Reproducible Web Corpora: Interactive Archiving with Automatic Quality Assessment

JOHANNES KIESEL, Bauhaus-Universität Weimar, Germany
FLORIAN KNEIST, Ulm University, Germany
MILAD ALSHOMARY, Paderborn University, Germany
BENNO STEIN, Bauhaus-Universität Weimar, Germany
MATTHIAS HAGEN, Martin-Luther-Universität Halle-Wittenberg, Germany
MARTIN POTTHAST, Leipzig University, Germany

CCS Concepts: • **Information systems** → **Information storage systems**; **Digital libraries and archives**; **Web mining**; Data mining; Information retrieval; • **Software and its engineering** → *Software notations and tools*;

ACM Reference Format:
Johannes Kiesel, Florian Kneist, Milad Alshomary, Benno Stein, Matthias Hagen, and Martin Potthast. 2018. Reproducible Web Corpora: Interactive Archiving with Automatic Quality Assessment. *ACM J. Data Inform. Quality* 0, 0, Article 0 (August 2018), 23 pages.
https://doi.org/10.1145/3239574

1 INTRODUCTION

The World Wide Web compiles the most complete collection of mankind's knowledge that has ever been created. It also includes extensive information about all kinds of human interactions, as well as about the traits and states of mind of individuals who share about themselves, increasingly allowing for studying societies at large. This, and the fact that much of the information can be obtained at virtually no expense just by downloading it, renders the web an invaluable source of raw data for various disciplines of science. In particular, many forms of "applied" computer science benefit and hence rely heavily on web data, such as data mining and machine learning, web science and social network analysis, computer linguistics and natural language processing, computational social science and the digital humanities, data science and data journalism, and not least of all, information retrieval.

Meanwhile the web has evolved into a maelstrom of information, constantly published, shared, revised—and eventually lost. Despite the ease of copying and preserving digital information without loss of quality, digital media have proven the most volatile of all. Unless some serious effort is invested in keeping it safe, the web must be considered ephemeral. It is therefore false to think of the web as an ever growing collection of knowledge. The web is merely a collection of *currently needed knowledge* plus leftovers from the past. It stands to reason that many scientific inquiries based on web data, whose authors did not ensure for that data to be archived, become irreproducible shortly after their publication.

However, simply downloading a web page's HTML source is not sufficient to ensure the reproducibility of a web page's "look and feel." Still, current web corpora like the ClueWeb09 [40] and the ClueWeb12 [41] used within TREC employ this strategy. The styles, scripts, and multimedia files not served inline with the HTML code of a web page are almost entirely missing, and most of them have long since disappeared from the open web. Viewing a web page without these resources results

in a wholly different experience than its authors intended, which has a detrimental effect on human perception when reviewing and judging pages from these corpora [18], jeopardizing the validity of TREC's evaluation results. Yet, even downloading externalized resources may not be sufficient. Nowadays, many web pages are a piece of software, composed of resources from several web servers, compiled and executed just in time in the browser when the user loads it, and dynamically modified and updated with additional content based on user interaction. This is becoming more and more the standard rather than the exception, as evidenced by the usage of JavaScript libraries in more than 70% of all websites [46], and the fact that Google has started rendering web pages during crawling [17]. Reproducing web pages, as in showing an archived page in a browser that looks and behaves like the original page at the time of archiving, has hence become increasingly difficult.

With this paper, we contribute the Webis Web Archiver, which implements (1) a flexible archiving tool based on scriptable user-page interactions, and (2) a component for the reproduction of archived web pages within a browser, replaying the interactions and thereby allowing for reproducible experiments based on such replays. To evaluate our tool, we construct the Webis Web Archive 17, an exemplary corpus of 10,000 archived web pages. The archived pages have been manually reviewed and annotated with regard to reproduction quality, recruiting 9 judges for each page via crowdsourcing to compare a reproduction to its original page. This marks the first systematic evaluation of the archiving quality of a web archiving tool and might serve as an example for future evaluations.

Despite our corpus being available for download including all annotations, it will be outdated before long, since the original web pages on which it is based will change or disappear, and since web technology continues to evolve. A reevaluation of an updated version of our tool, or the evaluation of a new tool, will hence require the renewed, costly annotation of a new corpus from scratch, rendering the results obtained only superficially comparable to ours. Nevertheless, we attempt to overcome this limitation of web archiving research by sidestepping it: we introduce the new task of *immediate automatic reproduction quality assessment* for web page archiving, where, given a recently archived web page, the task is to assess its reproduction quality compared to its original under a pre-defined user-page interaction. For this task, we evaluate three quality assessment approaches—the third of which an original contribution—, which resort to (1) estimating the number of missing resources and their impact on the web page's utility, (2) the pixel-wise visual difference between archived web page and original, and (3) deep learning on the visual differences. With this task, we disentangle the development of archiving technology from its evaluation, and our corpus may yet serve as a reference for future approaches to quality assessment, even after the original web pages have disappeared.

Altogether, after reviewing related work in Section 2, this paper presents the following four contributions:

(1) The Webis Web Archiver, a new and freely available web page archiving tool that incorporates browser automation, modern browsers, and web archive proxies to create standard WARC web archives, and to reproduce the archived pages (Section 3).
(2) The Webis Web Archive 17, a publicly available dataset of 10,000 carefully sampled, archived web pages, annotated for reproduction quality via crowdsourcing (Section 4), and constituting the first benchmark dataset for
(3) the new task of immediate automatic reproduction quality assessment, which helps to overcome the inherent irreproducibility of web archiving tool evaluations (Section 5).
(4) The first in-depth analysis of web page reproduction quality, comparing reproduction software and human quality annotations with three approaches to measure reproduction quality that are based on missing resources, screenshot differences, and learned screenshot comparisons, where the latter constitutes a novel application of deep learning (Section 6).

2 RELATED WORK

Web corpora are frequently used for scientific evaluations. However, the currently available ones have been criticized for not allowing the reproduction of web pages collected, undermining human annotation and user studies [18]. These issues have not been resolved until today. The two most widely used corpora for evaluating information retrieval systems, the ClueWeb09 [40] and the ClueWeb12 [41], contain only HTML files (excluding style sheets, scripts, and images), many of which are truncated to save space. For the TREC Web Tracks alone, more than 125,000 ClueWeb pages have been manually annotated, whereas most of them will not have been displayed as originally intended. Moreover, any algorithm (e.g., for main content extraction) relying on analyzing the appearance of a page and employed as part of a retrieval pipeline by a Web Track participant may as well not have worked as intended. Though it has been tried to fix existing web corpora by identifying, downloading, and extracting information from missing resources [31], this post hoc improvement suffers from the fact that most of the original web pages have since disappeared. The Common Crawl [36] is also missing many of such resources. Other small web corpora only contain plain text without reference to the original web pages [8] or HTML files without associated resources [24].

Outside science, however, web archiving has been a focus of many initiatives for a long time [5, 16]. Most prominently, the Internet Archive works since 1996 on archiving as much of the web (and the greater Internet) as possible, with the goal of building a comprehensive library of the web. The Internet Archive's well-known Wayback Machine is a web service allowing to access and browse past versions of all web pages archived, with an emphasis on actually maintaining their look and feel. This service is frequently used [2]. Both the Internet Archive and Perma.cc allow its users to create snapshots of web pages on demand, e.g., so that one can prove at a later time how a given web page looked like at the time of archiving. As of recent, Wikipedia makes use of these services to archive linked sources [12] in an attempt to fight link rot [44]. Much of the tooling developed by the Internet Archive has been shared open source, forming the basis for our tool.

Typical web archiving tools comprise two components, one for archiving a web page in a web archive file (WARC), the de facto standard for web archives, and another for reproducing an archived web page. Regarding the former, they feature the ability to store streaming data [20], to tailor the archiving to certain websites [15], and to simulate basic user interactions [3, 37, 38]. We build upon these efforts, using the Internet Archive's man-in-the-middle archiving proxy warcprox [39] within our archiving tool. Several approaches exist to reproduce an archived web page. The Memento framework [42] is available as a browser plugin and uses an approach comparable to HTTP content negotiation to access a past snapshot of a requested resource. The Memento protocol provides a common interface for tools that reproduce web pages from an archive, and is implemented by the two most widespread such tools, OpenWayback [11] and Python WayBack [25]. Both allow to access archived web pages, either via specific URLs, or by acting as a proxy server serving responses from the archive file instead of from the web. Besides the Wayback Machine, other tools to browse web archives have also been developed for specific user groups, such as Warcbase [28] for historians.

Only few studies have dealt with assessing archiving quality so far. Even though the Web Curator tool exists for manual assessment [29], it has not been used to construct web corpora. Regarding automatic assessment, the CLEAR method tries to predict how well an entire website could be archived based on its accessibility, compliance with standards, cohesion, and use of metadata [6, 7]. Brunelle et al. [9] score the archiving quality after archiving, considering the estimated importance of missing images, multimedia objects, and style sheets. They find that quality estimation can be improved by differentiating the importance of missing resources (e.g., based on image size) instead of treating each missing resource the same. We reimplement this approach for our evaluation to compare it with our own approaches.

Not directly related to web archiving, automated web page rendering is also used to detect changes in web pages to optimize the crawling process. Pagelyzer [43] renders a web page, segments it, and compares these segments to the segments of other web pages. Such tools are designed to enhance the crawling or archiving process. While change detection is not in the focus of this work, the Webis Web Archiver that this paper presents offers the flexibility to integrate change detection algorithms as part of a user simulation script.

