages too. Hash functions have no concept of ”distance” - if two pages differ only in comments in their HTML they are essentially the same, but hash to different values.

Defining ”page equality” is a theoretical issue with no ”correct” solution. Should we consider pages that only differ in advertisements to be the same? Should we consider a page that contains a clock to change every second?

Let's consider how the choice of a method used to compare pages can affect the HTTPS popularity problem introduced in Section 2.3. If **hash-based deduplication is used**, near-duplicates (i.e. pages that are nearly the same) cannot be detected. Therefore, many near-duplicates might be included in the crawl. If those pages predominantly use one of the protocols, then the results can be biased in an unpredictable way.

For purposes of this project, I use a hash-based approach without any attempt to address the aftermentioned issues. I made that decision because (1) this is the approach used by Common Crawl and (2) it allows me to focus on my research question without having to compute any similarity measure on the multi-terabyte dataset (which would present a great technical challenge).


This domain is expired. For renewal instructions please click here.

toptraveltours.com

This domain name will be available soon.
Inquire now!

Related Links

Car Insurance

Credit Cards

Life Insurance

Comcast

Cheap Flights

sedo This webpage was generated by the domain owner using Sedo Domain Parking. Disclaimer: Sedo maintains no relationship with third party advertisers. Reference to any specific service or trade mark is not controlled by Sedo nor does it constitute or imply its association, endorsement or recommendation.

Search Ads

Privacy Policy

Figure 2.1: An example of a parked domain

2.5 How many webpages are there?

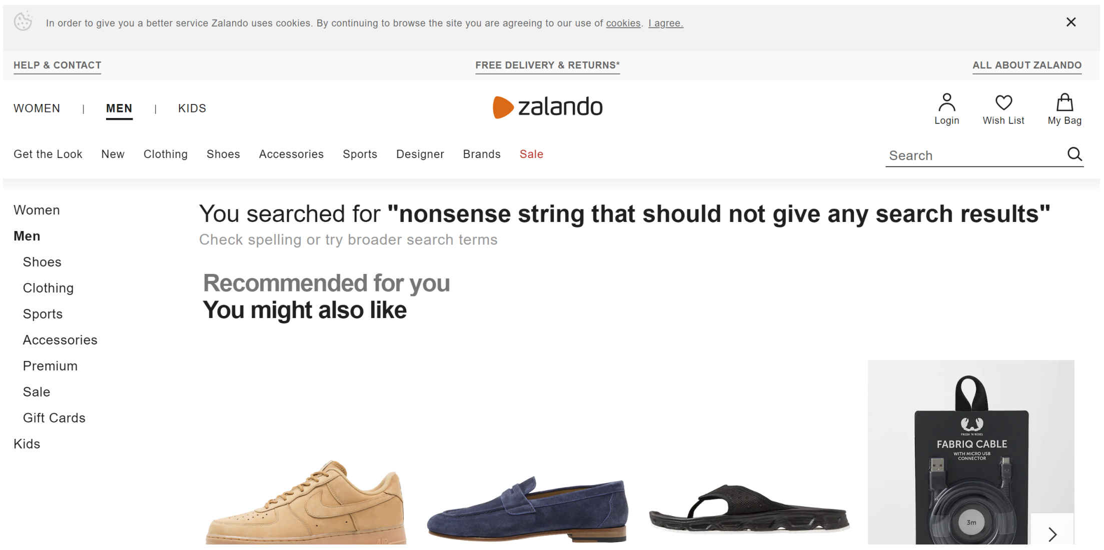
Section 1 mentions that it is not possible to establish how many webpages there are on the Web. Here I explain why.

The question of ”How many pages can be found on the Web?” is closely linked with the question of how we define two pages to be the same. As discussed in section 2.4, I use a hash-based method. This, however, does not help us address the issues of spam pages and the content dynamically generated based on the URL query component (or other parameters of the HTTP request).

2.5.1 Content generated based on the URL

Some websites automatically generate the content of the page based on the URL - more specifically, the query component. Consider a webpage such as the one shown in Figure 2.2:

```
https://www.zalando.co.uk/men/?q=<query-string>
```



You searched for **"nonsense string that should not give any search results"**
Check spelling or try broader search terms

Figure 2.2: An example of page that is generated based on the URL

This URL can be modified by changing `<query-string>` to a string of arbitrary length, meaning that we can generate as many such URLs as we please. As the resulting page contains `<query-string>` in it's HTML (see Figure 2.2), every such page would be considered unique if hash-based equality testing is used.

The problem becomes apparent: research questions such as ”How many pages are there on the Web?” or ”How many pages are there under `zalando.co.uk` **domain**?” do not have a meaningful answer as the URL above alone can be used to generate an unbounded number of pages.

2.5.2 The role of spam pages

Figure 2.1 shows an example of a so-called ”**parked domain**” - a website hosted under an otherwise unused **domain** that exclusively contains advertisements. **Spam pages** can be defined in different ways, but parked domains are a clear example of spam as they provide no value to visitors whatsoever.

Spam pages are an integral part of the Web. Therefore, it is clear that a representative sample of the Web should contain some amount of spam pages. However, those pages sometimes use techniques such as *sitemap spam* - they announce an unusually high number of sitemap URLs. Treating URLs obtained from those sitemaps in the same way as other URLs could cause spam content to dominate the crawl.

Spam and malicious pages are a common object of research; however, since November 2017 Common Crawl began to actively fight spam to prevent it from becoming a majority of the crawl [50]:

> (...) every host of one spam cluster announced 20,000 sitemap URLs in its robots.txt (...) in total, 2.5 billion sitemaps. (...) Penalising spam domains is the easiest way for us to avoid further issues and also to ensure that these spam clusters do not start to dominate future crawls. [50]

Let's consider how the HTTPS popularity problem introduced in Section 2.3 is going to be affected by those issues. If we create a sample that **excludes spam pages**, then we are intentionally omitting a significant part of the Web - the popularity of HTTPS estimated based on that sample is not going to be representative of the Web as a whole.

Many of the pages dynamically generated based on URL query parameters can be considered near-duplicates (for example, pages generated using the `zalando.co.uk` URL discussed in Section 2.5.1). If the crawl contains a high number of those pages, then the HTTPS popularity computed using that crawl is going to be biased. In Section 5.6 I showed that Common Crawl contains many URLs that only differ in query parameters so this issue has practical significance and non-negligible impact on Common Crawl.

2.5.3 Personalisation

Personalisation is "individualising products, services, and contents according to customer interests and preferences" [35]. In recent years, personalisation has become more and more prevalent, which poses a challenge for Web crawling. If `http://youtube.com` looks differently for everyone, which version of the page should be included in the crawl?

Arguably, the crawl should not contain content based on previous browsing history - ideally, we want to crawl the page that a "first-time visitor" would see. Sadly, this might not be possible. Personalisation is often implemented by using a *cookie* - a small piece of data stored by the browser and included in HTTP requests to the server in order to identify the user [73]. However, even if no cookies are included in the request, other information might be used to identify the user, for example user's IP address. It's even possible that different content will be served to web-crawlers compared with what users would usually see.

It can be argued that removing cookies from the request should be sufficient to avoid personalisation. As detailed in Section 2.7.1, Common Crawl does not include cookies in HTTP requests. However, the optional header `Accept-Language` is included and informs servers that English content is preferred and should be served if available. This can be considered a possible source of bias towards English content.

2.6 What is a ”good” sample?

When conducting research, it is often the case that the population under the investigation cannot be analysed in its entirety due to its size, and it is necessary to choose a sample - a subset of the population that we can use to draw inferences about the properties of that population.

When selecting a sample we should [63]:

- Choose an appropriate sample size
- Choose the sample randomly while keeping the heterogeneity of the population in mind

Using a sample that is too small might result in drawing erroneous conclusions, as it increases the probability that the patterns that we observe are a result of sample's peculiarity and not the actual trends in the data [63]. A bigger sample would give us more confidence in our finding, but data collection can be costly.

2.6.1 Is Common Crawl big enough?

It's a difficult question to answer, as the required size of the sample used for a particular study depends on the purpose of that study, the size of the population under the investigation, the acceptable sampling error, and so on [32]. A sample that is sufficiently big for a particular study can be unacceptable for others.

Moreover, it's nearly impossible to estimate how big the Web is, due to the decentralised nature of the Internet as well as the prevalence of dynamically generated content. However, if we narrow down our focus to indexable Web (a subset of the Web accessible using traditional ”link-to-link traversal of Web content” [59]), it is possible to give a lower bound of the number of webpages - this was attempted by multiple authors [69, 29]. Gulli and Signorini estimated that the indexable Web contains at least 11.5 billion webpages [29] .

We can use a simplified formula published by Yamane [74] to compute an estimate of how big a sample of the Web should be:

$$n = \frac{N}{1 + Ne^2} \tag{2.1}$$

Where e is the desired sampling error and N is the size of the entire population. Assuming 95% confidence level and a level of precision (sampling error) of just 1% ($e = 0.01$), this formula tells us that we would only need roughly 10 thousand pages to be included in the sample, even if the Web was actually a 100 times bigger than that lower bound.

However, this is not a useful result due to the heterogeneity of the Web. Using a sample of that size would make investigating ”sub-populations” (for example, pages of a particular **domain**) impossible. Therefore, a ”general purpose” sample, like Common Crawl, should be as big as possible to make different kinds of research possible.

2.6.2 Sample selection method

In Section 2.7.1, I describe how Common Crawl gathers data. That process is not equivalent to random sampling, and it involves different kinds of content filtering. Moreover, as detail in Section 2.7.1.3, Common Crawl is (intentionally) biased toward popular websites.

2.7 Datasets used and their crawling process

The way in which the crawler explores the Web influences which pages are (and are not) included in the dataset. As detailed in Section 3.2, depending on the crawling strategy used, some **domains** can be exhaustively crawled while others remain undiscovered [33, p.1]. In some cases, this might lead to the crawl being dominated by spam, dynamically generated content, near-duplicates, and so on. In previous sections, I used the example of HTTPS popularity (see Section 2.3) to show that this can affect our ability to make inferences about the Web as a whole. As a result, detailed knowledge of the data collection process is necessary. This section outlines the crawling strategies employed by the web crawls that I used in the course of this project.

2.7.1 Common Crawl

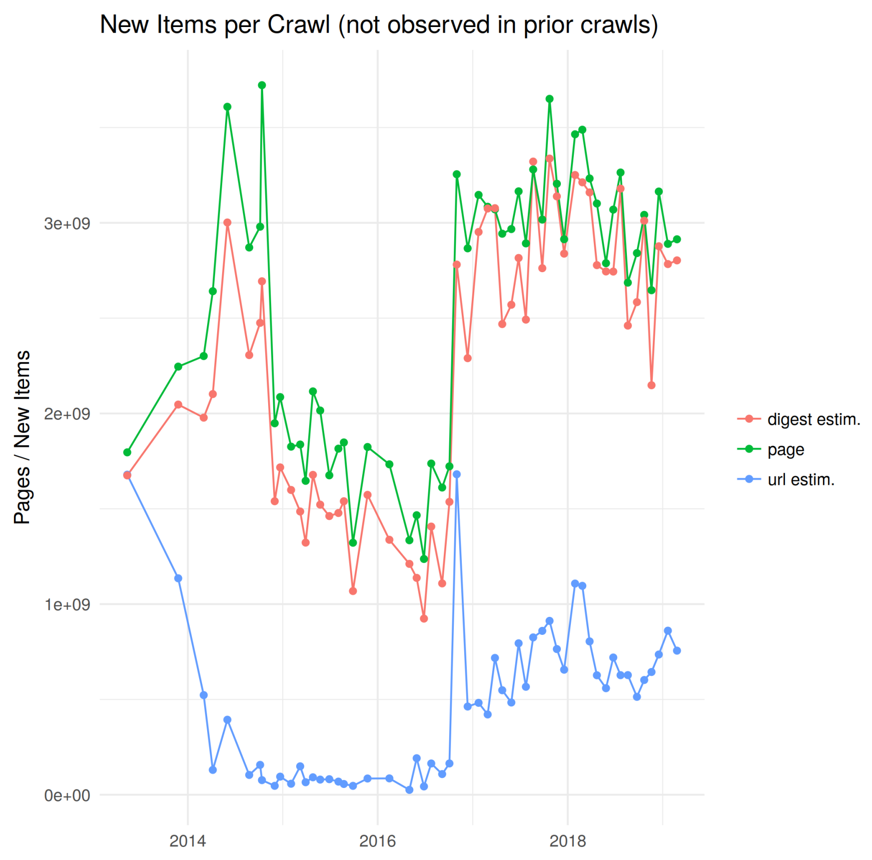


Figure 2.3: Number of new items per crawl. Source: Statistics of Common Crawl Monthly Archives [16]

.

Common Crawl lacks centralised, comprehensive documentation. Moreover, contradicting descriptions of the crawling process can be found [33, 54]. Common Crawl regularly updates its data collection methods, hence crawls from different months were created using different methodologies. This can make Common Crawl unusable for longitudinal studies, as crawls from different months might be subject to different biases. The impact of changes to the crawling policy is apparent when looking at the variation of the number of new elements added to the crawl per month (Figure 2.3).

We can observe three distinct ”phases” - between mid-to-late 2014 and late 2016 very few new pages were added compared to other months [44].

A *crawler* is a program that visits webpages to perform some task, such as data collection or indexing. Typically, crawlers start their exploration with a set of URLs that we call a **seed**. Crawlers fetch pages included in the seed, and then follow all URLs found on those pages in a breadth-first fashion. This procedure leads to an ”exhaustive collection of locally connected subcomponents of the web graph, while possibly leaving other interesting material in other parts of the graph unvisited” [33, p.1]. Common Crawl’s method of data collection differs significantly from this standard process.

2.7.1.1 Fetch list generation

Common Crawl’s crawler does not follow links directly in order to avoid duplication and spam. Instead, the data is collected by fetching URLs from a ”fetch list” generated in advance [55].

Each fetch list is generated by choosing URLs from a database. The database is updated every month by adding new links and removing obsolete URLs - by the end of 2017 it reached 17.5 billion entries [16]. The source of URLs changed over time: the database was populated with donations of lists of URLs from blekko (a company that used to provide Web search functionality) until 2015. Since September 2016, the database is updated with URLs found in sitemaps. Since January 2017, additional ”seed crawls” are used to gather URLs. Since September 2017, WAT files are used to randomly sample URLs [54].

Using WAT files to gather URLs makes Common Crawl’s methodology more similar to how ”traditional” crawlers work. However, as the URLs are chosen (pseudo)randomly from the database, Common Crawl is less likely to exhaustively crawl some part of the Web graph like crawlers that use breadth-first strategy might do. This can be seen as ”a bug or a feature” depending on how the dataset is used.

Each crawl is performed in 100 chunks that we call *segments*. A fetch list is generated (pseudo)randomly for each segment from a database. All URLs are chosen such that:

- ”every fetch list contains a (pseudo)random selection” [54]
- ”pages are also distributed (pseudo)randomly over the WARC files of one segment [data formats are explained in Section 2.7.1.7], so that every WARC file is

a sample by its own" [54]

- "pages with a high score are generated more often (monthly) while low-scoring pages are included in less often" [54]

The crawler does not run JavaScript - it simply saves the HTTP response to a file. If some content is only available through client-side events, then it is not going to be captured.

