Discovering Refactoring Opportunities in Cascading Style Sheets Artifact Submission

Davood Mazinanian, Nikolaos Tsantalis, and Ali Mesbah

July 11, 2014

1 Information about the Virtual Machine

We have installed Ubuntu 14.04 LTS (x64) on the Virtual Machine, and every software that is necessary for running and evaluating our tool, and the R scripts producing the plots shown in the paper. The VM has been compressed in a tar.gz file (it can be decompressed using 7-Zip in Windows, or the regular archive tools in Linux and Mac). The VM is setup to use 4 GB of memory.

The VM can be downloaded from this link and the download size is 1.9 GB. The user name with admin privileges is `fse14` and the password has also been set to `fse14` (the operating system is set to log-in automatically to this account when you turn it on).

2 Tool overview and configuration

We have developed a tool that analyzes CSS files, finds potential refactoring opportunities, and applies those refactorings that preserve the presentation of the web documents. The tool has been developed in Java, and its source code is publicly available on GitHub[1]. Additionally, we made a runnable JAR from the source code, named as `css-analyser.jar`. In the following subsections we provide all necessary information for running the tool.

2.1 Modes of operation

Our tool runs in three different modes:

1. **Crawl Mode**: The most common scenario is to analyze a web application that is currently running online. Given the main URL of a website, our tool opens that webpage in a web browser, downloads the HTML code and corresponding external CSS files using Crawljax[2], analyzes the

[1] `https://github.com/dmazinanian/css-analyser`
[2] `http://crawljax.com/`

downloaded CSS files, and applies the detected presentation-preserving refactoring opportunities in the CSS files. As we have described in the paper, the refactoring application is done in a multi-step process. In every step, the refactoring causing the maximum size reduction is applied and the resulting CSS file is saved on the hard disk.

2. **Folder Mode**: This scenario is suitable when we have already used our tool to crawl a web application in the past and saved the crawled data on the hard disk (our tool creates a folder hierarchy with the HTML code and the linked external CSS files for every analyzed DOM state). This mode essentially skips the crawling phase, and operates directly on a file path specified by the user that contains the crawled data.

3. **No DOM Mode**: In this mode, the tool analyzes only the CSS files in a given folder without taking into account the HTML code making use of the styles defined in them. This mode is useful when we have CSS files that we just created and we want to find potential refactoring opportunities. However, if the CSS files given as input are already used in some web applications **this mode may find refactorings that are not presentation preserving**.

2.2 Running the tool with sample data

The dataset of the websites that have been analyzed in the evaluation section of the paper are provided on the Desktop inside folder `FSE'14-crawled`. To run the tool on this dataset (in the Folder Mode), double click on the script named `run-precrawled`, and then click on `Run in Terminal`. Alternatively, you can open a Terminal window, and run the command `cd Desktop` followed by the command `./run-precrawled`. For the sake of demonstration, we included only 10 out of the 38 in total examined web applications. The total execution time for this script is **8 minutes** (the host system we used has an Intel Core i7 CPU @ 2.70 GHz, and 8 GB of RAM).

When the processing is finished, you can explore the results in the dataset directory `FSE'14-crawled`. There is a separate folder for each analyzed web application (named after the web application) that further contains a sub-folder named as `css`.

The `css` sub-folder contains a folder named as `CSS-FILE-NAME.analyse` for each analyzed CSS file of the web application, which includes all refactored CSS file versions generated in every step of our approach.

We have also created a bash script that can be used to run the tool in the Crawl Mode, named as `run-crawl`. By default, it is configured to crawl `http://en.wikipedia.org` and store the results of the analysis inside folder `output` on the Desktop. The total execution time for this script is **3 minutes** including the crawling phase.

2.3 Detailed Tool Guide

The `css-analyser.jar` file can be used with proper arguments for custom data. From a terminal window, run this command:

```
java -Xmx4g -jar PATH/TO/css-analyser.jar ARGUMENTS      (1)
```

As we have set the VM's memory to be 4GB, we use `-Xmx4g`. You can increase this value and also VM's memory accordingly. `ARGUMENTS` define the mode of operation for the tool, as well as additional arguments required to run the tool, which will be described in the following subsections.

2.3.1 Crawl Mode

This is probably the most popular feature of the tool. You can analyze a single website by crawling it, or you can provide a list of websites to be analyzed by the tool.

Single Mode In this mode a single website URL is provided to the tool for the analysis. For running the tool in this mode, replace `ARGUMENTS` in Command 1 with:

```
--mode:crawl --url:WEBSITE_URL
--outfolder:"PATH/TO/OUTPUT/FOLDER"
```

Batch Mode In the batch mode, you have to provide a path to a text file which contains a list of webpage URLs to be analyzed. Every URL must be written in a separate line. For every URL, a separate folder is created in the specified output folder. For using the tool in this mode, replace `ARGUMENTS` in Command 1 with:

```
--mode:crawl --urlsfile:"PATH/TO/URLs/FILE"
--outfolder:"PATH/TO/OUTPUT/FOLDER"
```

An example of such a file exists on the given VM's Desktop, named `sites`.

Please note that, existing data in the specified output folder will be overwritten.