3 THE WEBIS WEB ARCHIVER

This section presents the Webis Web Archiver in terms of its envisioned use cases, a requirements analysis derived from that, its software architecture, and details on user-page simulations and fuzzy request-response matching during reproduction plus a number of other technical challenges. Our web archiver is available as a readily executable Docker image,[1] and open source.[2]

3.1 Target Use Cases

Reproducible Web Corpora. A key use case of our archiver is the construction of reproducible web corpora, where each web page archived can be reproduced as close to its original as possible. This has two benefits: human inspection and annotation of web pages is not harmed, rendering conclusions drawn more realistic and hence experimental results based on them more valid. At the same time, algorithms can be developed against a static, yet close-to-realistic source of raw data. In particular, information can be extracted from all of a web page's resources, including features extracted from a web page's visual representation, e.g., when relying on recent advances in deep learning for computer vision. Even if the page appearance is not used in the development of some algorithm based on web data, one should not discard the supposed overhead, since new ideas at solving the algorithm's task may emerge at a time when it is too late to re-crawl the previously omitted data.

Reproducible User Experience. In laboratory user studies, participants often have to accomplish a task that involves using a web service (e.g., a search engine) and web browsing (e.g., browsing search results). For instance, one may wish to analyze the reaction of different users to a specific situation without explicitly spoiling it to participants. Since using live web services and live web pages may allow for high variation, it has been technically challenging to ensure that participants will independently experience the situation in question exactly the same, while not changing the general user experience they are used to (e.g., the web browser). Hence, another use case for our archiver is the creation of a reproducible user experience by allowing to manually or automatically pre-record the usage of web services as well as web browsing all in one web archive, including alternative predicted user behaviors. This way, participants of a user study may not notice a difference when actually running through an experiment, and the experiment itself can be repeated any time, even when the original web services and web pages involved have changed or vanished.

Reproducible User Behavior. In observational user studies, participants have to accomplish a task on their own, usually with as little guidance as necessary to set them in the right state of mind. It may even be the case that participants are asked to use a web service in any way they please (e.g., an entertainment system). In cases where the degrees of freedom for users are too large to be anticipated in advance, user sessions can be archived using standard tools (optionally including clickstream logging [30]). With our tool, the recorded sessions can be algorithmically revisited for an analysis at any time after the study, even when the web service or web pages underlying the study have meanwhile changed.

[1] Docker image: https://hub.docker.com/r/webis/web-archive-environment
[2] Instructions, binaries, code: https://github.com/webis-de/webis-web-archiver

Reproducible User Simulation. The simulation of users has gained significant interest in information retrieval as of recent [4], where users of search engines are simulated to avoid the costs of collecting click data from real users. While recorded user sessions may serve as training data for user simulation, the simulations themselves become increasingly more sophisticated and dynamic. Conceivably, a simulated user may be envisioned which reacts to content it "sees" on a web page, so that the capability to reproduce the look and feel of a web page becomes important. Here, an additional use case for our archiver is to record user simulation runs, rendering them reproducible.

3.2 Requirements Analysis

Based on its target use cases as well as our review of related work, we derive the following key requirements for our web archiver:

(1) *Scriptable user-page interactions.* Users of our archiver want to be able to easily specify the user-page interactions that happen during the archiving or reproduction of web pages. This includes taking screenshots, manipulating and exporting a web page's DOM tree, firing any kind of event that occurs when a user interacts with a web page, such as click, tap, scroll, etc., entering text into and submitting forms, and so on.
(2) *Scalable archiving and reproduction.* Since web corpora nowadays contain millions or even billions of pages, parallel archiving and reproduction are mandatory. The archiving and reproduction of an individual page must be seamless, not blocking human user interaction, and not taking longer than loading the original web page did.
(3) *Long-term reproducibility.* To enable reproducible science on web pages, the reproduction of the web pages must not change over time. For example the rendering of a web page can change with newer browser versions, which has a negative impact on the reproducibility.
(4) *Adjustable level of archive granularity.* Web archivers typically store pages in archive files, each collecting hundreds of pages. This makes handling individual pages difficult, which is a requirement when constructing small, focused web corpora for specific research questions. The level of archive granularity must be adjustable, the lower boundary being exactly one page. This also makes splitting web corpora into sets for training and testing easier.
(5) *Local execution and storage.* One must be able to create and store web archives locally, without reliance on third parties. Possible reasons include that to-be-archived web pages are not accessible from the open web, privacy issues, or that experiments demand to process and manipulate the web pages before the reproduction.
(6) *Compatibility with other web archiving software.* Given the buzzing web archiving ecosystem, the web archives created by our web archiver must be compatible to be processed with third party tools. In this regard, the web archive file format (WARC) is the proper choice to ensure compatibility.
(7) *Sustainability.* The software stack of our web archiver must be maintainable, and easily adjustable by everyone, also when the original authors are unable to provide support. This is in fact a requirement for all of software development in science. Hence, the dependencies of our web archiver must be chosen to ensure long-term support, proprietary dependencies should be avoided, and slick APIs must be defined to integrate and orchestrate third party components.

The Webis Web Archiver is designed to meet all of these requirements. Below, we overview its software architecture.

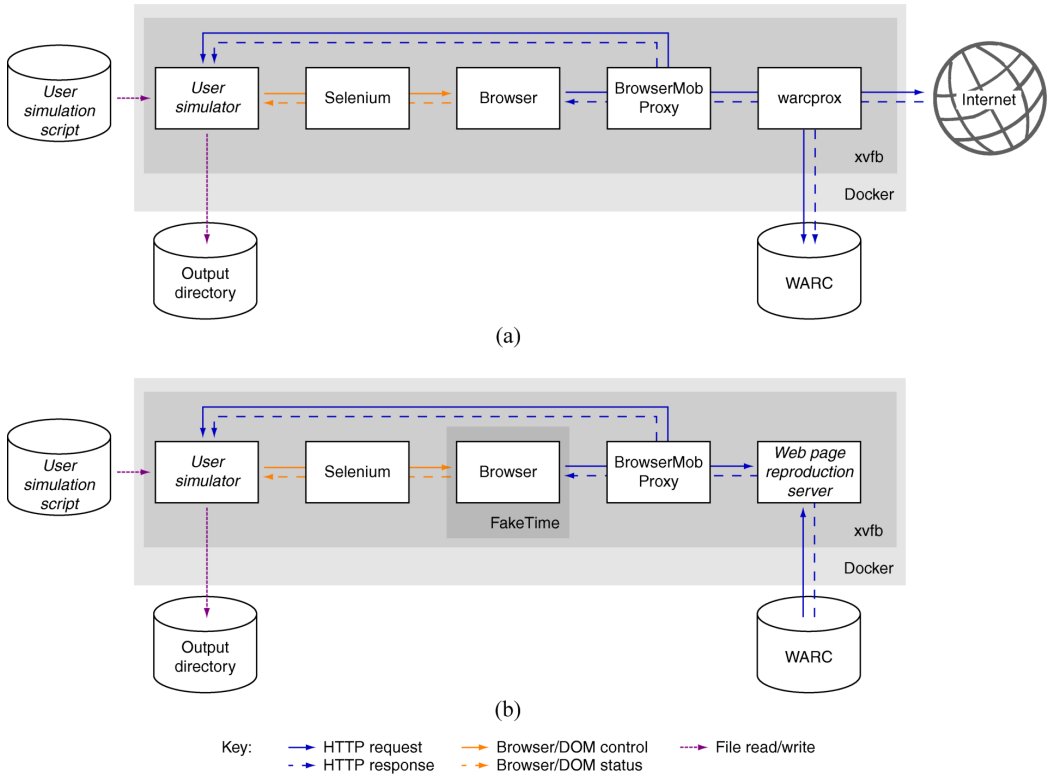


Fig. 1. Software architecture of the Webis Web Archiver. Depicted are the building blocks, dependent libraries, and communications between them (a) for archiving and (b) for reproduction. Libraries are shown as rectangles, files and directories as cylinders. The components originally created for the Webis Web Archiver are highlighted in italics.

3.3 Software Architecture

Web archiving software typically comprises two major components, one for archiving a web page, and another for reproducing a web page from a pre-recorded archive. Figure 1 overviews the two components, and the respective processes for archiving and reproduction they implement.

The Webis Web Archiver depends on a number of carefully selected software libraries. A readily executable configuration of the archiver is encapsulated in a Docker image [14]. Docker allows to robustly run our archiver in parallel on any host where Docker is installed. It helps to separate the configuration and orchestration of our archiver's dependencies without affecting, or being affected by other software that may be installed on a given host. Docker also ensures the reproducibility of our archiver's execution environment by fixing the versions of each software library and especially the browser. Moreover, all of 2 GB worth of fonts available to Ubuntu are installed in the Docker image, which ensures that virtually all characters are properly displayed when taking screenshots, regardless the language. This way, using our archiver boils down to executing a Docker command with a small number of parameters. To avoid caching effects, a new docker container is started for each URL. Also, Unix shell scripts are provided to allow for easy execution of our archiver outside Docker.