2.7.1.2 HTTP redirects

An **HTTP redirect** is a special kind of an HTTP response that informs the client that the requested resource is available under a different URL than the one requested. Upon receiving a **redirect response**, the client should immediately load the new URL provided in that redirect response. Before April 2016, the crawler followed redirects immediately. This resulted in duplication, mostly due to shared error pages, as well as URLs differing only in query parameters leading to the same page [68, 45]. In April, May and June 2016 redirects were not followed. In later crawls, redirects were followed again, but deduplication based on URLs was used. Since November 2016 deduplication is also done by computing MD5 hash on the binary content of the response [54].

2.7.1.3 Guiding the crawl

Common Crawl aims to include not only pages from well-known, frequently visited **domains** but also lesser known ones. However, those lower-ranking **domains** are included with a lower probability. The rationale behind this decision is that the sample should consist of content that a "random surfer", a typical internet user who only occasionally visits low-ranking **domains**, would most likely come across [55].

The previous paragraph makes a distinction between high- and low- ranking **domains**. The way Common Crawl makes this distinction has changed over the years:

For crawls performed in 2008 - 2012 PageRank algorithm [61] was used to rank pages.

From 2013 and until February 2016, Common Crawl used scores from the list of URLs donated by Blekko with "a few smaller additions" [54].

From August 2016 and until January 2017, the page rank list published by Common Search has been used, combined with Alexa's ranking of top million webpages [54].

From February 2017 and until April 2017, Harmonic Centrality scores were computed on "smaller web graphs" [54]. Those scores were not published, however, they were used to guide the crawler. Since May 2017 Common Crawl uses the Harmonic Centrality scores (which are computed and published quarterly) to guide the crawler [54].

2.7.1.4 Crawling limits

For each website, Common Crawl checks for so-called `robots.txt` file and obeys any access restrictions or download-rate limits included in that file. For example, Common Crawl does not include any Facebook pages, as `https://www.facebook.com/robots.txt` contains the following:

```
User-agent: *
Disallow: /
```

This disallows any crawling unless a particular crawler has been made exempt.

Additionally, since February 2019 a "dynamic" limit of a number of pages per **domain** was introduced. The limit ranges from 100 to 500 000, depending on the **domain**'s rank [52].

2.7.1.5 Spam detection

Since November 2017 Common Crawl began to actively fight spam to prevent it from becoming a majority of the crawl [50].

AntiTrustRank algorithm (as described in [27]) and the fact that spam **domains** tend to use many **subdomains**, combined with manual flagging, was used to identify 370 000 **domains** which are now blacklisted and are not crawled at all [54].

2.7.1.6 HTTP headers

I analysed April 2016 crawl to establish that the crawler always uses an HTTP GET request with the following headers:

```
Accept-Language:"en-us,en-gb,en;q=0.7,*;q=0.3",
Accept-Encoding:"x-gzip, gzip, deflate",
User-Agent:"CCBot/2.0 (http://commoncrawl.org/faq/)",
Accept:"text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8"
```

I think it's interesting to note that the request does not include Cookies. The authors of Common Crawl decided to include the optional header `Accept-Language` and set it such that English content is preferred and should be served if available.

2.7.1.7 Common Crawl data formats

Common Crawl uses multiple data formats. Each crawled page is saved in three different files: WARC, WAT, and a WET file. This, of course, increases the total size of the crawl as some of the information is repeated; however, it makes working with the crawl easier, as often we can avoid processing raw HTML and simply use extracted metadata conveniently provided by Common Crawl. A description of each file format follows [39]:

WARC files

WARC files contain "raw" data collected by the crawler. WARC files contain 3 types of records (as indicated by `WARC-Type` field of each record):

1. HTTP request sent to the server (`WARC-Type = "request"`)
2. HTTP response from the server (`WARC-Type = "response"`) which includes:
    - HTTP headers
    - the response "body" (e.g. the HTML code)
3. Crawling process metadata (`WARC-Type = "metadata"`)

WAT files

WAT files, similarly to WARC files, also contains three types of records (request, response, metadata). However, WAT files do not contain the raw HTML and instead include additonal metadata stored in JSON format (for example, a list of URLs extracted from HTML).

WET files

WET files contain the text extracted from crawled pages. WET files only include HTML pages that have textual content [54].

2.7.2 ArabicWeb16 and ArClueWeb09

ArabicWeb16 is a publicly available web crawl of around 150.9 milion Arabic webpages (10.8TB of uncompressed data) that was created to provide researchers with a comprehensive dataset of documents in Arabic [67].

It is a sizable dataset which might be a very good source of data for comparisons, as its crawling process is fundamentally different from how Common Crawl collects data.

The initial seed size was small compared to Common Crawl - a set 27 million URLs collected through a variety of methods was used. The crawling software ran for 30 days and was allowed to follow links. ArabicWeb16 used a language detection system and some other heuristics aiming to maximize the amount of Arabic content crawled [67, Section 2.2].

ArClueWeb09, a subset of **ClueWeb09** - a dataset collected in January and February 2009, focuses on Arabic content and contains over 29 milion pages. ClueWeb09 was collected using a crawler that was allowed to follow links, albeit with some restrictions including a depth limit (depth 5 for dynamic pages and depth 15 for static pages) and limiting the number of pages from any site to 25 000 [12].

Table 2.1 compares the size of the three datasets, as well as the number of pages and **hostnames** that they include.

	Common Crawl (Dec 2017)	ArabicWeb16	ArClueWeb09
Data Size	241.9 TB	10.8 TB	0.97 TB
Pages	2.6 billion	150.9 million	29.2 million
Hostnames	50,537,187	768,516	196,776
Average number of pages per hostname	51	196	148

Source: Data for ArabicWeb16 and ArClueWeb09: [67, Page 3, Table 1]
Data for Common Crawl based on [48]

Table 2.1: Comparison of the size of a single month of Common Crawl data, ArabicWeb16, and ArClueWeb09.

2.7.3 Crawls of ”national” Web

Researchers often need to investigate the properties of a ”national Web” of some country. As a result, many crawls restricted to a particular country’s Web have been created. The datasets themselves are usually not publicly available, but relevant and interesting results were published for some of them. In this report, I compare some of my experimental results to published statistics based on crawls from 12 different countries including Spain, Greece, Chile, Brazil, Thailand, and others [2, 25, 75, 42].

I'll briefly describe two of those datasets: crawls of Spanish and Greek Web. The former dataset contains 16 million pages that were collected by crawling **hostnames** under the `.es` Top-Level Domain, as well as pages hosted on servers with IP addresses physically located in Spain. The crawler was allowed to follow links with few restrictions, such a limit of 400 pages per **hostname** for **hostnames** under `.com` TLD [42].

The latter crawl only considered pages under the `.gr` Top-Level Domain and contains 4 million Greek pages. The crawler followed links with the same depth limit as ClueWeb09 (depth 5 for dynamic pages and depth 15 for static pages) [25].

Chapter 3

Related work

This chapter contains a literature review and a discussion of published results related to my methodology, as well as Common Crawl's representativeness and completeness.

3.1 Representativeness of Common Crawl

Properties of the Web are a popular area of research, and there are many resources available on the topic. The web can be studied on many levels of granularity. We can define those levels as micro, meso, and macro [6] or as the level of bytes, words, pages, hostnames, domains, TLDs, and as a whole [33]. Regardless of our choice, to establish if Common Crawl accurately represents contents of the Web, we need to look at multiple (preferably all) levels of granularity.

Multiple authors addressed the question of whether Common Crawl is representative of the Web. Du et al. [24] performed an analysis using natural language understanding techniques. Authors compared topics estimated using (1) a complete crawl of a "very active car owner online forum" [24] and (2) pages of that online forum included in Common Crawl. That analysis yielded similar results on both datasets and authors showed that the results from Common Crawl were "no worse" than the results based on a sample obtained from the complete crawl via true random sampling. This paper, however, did not consider the issue of whether Common Crawl is representative of the Web as a whole at all.

Based on the conclusions of this paper, we can hypothesise that statistics regarding just a single website and its pages can use Common Crawl as a reliable, representative sample - provided that the coverage of that website and the number of its pages included in Common Crawl is sufficiently large (which is not always the case).

Stolz and Hepp [66] addressed the question of Common Crawl's completeness. Authors selected a set of e-commerce websites and showed that for the domains under the investigation a simple crawl using sitemaps yields coverage 2 orders of magnitude higher than what can be found by looking for the same domains in Common Crawl.

Thompson and Tong [68] indirectly addressed the question of how useful Common Crawl is for analysis of technology usage patterns. Authors discussed the issue of page duplication in the dataset. As described in Section 2.7.1, the set of URLs chosen for fetching does not contain duplicates. However, prior to April 2016, the crawler followed redirects immediately, which resulted in duplication [68, 45]. This is a significant problem with older crawls because (1) varying amount of duplication poses a challenge for longitudal studies and (2) as shown in Table 3.1, the amount of duplication in 2014 crawl is so high that in some cases using the dataset might require performing deduplication first.

Spiegler [65] investigated Common Crawl on the macro-level by looking at over- and under-represented Top-Level Domains. He compared the relative frequencies of each TLD in Common Crawl with relative frequencies provided by W3Techs Web Technology Surveys [72]. It should be noted that this comparison only includes "the top 10 million websites (...) to limit the impact of domain spammers" [72]. His analysis showed that some TLD domains were over-represented by a factor of 2.5 and others were under-represented by a factor of 5 [65]. Interestingly, top 3 most over-represented TLDs include `.gov` and `.edu` - TLDs that are restricted to selected organisations. The second most popular TLD in the dataset (`.org`), with over 230 million webpages included in the sample, is overrepresented by 50%.

Those strong biases towards some Top-Level Domains should be investigated further. Spiegler used a very old 2012 dataset [65] which might 1) contain a high level of duplication which might disproportionately affect some of the TLDs and 2) as crawling process changed over time it might not reflect the properties of new crawls.

After-mentioned studies investigate the representativeness of Common Crawl. However, to the best of my knowledge, nobody has previously attempted to establish if the dataset is representative of the Web **as a whole** on levels of granularity other than Top-Level Domains.

One of the objectives of this project was to address the shortcomings of existing literature by creating a multi-level, in-depth, comparative analysis of Common Crawl and alternative datasets.

Crawl month	Duplicated pages
2014-04	14.80%
2017-04	0.90%

Table 3.1: Comparison of amount of duplication in different crawls. Source: [68]

3.2 Impact of the crawling process

It is not possible to study the Web in its entirety due to its size and structure. We can only analyse samples of the Web that we call *Web crawls*. Bennouas and Montgolfier define a Web crawl as ”a (biased and partial) image of the Web.” [4]

A *Web crawler* is software that automatically browses the Web and saves relevant information - typically HTML content of visited pages [4]. The Web can be represented as a directed graph where vertices are pages, websites, or domains, and the edges are links between them. Crawlers differ in how they explore that graph - we say that they employ different *crawling strategies*. For example, a crawler might traverse that graph using breadth-first search, depth-first search, or somewhat randomly [4]. The choice of that strategy will affect which pages are (and are not) included in the final crawl. For example, crawlers using a bread-first search tend to ”discover” high-ranking pages early on in the crawl [57].

Parameters of the crawling process such as the choice of a *seed* will affect the crawl as well. A *seed* is the initial set of ”starting” URLs (vertices) from which the crawler starts its exploration. As the Web graph most likely contains disconnected components unreachable from other parts of the Web, and as crawlers explore the Web graph by following links, if the initial *seed* does not include pages located in some disconnected component, then that component will not be crawled. It’s difficult to establish how many disconnected components exist (or what they contain), as they might not be included in any published Web crawls. This is a significant problem as the only way to analyse properties of the Web as a whole is to make inferences based on the properties of some crawl.

An excellent example of how artefacts of the crawling process can lead to erroneous conclusions is the ”bow-tie structure” of the Web described by Broder et al. [9] which was even published by Nature [4]. According to Broder et al. [9] the Web follows a structure shown in Figure 3.1 with a large, strongly connected component (labeled SCC) in the middle which resembles a bow-tie. Meusel et al. [40] showed that some of the properties of the Web graph ”are very dependent on artefacts of the crawling process, whereas others appear to be more structural” [40] and that the bow-tie structure seems to be crawl-dependant and might not be a

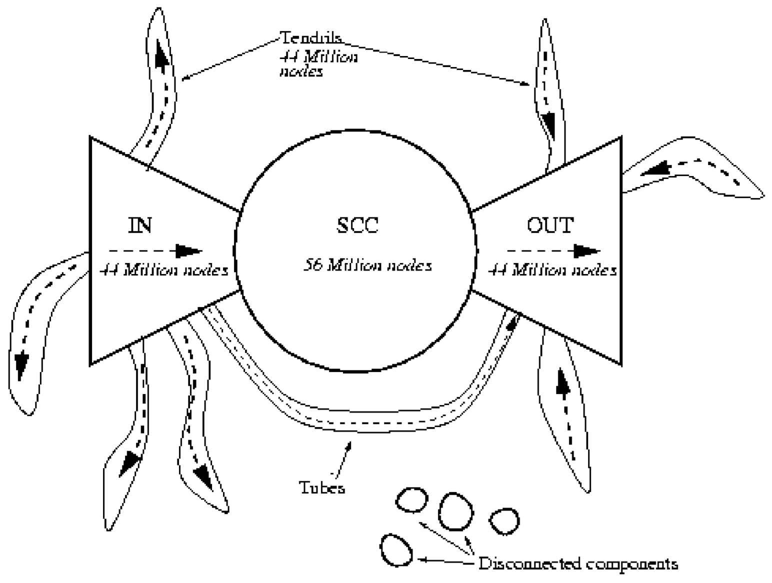


Figure 3.1: Bow tie structure of the Web. Source of the image: Broder el al. [9]

structural property of the Web.

This issue is relevant to my research question, as if a certain statistic is crawl-dependant, then we cannot use Common Crawl (or any other crawl) as a reliable source of that statistic. In this project, I (indirectly) investigated what properties of the Web are crawl-dependant, which is described in detail in Chapter 5.

3.3 Crawlers and client-side events

As detailed in Section 2.7.1, Common Crawl does not run JavaScript code, and it simply saves the HTTP response from the server. Brunelle et al. [10] developed a solution that allows crawlers to access content only reachable through clientside events - so called *deferred representations*. The study estimates that if the July 2015 Common Crawl dataset (145 TB) was collected using the after-mentioned solution, it would contain additional 597.4 TB of data. Just because it does not handle client-side events, Common Crawl ”will miss 7.17 PB of data per year” [10].

It is interesting to consider what might be missing from Common Crawl due to its inability to run client-side code. It is very difficult to establish what kinds of content might be missing, as to the best of my knowledge no large and freely-available crawls exist that could be used to perform comparisons. Due to time constraints, I did not address this issue.

3.4 Comparative analysis of Web Crawls

The methodology used in this project involves making comparisons between web crawls. Baeza-Yates, Castillo, & Efthimiadis [2] used a similar approach and analysed crawls of ”national Web” from 11 different countries. The results show significant variations in certain statistics between the crawls. For example, the average total size of all pages per website varies between 6.4MB (Brazil) and 18.1MB (Greece).

The authors do not attempt to address the representativeness of any of the crawls, but the results show that many statistics can be affected by the crawling process.

Chapter 4

Processing large-scale data

This chapter discusses the reasoning behind my choice of a distributed data-processing framework, its performance, and the issues related to processing large-scale data.

4.1 Frameworks for distributed data processing

Common Crawl is a massive dataset. For example, a crawl performed in December 2017 contains over 263 Terabytes of uncompressed data [48]. Even just the WAT files require over 21TB of disc space [48]. It is impossible to process this amount of data on a single machine in a reasonable amount of time, and the computation needs to be distributed on many machines. Distributed processing of large-scale data poses many challenges [41] and many relevant frameworks exist. I considered a few alternatives before making the final decision.

