Saving Data Journalism

Using ReproZip-Web to Capture Dynamic Websites for Future Reuse

Katherine Boss
New York University
USA
katherine.boss@nyu.edu
0000-0003-2148-8386

Vicky Steeves
New York University
USA
vicky.steeves@nyu.edu
0000-0003-4298-168X

Rémi Rampin
New York University
USA
remi.rampin@nyu.edu
0000-0002-0524-2282

Fernando Chirigati
New York University
USA
fchirigati@nyu.edu
0000-0002-9566-5835

Brian Hoffman
Independent
USA
brianjhoffman@gmail.com
0000-0003-2333-0603

Abstract – While dynamic and interactive Web applications are becoming increasingly common to convey news and stories to people all around the world, their technological complexity makes it hard to archive and preserve such applications, and as such, they are being lost. We present ReproZip-Web, an open-source prototype aimed at saving these news applications from extinction. ReproZip-Web leverages ReproZip, a computational reproducibility tool, and Webrecorder, a tool for recording Web resources, to automatically and transparently capture and replay dynamic Websites. The prototype creates a bundle that contains all the information needed to reproduce a news application, and its lightweight nature makes it ideal for distribution and preservation. We will present our ongoing work on the prototype, and also discuss some use cases and avenues for future development.

Keywords – Web archiving, Emulation, Data Journalism, Emulation-based web archiving, ReproZip

I. Introduction

Data journalism stories are among the most complex, innovative, and original stories being produced by newsrooms today. These projects, created by news organizations in dozens of countries, are custom-built websites that display content dynamically in the browser. On the back end, many of these works also allow readers to explore, query, and inspect data related to a news story. Iconic examples of data journalism projects include the Panama Papers, "The Color of Debt" by ProPublica, and "Gun Deaths in Your District" by *The Guardian*[1]–[3]. These works are often called interactives or "news applications," and are a continuation of a larger branch of journalism, previously called computer-assisted reporting, which has increased dramatically in production and popularity as societies have become more data-driven [4, p. 154]. Yet because of their technological complexity, these sites cannot be fully or systematically captured by current web archiving tools. Current web archiving technologies, which have been successful in capturing snapshots of static news content, fail to capture the look, feel, and functionality of dynamic content such as social media feeds, interactive maps, visualizations, and database-reliant websites. Web archiving of dynamic content would require the capture of assets located on the web server, many of which are protected, rightly, by firewalls and other security measures. As such, theses websites are being lost [4]–[6].

Beyond the technical challenges of capturing and archiving dynamic websites, there are organizational barriers to advancing a solution. In the majority of newsrooms in the United States, newsroom libraries have long been shuttered [4], [5]. This trend, which accelerated in the 2000s as newspapers became less profitable and budget cuts became common, has left newsrooms bereft of any archiving or preservation expertise within the organization. Few, if any, newsroom staff wake up each day thinking about how to save their digital content. Consequently, web archiving has always been an afterthought; in 2002 only 7% of newsrooms with libraries (already a minority) were conducting any sort of web archiving [4, p. 44]. Though outside organizations such as the Internet Archive have stepped in to save millions of pages of articles [6], much more content has been lost.

These losses have a detrimental impact on the collective cultural record and the future of research based on journalism [7, p. 1208]. News stories are known as the "first draft of history," and this makes

them an important and frequent object of study for scholars across the academy. News websites are an important research artifact [8], and demand for them will likely only grow, given that the way the public finds, reads, and shares news is increasingly online [9], [10].

To address this problem and save interactive news websites, our research team has built an emulation-based web archiving tool, which to our knowledge, is the first of its kind.

II. About the Project

Emulation as an archiving and preservation strategy was introduced as a concept decades ago by Rothenberg [11], though the infrastructure, skill, and knowledge to create emulators has only recently made it a feasible, economical, or practical option [12, p. 2]. Advances in cheaper and more abundant digital storage in the last decade have paved the way for emulation projects, and coincided with the belief in the digital archiving community that to save digital objects for the long term, we must emulate them in their original computational environment [12]–[16]. Preservation of encapsulated projects, rather than websites (both static and interactive), is currently underway at several institutions, including Rhizome, the Internet Archive, Carnegie Mellon, New York University, Yale and the Software Preservation Network, Deutsche Nationalbibliothek, and the British Library [12], [17], [18]. These pioneering projects have advanced the capture and preservation of system images and the frameworks that allow users to replay them on modern machines [7]. However, none of these initiatives have yet addressed a scalable, full-stack, emulation-based web archiving tool that could systematically capture the large volume of interactive news projects being published daily. Our prototype is the first tool of this kind.