The virtual screen software xvfb [47] is used to run the browser without requiring a physical screen, thus allowing for server-side execution. Apart from two exceptions, the archiving process and

Algorithm 1: Built-in user simulation script

```
Data: Browser browser, start URL url, output directory directory
Result: Screenshot and HTML snapshot of web page at url in directory
begin
    window = browser.openWindowWithUrl(url)
    // Scroll down to load all content
    for i ∈ {1, ..., 25} do
        if ¬window.canScrollDown() then break
        window.scrollDown(window.viewport.height)
        browser.WaitForNetworkTrafficToStop()
    // Resize browser viewport to page size for screenshot
    window.scrollTo(window.page.top)
    window.resizeViewportHeight(window.page.height)
    browser.WaitForNetworkTrafficToStop()
    // Save current state to output directory
    window.saveHTMLSnapshotTo(directory)
    window.saveScreenshotTo(directory)
```

the reproduction process are exactly the same: a user simulation script is read from disk and started. The Selenium browser automation software [32] serves as an interface between the script and the browser. As browser, we employ a current version of Google Chromium, but others are supported as well. During reproduction, the FakeTime Preload library [48] is used to pretend to the browser that it runs at the time of archiving, which affects all JavaScript calls that use the current date. The browser is set up to communicate with an instance of BrowserMob Proxy [27], which is used by the script to learn when network traffic ceases. During archiving, the BrowserMob Proxy communicates with the Internet via an instance of the warcprox proxy [39], which stores all requests and the corresponding responses that pass through it in a standard WARC archive file. During reproduction, a local server is started that pretends to be a proxy, but actually attempts to retrieve the previously recorded responses to requests made by a to-be-reproduced web page from its WARC files.

Important parameters include the start URL for the user simulation script, the script itself, the output directory, whether to archive or to reproduce, and where WARC files are to be stored or located. For debugging purposes, an extra mode allows for applying user simulation scripts to the live web without archiving. The following sections detail two key features of our archiver, namely its unique user simulation scripts and fuzzy request-response matching.

3.4 User Simulation Scripts

A key feature of the Webis Web Archiver is its capability of controlling the browser during archiving and reproduction using the Java-based API. Among other things, the scripts API allows to load any URL into the browser, to scroll, to resize the browser window, to manipulate the DOM, to click on elements, to print elements, to take screenshots, to wait for network traffic to stop, and to execute JavaScript code. Instead of implementing this kind of browser automation ourselves, we rely on Selenium [32], a software-testing framework for web applications. Selenium has been under development since 2004, and has since risen to become the single most important testing framework of its kind, integrating all major web browsers. Despite its obvious usefulness for web archiving tools, Selenium has never been considered for this purpose until now. Rather, most of the existing solutions

employ PhantomJS [19], a comparably limited scriptable browser which has not been updated for months at the time of writing. The interface between Selenium and the web browser will become the W3C WebDriver standard [35], further increasing the cross-browser compatibility of Selenium and, consequently, our archiver.

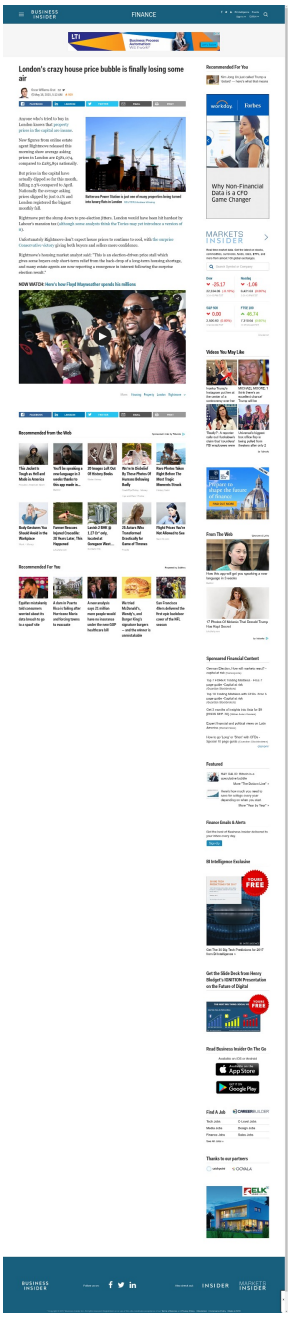
We call the code used during archiving *user simulation scripts*, since they can be thought of as modeling a real user's behavior. Nevertheless, more can be accomplished within the use cases outlined above; for example, a user's behavior may be influenced in real-time by changing a web page while the user uses it in a recorded session, a walk on the link graph can be implemented, the web page may be classified, or information extracted from it. Our archiver ships with the built-in user simulation script shown in Algorithm 1: it simulates a user who wants to reach the bottom of a page, but gives up after a predefined maximum number of page down scrolls. During the process, a screenshot of the original page as well as its HTML source are stored. Any user simulation script used during archiving can be used without change during reproduction so as to allow a web page served out of an archive to reach the same state as its original did.

Parameters of a user simulation script are a browser object, a start URL, and an output directory where any extraneous script output can be stored (e.g., a screenshot). The browser object allows opening new browser windows. By default, browser windows have a size of 1366x768, which is the most widespread screen resolution for desktops as of 2018 [34, 45]. Selenium also allows to emulate mobile devices, but this functionality is not mature enough as of yet. Additionally, user simulation scripts have access to the directory in which their source code resides for accessing resources like dictionaries or machine learning models if needed. Our user simulation scripts are currently limited to Java, but Selenium offers more language bindings, and we plan on following suit in the future.

3.5 Fuzzy Request-Response Matching

Although archiving and reproducing web pages the way described above may seem straightforward, the devil is in the details, and reproducing web pages from an archive is prone to errors. It is important to understand that a web page is not just a collection of static plain text files that only need to be systematically found and copied. Rather, it is a distributed piece of software with a client-server architecture. What is displayed in a browser is the user interface of such a software, which is assembled and executed just in time as the user visits a page. In this regard, web archiving is essentially the same as taking snapshots of the states of the user interface, hoping that the user simulation script visits all, or at least the important states of a page a real user can reach. As this view on web pages shows, archiving and reproducing a web page is a non-trivial task.

During our developments, we identified and tackled many technical challenges related to technologies used at client side, at server side, and in-between. For brevity, this section describes only one of the more interesting solutions: fuzzy request-response matching. When reproducing a web page, all requests sent by the archived web page must be answered with the same responses that have been captured during archiving. However, the requests sent during reproduction may differ in subtle ways from the ones originally sent, requiring a fuzzy match between request and response. For example, requests made via JavaScript that involve random numbers or that are based on the current time will be dissimilar from their original. Furthermore, requests can change the server's state, so that other requests result in different responses depending on whether they are issued before or afterwards. As the server is essentially a black box to the archiving process, no information exists on which requests have effects on responses, or which responses are affected. Furthermore, as requests are usually sent in parallel by the browser, race conditions will occur. The order in which requests are issued can drastically vary between two runs of the same user simulation script as this order often depends on which responses are received first. For illustration, Figure 2 shows an example where the screenshots taken by our archiver differ as a result of failed matches between requests and responses.



(a)

(b)

(c)

Fig. 2. Screenshots of a web page archived under the user simulation script of Algorithm 1 (a) during archiving (i.e., the original page) and (b) during reproduction, as well as (c) highlighting differences between (a) and (b). The web page has been selected to show typical errors, namely visually shifted content due to missing advertisements and missing recommendations for further reading.

Our archiver currently allows to choose from three different tools for reproducing web pages (the "web page reproduction server" in Figure 1): warcprox, Python WayBack (pywb), and an implementation of our own. Warcprox [39] focuses mainly on archiving and implements only a very basic request-response matcher based on URL normalization.[3] It serves as our baseline. Python WayBack [25] is one of the two most widely known web page reproduction tools available.[4] It implements a rule-based matcher, engineered specifically for this task, which aims at resolving differences in requests that likely occur in a reproduction scenario, for example, by ignoring strings that resemble session IDs. In addition, Python WayBack offers support for organizing multiple archives and handling repeated archiving of the same page, which is not needed in the research scenarios we consider. However, by default, Python WayBack injects code into reproduced web pages, for example, to change the time reported by JavaScript to the time of archiving. We deemed code injection problematic for our purposes—since injected code may interfere with user simulation scripts—as well as for maintaining integrity of the archived data, so we disabled this feature. In preliminary tests, we noticed that some images are missing in the reproduction of Python WayBack without obvious reason. For an easier analysis, we thus implemented a custom request-response matching cascade that tries to find a request match by identity checking including URL normalization, then by partial matching ignoring headers, and if that also fails, by rules very similar to those used in Python WayBack. We apply these three reproduction tools to the Webis Web Archive 17, which is presented next, to analyze how well web pages are reproduced with the current state-of-the-art.

4 CROWDSOURCED EVALUATION OF WEB ARCHIVING TOOLS AND DATASET FOR AUTOMATIC REPRODUCTION QUALITY ASSESSMENT

This section serves three purposes at once: it presents the results of a comprehensive evaluation of the Webis Web Archiver; it introduces for the first time the methodology to evaluate web archiving tools by means of crowdsourcing; and it presents the dataset compiled, the Webis Web Archive 17,[5] which lays the foundation for the novel downstream task of automatic reproduction quality assessment.

Unlike traditional evaluation setups, the evaluation of web archiving tools cannot rely on benchmark datasets or corpora: compiling a dataset of static web pages is insufficient, since an archiving tool requires a functional web server that hosts web pages in order to archive them. Even if a sufficiently large set of web servers hosting a sufficiently large number of different web pages, each built with a diversity of web technologies, were conserved and made available, the rapid pace at which web technology evolves would soon render this resource outdated and evaluation results gained from it irrelevant. Therefore, direct comparative evaluations of web archiving tools that are developed by different parties at different points in time are virtually impossible at reasonable costs.

A potential solution to this problem can be found in a standardized, manual evaluation procedure to be followed minutely by independent parties that develop web archiving tools. The operationalization of sensible notions of web archiving and reproduction quality criteria, however, has its own pitfalls as outlined below. Further, the comparability of such evaluations conducted at different points in time is weaker than with a direct comparative evaluation. Nevertheless, at least scale can be attained via crowdsourcing, which is how we were able to conduct a large-scale evaluation the Webis Web Archiver, demonstrating the use of Amazon's Mechanical Turk for this purpose for the first time.