Initially, a solution based on bash scripts that was developed by my supervisor Henry S. Thompson was used. It allowed me to divide the input files evenly among the worker machines and run scripts in a distributed manner.

However, as I have previous knowledge of other data processing frameworks, I decided to investigate whether alternative solutions, in particular Apache Hadoop, have other capabilities that might be useful for my use-case.

4.2 Apache Hadoop

Hadoop is a collection of utilities that allow using clusters of computers for distributed processing of large-scale datasets [31]. Hadoop uses MapReduce - a programming model that allows writing distributed, parallel programs using very simple programming primitives [21].

In MapReduce, the programmer only needs to define two methods: a *map* function that processes the input data and generates a set of intermediate key/value pairs and a *reduce* function that merges all key/value pairs associated with the same key [21]. The framework then abstracts away the complexity of dealing with distributed systems. MapReduce is a data parallel programming model which means that executing map or reduce tasks in parallel is always equivalent to executing them sequentially, and hence it is easy to reason about the code.

Another advantage of Hadoop is that it has a built-in integration with Amazon S3 - a cloud storage solution used to host Common Crawl dataset. Hadoop enables the programmer to work with files located on an Amazon S3 over **s3a** protocol as if they were stored on the cluster he/she is using. The framework handles transferring data from Amazon S3, decompressing the files, and running the computation, all in a distributed and efficient manner.

Additionally, I considered Apache Spark, a data processing framework that can outperform Hadoop in some iterative machine learning algorithms by a factor of ten [76]. However, as it was not completely clear if there would be any performance advantages in my use case (batch processing of large-scale data) and as I had extensive experience with writing and debugging MapReduce jobs, I decided to choose Hadoop.

4.3 Performance

I performed tests to make sure that the system scales well with the increasing size of input data. I used **HDInsight** - a pre-configured cluster that allows running popular services related to data analytics such as Apache Hadoop, Spark, and Kafka [22]. The purpose of HDInsight is to allow programmers to quickly and easily create a cluster, perform all necessary computation and then deallocate the cluster after retrieving the output to minimise the costs. The main advantage of using HDInsight over using a cluster of ”manually” created Linux Virtual Machines (VMs) is that HDInsight automatically installs all required software and configures all **worker** and **master** nodes.

Slave (or **worker**) nodes are VMs that perform the actual computation (in Hadoop they execute map and reduce tasks). **Master** (or **head**) node is in charge of the ”coordination” of the cluster. It assigns tasks to worker machines, handles hardware failures, communication between workers, and so on.

For my tests, I used the default number and type of Virtual Machines. The cluster had 2 master nodes and 4 worker machines with the specification described in Table 4.1.

Master nodes

VM Type	No. of CPU cores	RAM	Local storage
D12 v2	4	28.00 GiB	200 GiB

Worker nodes

VM Type	No. of CPU cores	RAM	Local storage
D4 v2	8	28.00 GiB	400 GiB

Table 4.1: Specification of VMs used for testing

I tested that cluster by running a job performing a simple task (counting how many crawled URLs use HTTP protocol and how many use HTTPS instead) on different numbers of WAT files (data formats are explained in Section 2.7.1.7). The results are summarized in the Figure 4.1.

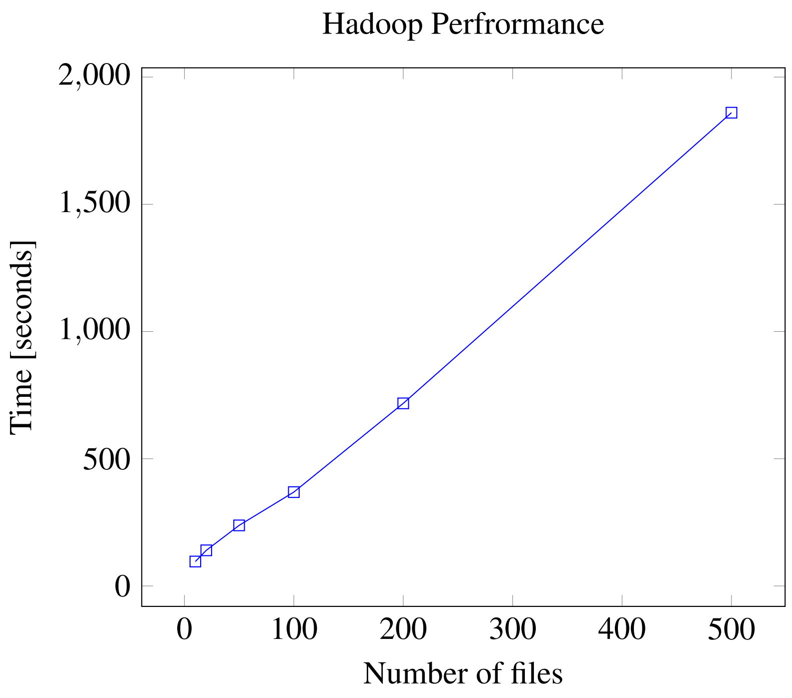


Figure 4.1: Impact of the input size on performance of Apache Hadoop

The graph shows a linear relationship between the size of the input and the time it took to process the data. Downloading, decompressing, and processing 500 WET files (approximately 160GB of compressed data) took 31 minutes. This result suggests that Hadoop should be able to process big input files without negative impact on performance. No further performance testing was done, however, a similar cluster with four more powerful worker machines was successfully used throughout the project. Specification of that cluster is shown in Table 4.2. Processing the entire dataset from April 2016 (22200 WAT files) on that Hadoop cluster was completed in less than 20 hours.

Master nodes

VM Type	No. of CPU cores	RAM	Local storage
D12 v2	4	28.00 GiB	200 GiB

Worker nodes

VM Type	No. of CPU cores	RAM	Local storage
D13 v2	8	56.00 GiB	400 GiB

Table 4.2: Specification of VMs used for running experiments

4.4 Problematic records

One of the difficulties related to long-running jobs is the danger of coming across ”rare” edge cases. For example, while working with ArabicWeb16, despite rigorous testing and the job running successfully for over 2 days with no issues, 22 map tasks failed due to an unhandled edge case - I did not anticipate that WARC files might be corrupted. Despite the problem affecting just 0.7% of all files in the dataset, this issue caused the whole job to fail.

To safeguard myself from such problems and to avoid wasting significant amounts of time and computing resources, I wanted the job to successfully finish even if up to 5% of map tasks fail. I achieved that by setting Hadoop’s `mapred.max.map.failures.percent` configuration parameter to 5%.

The output of failed tasks is ignored, and hence the output of the job will be affected. However, if the number of failed tasks is very small, the impact on the results should be negligible. In the case of experiments in which there were some failed tasks, I include a discussion of their potential impact on the results.

Chapter 5

Experiments

5.1 Methodology

As discussed in Section 3.2, when a dataset is collected through crawling, the policy employed by the crawler influences what pages will (or will not) be included in the dataset. As a result, some biases may be introduced into the data.

As it's impossible to obtain a complete crawl of the Web, it is difficult to analyse those biases. As every crawl is subject to some unknown biases, there is no source of ”absolute truth” - if we analyse two datasets and get the same results on both, then it still is possible that both datasets have the same biases, and neither is representative of the Web.

However, if we analyse crawls with widely different crawling strategies, that use a different seed (a starting set of URLs), and that are focused on different, nonoverlapping ”parts” of the Web and we still get similar results on those datasets, then that gives us more confidence that the results of our analysis are not an artefact of the crawling process and that they reflect the properties of the Web accurately. On the other hand, if the results are vastly different, this would suggest existences of some biases in one of the samples.

The methodology used in this project is based on that idea. I performed a series of experiments in which I computed a range of statistics on Common Crawl and other web crawls. I then analysed the results and attempted to explain any differences between the datasets.

5.2 Number of pages per hostname

I started my research with an investigation of the number of pages that were crawled for each **hostname**. As I discuss in Section 2.5.1, there might not be a meaningful answer to the question of how many pages are available under a particular **hostname**. Despite that, I decided to investigate this statistic for the following reasons:

1. This statistic was used by Suwaileh et al. [67] to compare ArabicWeb16 and ArClueWeb09 which gave me a chance to replicate the comparison between the two datasets and verify that my code works correctly.
2. Initially, I assumed that the issues mentioned in Section 2.5.1 have a negligible impact on real-life crawls and I performed the experiments described in this section under that assumption. Unfortunately, later in the project, I found evidence suggesting this assumption is, in fact, incorrect (see Section 5.6). Regardless, I believe that the following results provide a useful insight into the impact of the crawling process on the resulting dataset.

The average number of pages crawled for each **hostname**, showed in the Table 5.1 below, was computed for each crawl by diving the total number of pages in that crawl by the number of **hostnames** it contains. I was surprised to see that, in comparison with CommonCrawl, the value of this statistic is over 3.8 times higher for ArabicWeb16 and 2.9 times higher for ArClueWeb09. For crawls of national Webs, that I also included in the table below, the statistic differs between some crawls by a factor of 10.

Crawl	**Average**	**Crawl**	**Average**
Indochina	549	ArClueWeb09	148
Italy	410	Brazil	66
United Kingdom	248	Chile	58
South Korea	224	Spain	52
ArabicWeb16	196	Common Crawl	51
Greece	150		

Data for national crawls: Baeza-Yates and Castillo [75]. Data for ArabicWeb16 and ArClueWeb09: [67, Page 3, Table 1]. Data for Common Crawl based on [48]

Table 5.1: Average number of pages per hostname for ”national” crawls of the Web, Common Crawl, ArabicWeb16, ArClueWeb09

If we assume that those datasets do not contain significant biases, we would expect the value of this statistic to be similar for all of them. Different datasets in this comparison focus on different subsets of the Web, however, language-specific properties of the crawl should not affect the result so drastically.

I believe those discrepancies are due to the differences in how the datasets are collected. In the next section, I look at the distribution of the number of pages per **hostname** to illustrate the impact of the crawling process.

5.2.1 Distribution of the number of pages per hostname

The objective of this experiment was to investigate the distribution of the number of pages crawled for each **hostname** in Common Crawl and to compare it with other datasets. This was inspired by Yates, Castillo, and López [75] who compared a similar statistic computed on crawls of Spanish, Brasilian, Chilean, and South Korean Web [75, p. 21] and found the distribution to be similar for all of them.

In their paper, the creators of ArabicWeb16 included a histogram comparing the distribution of the number of pages crawled for each **hostname** in ArabicWeb16 and ArClueWeb09 (see Figure 5.1). I started by reproducing that histogram to make sure that my method of computing the statistic is sound.

Based on the data provided by Mucahid Kutlu, one of the authors of this paper, I confirmed my suspicion that when authors use the word ”domain” they don’t actually refer to the term **domain** as defined in Section 2.1.1, but they refer to the term **hostname** (as defined in Section 2.1.1) instead.

I counted the number of pages for each **hostname** and reproduced the aftermentioned histogram - the result is shown in the Figure 5.2 and is very similar to the original figure from ArabicWeb16 paper (see Figure 5.1). The X-axis shows page counts and the Y-axis shows the number of **hostnames** for which that many pages were crawled (on a logarithmic scale).

I then created a histogram comparing the distribution of the number of pages per **hostname** shown in Figure 5.3. Y-axis shows what percentage of hosts have a particular number of pages. As some of the bars are very small and are not clearly visible in that figure, I include the results in a tabular form as well (see Table 5.2).

The number of crawled pages per **hostname** for ArabicWeb16 and ArClueWeb09 was computed using the data shared by Mucahid Kutlu. For Common Crawl, I used so-called ”count” files available at `/crawl-analysis/CC-MAIN-2017-51/count/part-XXXXX.bz2` in Com-

In each of the the following figures 5.1 and 5.2, X-axis shows the number of pages per ***hostname*** *and Y-axis shows the number of* ***hostnames*** *that have a particular number of pages.*

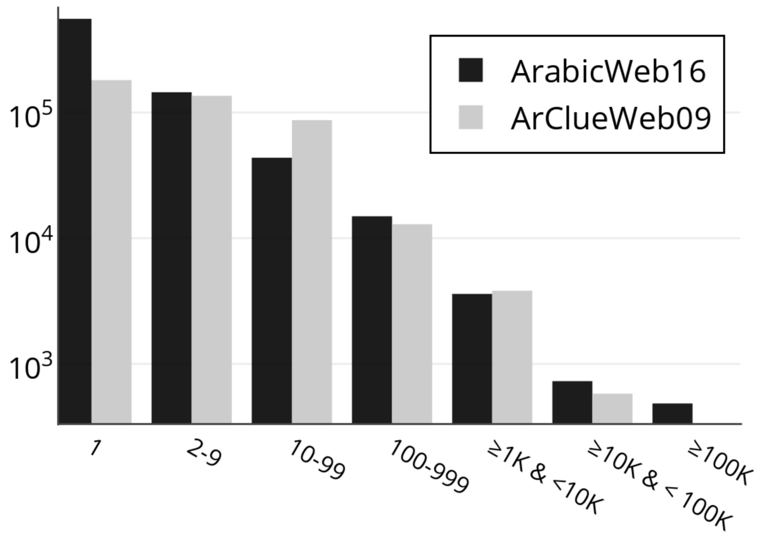


Figure 5.1: The original figure comparing the distribution of pages per hostname in ArabicWeb16 and ArClueWeb09. Source: [67, Figure 1]

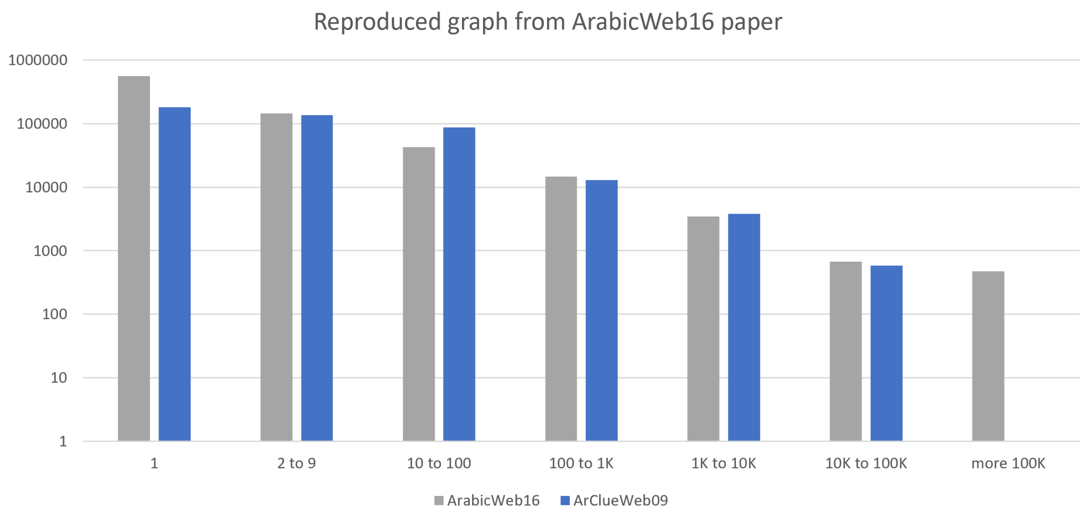


Figure 5.2: Reproduced histogram of the number of pages per hostname

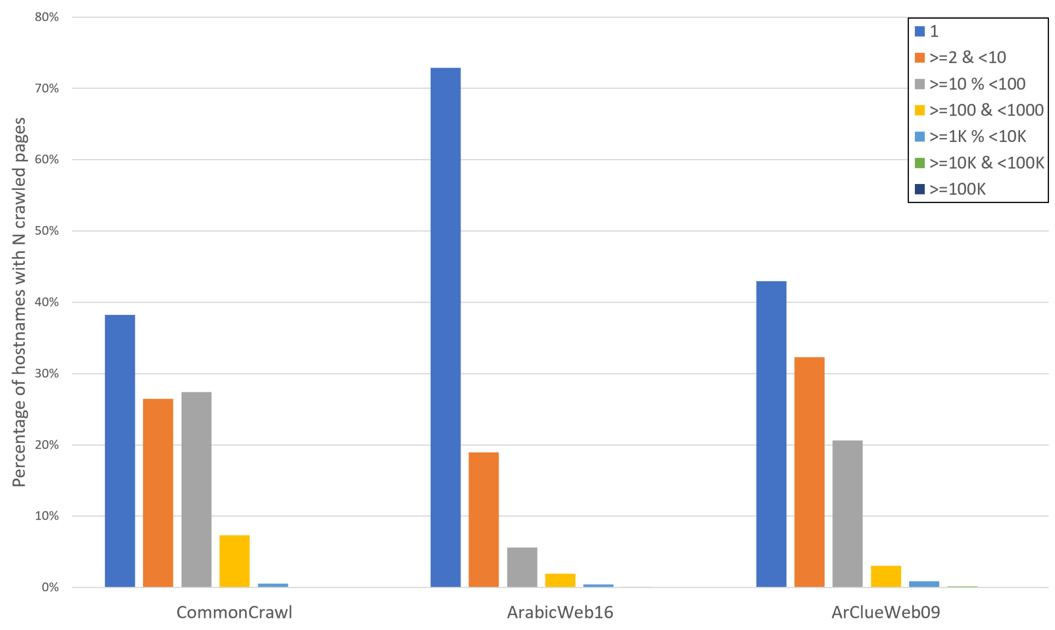


Figure 5.3: Comparison of distribution of the number of crawled pages per hostname in CommonCrawl, ArabicWeb16, and ArClueWeb09

