Web page comparison on post-load DOM trees

Jonathan Thomas (Honors Capstone), Ayush Goel and Prof. Harsha Madhyastha

University of Michigan

Problem

Code is always being rewritten and in constant need of further improvement. As such, source code for web pages changes drastically over time, and it is important to know what it really means for a web page to be the same as another or the same as a previous version of itself. In order to proceed, we introduce a core browser concept.

The main entity we are concerned with is the Document Object Model (DOM) tree, which keeps track of the structure of the web page at a high level and is modified as the page loads.

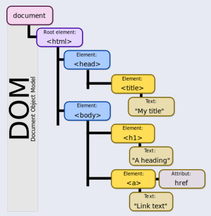

Figure 1: An example DOM

There is a need for an approach and implementation of a more useful way to check differences between two web pages than a naive diff of the source code text. We also wish to get a better understanding of how the DOM differs over different sites as well as environments, which will be explained below.

Objective

In this project, I wrote a tool to determine how similar two web pages are from the perspective of a user visiting either page. This was achieved by comparing the resulting post-load DOM trees of loading two web pages and quantifying how similar they were. Using this tool, we aim to form a better idea of how real web pages change over time.
The major steps for this project included:

- Navigating to a site and dumping the DOM after loading.
- Comparing two loads and reporting the differences.
- Automating the process across several websites and graphing results.

Methodology

To evaluate the appearance of the DOM in addition to the structure, we make a first pass over the source code and serialize the computed CSS styles into the DOM. We also make use of a more discerning algorithm for comparison than a naive tree diff.
In addition to comparing two loads of the same web page in a standard live setup, we wanted to explore how the DOM changes in a replay environment using the mahimahi tool [1]. After recording replay data for $\sim$ 100 websites randomly chosen from the Alexa top 1000, we performed the comparison within these replay environments. We also constructed modified versions of the replay environments by manually removing sources of non-determinism from any included JavaScript files.

Results

First, we justify why a smarter DOM comparison is necessary. The following graph compares the differences detected in a naive algorithm vs. our algorithm in comparing a suite of live web pages.

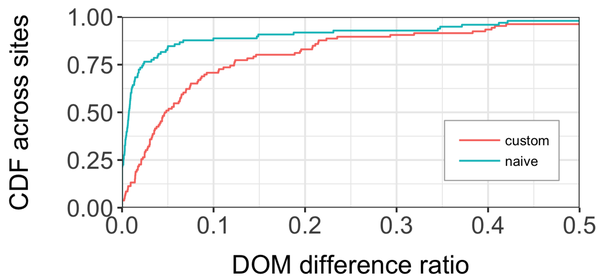

Figure 2: How our comparison performs compared to a naive approach

As seen, our approach identifies many additional differences that the naive approach suppresses to determine if a web page has changed.
Having established the viability of this approach, we now compare the differences between the three classes of comparisons: comparison with live loads, loads within a replay environment, and finally loads within a modified replay environment.

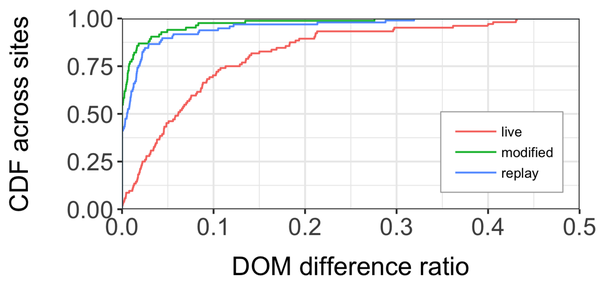

Figure 3: DOM comparison across environments

We can see that the DOM differs significantly more in the live loads than those done in a replay environment.
Finally, we show how the logged network requests change across the different environments for reference.

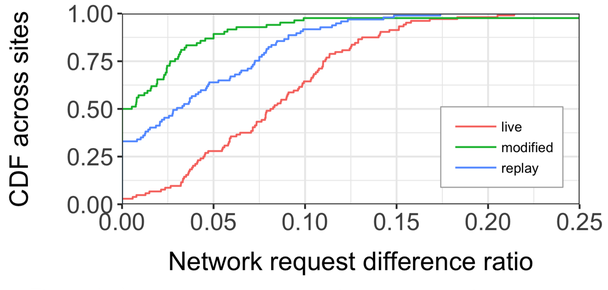

Figure 4: Network request comparison across environments

Discussion

The results show that almost all live web pages sampled change across back-to-back loads with this comparison. Existing work suggested a general trend of only roughly 22% of websites having the same DOM structure on back to back loads [2], but this figure did not account for changes in style or attributes of the DOM elements.
Additionally, while the replay environment helps decrease the detected changes in both the DOM and the network requests sent, there are still sources of non-determinism that cause changes. In the modified replay environments supplied, we see less change, but there are still sources of differences present that are currently being investigated.

Next Steps

Some areas for future exploration include optimization of the existing tool, a better characterization of the types of differences that appear, and extension of the modified replay environment to account for more sources of non-determinism.
This tool can also be applied in web page optimization research, where it could be used to evaluate the correctness of existing work. Rewriting source code is common in this field to gain improvements in speed and memory usage. This project would provide useful information to determine how users' modifications have affected the content of the rewritten web pages.

References

[1] Ravi Netravali, Anirudh Sivaraman, Keith Winstein, Somak Das, Ameesh Goyal, and Hari Balakrishnan.
Mahimahi: Accurate record-and-replay for http.
ACM SIGCOMM Computer Communication Review, 44(4):129âĂŞ130, 2014.

[2] Vaspol Ruamviboonsuk, Ravi Netravali, Muhammed Uluyol, and Harsha V. Madhyastha.
Vroom: Accelerating the mobile web with server-aided dependency resolution.
Proceedings of the Conference of the ACM Special Interest Group on Data Communication - SIGCOMM 17, 2017.

Acknowledgements

I would like to thank Prof. Harsha Madhyastha, Ayush Goel, and the Engineering Honors Program for mentorship and guidance throughout this project.

- Email: jonthoma@umich.edu

ENGINEERING
HONORS PROGRAM
UNIVERSITY OF MICHIGAN