We compiled the web pages archived with our tool and the human annotations collected as part of our evaluation into the Webis Web Archive 17, which is publicly available. While this dataset, too, will soon be technology-wise outdated, it may still serve as a basis for the development of automatic

[3] Reproduction in warcprox was buggy when we first tried it for this purpose, but we helped resolve this issue.

[4] Since the most prolific contributors to Python WayBack and OpenWayback are the same people, but the development of Python WayBack is more active, we do not expect Open Wayback to achieve a higher reproduction quality.

[5] The dataset is publicly available at https://doi.org/10.5281/zenodo.1002203.

reproduction quality assessment technology. Instead of manually reevaluating each web archiving tool (or each version of a web archiving tool), the quality criteria underlying the human assessment of reproduction quality may be learned with machine learning based on this dataset. If sufficiently successful, the resulting reproduction quality model may be applied in real time as a web archiving tool is deployed, used, and developed, allowing for severely shorter turnaround time and significant cost-savings compared to a manual evaluation.

After sampling web pages for our evaluation from a carefully selected mixture of high-ranked and low-ranked websites (Section 4.1), we used our tool to create the Webis Web Archive 17, comprising 10,000 archived web pages. We then defined our operationalization of reproduction quality assessment criteria (Section 4.2), and proceeded to conduct a corresponding crowdsourced quality annotation of our dataset (Section 4.3). This formed the basis both for the direct evaluation of the Webis Web Archiver, as well as for our experimental setup for the task of automatic reproduction quality assessment in Sections 5 and 6.

4.1 Sampling of Web Pages

To build a solid benchmark dataset for web reproduction quality assessment, we carefully sampled web pages with the goal of representing a wide cross-section of the different types and genres of web pages found on the web. As a population of web pages to draw a sample from, we resort to the recent billion-page Common Crawl 2017-04 [36]. From there, we primarily sampled pages from most of the well-known sites—as defined by the website's Alexa traffic rank [1][6]—to ensure that our sample encompasses pages using the most recent web technologies and design standards. Moreover, pages from a number of less well-known sites have been included. Altogether, the Webis Web Archive 17 comprises 10,000 web pages.

We drew a stratified sample of web pages from websites listed in the Alexa ranking according to the following restrictions: To avoid overrepresentation of organizations that host similar pages under several domains, we disregard pages from sites that are different-language versions or subdomains of higher-ranked sites.[7] Further, the sampling is restricted to yield exactly 50 pages from each of the top 50 sites, 10 pages from each of the next 100 sites in the ranking, 5 pages from each of the next 500 sites, and finally, 1 page from each of the next 1000 sites. In total, we thus sample 7000 web pages from the top 1650 sites. If the Common Crawl did not contain enough pages for a site, this site was skipped altogether and the next site in the ranking was used instead.[8] Not all web pages from the Common Crawl still exist, so we continued to archive pages of a site until the desired amount of web pages could be archived successfully. However, as the sample for a website would be hardly representative of the site when most web pages fail archiving, we again skip the website if twice the amount of required web pages have been tried without yielding the desired amount of successes. In total, 429 websites have been skipped. To also include pages from a sample of less-known sites in the dataset, we include an additional random sample of 3000 pages from the remainder of sites.

The final sample of pages was re-crawled and archived using the Webis Web Archiver presented in Section 3, employing the user simulation script given in Algorithm 1. On average, it took 86 seconds to archive a web page using this script. Reproduction was a bit faster, taking 67 seconds on average with only minor differences between the different reproduction approaches described in Section 3.5. For each page, the dataset contains on average 10.4 MB of data: 4.3 MB for the archive, 5.4 MB for screenshots (one screenshot of the live page as seen during archiving, and one when using each reproduction approach), and 0.7 MB for HTML snapshots (distributed like the screenshots).

[6]Ranking as of March 29th, 2017

[7]For example, files.wordpress.com is subordinate to wordpress.com.

[8]By manual analysis, we found that top-ranked sites with an insufficient amount of pages have either very restrictive crawling conditions (e.g., tmall.com), are link services (e.g., t.co), or are related to advertisement (e.g., onclkds.com).

Table 1. Number of sites and pages in each category and their most frequent sub-categories according to the Alexa Web Information Service. Sites and pages can be in multiple (sub-)categories.

Category	Sites	Pages	Most frequent site sub-categories	Most frequent page sub-categories
Adult	40	129	Computers (25), Image_Galleries (8), Society (2)	Computers (94), Image_Galleries (27), Society (3)
Arts	187	471	Television (59), Movies (19), Music (18)	Television (152), Movies (75), People (50)
Business	229	489	Financial_Services (39), News_and_Media (24), Arts_and_Entertainment (22)	Financial_Services (116), News_and_Media (84), Arts_and_Entertainment (43)
Computers	444	1502	Internet (184), Software (143), Companies (60)	Internet (821), Software (371), Companies (261)
Games	58	150	Video_Games (48), Gambling (6), Board_Games (1)	Video_Games (139), Gambling (6), Puzzles (2)
Health	28	36	Medicine (15), Conditions_and_Diseases (7), Nursing (5)	Medicine (21), Nursing (11), Conditions_and_Diseases (7)
Home	42	88	Consumer_Information (19), Cooking (9), Personal_Finance (7)	Consumer_Information (44), Cooking (17), Personal_Finance (15)
Kids_and_Teens	55	99	School_Time (21), Games (13), Computers (7)	School_Time (43), Games (28), Computers (10)
News	149	303	Newspapers (77), Breaking_News (13), Magazines_and_E-zines (8)	Newspapers (110), Headline_Links (55), Weather (24)
Recreation	70	106	Travel (32), Autos (8), Humor (5)	Travel (63), Autos (10), Pets (7)
Reference	164	275	Education (105), Dictionaries (25), Libraries (21)	Education (128), Ask_an_Expert (51), Dictionaries (47)
Regional	605	1182	North_America (395), Europe (112), Asia (78)	North_America (829), Europe (240), Asia (144)
Science	74	116	Technology (19), Social_Sciences (16), Biology (12)	Technology (34), Social_Sciences (27), Biology (20)
Shopping	182	413	General_Merchandise (30), Clothing (26), Entertainment (16)	Entertainment (108), General_Merchandise (105), Auctions (51)
Society	100	125	Issues (20), Religion_and_Spirituality (15), Government (12)	Issues (27), Religion_and_Spirituality (18), Government (16)
Sports	73	123	Resources (18), Soccer (9), Baseball (6)	Resources (48), Soccer (16), Cricket (10)
World	1453	3437	Chinese_Simplified_CN (305), Russian (245), Deutsch (213)	Chinese_Simplified_CN (866), Russian (731), Deutsch (593)

For each page's site in the dataset, we queried the Alexa Web Information Service API to retrieve it's category and dominant language. Table 1 shows the distribution of categories of the websites and their corresponding web pages. All categories are present in the dataset and no single genre is dominating. Indeed, the categories with the most sites are "World" (which can be considered a "miscellaneous"-category), "Regional," and "Computers," all of which are rather generic categories enclosing several sub-topics. As Figure 3 illustrates, English is the dominant language in the dataset: about 70% of the pages for which the Alexa Web Information Service API returned site-wise language information are English, followed by Chinese (13%) and Russian (4%). While the dominance of English pages is unsurprising, the dataset still covers a reasonable variety of languages.

4.2 Defining Reproduction Quality

Intuitively, if a reproduced version of a web page looks and reacts exactly as the original one did at the time of archiving, the reproduction quality is maximal. But quantifying how much an archived web page differs from this ideal is far from trivial. Operationalizing the notions of "looking" and

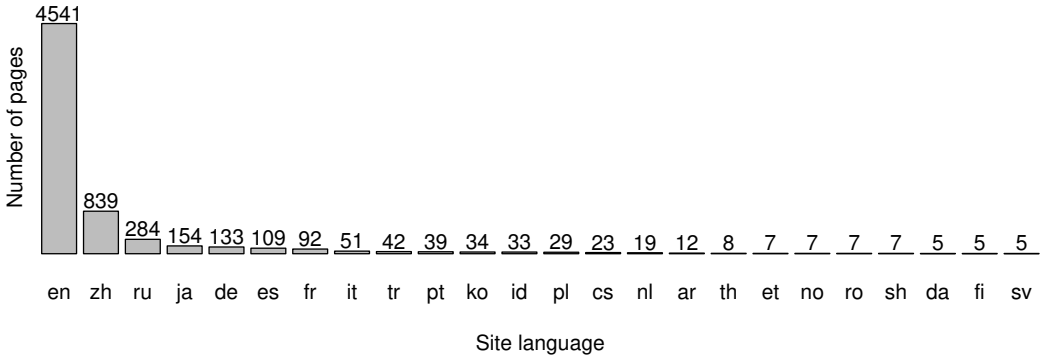


Fig. 3. Number of web pages by their website's language as determined by the Alexa Web Information Service (successful for 6,495 out of 10,000 pages). Languages with at least 5 pages are shown.

"reacting" the same or similarly is where the difficulties arise, especially when some differences in look or reaction might be negligible (e.g., a missing advertisement) while others render a reproduction unusable (e.g., a missing image in a web comic). Furthermore, the importance of any given difference depends on the context of a web page. For example, missing an image advertisement for a pair of shoes on a discussion board only marginally degrades its reproduction quality, whereas missing the same image on a shoe shopping site degrades its quality more significantly. For another difficulty, as the number of possible interactions with a web page can be enormous, it is infeasible to include all possible interactions in a quality assessment measure, let alone weighing the relative importance of each interaction, which would require a prediction of the likelihood of each interaction.