	CommonCrawl	ArabicWeb16	ArClueWeb09
1	38.24%	72.87%	42.97%
2 to 9	26.47%	18.98%	32.29%
10 to 100	27.41%	5.62%	20.63%
100 to 1K	7.34%	1.93%	3.06%
1K to 10K	0.52%	0.45%	0.90%
10K to 100K	0.03%	0.09%	0.14%
more 100K	0.00%	0.06%	0.00%

Table 5.2: Comparison of distribution of the number of crawled pages per hostname in CommonCrawl, ArabicWeb16, and ArClueWeb09

mon Crawl's Amazon S3 bucket where `XXXXX` takes values from `00000` to `00009`.

Those files do not have standalone documentation and to understand the data format I had to inspect the comments in the source code from `commoncrawl/cc-crawl-statistics` repository [17] to decode the meaning of each record. The data format is briefly described in Appendix B.

Conclusions

A few observations can be made based on Figure 5.3. Firstly, I expected the number of pages to follow a similar distribution for all three datasets as that would be consistent with findings of Yates at al. [75, p. 21] and Modesto et al. [42, Fig. 7]. However, this was not the case - the distribution is somewhat different for each dataset.

Secondly, over 70% of **hostnames** in ArabicWeb16 only have one crawled page. This is similar to what Yates at al. [75] observed in the crawl of Spanish Web where about 60% of sites only have one crawled page. Authors concluded that this was due to their crawler not being able to follow links due to technical obstacles [75, p. 21-22].

I believe those results can be attributed to differences in the crawling method. **Hostnames** with a single page might be overrepresented in ArabicWeb16 as it actively ”pursuits” Arabic content, even if it is only linked to from pages in different languages, which might result in including only a single page from a **hostname** that has thousands of pages in different languages (technical issues with following links are a possible factor as well).

Note that Common Crawl has a higher number of **hostnames** with between 2 and 1000 pages compared to two other datasets. Common Crawl includes a particular page with a probability proportional to the rank of its **hostname**. As a result, high-ranking **hostnames** are overrepresented, and as those **hostnames** are unlikely to have just a handful of pages[1], **hostnames** with many pages are overrepresented.

This is an important result as it confirms that certain properties of the crawled content will be heavily affected by the choice of the crawling method and its parameters.

[1]This is an assumption that I manually verified by inspecting the list of 100 top ranking **hostnames** available at `http://commoncrawl.org/2017/05/hostgraph-2017-feb-mar-apr-crawls/`

5.3 Most commonly used URI schemes

The objective of this experiment was to compare the distribution of the most commonly used URI schemes. The question of how popular are different schemes was addressed by researchers in the past, for example by Felt at al. [26] who attempted to measure HTTPS adoption on the Web. However, to the best of my knowledge, no published papers attempted to measure the usage of other schemes besides HTTP and HTTPS.

I analyzed links extracted from the contents of web pages and not the list of crawled pages themselves. This is because the list of crawled URLs directly depends on the seed generation method - depending on how the seed was harvested, the same page might be crawled using `http://` or `https://`, regardless of which scheme is most commonly used to link to that page.

Similarly to Common Crawl, ArabicWeb16 uses a WARC file format. However, in the case of ArabicWeb16, those files do not contain any metadata extracted from crawled pages. This means that to extract the necessary data I had to parse HTML contents of each page, find the value of `href` attribute of each `a` tag, and then determine which scheme the link is using.

HTML code can be, and often is, syntactically incorrect. To parse HTML I used a python library `BeautifulSoup` that, to some extent, is capable of parsing ”broken” HTML. To install `BeautifulSoup` on all worker machines, I used the contents of `/etc/hadoop/conf/slaves` Hadoop configuration file to retrieve the list of worker machines, and then used `pssh` Linux command to perform the installation on all machines in parallel.

The next step was to determine which scheme is used. I developed the following ”algorithm” based on the specification of URL format published in RFC 3986 [5]. First, I used a regular expression to check if the URL is of the form:

```
<some_string_A>://<some_string_B>
```

If that was the case, I assumed that `some_string_A` is the scheme. Otherwise:

1. I assumed `href` attributes beginning with # are same-document references
2. I counted tags with an empty `href` attribute separately
3. I classified non-empty `href` attribute values that did not contain a scheme (and did not start with a #) as relative URLs

I created a Hadoop job that processed ArabicWeb16. Each map task downloaded a single WARC file and counted the URL schemes as discussed in the previous paragraph. A single

ArabicWeb16			Common Crawl		
scheme	count	Relative frequency	scheme	count	Relative frequency
http	9E+09	54.956%	relative_url	1.005E+11	44.882%
relative_url	8.6E+09	35.722%	http	6.531E+10	46.848%
https	8.7E+08	4.870%	https	8.905E+09	4.559%
#	6.3E+08	3.158%	#	5.774E+09	3.274%
empty_href	4.4E+07	0.696%	javascript	1.273E+09	0.198%
javascript	3.8E+07	0.581%	empty_href	1.061E+09	0.230%
file	616111	0.005%	ftp	9452205	0.000%
whatsapp	360102	0.004%	whatsapp	7106688	0.002%
hhttp	183247	0.002%	tel	3709570	0.000%

Table 5.3: Ten most commonly used schemes in Common Crawl and ArabicWeb16

reduce task merged the outputs from all map tasks. As parsing HTML is computationally expensive, a single map task took 30 mins to finish. However, Hadoop was able to run many map tasks in parallel and the job was completed within 68 hours.

ArabicWeb16 data was collected in January 2016. As Common Crawl data is not available for that month, I used a crawl from April 2016. Common Crawl data is also available for February 2016, however, that crawl contains almost 10% of duplicated pages [46] so I decided to avoid using it.

Processing Common Crawl was much quicker compared to ArabicWeb16, as Common Crawl metadata includes a list of all outgoing URLs for each crawled page (parsing HTML was not necessary). Despite its massive size, processing Common Crawl took only 21 hours.

5.3.1 Results

Ten most commonly used schemes in both datasets are shown in Table 5.3. `relative_url` denotes the number of relative URLs, `empty_href` denotes the number of URLs that are empty strings.

Immediately, the differences between the datasets become apparent: according to ArabicWeb16, the number of relative URLs is roughly the same as the number of URLs using `http://` scheme. However, in Common Crawl, HTTP links are 54% more common than relative URLs.

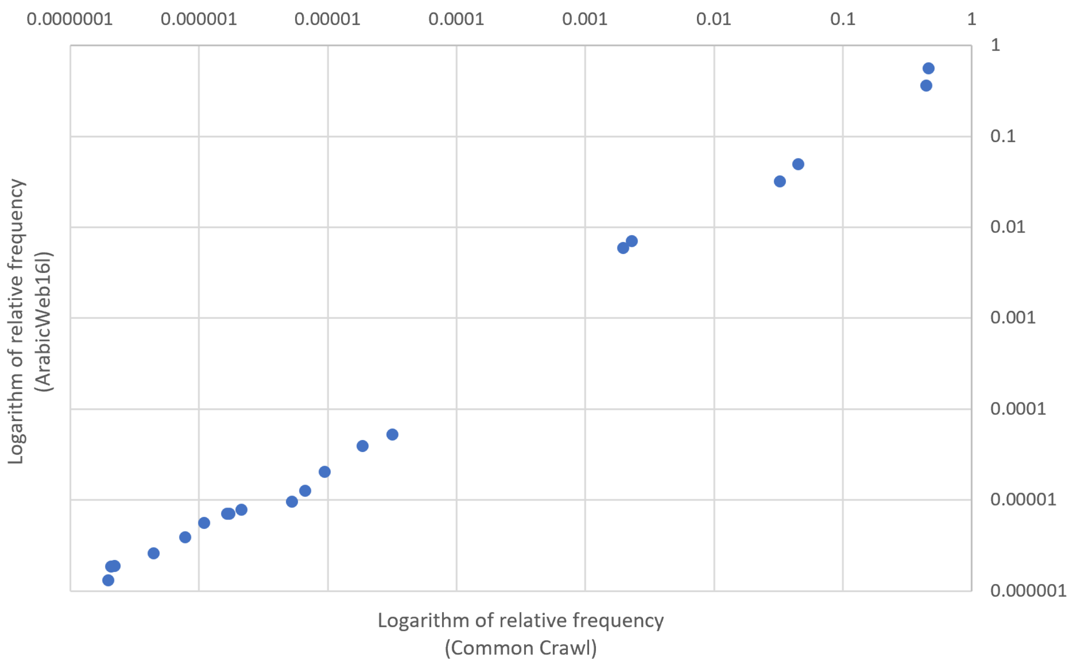


Figure 5.4: Dot i shows the relative frequency of ith most common scheme in Common Crawl and the relative frequency of ith most common scheme in ArabicWeb16.

To quantitatively measure how similar are the results from both datasets, I computed relative frequencies of each scheme. Then 1) I compared the distribution of the TOP 20 schemes in each dataset and 2) compared relative frequencies of the TOP 20 schemes in Common Crawl with their relative frequencies in ArabicWeb16.

5.3.1.1 Distribution of the TOP 20 schemes in each dataset

Every data point in Figure 5.4 compares the relative frequency of the ith most common scheme in Common Crawl with the relative frequency of the ith most common scheme in ArabicWeb16 (note that those might be two different schemes). The x-axis shows relative frequencies of schemes in Common Crawl while the y-axis shows relative frequencies of schemes in ArabicWeb16. The graph is a (nearly perfect) straight line which means that both distributions are nearly identical. The correlation coefficient for this data is 0.98.

5.3.1.2 TOP 20 schemes in Common Crawl

Given some scheme `s`, we would like to know if both datasets will ”give us” a similar relative frequency of `s`. For the next comparison:

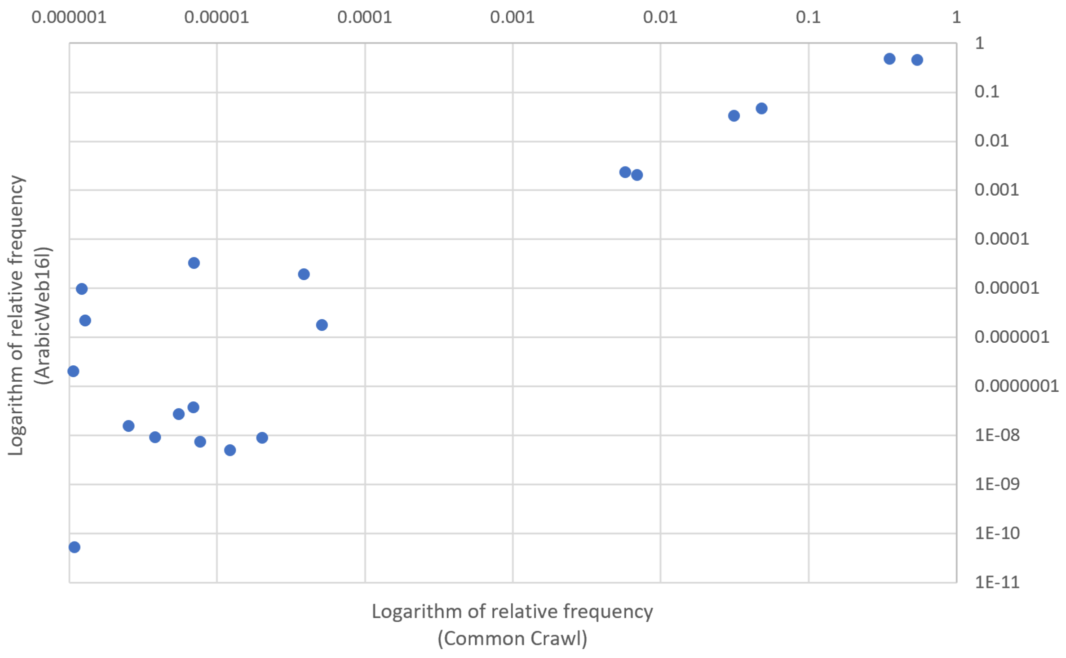


Figure 5.5: Dot i shows the relative frequency of ith most common scheme p in Common Crawl and the relative frequency of that scheme p in ArabicWeb16.

1. I narrowed my analysis to the schemes that can be found in both samples
2. From those schemes, I selected 20 most common schemes in Common Crawl
3. I computed relative frequencies of those schemes in Common Crawl and ArabicWeb16

The results are plotted in figures 5.5 and 5.6.

Each data point in Figure 5.5 corresponds to some scheme `s`. The x-axis shows the relative frequency of that scheme in Common Crawl, while the y-axis shows the relative frequency of that scheme in ArabicWeb16. The distribution in both datasets is somewhat similar, but the data points are more scattered compared to Figure 5.4. To investigate the issue further, I created a bar chart showing relative frequencies of the most commonly occurring schemes (see Figure 5.6).

The relative frequencies of some protocols differ significantly between the datasets. For example, schemes such as `tel` or `webcal` occur in Common Crawl significantly more frequently than in ArabicWeb16.