III. A Prototype to Preserve News Apps

To this end, we extended an existing open-source project, ReproZip, originally designed for computational reproducibility [19]. ReproZip is a tool that automatically captures all the dependencies of a software application originally run in a Linux environment, and creates a single, distributable bundle that can be used to reproduce the entire experiment in another environment (e.g., on Linux, Windows, or Mac). ReproZip works in two steps:

Packing. In the *packing* step, the tool traces all the system calls related to the execution of the application, capturing all of the dependencies at the OS level, including software, data files, databases, libraries, environment variables, and OS and hardware information. Using this information (which can optionally be customized by the user), ReproZip creates a bundle for it: an `.rpz` file containing all of the dependencies.

Unpacking. In the *unpacking* step, given an `.rpz` file, other users can use ReproZip to automatically and transparently set up the packaged application in their environment, even if their OS is different than the one used for the creation of the application. This is possible thanks to emulation- and container-based tools leveraged by ReproZip.

ReproZip successfully captures and reproduces the software environment, including involved scenarios such as the client-server ones that are common to news apps. One of the current limitations of ReproZip, however, is that it cannot capture front-end remote dependencies. As we discovered in the course of this research, news apps often depend on remote front-end files, e.g., JavaScript, cascading style sheets (CSS), fonts, and other resources. Consequently, the bundle created during the packing step is incomplete. When unpacking, these front-end files will only work assuming they are still accessible and live: if they become inaccessible, the look, feel, and interactivity of these news apps is entirely lost.

Our extension to ReproZip, called ReproZip-Web, aims to address this limitation, and to therefore fully capture and preserve news apps. To capture these remote resources and add them to the `.rpz` bundle, we leverage the Core Python Web Archiving Toolkit (pywb) software library from Rhizome's Webrecorder project, which records resources as they are loaded by the browser and stores them as Web ARChive (`WARC`) files [20], [21].

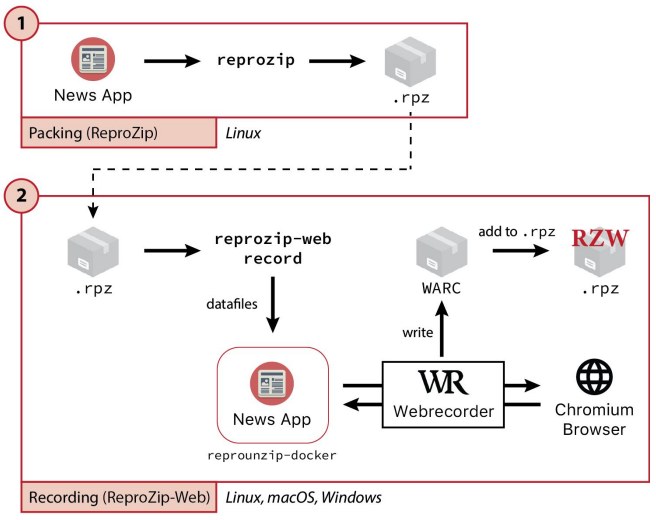

Figure 1: Packing and recording a news app with ReproZip-Web.

Packing and Recording the News App. The prototype assumes that the back-end of the news app (e.g., databases, web framework) has already been packed by ReproZip, thus creating a `.rpz` bundle. Given this `.rpz` file as input, our prototype simultaneously launches the emulated news app, a Webrecorder server, and an instance of the Chromium browser, which is controlled via the Chrome DevTools Protocol (CDP). With the application unpacked and being reproduced, the browser makes requests to Webrecorder, which acts as a proxy while it builds a `WARC` archive containing all the resources that were requested. Once the browser has finished loading the news app, our tool consolidates the `.rpz` and `WARC` data into a single package: a new `.rpz` file. Figure 1 depicts the full workflow for packing and recording a news app.