In this paper we therefore adopt a pragmatic definition of reproduction quality based on our basic scroll-down user model (as in Algorithm 1) and on visual differences between an original page and its reproduction. We chose this user model as scrolling down results in a single image for a web page, whereas other interactions (such as clicking links or buttons) results in a more complex, multi-state representation. Extensions to this user model are straightforward to implement with the Webis Web Archiver, but require a separate investigation into typical or "worst-case" user behavior. Under the scroll-down user model, we define the quality of a web page reproduction as follows:

> The more individual users that scroll down a web page are affected in their perception or use of the web page by visual differences between the original web page and its reproduction, the smaller the reproduction quality for that web page.

4.3 Human Annotation of Reproduction Quality

Employing the above definition of reproduction quality, we recruited human annotators to manually assess the 6,348 of the 10,000 pages in the Webis Web Archive 17 with reproduction errors, which were identified by comparing the screenshot taken during archiving with the ones taken during reproduction. For every web page where no reproduction screenshot matches the archiving screenshot perfectly, we asked humans to annotate the reproduction quality. Only the reproduction screenshot with the smallest pixel-based difference[9] to the archiving screenshot has been annotated. To match the scroll-down user model, annotators were shown the screenshots of the web page with the possibility to scroll down to the bottom of the page. To enable annotators to assess the effect of visual differences on their perception, the archiving and reproduction screenshots were shown side by side, while

[9] As measured by ImageMagick's RMSE, see Section 5.

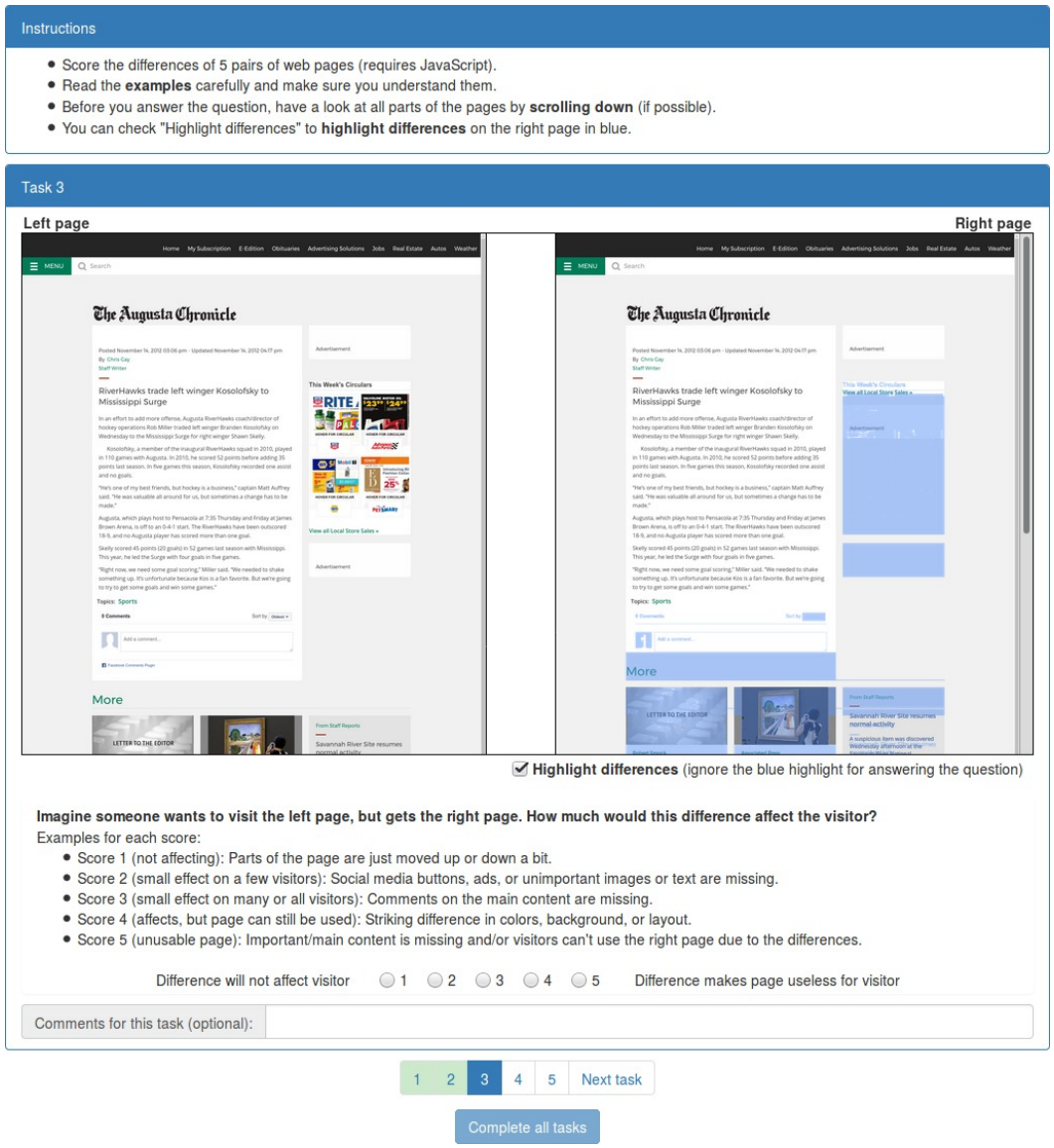


Fig. 4. Interface used by the human annotators to judge the reproduction quality of five web pages in a row, showing one at a time.

synchronizing their scrolling behavior and thus allowing for visual content matching. As some differences were rather small and might otherwise have been overlooked during annotation, the annotators were able to highlight the differences between the screenshots. The reproduction quality was assessed on a 5-point Likert scale to account for different levels of perceived severity, ranked from no effect (score 1) to unusable reproduction (score 5). The interface provided annotators with a short textual description and one example for each of the five quality scores. Figure 4 shows the annotation interface with activated highlighting.

Table 2. Number of times each median reproduction quality score occurs in the final annotation, ranging from no effect on visitor (1) to useless as a reproduction (5), when considering only reproductions annotated by humans (Annotated), and including the ones with no difference in the screenshots (All).

Reproductions	**Score**					$\sum$
	1	2	3	4	5	
Annotated	1942 (30.6%)	3307 (52.1%)	422 (6.7%)	318 (5.0%)	359 (5.7%)	6348
All	5594 (55.9%)	3307 (33.1%)	422 (4.2%)	318 (3.2%)	359 (3.6%)	10000

To annotate each web page, we hired 9 annotators at Amazon's Mechanical Turk. The annotators received their tasks in batches of five web pages; each annotator could work on several batches, but not on the same one twice. To avoid order biases, we randomized the order of pages within a batch as well as whether scores (and examples) are in ascending or descending order. However, to not confuse annotators who annotate several batches, we used the annotator's Mechanical Turk ID to seed the random number generator, essentially causing the order to be fixed across batches for each worker. We manually reviewed all web pages where not all 9 annotators agreed on a score within one point of the median score for that web page. Annotation batches for which a score clearly does not fit the screenshot differences as per the provided example for the score were rejected. Overall, we rejected 1234 of 11,430 batches (10.8%). In order to assess the annotation, we calculated the majority agreement among the annotators. In 81% of the cases, more than half of the accepted annotators agreed on a score. Due to this high annotator agreement, we deem the annotations reliable. To derive a ground truth annotation from the set of individual annotations for each page, we use the median of the scores that were assigned to a reproduction, which is a stable measure for such a small number of annotations as well as robust against outliers.

As high-quality annotations are very important for benchmark datasets, we then manually curated the web pages where the Mechanical Turk annotators disagreed (regarding only approved batches). Specifically, we took another look at all web pages where annotators gave at least one 1 and one 5, or where at least one annotator gave a score with an absolute difference of 3 from the median score of all annotators (e.g., score 4 for a median of 1). We found these web pages to be difficult cases that were not fully covered by the examples we provided. For example, these pages included videos or assets that were seemingly still loading in the reproduction screenshot (which we count as not reproduced), missing promotions for related products on shopping pages (which we count as content, not advertisement), or missing overlays that ask to accept cookies (which we count as a small effect as they can usually be dismissed easily). Additionally, we looked at all web pages from domains with many similar pages, and ensured that the quality is judged consistently across these pages. In total, this manual curation changed the ground-truth score for 717 of the 6348 reproductions (11%). For an overview, Table 2 gives the distribution of final scores, while Figure 5 shows one typical reproduction example for each score. As the table shows, the vast majority of all web pages (89.0%) were reproduced with a score of 1 or 2, demonstrating a sufficiently high reproduction quality for many applications. For example, in main content extraction, a quality score of 2 (small effect on few users) is likely sufficient. Nevertheless, reproduction is still far from perfect. To facilitate faster improvements to reproduction technology at development time, in what follows, we analyze methods for an automatic assessment of reproduction quality in real time, thus allowing for the immediate quantification and assessment of software improvements without having to repeat manual assessment.