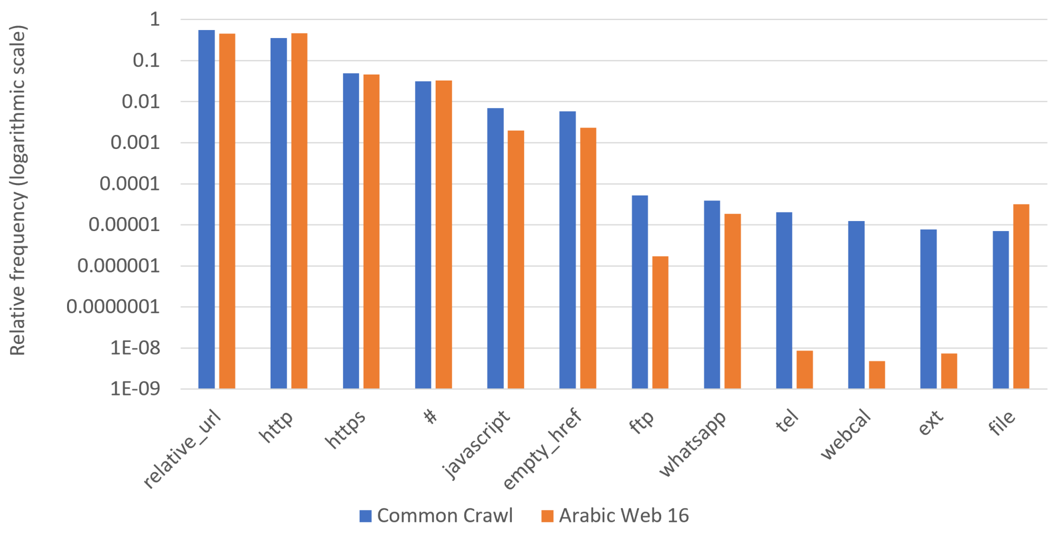


Figure 5.6: Relative frequencies of the most popular URI schemes

5.3.2 Conclusions

Getting similar results from multiple datasets would give us more confidence in that the results are representative of the Web as a whole. Distributions of relative frequencies in both datasets are nearly the same (see Figure 5.4), but what if we tried to use the results from one of the datasets? Let's assume that I need to collect 10 million relative URLs to conduct some analysis, and I want to estimate how many pages I need to fetch to collect those URLs. Using frequencies from ArabicWeb16 would give us an answer 24% higher than using Common Crawl, causing a significant increase in the cost of data collection. The differences between the datasets are even more striking if we consider less common schemes such as `webcal`.

ArabicWeb16 is limited to Arabic webpages, which introduces a variety of biases into the sample. More comparisons are needed to draw any conclusions as it's possible that 1) the differences that I observed are a result of random noise in the data or 2) both datasets are representative of the subsets of the Web which they crawl, however, the properties of those subsets might be inherently different. Using a third dataset, or repeating this comparison for an Arabic subset of Common Crawl and ArabicWeb16 could shed more light on whether 1), 2), or some combination of both is true. This comparison was not performed due to the computational cost and time limitations.

5.4 Analysis of Last-Modified HTTP header

HTTP headers are used to provide additional information besides what is included in the body of the response from the server. A commonly used header field called `Last-Modified` tells us when, according to the origin server, the requested resource was last modified [37]. This header field is not a perfect source of information - the value of this field might not accurately reflect the actual time of when the document was changed. Moreover, this header field is often left empty, contains a malformed date string, or is omitted altogether, however, for most pages this is the only source of information about their age available.

The objective of this experiment was to perform an in-depth analysis of the `Last-Modified` field and of the distribution of age of webpages in Common Crawl, ArabicWeb16, and a crawl of Greek Web from May 2004 (containing around 4 million pages). The last crawl is relatively small and a bit dated, however, relevant results were published for that dataset and its inclusion allowed for a more complete comparison.

I created two nearly identical Hadoop jobs: one that processed ArabicWeb16 and one that processed Common Crawl. I arbitrarily decided to use Common Crawl data from April 2016. Both jobs:

- extracted the value of `Last-modifed` header (for each crawled page that included that header)
- discarded the time of the day (hours, minutes, and seconds) to reduce the amount of ”noise” in the data. Additionally, that decision made it possible to merge the output of map tasks using so-called *combiners*, reducing the need for data transfer between worker machines
- aggregated results to obtain the following information:
  - for each month (e.g. September 2013), the number of pages that have been last modified in that month
  - for each day of the week, the number of pages that were last modified on that day of the week
  - for each day of the month, the number of pages that were modified on that day of the month

5.4.1 Date format

RFC1123 [7] specifies the following data format:

```
Last-Modified: Tue, 15 Nov 1994 12:45:26 GMT
```

RFC2616 [58] specifies that 2 additional date string formats should be accepted by HTTP application for backwards-compatibility. However, the document also states that HTTP applications "MUST only generate the RFC 1123 format for representing HTTP-date values in header fields" [58, Section 3.3.1]. I, therefore, decided to only include websites that use RFC1123-compliant date strings. I collected 293 million such date strings from Common Crawl. Processing all of the WAT files from April 2016 dataset on the available Hadoop cluster took less than 20 hours. Processing ArabicWeb16 took roughly 8 hours.

5.4.2 Data cleaning

In order to make meaningful comparisons, I had to "clean" the data by removing malformed records. I decided to ignore timestamps that were obvious outliers. Exclusion criteria were as follows:

- The time stamp string is empty or is not formatted correctly
- The date is later than the day the crawl has finished (date is in the future)
- The date is earlier than 1970
- The time stamp is **not** in GMT time (RFC2616 states that all "time stamps MUST be represented in Greenwich Mean Time (GMT), without exception" [58])

In comparisons in which the date is aggregated by month, I ignored months that had less than one thousand pages, as that corresponds to the relative frequency that is nearly 0.

5.4.3 Distribution of days of the week and days of the month

The dynamics of how the Web changes over time are an interesting topic of research and have practical implications, such as scheduling when web crawlers should run [8, p. 18].

I suspected that analysing distributions of days of the week and days of the month might reveal something interesting about when "content creators" - people who, in some way, publish content online are most active (for example, during the weekdays), and should be similar in both datasets. This would be consistent with finding of Brewington and Cybenko [8] who showed that changes to the Web occur most frequently on weekdays, during US work hours, and diminish on the weekends and at night.

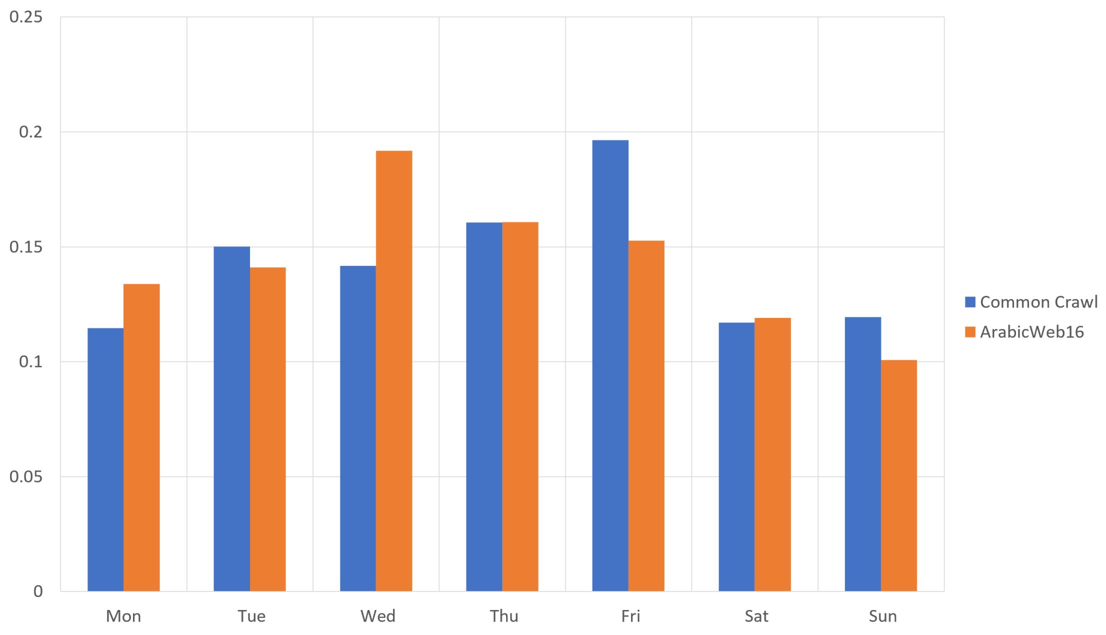


Figure 5.7: Relative frequency of each day of the week

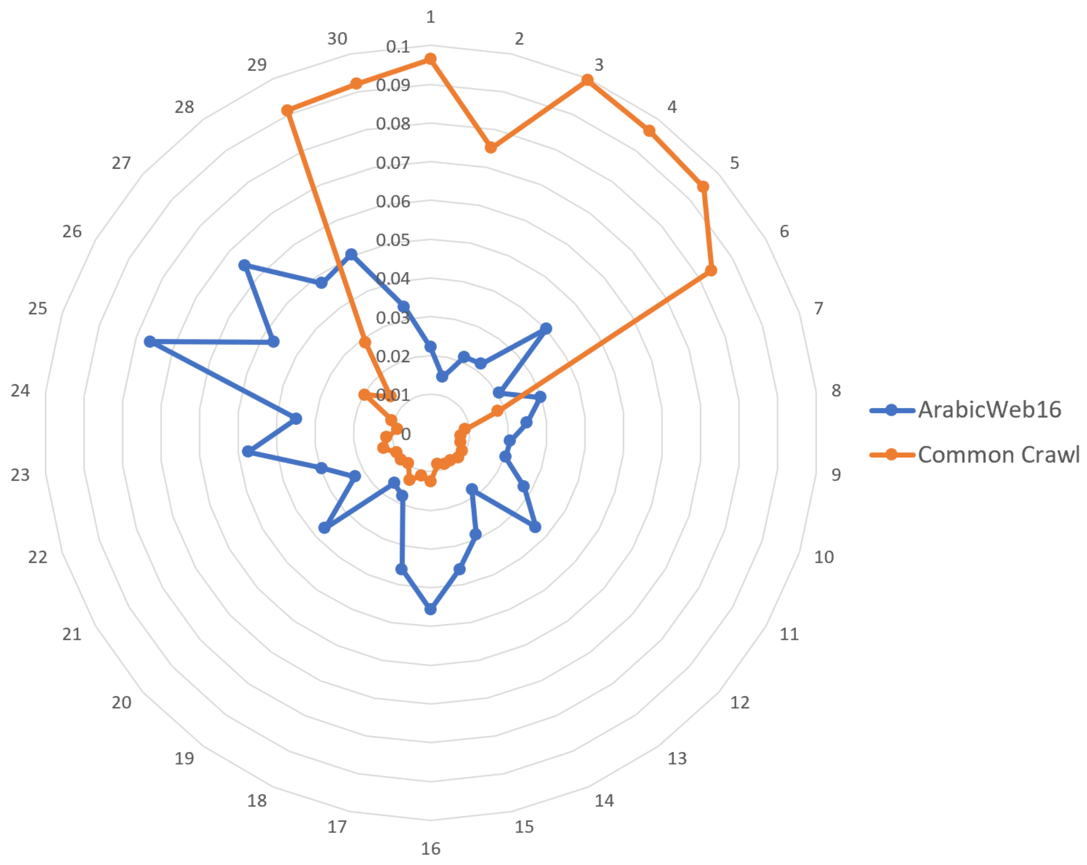


Figure 5.8: Relative frequency of each day of the month

I plotted the results in figures 5.7 and 5.8. Figure 5.7 is a ”standard” bar char, however, figure 5.8 uses so-called *radar chart* to graphically show the distribution of days of the month. I decided to use a radar chart as it was significantly easier to read. Each data point on the graph represents the relative frequency of pages that were published on a particular day. The further the data point is from the centre of the graph the higher the relative frequency.

Brewington and Cybenko [8] found that more pages are modified on weekdays (with roughly the same amount of pages modified each weekday), so I expected to see similar relative frequencies for all weekdays and a sharp drop-off on the weekend. My results, shown in Figure 5.8, are somewhat different from the results presented by Brewington and Cybenko. For ArabicWeb16 Saturdays and Sundays have the lowest relative frequencies, but in Common Crawl fewer pages were modified on Mondays than on the weekend.

To understand why this might be the case, I looked at the number of pages that were last-modified on each day of the month (Figure 5.8). The results are very surprising, as both datasets exhibit completely different distributions. I expected roughly the same number of pages to be modified on each day of the month (i.e. Figure 5.8 should show a uniform distribution). However, this is not the case.

In Common Crawl, the vast majority of pages in the dataset were modified after the 28th day of the month or before the 6th day of the month. Some days of the month outside of this range have nearly zero relative frequency. This cannot possibly be a structural property of the Web - this result shows that there is some mysterious, strong bias in Common Crawl. The following section describes how I analysed the page age distribution to identify what might be the cause.

5.5 Distribution of page age

To explain my surprising findings from the previous section, I decided to look at page age, its distribution, and how it might be affected by the crawling process.

I started by creating a comparison of the distribution of page age in Common Crawl, ArabicWeb16, and a crawl of Greek Web. I included the last crawl, as the necessary data was published by Efthimiadis and Castillo [25].

I used the procedure described in Section 5.4 to collect and clean the data from ArabicWeb16 and Common Crawl. I compared the distribution of page age of those datasets and visualised it in Figure 5.9 on page 46. The x-axis shows the page age in months, and the y-axis shows the logarithm (base 10) of the number of pages.

Page age should follow an exponential distribution [8]. My results (unsurprisingly) confirm

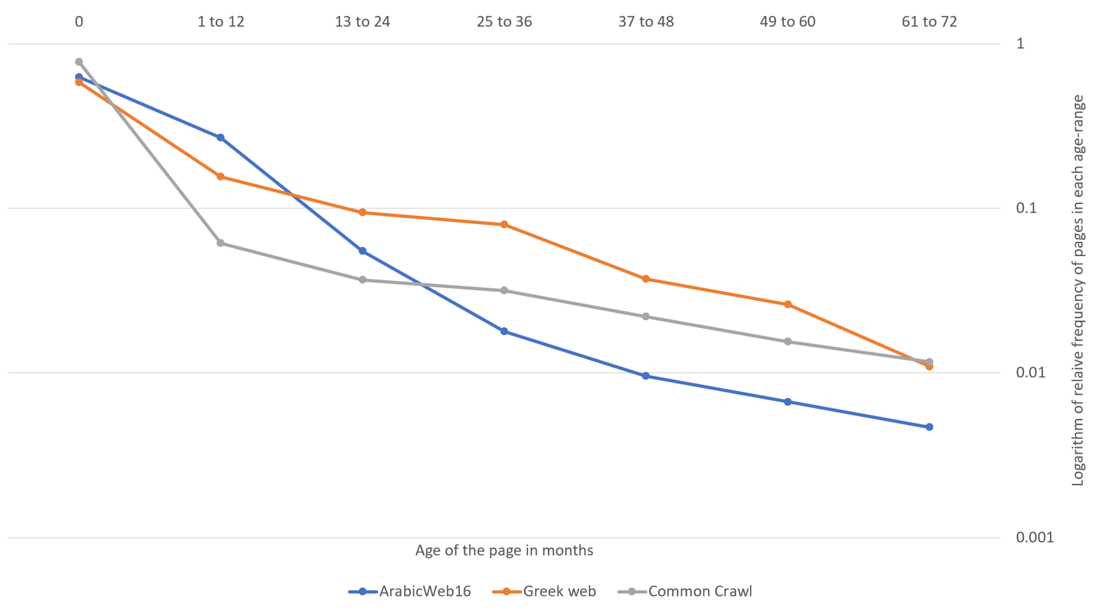


Figure 5.9: Distribution of age of pages. Source of data for Greek Web: [25]

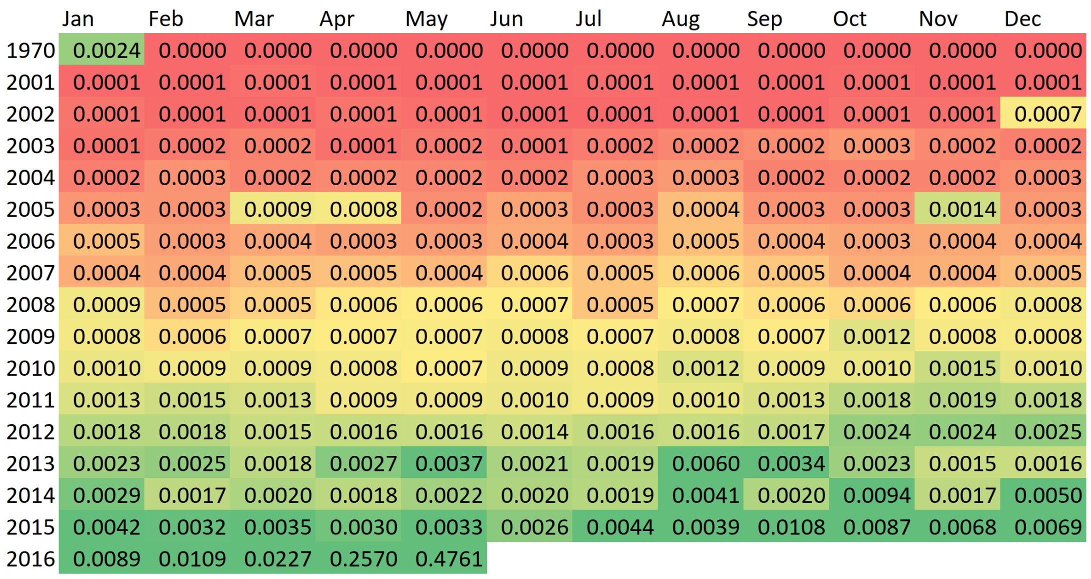
this for all three datasets. The correlation coefficient between the data from Common Crawl and Greek Web is 0.98, between the data from Common Crawl and ArabicWeb16 is 0.93.

