Enabling Correct and Efficient Web Archiving in the Era of JavaScript

Two key goals of any web archive are to serve archived pages with high fidelity to users and to minimize operational costs. However, the widespread use of JavaScript (JS) on the web is currently a significant hindrance. Today, web archives crawl and save large amounts of JS and, yet, the snapshots of many JS-heavy pages fail to accurately mimic how those pages originally looked and functioned. To address these seemingly contradictory problems, University of Michigan requests $398,714 in funding for a three-year Applied Research project which aligns with the NLG-L program's Objective 3.2 under Goal 3. We seek to develop methods which will 1) ensure that archived pages load without any errors for users, and 2) enable budget-constrained libraries and museums to archive many more pages than they can today. Our work will be guided by an advisory board comprising representatives from a variety of organizations that archive web pages, and we seek to inform the development of next-generation archival software and services.

Project Justification

If one compares the web of today with that from a decade ago, one dramatic change that is apparent is the increased use of JavaScript, up from around 150 KB on the median web page in 2010 to over 500 KB today. To cope with this increase, browser-based crawlers such as Brozzler and Browsertrix have been developed to enable web archives to save the resources on any page that are fetched by JS. In addition, when a user interacts with a page (e.g., clicks a button or hovers over an image), systems such as Conifer enable archival of the content fetched by the JS which gets executed to handle the user's interaction.

But, existing solutions have two significant shortcomings, both of which stem from properties of modern JavaScript which web archives currently do not account for.

Non-deterministic JS execution breaks archived pages. First, across multiple loads of a page, the manner in which JS code on the page gets executed can vary based on: 1) characteristics of the client device, e.g., Chrome or Safari, 3G or WiFi, or even the resolution of the display, and 2) values returned by the web browser in response to requests for random numbers, the current time, etc. As a result, when a user loads an archived page snapshot, the browser's execution of JS on the page might cause it to request resources which were never crawled.

Non-functional JS hampers archival efficiency. Second, a significant fraction of the JS on a typical page relies on the user's device interacting with the page's servers, e.g., JS that enables a page's provider to push real-time notifications to the user and JS that enables users to post comments. However, such functionality cannot work on an archived page. Therefore, crawling and storing the JS that powers these capabilities consumes network, compute, and storage, without any resultant benefits to preserving page fidelity.

Example of adverse impact in practice. Based on our review of the literature and our conversations over the last two years with various stakeholders involved in web archiving, there is not much empirical data or widespread awareness about the aforementioned problems. Therefore, drawing upon our prior work in programmatically instrumenting and analyzing JS code, we showed in a recent paper that the Internet Archive's snapshots for many JS-heavy pages render with missing page resources and runtime errors reported by the browser. We showed that web archives cannot sidesteps these problems by archiving pages as screenshots because modern web pages often include interactive features (such as menu bars and image carousels) that are powered by JS. Moreover, in a corpus of 1 million page snapshots (roughly 3500 page snapshots each for 300 websites) that we downloaded from the Internet Archive, we were able to safely discard 36% of the JavaScript bytes without any loss of fidelity.

Project Work Plan

To address the impact of JavaScript on web archives, we propose to tackle the following three research questions.

- *RQ1:* Given a set of archived resources (i.e., HTML, CSS, JS, images, etc.) for a web page, how can we programmatically assess whether page fidelity might be compromised in *some* future load of the page?
- *RQ2:* How should archival systems be modified so that the archived copy of a page is guaranteed to mimic the original page, as it existed at the time of archival, in *all* loads of that copy?
- *RQ3:* How can an archival crawler identify which subset of the JavaScript code on a page will *never* be used in *any* future load of the page's archived copy and is, therefore, optional for it to fetch and save?

Since a web page is essentially a computer program, it is challenging for a human to reason about all feasible executions of a program. Therefore, to answer *RQ1*, we will use well-known techniques for analyzing JS code such as concolic execution (as we have done in our prior work), and develop a principled methodology to reason about all potential loads of an archived page. By applying this methodology to a diverse collection of web pages, we will assemble a corpus of example JS-heavy pages that are prone to fidelity compromises. We will advertise these examples in a variety of ways (e.g., emails to mailing lists used by the web archival community, talks at IIPC's general assembly, blog posts advertised on social media) to raise awareness of the adverse impacts of JavaScript on archival fidelity.

Second, for *RQ2*, we will study how existing browser-based crawlers such as Brozzler and Browsertrix can be modified to analyze and instrument the JS code that they fetch so that they can make strong guarantees about page fidelity. Today, web archives already rewrite crawled JS to ensure that any resources fetched via JS are requested from the archive and not from the page's original servers. We will extend this practice so that, when archived JS is executed in a user's browser, it will only request resources that were crawled and saved by the web archive. A key goal in doing so will be to preserve any non-determinism which is important for the functionality on the page and does not impact the set of resources fetched, e.g., on an archived copy of NYT's Spelling Bee, users should still be able to shuffle the letters.

Lastly, for *RQ3*, we will develop methods for generating a *filter list* that can be used by archival crawlers to identify JS files that they do not need to fetch. This filter list will comprise rules akin to those used by browser add-ons to block advertisements and trackers on the web, except that the rules we aim to identify will filter out page resources which are important on the live web but serve no functional purpose on archived page copies, e.g., on news sites, code for tracking the number of articles a user has read. In contrast to our recent work, where we manually compiled such a filter list for the previously mentioned corpus of 1 million page snapshots, we seek to automatically generate the filtering rules for any particular site.

We will evaluate our methods by prototyping their use in existing browser-based crawlers such as Brozzler and Browsertrix and examining the impact along three dimensions: a) the rate at which we can crawl pages, b) the amount of storage and CPU needed to save the pages we crawl, and c) the number of missing resources and browser-reported errors when we load and interact with archived pages. Our choices of evaluation corpora and the practical considerations that we must account for will be guided by an advisory board. Jefferson Bailey, the Director of Web Archiving & Data Services at IA, has told us that he would be happy to serve in an advisory role on this project. In addition, we are currently in conversations with representatives from other institutions, and we anticipate recruiting four advisors spanning both developers (e.g., Rhizome project) and users (e.g., IIPC members who use Archive-It) of web archiving services.

Project Results

Via publications in top-tier conferences/journals and talks at venues such as ARLIS/NA, iPRES, and DLF, we will seek to make libraries and museums aware of our results as well as draw more computer scientists to the field of web archiving. By pointing out that all JS code on crawled pages need not be saved and the subset that is archived cannot be saved as is, the primary impact of our work will be to get web archives to rethink how they account for the presence of JavaScript on web pages. The methods we develop will enable archivists to be confident about the preservation of page fidelity and significantly reduce costs for those with limited budgets.

Beyond disseminating the ideas and methods that result our work, we will also strive for real-world impact. For example, our recent work has helped inform improvements in the tools used by the Internet Archive (IA) to patch broken links on Wikipedia. In a similar spirit, we will reach out to libraries with limited budgets and, by enabling them to safely discard JS files that match our filter list, reduce their costs on services such as Archive-It. In the long term, we hope that our work will enable services like Archive-It to give all of their customers the option to filter JS code that will never be executed.

Budget Summary

The estimated three-year budget of $398,714 includes one month of summer salary plus benefits for the PI in each year of the project ($58,799), tuition ($66,988) and stipend plus benefits ($128,346) for one graduate student researcher for all three years, cloud computing to evaluate the proposed work ($6,000), travel expenses to disseminate our findings ($12,000), a $500 honorarium per year for each of five advisory board members ($7,500), and indirect costs ($119,081) at the negotiated rate of 56% of eligible costs.