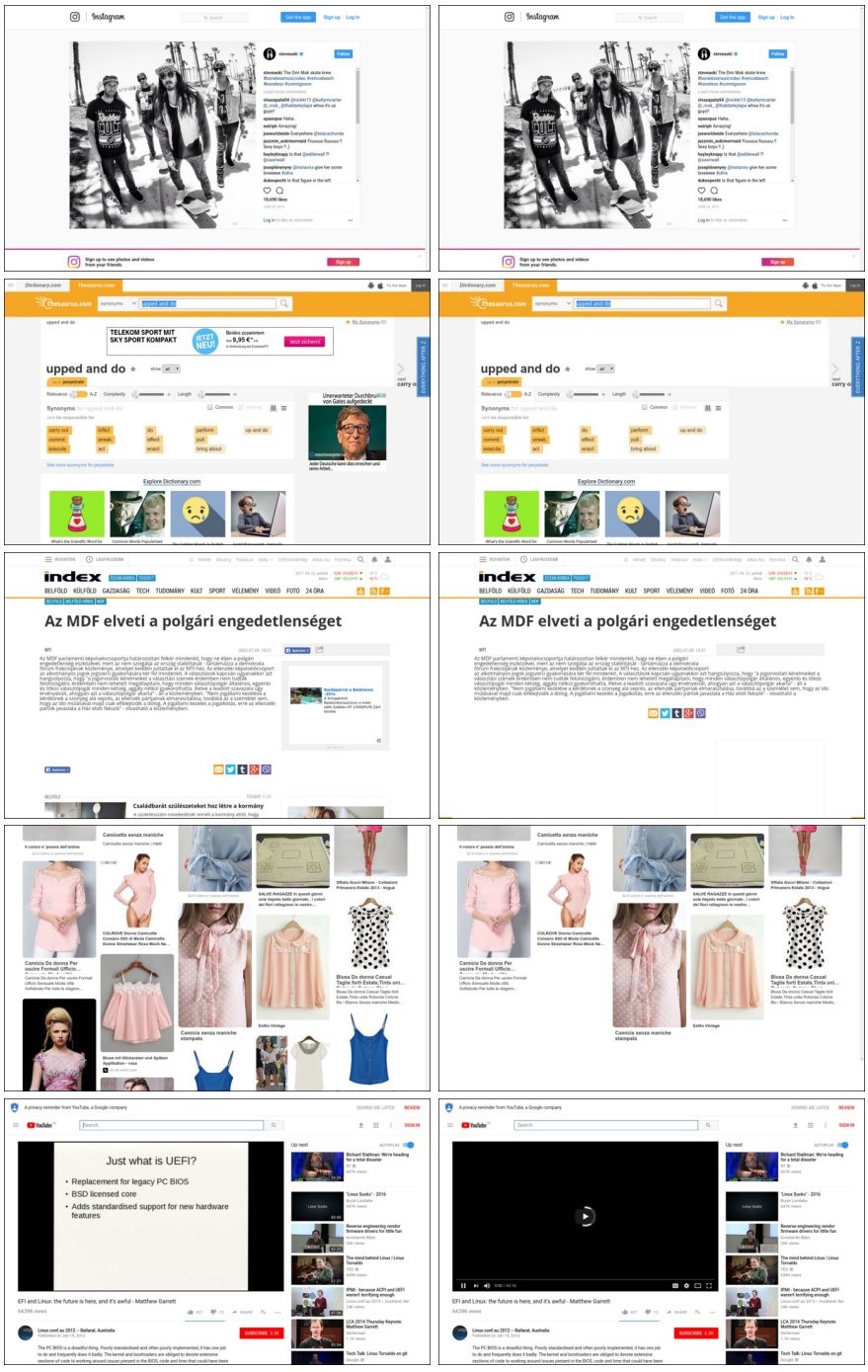


Fig. 5. Typical screenshots (or parts thereof) taken during archiving (left) and reproduction (right) for each of the five available quality scores from 1 (top) to 5 (bottom): (1) an animated button in a different state; (2) missing advertisements; (3) missing links to related pages and missing social media buttons; (4) missing additional content further down the page; and (5) missing main video.

5 AUTOMATIC REPRODUCTION QUALITY ASSESSMENT

The new task of automatic reproduction quality assessment aims at facilitating the fast debugging, quality assurance, and improvement of web archiving and reproduction technologies. The goal of solving this task it to integrate into existing archiving tools a feature that automatically evaluates the reproduction quality after archiving a web page, providing immediate feedback to the tool's user and developer. This section introduces three different approaches suitable for this task, one from related work, one standard measure, and one based on neural networks which is our own contribution.

Brunelle15. As a first measure of reproduction quality, we use the heuristic damage rating proposed by Brunelle et al. [9] for measuring the impact of missing resources. The impact of a missing resource is determined by it's importance for a page. For example, a large missing image has usually a greater impact than a small missing image. Different to the quality assessment setting in this paper, the authors consider the more general case where the archived page is the only available resource for the assessment. The approach calculates the damage for a web page as normalized value between 0 (no missing resource) and 1 (all resources are missing), where a resource is an image, a multimedia object, or a style sheet. The approach weighs resources according to a heuristically determined importance, which is based on image or multimedia object size or on the number of HTML classes that are not referenced by any of the retrieved style sheets. With the help of the original authors, we reimplemented the damage rating calculation as a user simulation script for our web archiver, which first scrolls down the page just like the script we used for creating the Webis Web Archive 17, and then calculates the damage using the current web page state in the browser. Since this measure is restricted to the archived page and exploits no information how the page looked when all resources where retrieved, we expect it to perform worse than the other measures. Also, the heuristics do not account for script files, which can have serious impact on the look and feel of a web page.

RMSE. The second measure uses the screenshots taken during archiving and reproduction, computes the squared color difference for each pair of corresponding pixels, and then uses the root of the mean as reproduction quality score.[10] Specifically, we use the corresponding option of ImageMagick [21] to calculate the difference, and then employ the normalized RMSE values that range from 0 (identical images) to 1 (for each pair of pixels in the two images with the same coordinates, one pixel is purely white and the other one purely black). If one screenshot is smaller than the other one, it is enlarged by adding white pixels at the bottom of the image so that pixels can be directly matched by their image coordinates. Note that all screenshots in the dataset have already the same width as they were taken using the same browser viewport width. While we expect some correlation with the ground-truth reproduction quality, RMSE does not consider the position of pixels nor the context of neighboring pixels. These features are useful for a quality analysis, as main content is often placed mid-screen. However, the shape of coherent regions of deviating pixels hint at what kind of content is missing. Furthermore, a single missing advertisement banner at the top of the page can shift up the entire web page, causing the RMSE to become unjustifiably high (see Figure 2 for an example).

Neural Network. The final measure uses machine learning to predict the reproduction quality from the screenshot differences. As the first machine learning approach applied to this task, we see this measure as a baseline for similar and more sophisticated measures that are yet to be developed. As a direct conclusion from the drawbacks of RMSE, we decided to employ a machine learning model that has been applied successfully to different image classification tasks, namely deep convolutional neural networks [26]. Since, however, fine-tuning a network to a new task is a research topic in its own right, we use the widespread and straightforward VGGNet network [33] instead. As it was shown that the fully connected layers at the end do not contribute to the performance [22], however,

[10]RMSE is the abbreviation for "root of the mean of the squared error".

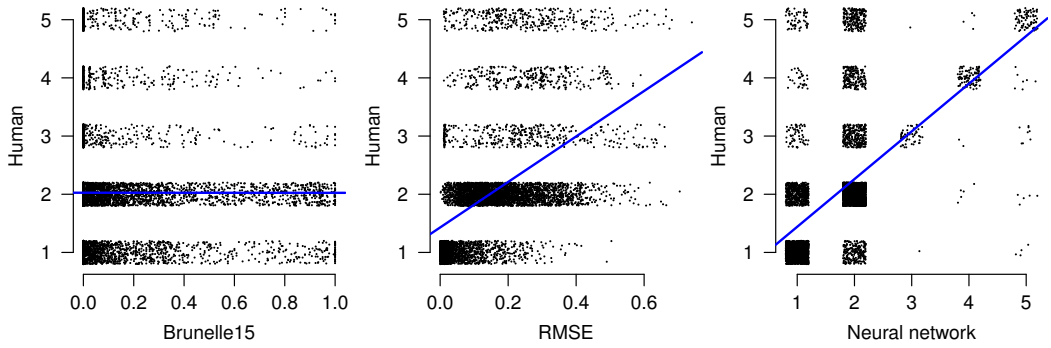


Fig. 6. Scatter plots of human reproduction quality annotations over scores computed by the three automatic measures for each annotated reproduction in the Webis Web Archive 17. A small jitter is applied for human annotations and neural network scores to provide a better impression on the number of dots at each coordinate. The straight lines correspond to fitted linear models; the closer they are to the main diagonal, the better the quality measures.

we omit them. The network structure and some implementation details are further described in Appendix A. As input images to the network, we create a single two-channel image, where one channel corresponds to the screenshot taken during archiving (in greyscale) and the other channel corresponds to the screenshot taken during reproduction (in greyscale). As the structure of VGGNet can only be trained and applied to images of a single size, we scaled down all created images to 384×128 pixels. For this size, icons with the usual width of 32 pixels are scaled down to a width of 3 pixels, which matches exactly the receptive field of the neurons in the convolutional layers of the network. In order to avoid skewing the images, we extended or cropped all screenshots to a height of 4098 pixels before the scaling. This height was chosen as the closest multiple of the image width (the currently most common browser window width of 1366 pixels) to the average image height of the archive images (4388 pixels). We converted all images to greyscale, since we do not expect color to play a major role for reproduction quality assessment, and found this to be indeed the case in our preliminary tests. We use the network in a 10-fold cross validation setting including all web pages[11] and using the annotations gathered in Section 4.3 as labels for both training and evaluation.