I then created heatmaps for both datasets. Each heatmap was created as follows:

- As described in Section 5.4, I computed the number of pages that were last modified in each month (for example, in Jan 2012).
- I then computed the relative frequency for each month.
- I placed those relative frequencies in a table. The colour of each cell reflects the value in that cell - cells with the values in the upper 10% are green and cells with values in the lower 10% are red.

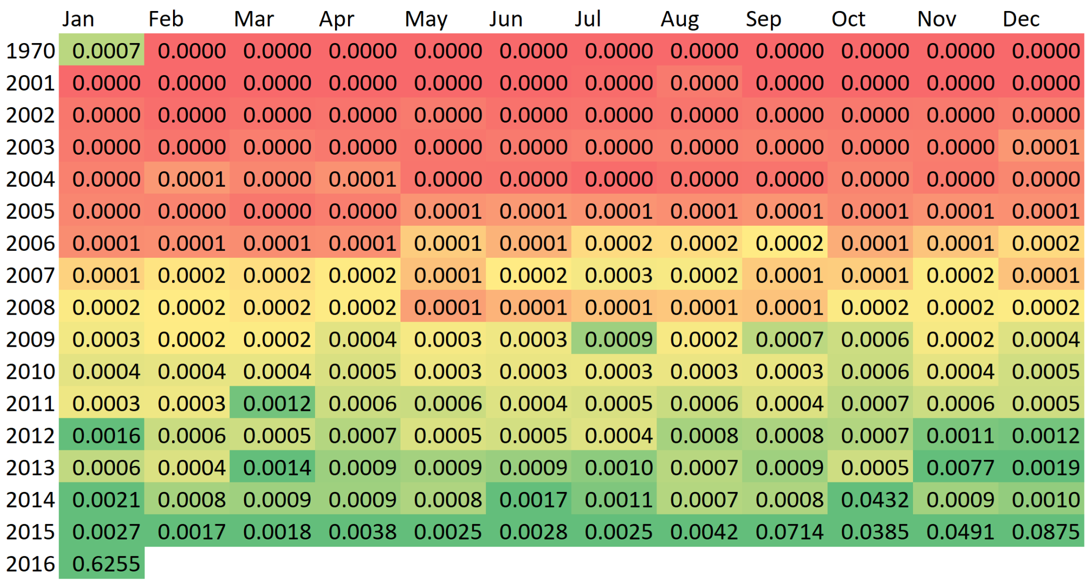
The resulting heatmaps are shown in figures 5.10 and 5.11. I only included years 2001-2016 in the heatmaps, as for those years both datasets contained a sufficient number of pages per month to draw meaningful conclusions. I also included the year 1970 to showcase how commonly January 1970 appears in `Last-Modified` headers.

What would we expect those heatmaps to look like? Page age follows an exponential distribution [8] which "can be explained by modeling page changes as a Poisson process" [2]. As a consequence, I would expect both heatmaps to be "smooth" and "gradient-like" - with green cells on the bottom and red cells on the top. However, closer inspection of figures 5.10 and 5.11 reveals "hotspots" - in both crawls we can find months for which the number of crawled pages is significantly higher than the number of pages from the proceeding and succeeding months.



	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1970	0.0024	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
2001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
2002	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0007
2003	0.0001	0.0002	0.0002	0.0001	0.0002	0.0001	0.0002	0.0002	0.0002	0.0003	0.0002	0.0002
2004	0.0002	0.0003	0.0002	0.0002	0.0002	0.0002	0.0003	0.0003	0.0002	0.0002	0.0002	0.0003
2005	0.0003	0.0003	0.0009	0.0008	0.0002	0.0003	0.0003	0.0004	0.0003	0.0003	0.0014	0.0003
2006	0.0005	0.0003	0.0004	0.0003	0.0003	0.0004	0.0003	0.0005	0.0004	0.0003	0.0004	0.0004
2007	0.0004	0.0004	0.0005	0.0005	0.0004	0.0006	0.0005	0.0006	0.0005	0.0004	0.0004	0.0005
2008	0.0009	0.0005	0.0005	0.0006	0.0006	0.0007	0.0005	0.0007	0.0006	0.0006	0.0006	0.0008
2009	0.0008	0.0006	0.0007	0.0007	0.0007	0.0008	0.0007	0.0008	0.0007	0.0012	0.0008	0.0008
2010	0.0010	0.0009	0.0009	0.0008	0.0007	0.0009	0.0008	0.0012	0.0009	0.0010	0.0015	0.0010
2011	0.0013	0.0015	0.0013	0.0009	0.0009	0.0010	0.0009	0.0010	0.0013	0.0018	0.0019	0.0018
2012	0.0018	0.0018	0.0015	0.0016	0.0016	0.0014	0.0016	0.0016	0.0017	0.0024	0.0024	0.0025
2013	0.0023	0.0025	0.0018	0.0027	0.0037	0.0021	0.0019	0.0060	0.0034	0.0023	0.0015	0.0016
2014	0.0029	0.0017	0.0020	0.0018	0.0022	0.0020	0.0019	0.0041	0.0020	0.0094	0.0017	0.0050
2015	0.0042	0.0032	0.0035	0.0030	0.0033	0.0026	0.0044	0.0039	0.0108	0.0087	0.0068	0.0069
2016	0.0089	0.0109	0.0227	0.2570	0.4761							

Figure 5.10: Heatmap of relative frequencies of pages from each month in Common crawl (April 2016)



	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1970	0.0007	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
2001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
2002	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
2003	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001
2004	0.0000	0.0001	0.0000	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
2005	0.0000	0.0000	0.0000	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
2006	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0002	0.0002	0.0002	0.0001	0.0001	0.0002
2007	0.0001	0.0002	0.0002	0.0002	0.0001	0.0002	0.0003	0.0002	0.0001	0.0001	0.0002	0.0001
2008	0.0002	0.0002	0.0002	0.0002	0.0001	0.0001	0.0001	0.0001	0.0001	0.0002	0.0002	0.0002
2009	0.0003	0.0002	0.0002	0.0004	0.0003	0.0003	0.0009	0.0002	0.0007	0.0006	0.0002	0.0004
2010	0.0004	0.0004	0.0004	0.0005	0.0003	0.0003	0.0003	0.0003	0.0003	0.0006	0.0004	0.0005
2011	0.0003	0.0003	0.0012	0.0006	0.0006	0.0004	0.0005	0.0006	0.0004	0.0007	0.0006	0.0005
2012	0.0016	0.0006	0.0005	0.0007	0.0005	0.0005	0.0004	0.0008	0.0008	0.0007	0.0011	0.0012
2013	0.0006	0.0004	0.0014	0.0009	0.0009	0.0009	0.0010	0.0007	0.0009	0.0005	0.0077	0.0019
2014	0.0021	0.0008	0.0009	0.0009	0.0008	0.0017	0.0011	0.0007	0.0008	0.0432	0.0009	0.0010
2015	0.0027	0.0017	0.0018	0.0038	0.0025	0.0028	0.0025	0.0042	0.0714	0.0385	0.0491	0.0875
2016	0.6255											

Figure 5.11: Heatmap of relative frequencies of pages from each month in ArabicWeb16 (January 2016)

For example, in Common Crawl, the number of pages from December 2002 is 7 times higher compared to November or January.

There are many possible reasons why some months are overrepresented: a commonly used software might erroneously return the same date for all of its pages, the crawl might include a website that has many pages created in a particular month (for example, a new online shop that ”went live” with thousands of product pages made available on the same day), and so on. Closer inspection of some of the hotspots could shed more light onto the issue, however, due to time constraints no further experiments were conducted.

January 1970 is a hotspot that can be easily explained. Servers often run some Unix-like operating system and hence use **unix time** (the number of seconds since 1st January 1970) to keep track of time. Hence, that date ”of zero seconds” is commonly used as a ”placeholder” value.

Both crawls contain some number of hotspots, however, those hotspots occur in different months. It is reasonable to assume that those hotspots are not an inherent property of the Web, but an artefact of the crawling process. Different months might be overrepresented depending on what websites the crawler ”stumbled upon”.

This experiment analysed `Last-Modified` date at a granularity of months. In the next section, I analyse page age on the granuality days and seconds.

5.5.1 Page age (granularity of days and seconds)

In Section 5.4.3, I showed that the vast majority of pages in Common Crawl had been modified after the 28th day of the month or before the 6th day of the month. Some days of the month outside of this range have nearly zero relative frequency.

I came up with the following hypothesis: as page age follows an exponential distribution, then at any time a significant percentage of pages on the Web are just a few hours old. Therefore, (1) the crawler would encounter mostly very ”young” pages. The conclusion would be that a crawl collected over some period would mostly contain pages last-modified during that period which would explain the unusual distribution from Figure 5.8.

I inquired on Common Crawl mailing group about how to determine when the crawl was running. Using provided instructions [53], I established that the crawl was running from the afternoon of 28th April until 7th May. Those dates correspond to the ”peak” in the distribution of the days of the month.

To verify assumption (1) I used a random number generator to select a sample of 50 WAT files

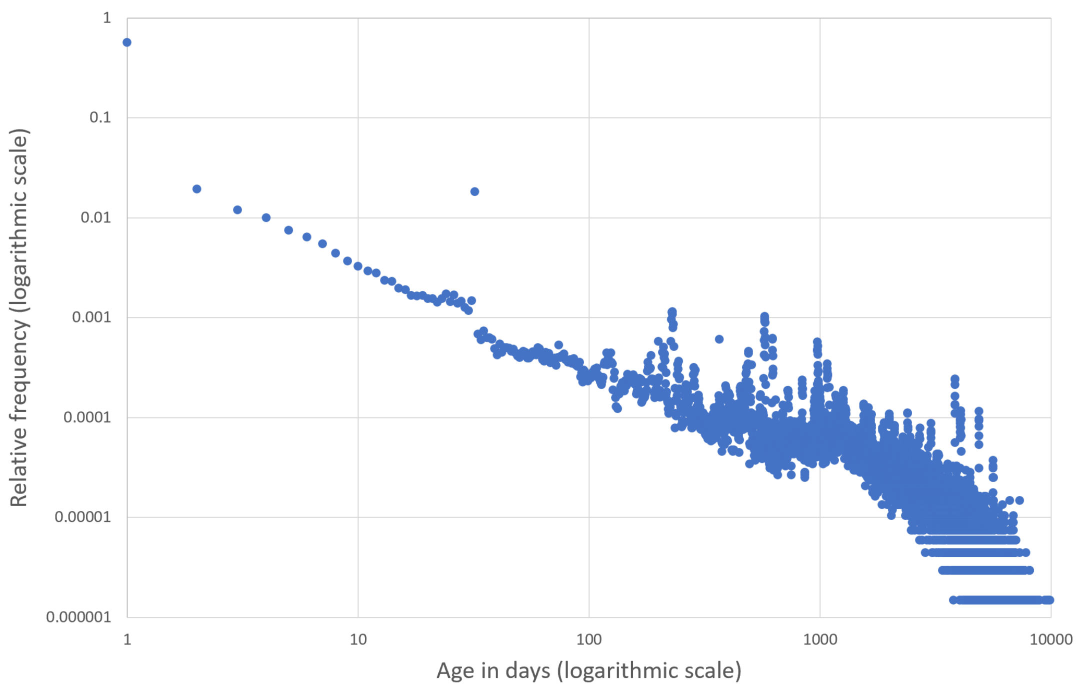


Figure 5.12: Distribution of age of pages in Common Crawl

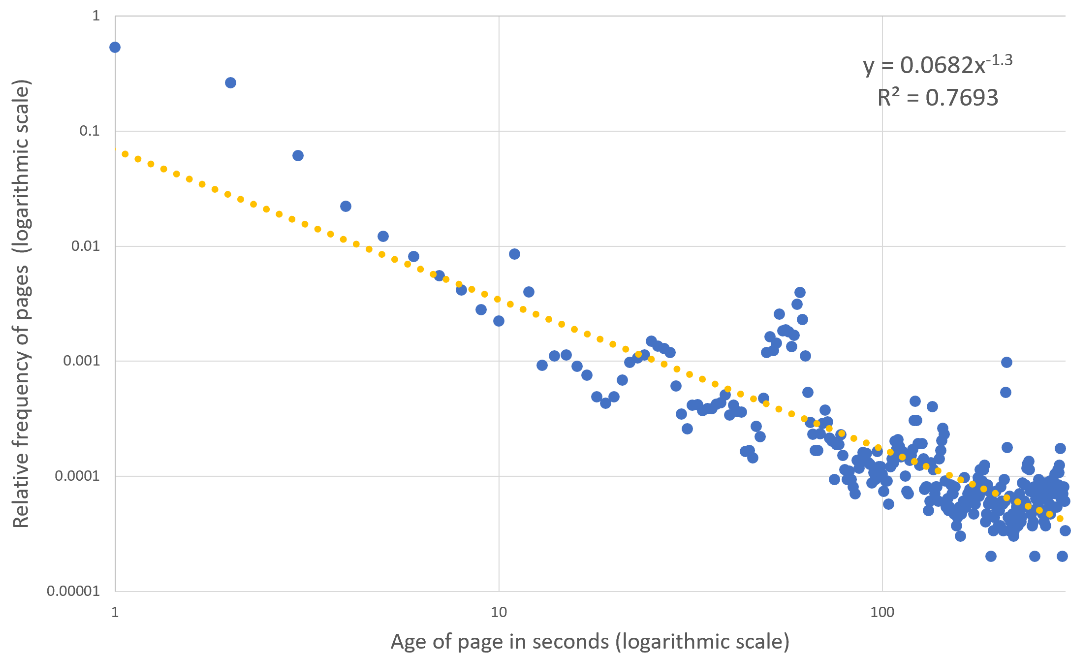


Figure 5.13: Distribution of age of pages that are no older than 5 minutes

from April 2016 crawl. A sample was used instead of the complete dataset, as when the following experiments were conducted Azure credit was temporarily unavailable and my personal machine was used to run the computation instead. Selected WAT files contained approximately 3 million Web pages (out of which over 676 000 pages had a valid `Last-Modified` timestamp). I used the procedure described in Section 5.4 to clean the data, and then I computed the time between the `Last-Modified` date and the time the page was crawled. The results are as follows:

The average age of a page is approximately 11.8 months. This is comparable to the average page age in crawls of the Web of Brazil (11.6 months), Chile (13.0 months), or Greece (17.7 months) [2]. Figure 5.12 shows the distribution of page age on the granularity of days. The x-axis is the page age, the y-axis shows the relative frequency of pages that are a particular number of days old. For example, the first data point from the left shows the relative frequency of pages that are less than 1 days old, the second data point corresponds to pages that are between 1 and 2 days old, and so on.