Replaying the News App. When a user replays the `.rpz` package with the prototype, the two servers are again launched simultaneously: the emulated news app in a Docker container (via `reprounzip-docker`), and a Webrecorder server providing the web resources archived in the package's embedded `WARC` data. A proxy server (Nginx) receives all network requests from the browser and routes them to the appropriate server, using the domain of the request URI to determine which server can fulfill the request. Our tool also has a mode in which the requests are handled directly by the Wayback server, allowing archival packages to be played back over the internet and without modification to the researcher's browser. Figure 2 depicts the full workflow for replaying a news app.

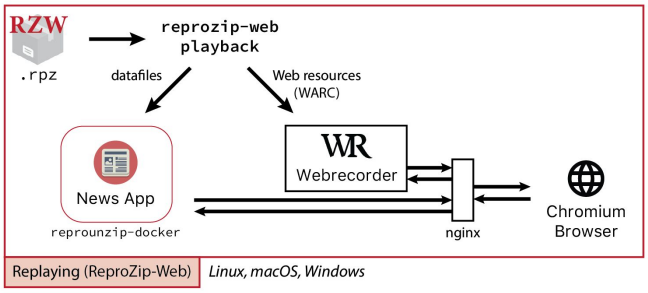

Figure 2: Replaying a news app with ReproZip-Web.

Our final product is a tool that effectively harnesses ReproZip and Webrecorder simultaneously to create a single archival package emulating both the state of the web server on which the news app runs, and the state of the relevant parts of the world wide web at the time the app was published. Our prototype is available at https://github.com/reprozip-news-apps/reprozip-web.

IV. Use Cases

Our primary test case for the tool was a data-driven news app called Dollar for Docs, from ProPublica [24]. This news app was built using Ruby on Rails, MySQL, and Elasticsearch on a Linux server. Our first step, prior to development, was to capture the app and its environment using ReproZip, and to review the emulated site. Our first observation was that the look and feel of the app broke entirely when the browser was restricted from accessing resources on the world wide web (as expected).

When we implemented ReproZip-Web, we found that we could indeed view the news app with the correct look and feel, and began to drill down to specific limitations. One unexpected issue was the inoperability of the paginated search feature, which led us to the realization that certain Ruby on Rails files had not been captured by ReproZip due to their "lazy loading": one of the search results pages raised a server error because the HTML template had never been captured in the original package. This led to our decision to modify ReproZip with additional rules aimed at capturing a complete Ruby on Rails application, and the recognition that such rules may be required for each major web publishing platform.

We also tested our prototype on The Guardian Elections Poll Projection, an open-source news app from The Guardian that was built using Node.js [22], [23]. We were able to successfully reproduce and replay this news app as well, even when restricting the browser from accessing external resources. A demonstration video of our prototype using this news app is available at http://bit.ly/2O3Q4Ee.

V. Future directions

Much work is needed in testing, developing, and generalizing the prototype. ReproZip was developed to only capture what is executed during an application, but many interactive websites require that all parts of the environment (e.g., gems for Ruby) be included, and front-end content to be recorded and packaged. To account for this, we implemented an extra rule in ReproZip-Web to detect and automatically capture Ruby gems even if they were not executed while the application was running. In the next phase of development, we plan to implement more rules for other languages, optimizing for languages commonly used in data journalism projects. This will generalize the tool for a wider range of dynamic websites.

In the course of this project, while assessing different news apps, we also found out that some news apps require access to external APIs and data (e.g.: sites that dynamically upload and download data from Amazon S3 stores). Reproducing this

scenario is challenging and we plan to investigate different solutions in the future.

We also plan to gather information about the needs of data journalists, system administrators, and other stakeholders that would use this tool. We will work closely with potential users in testing the prototype and gathering their feedback on its usability, the time commitment it would require, and the types of software that users would be willing and able to install on the production servers where ReproZip would be deployed. This feedback will be instrumental in building a finished tool that fits the needs and workflows of newsrooms.