6 EVALUATION

We discussed three different web page reproduction approaches—one of which we created for this publication—in Section 3.5, and we introduced three measures for automatic reproduction quality assessment in Section 5. Next in this section, we evaluate their performance.

6.1 Comparison of Reproduction Quality Measures

The main goal of an automatic measure of reproduction quality is to compute quality scores that correlate with the true reproduction quality as per human assessment. The higher the correlation of automatic scores and true quality, the better the measure. Therefore, to test the three measures described above, we analyze their correlation with the human annotations on the Webis Web Archive 17. Figure 6 gives a visual impression of the correlation. For a quantitative result, we also calculate the Pearson correlation r with the human annotations for each of the three measures. As shown by the horizontal line for the linear model in Figure 6, the Brunelle15 measure is uncorrelated with the human annotations ($r = 0.00$). We believe that the main reason for the low r is that the measure

[11] We also include the web pages without differences in the screenshots for training

Table 3. Left: Confusion matrix of predicted and ground-truth reproduction quality scores for the neural network. Right: Effectiveness of the neural network quality measure for identifying not acceptable reproductions depending on what scores are considered to be minimally acceptable. Effectiveness is measured as accuracy (Acc.), precision (Prec.), recall (Rec.), and the F-Measure (F_1). All values are calculated from the left-hand confusion matrix (including the 3,652 trivial cases for accuracy).

Truth	**Estimated reproduction quality**					∑
	1	2	3	4	5	
1	1707	230	1	0	4	1942
2	535	2762	0	5	5	3307
3	62	294	55	2	9	422
4	33	183	0	95	7	318
5	57	218	1	4	79	359
∑	2394	3687	57	106	104	6348

Min. Quality	**Acc.**	**Prec.**	**Rec.**	**F_1**
1	90.8%	94.1%	84.4%	89.0%
2	91.4%	94.4%	22.9%	36.9%
3	94.8%	88.1%	27.3%	41.7%
4	97.0%	76.0%	22.0%	34.1%

was developed for a setting at which no information on how the web page originally looked is available, restricting the measure to guessing the importance of resources on mere hints. Also, the employed heuristics may suffer from genre dependence, where single heuristics that work well for all genres of web pages might just not exist. For comparison, the conceptually much simpler RMSE, which uses information on visual changes for the reproduction, achieves a much stronger correlation of $r = 0.48$. Given the aforementioned restrictions on what RMSE considers, this good result is somewhat surprising. Therefore, the extent to which pixels changed in a reproduction seems to be a strong feature for reproduction quality, and the cases in which the changes are misleading (e.g., due to content shifting up, which leads to a lot of pixel changes) seem to be in the minority. Finally, the best correlation is achieved by the neural network ($r = 0.57$). This is not surprising, as it also incorporates the most information on the web page visuals, and since it employs labeled data and learning theory to reach an empirical understanding of reproduction quality.

Table 3 (left) provides a more detailed look at the effectiveness of the neural network measure, showing the scores underlying the respective scatter plot in Figure 6. For 4,698 of the 6,348 annotated reproductions of the Webis Web Archive 17, the measure agrees on the annotated score. This corresponds to a classification accuracy of 74.0%. For comparison, classifying all reproductions with the majority score (score 2, cf. Table 2) would yield an accuracy of 52.1%. Moreover, if not correct, the measure is most of the time very close to the human annotation: only for 578 pages (9.1% of reproductions) the estimated score is more than one off of the ground-truth. Note that these calculations already omit the trivial cases without a difference in the screenshots. If the 3,652 trivial cases are included, the accuracy, for example, increases from 74.0% to 83.5%.

For an application of the neural network for immediate automatic reproduction quality assessment, also the trivial cases are relevant, but usually only two classes exist: the reproduction quality is still acceptable or not. Which score is still acceptable, however, depends on the specific application. Table 3 (right) shows the achieved effectiveness as measured by standard metrics for different choices of a minimum desired quality. For debugging and fast improvement of the reproduction approach, the recall—percentage of all not acceptable reproductions that were identified—is the most important metric. A high recall (84.4%) is only achieved for the case of separating quality-1 reproductions from others. Thus the neural network can be employed for debugging reproduction approaches that aim at this quality—which are probably most if not all approaches. On the other hand, the good precision values indicate that the neural network can be used to automatically detect which web pages fail to be reproduced with acceptable quality, which is important for generating large-scale web archives.

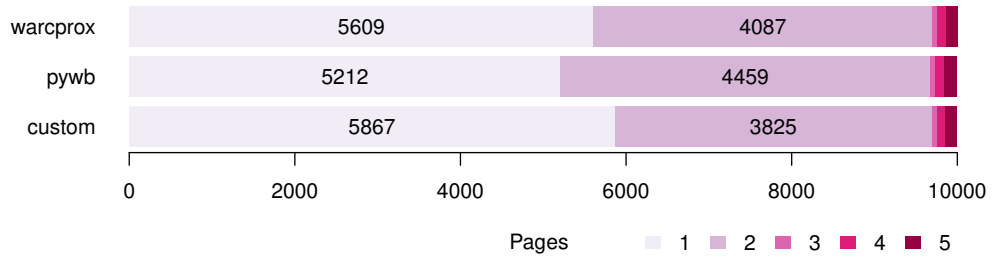


Fig. 7. Distribution of reproduction quality scores for each of the tested reproduction approaches as assessed by the neural network measure.

6.2 Comparison of Reproduction Approaches Using the Quality Measures

As described in Section 3.5, we noticed differences in the reproduction quality of the existing reproduction approaches. In order to substantiate this impression, we used the reproduction quality measures to automatically assess the reproduction approaches over the entire Webis Web Archive 17. Since the neural network reproduction quality measure performed best, we used it for the quality assessment, demonstrating its usefulness for a comparative evaluation of alternative web archiving and reproduction approaches. While the use of the still less-than-optimal model may introduce some bias, we believe that all three reproduction approaches are similarly affected, yielding a fair comparison. This saves us significant resources, since otherwise we would have had to repeat the crowdsourcing assessment for each of the three alternative reproduction approaches in question.

Figure 7 shows the distribution of the quality scores for each reproduction approach. The achieved quality of the three approaches is rather similar with the custom approach reproducing the most pages with the best score. This visual impression is confirmed by the average scores that the reproduction approaches achieve: 1.48 (custom), 1.51 (warcprox) and 1.55 (pywb). While this result does not suggest that a single reproduction approach is clearly preferable over the others, it allows for future analyses to pinpoint the common and different problems of these approaches.

7 CONCLUSION

Many branches of computer science rely heavily on web data, but the ephemeral nature of the web poses a challenge to the reproducibility of insights gained from such data. While tools for web archiving exist, they are not tailored to the needs of scientists, nor has their effectiveness been systematically evaluated to date. We develop a new archiving tool, define a new quality assessment task, provide a tailored dataset for it, and introduce an automatic evaluation measure.

The new Webis Web Archiver can be used to construct reproducible web corpora and user study setups, to replay recorded user behavior, and to run user simulations in a reproducible and authentic manner. However, as our analysis on 10,000 carefully sampled web pages shows, the reproduction of web pages from archives is far from perfect. To facilitate improvements in this regard, we cast the problem of assessing the effectiveness of web page reproduction software into the new task of immediate automatic reproduction quality assessment, where screenshots taken from the original web page and its reproduced version are used to judge the reproduction quality. With the Webis Web Archive 17, we provide the first benchmark dataset for this task, compare three measures for a respective automatic evaluation, and show that a neural network-based measure is able to automatically detect low reproduction quality, while achieving a high correlation with human annotations (Pearson's $r = 0.57$), and while reaching a recall between 22% and 84% as well as a precision between 76% and 94%, dependent on the desired reproduction quality threshold.

Even though the accurate archiving and reproduction of web pages has made impressive progress, the state of the art for scientific corpus construction is still simply downloading the HTML source code of a page. Archiving and reproduction both face several challenges that need to be addressed by future software engineering and research. For example, during archiving, some web content is tailored toward an assumed location of the client, so this location should be another parameter for an archiving tool. Some websites block clients that perform unusual request patterns (also across websites) in an attempt to avoid bots, which can be problematic for large-scale archiving. However, using proxy servers to address these two problems is a double-edged sword, as some websites block known proxy servers by default. As another improvement, popovers or splash screens should usually not be part of the archive, which could be handled by adding (possibly reactive) simulated interactions to the user simulation script that dismiss these overlays during reproduction. Furthermore, as content vanishes, archiving software should include a technique for detecting error pages that do not send the appropriate error code and alert the user if they are attempting to archive such an error page.

Current problems for reproduction that can be tackled by software engineering are data streaming based on HTTP range requests (necessary for videos), where ranges during archiving and reproduction may differ; and reproduction of server-sent events (known as HTTP push), where the web server sends additional information without a request from the client. Additionally, the approaches to fuzzy request-response matching need improvements, for example, by employing better heuristics, specifically tailored request-similarity measures, machine learning to classify which parts of a request are important, or a mixture of all these. Moreover, the off-the-shelf neural network used for automatic reproduction quality assessment—which presents a strong baseline—should be tailored more to the task. For example, extending and cropping the screenshots might not be optimal for the assessment. Instead, more sophisticated approaches that use recurrent networks after the convolutional layer to handle inputs of arbitrary size might be a better fit [13]. Finally, it may be worthwhile to fine-tune the number of layers and their parameters.