For pages younger than 100 days, the data seems to follow exponential distribution quite closely, however, there are two obvious outliers. The number of pages less than 1 day old and the number of pages 31 days old is much higher than expected. **Over 56% of the pages in the sample are less than 1 day old.** I began to suspect that the overrepresentation of 1-day old pages might be caused by webservers returning the current time as `Last-Modified` header, instead of the correct value. I noticed that nearly **40% of pages claim to be no older than 5 seconds** and 35% claim to be no older than 1 second. Additionally, 24% of pages have `Last-Modified` time equal to the time they were crawled (they are ”0 seconds old”).

I plotted the distribution of age of pages that are no older than 5 minutes in Figure 5.13 and I fitted an exponential distribution to the data. The first data point from the left corresponds to pages that are 0 seconds old, the seconds dot corresponds to pages that are 1 second old, and so on. If my suspicion was correct, I would expect pages that are just a few seconds old to be significantly overrepresented. Results are inconclusive - relative frequencies fit to a power-law distribution. The first few data points are significantly above the trendline, but it is hard to conclude anything definitive about why this might be the case due to the noise in the data.

5.5.2 Analysis of server software

To find further evidence confirming my suspicion that the current time is sometimes returned as `Last-Modified` header, I investigated whether there is some kind of server software that is ”responsible” for the majority of pages last-modified within the few seconds before being crawled.

Pages no older than 5 seconds		Pages older than 5 seconds	
`Apache`	23%	`GSE`	16%
`nginx`	12%	`Apache`	15%
`cloudflare-nginx`	12%	`Microsoft-IIS/7.5`	13%
`no_header`	8%	`nginx`	5%
`Microsoft-IIS/7.5`	6%	`no_header`	4%

Table 5.4: 5 most common values of `Server` HTTP header for pages older and younger than 5 seconds

I identified the server software by inspecting `Server` HTTP header string. I did not attempt to "merge" different versions of the same software together. For example, I considered `Apache 2.4 (Ubuntu)`, `Apache 2.4`, `Apache`, and `Apache Server` to be four different values of the header. This decision was due to lack of any standard of how this header field should be formatted - making arbitrary decisions on which "formats" to "parse" could introduce biases into the data.

For different values of t, I compared the most common `Server` header values for pages that are no older than t seconds, and pages older than t seconds. The results for $t = 5$ are summarised in Table 5.4. I used `no_header` to denote pages that do not contain `Server` header.

The results are inconclusive. GSE is found in 16% of pages older than 5 seconds, but only 0.05% of pages no older than 5 seconds use it. Apache and Nginx seem somewhat over-represented among "young pages" but there are no other "obvious outliers". Setting a threshold to a different value (e.g. 1 second) makes the results even more ambiguous. It's very unlikely that any particular software is responsible for the high number of "young" pages.

5.5.3 Conclusions

In Section 5.4, I described how I discovered crawling artefacts in Common Crawl related to the `Last-Modified` header field. In this section, I described how I investigated potential causes of those artefacts.

I showed that the "spike" in the distribution of relative frequencies of the days of the month corresponds to when the crawl was performed. I also showed that, surprisingly, 40% of pages are no older than 5 seconds.

I then investigated whether the prevalence of very young pages is a result of current time being used as the `Last-Modified` header. There is no convincing evidence to suggest that this is the case, however, pages that are just a few seconds old are overrepresented in the sample. There is also no convincing evidence to suggest that a particular server software might be responsible for the observed results.

The conclusion is that most likely the observed distribution of page age is an inherent property of the Web, and that it was captured by Common Crawl accurately. However, that means that certain statistics will be affected by the choice of the time when the crawl was performed - this is exemplified by the unusual distribution of days of month discussed previously. This needs to be taken into consideration when analysing data.

My findings do **not** explain the hotspots observed in heatmaps 5.10 and 5.11. Those hotspots should be inspected individually in the future, as they might shed more light on potential biases in Common Crawl.

5.6 Query and fragment URI components

As discussed in Section 2.5, pages generated based on the query parameters in the URL can be problematic when it comes to analysing properties of the Web, as in some cases they can be used to generate an unbounded number of different pages.

I initially believed that this is a theoretical issue with negligible impact on ”real-world” crawls of the Web. To verify this assumption I decided to check how many URLs that differ only in query parameters are included in February 2016 crawl. I used a sample of 50 randomly selected WAT files containing over 3 million crawled pages. I then:

- verified that there were no duplicate URLs in that sample
- counted URLs with a query component
- discarded query components from all URLs and computed the number of duplicates. If a URL was repeated 4 times, I would count it as 1 unique URL and 3 ”duplicates”
- checked how many URLs contain a fragment component - my sample contained no such URLs (this is expected, as URL fragments should be removed when constructing the GET request)

Out of 3 million URLs, 1.4 million (46%) have a query component. Surprisingly, out of those URLs, only 0.5 mln (35%) are unique if we disregard the query component (see Figure 5.14).

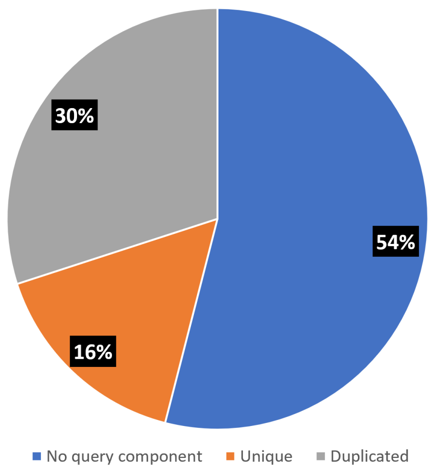


Figure 5.14: Disregarding URL query components results in a significant amount of URL duplication

Those results do **not** mean that Common Crawl contains millions of duplicates, they simply tell us that the issues discussed in Section 2.3 have a nonnegligible impact on real-world datasets, as there is some (unknown) number of crawled pages which might be similar to each other. This could be considered a type of duplication that cannot be detected using hash-based methods. To get a definitive answer on how commonly this occurs, we would need to compute some kind of distance measure between all pairs of webpages in the crawl, and decide on a threshold value for which we consider pages to be duplicates.

Table 5.5 lists URLs with the highest amount of duplication (after removing the query component). Best Buy's search page `http://www.bestbuy.com/site/searchpage.jsp` is an excellent example of a "problematic" URL - similarly to `zalando.co.uk` search page showed in Figure 2.2, this URL allows generation of an arbitrary number of page by varying the **search string** - in fact, my sample of Common Crawl data contains 1421 such URLs (that differ only by the search string).

Whether what I observed would be considered "a bug or a feature" of the dataset is unclear and depends on where we draw the line at how much dissimilarity we require to consider two pages "different".

URL (after removing query component)	Count
`http://www.dailytech.com/article.aspx`	1580
`http://www.tndeer.com/tndeertalk/ubbthreads.php`	1477
`http://www.fangraphs.com/leaders.aspx`	1461
`http://www.vesalia.de/`	1445
`http://orthodoxwiki.org/index.php`	1433
`http://www.bestbuy.com/site/searchpage.jsp`	1424

Table 5.5: URLs with the highest amount of duplication after removing the query component

Chapter 6

Future Work

In this report, I explored a variety of Web statistics. However, due to time limitations, I did not address many interesting questions related to Common Crawl's representativeness. During the second part of the project, I'm going to investigate the ”hotspots” that I found in the heatmaps presented in Section 5.5, as well as the unusually high number of pages that are 31 days old. My analysis of `Last-Modified` date does not distinguish between static and dynamic webpages. However, it should be investigated if those two types of pages have different distributions of `Last-Modified` date. Moreover, I will verify that the page metadata included in WAT files, and most importantly the list of outgoing links, accurately represents the actual contents of the crawled pages. This is important, as in my experiments I frequently used that metadata. Additionally, I will investigate how complete is Common Crawl, as this topic has not been adequately covered in research papers. Questions that should be addressed include: given a random **domain**, what is the probability that Common Crawl contains that **domain**? What coverage of that **domain** can we expect? How are the answers to those questions affected by the **domain**'s rank? Last but not least, I will analyze the distribution of Top-Level Domains.

Additional work can be performed if the previously mentioned tasks are completed on time.

Further comparisons (most notably, of distributions of the number of outgoings links and of the page size) would allow investigating biases introduced by the crawling process further. Comparisons with Microsoft Academic Graph can be used to establish if Common Crawl can be reliably used to investigate scientific publications. Pages which load embedded resources via JavaScript (so-called ”deferred representations”) should be investigated as well, as to the best of my knowledge, their impact on the representativeness of web crawls has not been addressed. Last but not least, in this report I did not investigate the graph properties of the Web. As there is a considerable amount of relevant literature on the topic, looking at those properties would be interesting as well.

Chapter 7

Discussion and Conclusions

In this project, I investigated the representativeness of Common Crawl. I explored theoretical issues such as the impact of spam pages and content generated based on the URL query parameters, and I empirically verified that the latter issue has practical significance. I showed that those issues imply that some questions regarding the properties of the Web are ill-posed and do not have a meaningful answer. I examined what are the desired properties of a representative sample of the Web, and demonstrated that Common Crawl contains inherent biases due to its data collection policy, and hence it is difficult to conclude if it can be considered representative of the Web. I also showed that Common Crawl's architecture underwent significant changes in the past years, which can be problematic if the dataset is used in a longitudinal study.

I performed a comparative analysis of Common Crawl and alternative datasets, and showed a range of inconsistencies and anomalies, part of which I explained by peculiarities of the data collection process, and part of which remained unexplained.

My methodology was based on performing comparisons between datasets with different crawling methods in the hope of finding similarities. Those similarities could give us more confidence in that the observed results are inherent properties of the Web, and not just artefacts of the crawling process. However, I found inconsistencies of varying significance for every statistic that I analysed.

In Section 5.2, I showed very significant differences between distributions of the number of crawled pages per **hostname** in Common Crawl, ArabicWeb16 and ArClueWeb09. I discovered that the average number of pages per **hostname** can vary between crawls by a factor of 10. I also revealed differences between distributions of most commonly used URL schemes in Common Crawl and ArabicWeb16 (see Section 5.5). I also discovered the existence of crawling artefacts in the distribution of `Last-Modified` date that I was not able to explain.

However, it should be pointed out that Common Crawl, even if it contains some biases, remains incredibly useful for Web research. It's the biggest available dataset of its kind and contains metadata which would be very computationally expensive to extract from raw HTML code. Its crawling process is focused on popular pages to reflect the kinds of content that an average user interacts with. Arguably, those kinds of content are what the researchers are most interested in.

The primary objective of this project is to establish if Common Crawl is a reliable source of Web statistic, and to what extent. Unfortunately, in the light of empirical results that I obtained, I cannot provide a definitive answer to this research question. Observations described in previous paragraphs do not imply conclusively that Common Crawl is not a useful source of Web statistics either, but that further research is needed.

I described some of Common Crawl's shortcomings, and tried to establish what kinds of statistics are crawl dependant. Throughout this report, I commented on how the data collection process affects different aspects of the sample. I established that Common Crawl might not be suitable for studies investigating `Last-Mofidied` date, as the exponential distribution of page age can result in unusual results for some statistics. If a crawl needs to be used to investigate trends and statistics related to `Last-Modified` header, then the data needs to be collected over a longer period of time. Otherwise, the patterns observed in the data might be caused by the timing of the crawl, which makes drawing any meaningful conclusions impossible. Common Crawl is typically collected within just a few days, which means that it might not be a useful dataset for such research. As discussed in Section 3.1, based on the results published by Du et al. [24], we can conclude that statistics regarding just a single website and its pages can use Common Crawl as a reliable, representative sample of that website, provided that its coverage and the number of its pages included in Common Crawl are sufficient. It is not clear if Common Crawl can be reliably used to source other types of statistics, and further work is needed.

However, despite its shortcomings, because of its sheer size and diversity of its contents, in my opinion, Common Crawl is the best readily available source of Web statistics.

Bibliography

[1] Sajjad Arshad, Seyed Ali Mirheidari, Tobias Lauinger, Bruno Crispo, Engin Kirda, and William K. Robertson. Large-scale analysis of style injection by relative path overwrite. *CoRR*, abs/1811.00917, 2018. Retrieved: April 4, 2019, from `http://arxiv.org/abs/1811.00917`.

[2] Ricardo A. Baeza-Yates, Carlos Castillo, and Efthimis N. Efthimiadis. Characterization of national web domains. *ACM Trans. Internet Techn.*, 7(2):9, 2007. Retrieved: September 23, 2018, from `https://doi.org/10.1145/1239971.1239973`.

[3] Namrata HS Bamrah, BS Satpute, and Pramod Patil. Web forum crawling techniques. *International Journal of Computer Applications*, 85(17), 2014.

[4] Toufik Bennouas and Fabien de Montgolfier. Random web crawls. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 451–460, 2007.

[5] Tim Berners-Lee, Roy T. Fielding, and Larry Masinter. Uniform resource identifier (URI): generic syntax. *RFC*, 3986:1–61, 2005. Retrieved: April 4, 2019, from `https://doi.org/10.17487/RFC3986`.

[6] Lennart Björneborn and Peter Ingwersen. Toward a basic framework for webometrics. *JASIST*, 55(14):1216–1227, 2004. Retrieved: April 3, 2019, from `https://doi.org/10.1002/asi.20077`.

[7] Robert T. Braden. Requirements for Internet Hosts - Application and Support. RFC 1123, October 1989. doi: 10.17487/RFC1123. Retrieved: March 20, 2019, from `https://rfc-editor.org/rfc/rfc1123.txt`.

[8] Brian E. Brewington and George Cybenko. How dynamic is the web? *Computer Networks*, 33(1-6):257–276, 2000. Retrieved: September 28, 2018, from `https://doi.org/10.1016/S1389-1286(00)00045-1`.

[9] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan,

Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web. *Computer networks*, 33(1-6):309–320, 2000.

[10] Justin F. Brunelle, Michele C. Weigle, and Michael L. Nelson. Archival crawlers and javascript: Discover more stuff but crawl more slowly. In *2017 ACM/IEEE Joint Conference on Digital Libraries, JCDL 2017, Toronto, ON, Canada, June 19-23, 2017*, pages 1–10, 2017.

[11] Volha Bryl, Christian Bizer, and Heiko Paulheim. Gathering alternative surface forms for dbpedia entities. In *Proceedings of the Third NLP&DBpedia Workshop (NLP & DBpedia 2015) co-located with the 14th International Semantic Web Conference 2015 (ISWC 2015), Bethlehem, Pennsylvania, USA, October 11, 2015.*, pages 13–24, 2015.

[12] Jamie Callan, Mark Hoy, Changkuk Yoo, and Le Zhao. Web08pr dataset (project planning document). Retrieved: April 3, 2019, from `http://boston.lti.cs.cmu.edu/Data/web08-bst/web08-bst-Sep26-08.pdf`.