Finally, a graphic user interface (GUI) is necessary to make the tool user-friendly and simple to deploy. While secondary users of ReproZip-Web have access to a GUI to replay the news applications from the `.rpz` file, there is currently no GUI for those originally packing the work, though there are in-depth command line utilities. Acknowledging that the command line is a high learning curve, adding a packing GUI would allow newsrooms to utilize ReproZip-Web to capture and archive their interactive news content at scale, potentially packing dozens of projects each month.

VI. Discussion

Cultural heritage institutions can leverage ReproZip-Web to create distributable and preservation-quality bundles of complex web applications that can be replayed in-browser or on desktop machines. ReproZip-Web not only captures the applications and all their dependencies, but also automatically re-configures these in any other environment. This has not only simplified the process of capturing these important cultural artifacts for posterity, but also allows anyone, on any type of computer, to access and replay them in their original computational environment -- seeing the applications as they were experienced from the onset.

The `.rpz` is ideal for preserving complex applications such as these because it is extensible, lightweight, and open. The ReproZip ecosystem is such that after a journalist or archivist captures a news application with ReproZip-Web, they can then use any current or future ReproZip unpacker to replay it. Currently, to replay a news app from an `.rpz` file, users can choose four unpackers: `reprounzip-directory`, `reprounzip-chroot`, `reprounzip-vagrant`, and `reprounzip-docker`. However, a fifth is on the way (`reprounzip-singularity`, to unpack via Singularity containers), and a sixth has been contributed to the project from an outside colleague (`reprounzip-benchexec`), only possible because ReproZip and it's file format are open source. ReproZip bundles can be unpacked and replayed with any virtual machine or container software; so as these software wax and wane out of popularity and use, so can the ReproZip ecosystem be adjusted to create new unpackers, or depreciate old ones, without compromising the ability to use and replay old `.rpz` files [25].

Additionally, unless the size of the input data for a news application is on the Terabyte scale, the `.rpz` files are quite small and easily distributable. To date in our testing, we have yet to create an `.rpz` file over 800MB. These archival bundles are easily shared and distributed, as well as stored at a much lower cost, without compromising on the ability to reuse, replay, and preserve the contents of the news applications.

ReproZip-Web allows newsrooms and archives alike to package complex news applications with all their dependencies into a single distributable and preservable `.rpz` bundle, from which users can replay the news application and archivists can ensure long-term preservation. Wide-scale adoption of this software and archiving practice would be a giant leap forward in saving data journalism projects for history, posterity, and the cultural record.

Acknowledgements

We wish to acknowledge the Institute for Museum and Library Services for their support of this project (LG-87-18-0062-18), as well as the PI of that grant Dr. Michael Stoller, and our project partner ProPublica for allowing us to test our prototype on their news apps. Thanks to Dr. Juliana Freire, the PI of the ReproZip project, for her continued support, and to Bofei Zhang, our research assistant, for testing the prototype and writing protocols. Thanks to Rhizome for their work on Webrecorder, without which this project would have had even greater barriers to overcome. Lastly, we would like to acknowledge the support from the Gordon and Betty Moore Foundation as well as the Alfred P. Sloan Foundation via the Moore-Sloan Data Science Environment for supporting continuing development of ReproZip.

References

[1] International Consortium of Investigative Journalists, "The Panama Papers: Exposing the Rogue Offshore Finance Industry," *International Consortium of Investigative Journalists*, 03-Apr-2016. [Online]. Available: https://www.icij.org/investigations/panama-papers/. [Accessed: 12-Mar-2019].

[2] P. Kiel, A. Waldman, and A. Shaw, "The Color of Debt," *ProPublica*, 08-Oct-2015. [Online].