A further improvement to the Webis Web Archiver would be the capability to capture the screen during archiving and to store the screen capture as a video. Such videos would add a new preservation element that enriches the archive files. Furthermore, such videos would assist users in writing and debugging user simulation scripts. Regarding the analysis of our tool, more detailed case studies for the different web page genres within our dataset will probably yield new insights into the difficulties and problems of current archive and reproduction technologies. While the automatic quality assessment is a solid first step towards an improvement of these technologies, an in-depth error analysis is needed to pinpoint the current flaws. However, as long as web technology continues to evolve, new challenges for web page archiving and reproduction will arise, demanding a continuous development of the tools and techniques that aim to create reproducible web corpora.

REFERENCES

[1] Alexa Internet, Inc. The top 500 sites on the web, 2017. https://www.alexa.com/topsites.

[2] Y. Alnoamany, A. Alsum, M. C. Weigle, and M. L. Nelson. Who and what links to the internet archive. In *Proceedings of TPDL 2013*, pages 346–357.

[3] B. R. Ayala, M. E. Phillips, and L. Ko. Current quality assurance practices in web archiving. Technical report, University of North Texas Libraries, Aug. 2014.

[4] L. Azzopardi. Simulation of Interaction: A Tutorial on Modelling and Simulating User Interaction and Search Behaviour. In *Proceedings of SIGIR 2016*, pages 1227–1230.

[5] J. Bailey, A. Grotke, E. McCain, C. Moffatt, and N. Taylor. Web archiving in the United States: A 2016 survey. *National Digital Stewardship Alliance*, 2017.

[6] V. Banos, Y. Kim, S. Ross, and Y. Manolopoulos. CLEAR: A credible method to evaluate website archivability. *Proceedings of iPRES 2013)*.

[7] V. Banos and Y. Manolopoulos. A quantitative approach to evaluate website archivability using the CLEAR+ method. *International Journal on Digital Libraries*, 17(2):119–141, June 2016.

[8] M. Baroni, F. Chantree, A. Kilgarriff, and S. Sharoff. CLEANEVAL: A competition for cleaning web pages. In *Proceedings of LREC 2008*.

[9] J. Brunelle, M. Kelly, H. SalahEldeen, M. C. Weigle, and M. L. Nelson. Not all mementos are created equal: Measuring the impact of missing resources. *International Journal on Digital Libraries*, 16(3–4):283–301, May 2015.

[10] F. Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

[11] I. I. P. Consortium. OpenWayback, 2017. http://netpreserve.org/web-archiving/openwayback.

[12] cyberpower678. InternetArchiveBot, 2017. https://meta.wikimedia.org/wiki/InternetArchiveBot.

[13] Y. Deng, A. Kanervisto, and A. M. Rush. What you get is what you see: A visual markup decompiler. *CoRR*, abs/1609.04938, 2016.

[14] Docker Inc. Docker – build, ship, and run any app, anywhere, 2017. https://www.docker.com.

[15] M. Faheem. Intelligent crawling of web applications for web archiving. In *Proceedings of WWW 2012 (Companion)*, pages 127–132.

[16] D. Gomes, J. Miranda, and M. Costa. A survey on web archiving initiatives. In *Proceedings of TPDL 2011*, pages 408–420.

[17] Google Webmaster Central Blog. Deprecating our AJAX crawling scheme, 2015. https://webmasters.googleblog.com/2015/10/deprecating-our-ajax-crawling-scheme.html.

[18] K. Gyllstrom, C. Eickhoff, A. P. de Vries, and M.-F. Moens. The downside of markup: Examining the harmful effects of CSS and JavaScript on indexing today's web. In *Proceedings of CIKM 2012*, pages 1990–1994.

[19] A. Hidayat. PhantomJS - scriptable headless webkit, 2017. https://github.com/ariya/phantomjs.

[20] H. Hockx-Yu, L. Crawford, R. Coram, and S. Johnson. Capturing and replaying streaming media in a web archive—A British Library case study. *Proceedings of iPRES 2010)*.

[21] ImageMagick Studio. Convert, edit, or compose bitmap images @ ImageMagick, 2017. https://www.imagemagick.org.

[22] A. Karpathy. CS231n Convolutional Neural Networks for Visual Recognition, 2017. http://cs231n.github.io/convolutional-networks.

[23] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[24] C. Kohlschütter, P. Fankhauser, and W. Nejdl. Boilerplate detection using shallow text features. In *Proceedings of WSDM 2010*, pages 441–450.

[25] I. Kreymer. Python WayBack for web archive replay and URL-rewriting HTTP/S web proxy, 2017. https://github.com/ikreymer/pywb.

[26] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computing*, 1(4):541–551, Dec. 1989.

[27] P. Lightbody. Browsermob proxy: A free utility to help web developers watch and manipulate network traffic from their AJAX applications, 2017. https://github.com/lightbody/browsermob-proxy.

[28] J. Lin, I. Milligan, J. Wiebe, and A. Zhou. Warcbase: Scalable analytics infrastructure for exploring web archives. *Journal on Computing and Cultural Heritage*, 10(4):22:1–22:30, July 2017.

[29] National Library of New Zealand and British Library. Web Curator Tool Project, 2014. http://webcurator.sourceforge.net/.

[30] H. Obendorf, H. Weinreich, E. Herder, and M. Mayer. Web page revisitation revisited: Implications of a long-term click-stream study of browser usage. In *Proceedings of CHI 2007*, pages 597–606.

[31] P. Schaer and M. Neumann. Enriching existing test collections with OXPath. In *Proceedings of CLEF 2017*, pages 152–158.

[32] Selenium Contributors. SeleniumHQ Browser Automation, 2017. http://www.seleniumhq.org.

[33] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[34] StatCounter. Screen resolution stats worldwide, June 2018. http://gs.statcounter.com/screen-resolution-stats.

[35] S. Stewart, D. Burns, and Mozilla. WebDriver W3C Candidate Recommendation, 05 June 2018. https://www.w3.org/TR/webdriver.

[36] The Common Crawl team. January 2017 Common Crawl archive, 2017. http://commoncrawl.org/2017/02/january-2017-crawl-archive-now-available.

[37] The Internet Archive. Brozzler, 2017. https://github.com/internetarchive/brozzler.

[38] The Internet Archive. Umbra, 2017. https://github.com/internetarchive/umbra.

[39] The Internet Archive. Warcprox – WARC writing MITM HTTP/S proxy, 2017. https://github.com/internetarchive/warcprox.

[40] The Lemur Project. The ClueWeb09 Dataset, 2009. http://lemurproject.org/clueweb09/.

[41] The Lemur Project. The ClueWeb12 Dataset, 2012. http://lemurproject.org/clueweb12/.

[42] The Memento Development Group. About the Memento project, 2017. http://mementoweb.org/about, RFC 7089.

[43] The Open Preserve Foundation. Pagelyzer, 2014. https://github.com/openpreserve/pagelyzer.

[44] P. Tzekou, S. Stamou, N. Kirtsis, and N. Zotos. Quality assessment of Wikipedia external links. In *Proceedings of WEBIST 2011*, pages 248–254.
[45] W3schools.com. Browser Display Statistics, 2018. https://www.w3schools.com/browsers/browsers_display.asp.
[46] W3Techs. Usage of JavaScript libraries for websites, October 2017. https://w3techs.com/technologies/overview/javascript_library/all.
[47] D. P. Wiggins and The Open Group, Inc. Xvfb - virtual framebuffer X server for X Version 11, 2017. https://www.x.org/releases/X11R7.7/doc/man/man1/Xvfb.1.xhtml.
[48] wolfcw. libfaketime (FakeTime Preload Library) - report faked system time to programs without having to change the system-wide time, 2014. http://www.code-wizards.com/projects/libfaketime/.

A NEURAL NETWORK ARCHITECTURE

As described in Section 5, we employ a neural network architecture based on VGGNet with 16 weight layers [33], but with the fully connected layers removed [22]. Table 4 lists all layers of the network, their parameters, and output sizes. All convolutional layers use a zero padding of 1 to preserve the output size. We train the model in 100 iterations with Keras [10], using the Adam optimizer [23] with default parameters and categorical cross-entropy as the loss function. In addition to the pre-processing described in Section 5, we normalize each image by subtracting the mean value and dividing by the standard deviation.

Table 4. Neural network architecture used for automatic estimation of reproduction quality.

Layer	**Layer parameters**			**Activation**	**Output size**
	Extent	Stride	Filters		
Input					1×128×128
Convolutional	3×3	1×1	64	Rectified Linear Unit	64×128×128
Convolutional	3×3	1×1	64	Rectified Linear Unit	64×128×128
Max Pooling	2×2	2×2			64×64×64
Convolutional	3×3	1×1	128	Rectified Linear Unit	128×64×64
Convolutional	3×3	1×1	128	Rectified Linear Unit	128×64×64
Max Pooling	2×2	2×2			128×32×32
Convolutional	3×3	1×1	256	Rectified Linear Unit	256×32×32
Convolutional	3×3	1×1	256	Rectified Linear Unit	256×32×32
Convolutional	3×3	1×1	256	Rectified Linear Unit	256×32×32
Max Pooling	2×2	2×2			256×16×16
Convolutional	3×3	1×1	512	Rectified Linear Unit	512×16×16
Convolutional	3×3	1×1	512	Rectified Linear Unit	512×16×16
Convolutional	3×3	1×1	512	Rectified Linear Unit	512×16×16
Max Pooling	2×2	2×2			512×8×8
Convolutional	3×3	1×1	512	Rectified Linear Unit	512×8×8
Convolutional	3×3	1×1	512	Rectified Linear Unit	512×8×8
Convolutional	3×3	1×1	512	Rectified Linear Unit	512×8×8
Max Pooling	2×2	2×2			512×4×4
Output				Softmax	5