[13] Jamie Callan, Mark Hoy, Changkuk Yoo, and Le Zhao. The ClueWeb09 Dataset, 2009. Retrieved: September 28, 2018, from `http://boston.lti.cs.cmu.edu/classes/11-742/S10-TREC/TREC-Nov19-09.pdf`.

[14] Common Crawl. Common Crawl (Homepage), n. d. Retrieved: April 3, 2019, from `http://commoncrawl.org`.

[15] LinkedIn Corporation. LinkedIn Corporation v. Doe Defendants, 2016. Retrieved: December 12, 2018, from `https://digitalcommons.law.scu.edu/cgi/viewcontent.cgi?referer=https://benbernardblog.com/&httpsredir=1&article=2261&context=historical`.

[16] Common Crawl. Statistics of common crawl monthly archives. Retrieved: September 28, 2018, from `https://commoncrawl.github.io/cc-crawl-statistics/plots/crawlermetrics`.

[17] Common Crawl. cc-crawl-statistics/crawlstats.py (source code), Last updated Dec 2018. Retrieved: September 28, 2018, from `https://github.com/commoncrawl/cc-crawl-statistics/blob/master/crawlstats.py`.

[18] Common Crawl. Curious about what we do?, n. d. Retrieved: September 28, 2018, from `http://commoncrawl.org/big-picture/what-we-do/`.

[19] Guilherme T. de Assis, Alberto H. F. Laender, Altigran Soares da Silva, and Marcos André Gonçalves. The impact of term selection in genre-aware focused crawling.

In *Proceedings of the 2008 ACM Symposium on Applied Computing (SAC), Fortaleza, Ceara, Brazil, March 16-20, 2008*, pages 1158–1163, 2008.

[20] Maurice de Kunder. Daily estimated size of word wide web. Retrieved: April 4, 2019, from `https://www.worldwidewebsize.com/`.

[21] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[22] Microsoft Azure Documentation. Hdinsight. `https://azure.microsoft.com/en-gb/services/hdinsight/`.

[23] Microsoft Azure documentation. Availability and reliability of apache hadoop clusters in hdinsight, March 2018. `https://docs.microsoft.com/en-us/azure/hdinsight/hdinsight-high-availability-linux`.

[24] Yuheng Du, Alexander Herzog, Andre Luckow, Ramu Nerella, Christopher Gropp, and Amy Apon. Representativeness of latent dirichlet allocation topics estimated from data samples with application to common crawl, 2017. Retrieved: April 4, 2019, from `http://alexherzog.net/files/IEEE_BigData_2017_Representativeness_of_LDA.pdf`.

[25] Efthimis Efthimiadis and Carlos Castillo. Charting the greek web. In *Proceedings of the Conference of the American Society for Information Science and Technology (ASIST), Providence, Rhode Island, USA*, 2004.

[26] Adrienne Porter Felt, Richard Barnes, April King, Chris Palmer, Chris Bentzel, and Parisa Tabriz. Measuring https adoption on the web. In *26th USENIX Security Symposium*, pages 1323–1338, 2017.

[27] Olivier Fercoq. Pagerank optimization applied to spam detection. In *2012 6th International Conference on Network Games, Control and Optimization (NetGCooP)*, pages 127–134. IEEE, 2012.

[28] R. Fielding and J. Reschke. Hypertext transfer protocol (http/1.1): Conditional requests. Technical Report 7232, RFC Editor, June 2014. ISSN 2070-1721, `https://tools.ietf.org/html/rfc7232#section-2.2`.

[29] Antonio Gulli and Alessio Signorini. The indexable web is more than 11.5 billion pages. In *Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005 - Special interest tracks and posters*, pages 902–903, 2005.

[30] Satinder Bal Gupta. The issues and challenges with the web crawlers. *International Journal of Information Technology & Systems*, 1(1):1–10, 2012.

[31] Apache Hadoop. Homepage, n.d. Retrieved: April 4, 2019, from `https://hadoop.apache.org/`.

[32] Glenn Isreal. Determining sample size, 2013. Retrieved: December 12, 2018, from `https://www.tarleton.edu/academicassessment/documents/Samplesize.pdf`.

[33] Vasilis Kolias, Ioannis Anagnostopoulos, and Eleftherios Kayafas. Exploratory analysis of a terabyte scale webcorpus, 2014. Retrieved: April 4, 2019, from `https://arxiv.org/abs/1409.5443`.

[34] John Kurkowski. john-kurkowski/tldextract (Github reposity), Last updated Jan 2018. `https://github.com/john-kurkowski/tldextract`.

[35] Chung Hun Lee and David A Cranage. Personalisation–privacy paradox: The effects of personalisation and privacy assurance on customer responses to travel web sites. *Tourism Management*, 32(5):987–994, 2011.

[36] Internet Library. Internet Archive v. Suzanne Shell, 2007. Retrieved: April 4, 2019, from `http://www.internetlibrary.com/cases/lib_case456.cfm`.

[37] MDN Web Docs. Last-modified. `https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Last-Modified`.

[38] Stephen Mendez. The difference between dynamic static web pages. Retrieved from http://smallbusiness.chron.com/difference-between-dynamic-static-pages-69951.html, August 2018.

[39] Stephen Merity. Navigating the warc file format, April 2014. Retrieved: April 3, 2019, from `http://commoncrawl.org/2014/04/navigating-the-warc-file-format/`.

[40] Robert Meusel, Sebastiano Vigna, Oliver Lehmberg, and Christian Bizer. Graph Structure in the Web Revisited, 2014. Retrieved: April 3, 2019, from `http://www.quantware.ups-tlse.fr/FETNADINE/papers/P4.9.pdf`.

[41] Kamal Sheel Mishra and Anil Kumar Tripathi. Some issues, challenges and problems of distributed software system, 2014. Retrieved: April 3, 2019, from `http://ijcsit.com/docs/Volume%205/vol5issue04/ijcsit2014050420.pdf`.

[42] Marco Modesto, Álvaro Rodrigues Pereira Junior, Nivio Ziviani, Carlos Castilho, and Ricardo Baeza Yates. Um novo retrato da web brasileira. 2005. In Proceedings of SEMISH, Sao Leopoldo, Brazil.

[43] Derek Monner. InternetDomainNameExplained, October 2017. https://github.com/google/guava/wiki/InternetDomainNameExplained.

[44] Tim Morris. Pages versus urls, and uniqueness of wat file entries. https://groups.google.com/forum/#!searchin/common-crawl/segments|sort:date/common-crawl/DdEjqaVRwfg/O2X6CVVNCAAJ.

[45] Tim Morris. Pages versus urls, and uniqueness of wat file entries. https://groups.google.com/forum/#!msg/common-crawl/DdEjqaVRwfg/q1qlACsrCAAJ.

[46] Sebastian Nagel. April 2016 crawl archive now available, May 2016. Retrieved: April 4, 2019, from http://commoncrawl.org/2016/05/april-2016-crawl-archive-now-available/.

[47] Sebastian Nagel. Javascript, December 2016. Retrieved: April 2, 2019, from https://groups.google.com/forum/#!searchin/common-crawl/javascript%7Csort:date/common-crawl/jOgV2RZCV2E/0sT-I9m9BQAJ.

[48] Sebastian Nagel. December 2017 crawl archive now available, 2017. Retrieved: April 3, 2019, from http://commoncrawl.org/2017/12/december-2017-crawl-archive-now-available/.

[49] Sebastian Nagel. List of root domains?, 2017. Retrieved: April 4, 2019, from https://groups.google.com/forum/#!topic/common-crawl/vsD4vBpDdG0.

[50] Sebastian Nagel. November 2017 crawl archive now available, November 2017. Retrieved: April 2, 2019, from http://commoncrawl.org/2017/11/november-2017-crawl-archive-now-available/.

[51] Sebastian Nagel. Now Available: Host- and Domain-Level Web Graphs, 2018. Retrieved: April 3, 2019, fromhttp://commoncrawl.org/2018/02/webgraphs-nov-dec-2017-jan-2018/.

[52] Sebastian Nagel. February 2019 crawl archive now available, March 2019. http://commoncrawl.org/2019/03/february-2019-crawl-archive-now-available/.

[53] Sebastian Nagel. How much time does the crawl take?, February 2019. Retrieved: April 2, 2019, from https://groups.google.com/forum/#!topic/common-crawl/S5EndpibI6Y.

[54] Sebastian Nagel. Commoncrawl's crawling process, December 2018. https://groups.google.com/forum/?hl=en#!topic/common-crawl/OJW0g2PBVeM.

[55] Sebastian Nagel. New to Common Crawl - Missing domains?, October 2018. https://groups.google.com/forum/#!msg/common-crawl/6iTPnQrtGLs/4bCw6HNzCAAJ.

[56] Sebastian Nagel. How complete is common crawl?, Sept 2016. https://groups.google.com/d/msg/common-crawl/xmSZX85cRjg/iqADMxH_AgAJ.

[57] Marc Najork and Janet L Wiener. Breadth-first crawling yields high-quality pages. In *Proceedings of the 10th international conference on World Wide Web*, pages 114–118. ACM, 2001.

[58] Henrik Frystyk Nielsen, Jeffrey Mogul, Larry M Masinter, Roy T. Fielding, Jim Gettys, Paul J. Leach, and Tim Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, June 1999.

[59] Edward T. O'Neill, Brian F. Lavoie, and Rick Bennett. Trends in the evolution of the public web: 1998 - 2002. *D-Lib Magazine*, 9(4), 2003.

[60] AdSense Help page. Ads troubleshooting. enable javascript in your browser to see ads on your site. Retrieved: April 2, 2019, from https://support.google.com/adsense/answer/12654?hl=en.

[61] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[62] M. Angeles Serrano, Ana Maguitman, Marian Boguna, Santo Fortunato, and Alessandro Vespignani. Decoding the Structure of the WWW: A Comparative Analysis of Web Crawls, 2007. Retrieved: April 2, 2019, from http://sites.google.com/site/cxnets/acm_tweb.pdf.

[63] Ajay Singh and Micah Masuku. Sampling techniques & determination of samplesize in applied statistics research: An overview, 2014. International Journal of Economics, Commerce and Management, Vol. II, Issue 11, Nov 2014, United Kingdom, ISSN 2348 0386, Retrieved: April 4, 2019, from http://ijecm.co.uk/wp-content/uploads/2014/11/21131.pdf.

[64] Abu Bakr Soliman, Kareem Eissa, and Samhaa R El-Beltagy. Aravec: A set of arabic word embedding models for use in arabic nlp. *Procedia Computer Science*, 117:256–265, 2017.

[65] Sebastian Spiegler. Statistics of the common crawl corpus 2012, June 2013. Technical Report. Retrieved: April 4, 2019, from https://docs.google.com/file/d/1_9698uglerxB9nAglvaHkEgU-iZNm1TvVGuCW7245-WGvZq47teNpb_uL5N9/edi.

[66] Alex Stolz and Martin Hepp. Towards crawling the web for structured data: Pitfalls of

common crawl for e-commerce, 2015. Retrieved: April 2, 2019, from `http://www.heppnetz.de/files/commoncrawl-cold2015.pdf`.

[67] Reem Suwaileh, Mucahid Kutlu, Nihal Fathima, Tamer Elsayed, and Matthew Lease. Arabicweb16: A new crawl for today's arabic web. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR16)*, pages 673–676. ACM, 2016.

[68] Henry S. Thompson and Jian Tong. Can common crawl reliably track persistent identifier (PID) use over time? *CoRR*, abs/1802.01424, 2018. Retrieved: April 4, 2019, from `http://arxiv.org/abs/1802.01424`.

[69] Antal Van den Bosch, Toine Bogers, and Maurice De Kunder. Estimating search engine index size variability: a 9-year longitudinal study. *Scientometrics*, 107(2):839–856, 2016.

[70] Benjamin Vandersloot, Johanna Amann, Matthew Bernhard, Zakir Durumeric, Michael Bailey, and J. Alex Halderman. Towards a complete view of the certificate ecosystem. *Proceedings of the 2016 ACM on Internet Measurement Conference - IMC 16, Pages 543-549*, 2016.

[71] W3C. The World Wide Web Consortium (W3C), W3C homepage, 2018. Retrieved: April 4, 2019, from `https://www.w3.org/WWW/`.

[72] W3Techs. Web technologies statistics and trends. Retrieved: April 3, 2019, from `https://w3techs.com/technologies`.

[73] MDN web docs. HTTP cookies. Retrieved: April 4, 2019, from `https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies`.

[74] Taro Yamane. Elementary sampling theory. 1967.

[75] Ricardo Baeza Yates, Carlos Castillo, and Vicente López. Characteristics of the web of spain. *Cybermetrics: International Journal of Scientometrics, Informetrics and Bibliometrics*, (9):3, 2005.

[76] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. In *2nd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud'10, Boston, MA, USA, June 22, 2010*, 2010.

Appendix A

Challenges of Web crawling

Web crawling possesses a multitude of ethical, and legal implications and presents significant technical challenges [2, 3, 30]. Those challenges include but are not limited to:

- **Extracting metadata** - Web crawling is usually performed to extract some kind of ”metadata” from the website. This might include establishing what is the language of the text on the page, finding all links included in the page, and so on. This process involves parsing HTML (that might, and often is, syntactically incorrect) and significant amount of computation (especially if other file formats, such as PDF, are processed as well). When performed on a large dataset, metadata extraction is very demanding in terms of resources and time required to complete the task.

- **Legal implications** - Obtaining data through automated crawling might be a violation of copyright laws or website‘s Terms of Service. In the past, there were cases in which the owner of the site decided to pursue legal action against individuals [15] or organizations [36] because of unauthorized crawling.

- **Long download time** - Aggressive crawling of web pages can lead to unintentionally causing so called Distributed Denial of Service (DDoS) attack - in case of some websites, especially those hosted on low-end hardware, sending too many HTTP requests too quickly might make the website slow, or even unavailable for other users. The solution is to increase the interval between requests but this drastically increases the time it takes to complete a crawl.

- **Crawlability** - To avoid performance of the website being affected by aggressive crawling, many sites limit or forbid web-crawler access for some, or all of their pages. This is sometimes enforced by blacklisting IPs of users that send too many requests in some period of time and might make obtaining necessary data more difficult or impossible [24].

- **Need for filtering** - When performing a web crawl, filtering crawled content is necessary. For example, a single page might be crawled multiple times for a variety of reasons. This duplication is not desired and hence filtering is absolutely necessary. Other types of filtering include adult content filtering, or removing spam websites. Filtering is yet another operation that increases the computational cost of web crawling.

Appendix B

Relevant data types

Common Crawl `count` files contain many useful statistics. The records in `count` files have the format [49]:

`[<record_type>, <key>, <crawl_id>] [pages, URLs, host, domain].`

The relevant values of `record_type` are listed in table B.1 below. The value of `key` depends on the record type. The value of `crawl_id` is the same for each record and can be ignored.

Counts have the format `[page, URL, host, domain]`, however, repeated trailing numbers are skipped [49].

ID	Name	Description
0	url	(Unique) URL
1	digest	(Unique) Content Digest (MD5)
2	host	Hostname (e.g. www.example.com)
3	domain	Pay-level domain or private domain (e.g. example.com)
4	tld	Public sufix (e.g. ".com", "com.au") *
6	scheme	URI Scheme (e.g http or https)

* Not necessarily a top-level domain (TLD) [43] as [34] is used for extraction

Table B.1: Relevant types of records in Common Crawl's "count" files. Source: [17]