Available: https://projects.propublica.org/garnishments/. [Accessed: 12-Mar-2019].
[3] K. Davis, R. Harris, N. Popovich, and K. Powell, "Gun deaths in your district: what have your elected representatives done?," *The Guardian*, 2015. [Online]. Available: http://www.theguardian.com/us-news/ng-interactive/2015/dec/14/gun-control-laws-congress-shooting-deaths-nra-lobby-campaign-donations. [Accessed: 12-Mar-2019].
[4] N. Paul and K. A. Hansen, "Reclaiming News Libraries," *Library Journal*, vol. 127, no. 6, p. 44, 4/1/2002 2002.
[5] K. A. Hansen and N. Paul, *Future-Proofing the News: Preserving the First Draft of History*. Lanham: Rowman & Littlefield Publishers, 2017.
[6] The Internet Archive, "Wayback Machine Hits 400,000,000,000!," *Internet Archive Blogs*, 09-May-2014. .
[7] M. Broussard and K. Boss, "Saving Data Journalism," *Digital Journalism*, vol. 6, no. 9, pp. 1206–1221, Oct. 2018.
[8] N. Brügger, "Website history and the website as an object of study," *New Media & Society*, vol. 11, no. 1–2, pp. 115–132, Feb. 2009.
[9] Pew Research Center, "Where People Get Their News," *Pew Research Center | Global Attitudes & Trends*, 04-Oct-2007. [Online]. Available: https://www.pewglobal.org/2007/10/04/chapter-7-where-people-get-their-news/. [Accessed: 20-Mar-2019].
[10] Pew Research Center Journalism Project, "The Growth of Digital Reporting," *Pew Research Center's Journalism Project*, 26-Mar-2014. [Online]. Available: http://www.journalism.org/2014/03/26/the-growth-in-digital-reporting/. [Accessed: 26-Mar-2014].
[11] J. Rothenberg, "Ensuring the Longevity of Digital Documents," *Scientific American*, vol. 272, no. 1, pp. 42–47, Jan. 1995.
[12] D. S. Rosenthal, "Emulation & Virtualization as Preservation Strategies," 2015.
[13] S. Granger, "Emulation as a Digital Preservation Strategy," *D-Lib Magazine*, vol. 6, no. 10, Oct-2000.
[14] L. Johnston, "Preserving News Apps | The Signal," 11-Mar-2014. [Online]. Available: //blogs.loc.gov/thesignal/2014/03/preserving-news-apps/. [Accessed: 16-Jul-2017].
[15] K. Rechert, I. Valizada, S. D. von, and J. Latocha, "bwFLA – A Functional Approach to Digital Preservation," *PIK - Praxis der Informationsverarbeitung und Kommunikation*, vol. 35, no. 4, pp. 259–267, 2012.
[16] D. von Suchodoletz and J. van der Hoeven, "Emulation: From Digital Artefact to Remotely Rendered Environments," *International Journal of Digital Curation*, vol. 4, no. 3, pp. 146–155, Jul. 2009.
[17] S. Anderson, E. Cochrane, E. Gates, and J. Meyerson, "About EaaSI – Saving Software Together," *About EaaSI – Saving Software Together*, 2018. [Online]. Available: https://www.softwarepreservationnetwork.org/eaasi/. [Accessed: 13-Mar-2019].
[18] J. Kim, "Access and Discovery of Born-Digital Archives," New York University, Jun. 2015.
[19] F. Chirigati, R. Rampin, D. Shasha, and J. Freire, "ReproZip: Computational Reproducibility With Ease," presented at the 2016 ACM SIGMOD International Conference on Management of Data (SIGMOD), San Francisco, USA, 2016, pp. 2085–2088.
[20] I. Kreymer, *Core Python Web Archiving Toolkit for replay and recording of web archives: webrecorder/pywb*. Webrecorder, 2013.
[21] I. Kreymer and D. Espenschied, "Webrecorder: A project by Rhizome." [Online]. Available: https://webrecorder.io/. [Accessed: 10-Jan-2018].
[22] C. Zapponi *et al.*, "Election 2015: The Guardian poll projection," *The Guardian*, 07-May-2015. [Online]. Available: http://www.theguardian.com/politics/ng-interactive/2015/feb/27/guardian-poll-projection. [Accessed: 16-Mar-2019].
[23] C. Zapponi, C.-J. (Apple) C.Fardel, and S. Clarke, *Tracking the UK election polls for 2015*. The Guardian, 2016.
[24] L. Groeger, C. Ornstein, M. Tigas, and R. G. Jones, "Dollars for Docs," *ProPublica*, 2010. [Online]. Available: https://projects.propublica.org/docdollars/. [Accessed: 12-Dec-2015].
[25] V. Steeves, R. Rampin, and F. Chirigati, "Using ReproZip for Reproducibility and Library Services," *1*, vol. 42, no. 1, pp. 14–14, 2018.