2.3.2 Folder Mode

As we explained in Section 2.1, the crawled data that was created using the Crawl Mode can be used later on by the tool for re-analysis. Again, the tool can operate in the single or batch mode.

Single Mode A folder path must be provided to the tool. The results will be written to the same folder and any existing results will be overwritten. In Command 1, replace `ARGUMENTS` with:

```
--mode:folder --infolder:"PATH/TO/INPUT/FOLDER"
```

Batch Mode It is possible to define a text file containing a list of paths to previously crawled data. In this file, every line is a relative or absolute path to the crawled data of a web application. For running the tool in this mode, replace `ARGUMENTS` in Command 1 with:

```
--mode:folder --foldersfile:"PATH/TO/FOLDER/FILE"
```

AN example of such a text file can be found at ∼`/Desktop/FSE'14-crawled/FSE'14/folders`.

2.3.3 No-DOM Mode

As we mentioned, this mode could be used to analyze CSS files without considering the HTML code making use of the styles defined in them. However, this mode may apply refactoring opportunities which are not presentation-preserving.

You can run this mode by replacing `ARGUMENTS` in Command 1 with:

```
--mode:nodom  --infolder:"PATH/TO/FOLDER/CONTAINING/CSS/FILES"
```

Note that you have to provide a folder containing CSS files to the tool. All files in this folder having a `".css"` extension will be analyzed.

2.4 Using tool results

In every mode of operation, when the tool analyzes each CSS file, it creates a folder named `CSS-FILE.analyse` in which all intermediate results are stored. These folders contain the following files:

- `formatted.css`: This is the formatted version of the CSS file. This file is useful for achieving readability when we are dealing with real-world CSS code which might be compressed and thus not human-readable.
- `orderDepenedncies.txt` All the order dependencies between the selectors of the original CSS file being analyzed are listed in this file.
- `fpgrowth.txt`: The results of the FP-Growth algorithm on the CSS files, as described in the paper.
- `fpgrowth-subsumed.txt`: A filtered version of the previous file, in a way that subsets have been removed (please refer to the paper for more information).
- `typeI`, `typeII`, and `typeIII.txt`: Each of these files contains the corresponding duplication type instances in the analyzed CSS file. These types are defined in the paper.
- `refactored[k].css`: These files (where k is an integer greater than 0) are the refactored versions of the original analyzed CSS file. Each of them is created by applying exactly one refactoring opportunity.

- `dependency-differences[k].txt`: These files are created if, after applying a refactoring at step k, there exist differences between the dependencies in the oiginal CSS file and `refactored[k].css`. They contain the type of dependency violations.

- `refactored-reordered[k].css`: If, at any step, a reordering of selectors is needed (because a dependency is violated), the tool will create a file with this name, which contains the same CSS file as in `refactored[k].css`, but the selectors are re-ordered to satisfy all dependencies. All these files make the tracing of changes in the original CSS file simpler.

- `clone types.txt`: This file contains the statistics about the clone types and the clone sets, which are used in the evaluation section of the paper.

When the analysis process is finished, one may obtain the final refactored CSS files by getting the file `refactored[k].css` with the highest k. If there exist some `refactored-differences[m].css` files, select either the `refactored.css` or the `refactored-differences.css` that has the highest index.

Also, in the folder that contains multiple CSS files, a file named `analytics.txt` is created. This is a pipe-separated ("|") file which contains the global statistics about all the analyzed CSS files. This file can be fed into any statistical software, like R. The columns in this file and their descriptions are shown in Table 1.

Table 1: Columns in the `analytics.txt`.

Column	**Description**
file_name	Name of the analyzed CSS file
size	Size of the analyzed CSS file (KB)
sloc	Source lines of the code, after formatting CSS file
num_selectors	# selectors in the original CSS file
num_base_sel	# base selectors in the original CSS file
num_grouped_sel	# groped selectors in the original CSS file
num_decs	Total # declarations in the original CSS file
IOnly, IIOnly and IIIOnly	# clone sets containing only a specific clone type instances
I_II, II_III, I_III, I_II_III	# clone sets containing all the specified clone type instances
number_of_duplicated_declarations	# duplicated declarations (of any type)
selectors_with_duplicated_declaration	# selectors containing at least one duplicated declaration
longest_dup	Size of the largest set of duplicated declarations (clone set)
max_sup_longest_dup	# selectors which share the previous clone set
clone_sets	# distinct clone sets
refactoring_opportunities	# all possible refactoring opportunities
applied_refactorings_count	# applied refactoring opportunities
number_of_positive_refactorings	# applicable refactoring opportunities with positive effect
size_after	Size of the final CSS file (KB)
number_of_order_dependencies	# order dependencies in the original CSS file
refactoring_opportunities _excluded_subsumed	# refactoring opportunities excluding subsumed ones
positive_excluded _subsumed	# refactoring opportunities excluding subsumed ones which have positive effect

3 Using R scripts

We have also provided some R scripts that were used in the evaluation section of the paper. They are located on the Desktop of VM, in `R-scripts` folder.

To run these scripts, you can use R Studio[3], an IDE for R, which has already been installed on the VM. You can open it by clicking on its icon in the Ubuntu Launcher, or running `rstudio` in the Terminal. For executing the provided scripts, from R Studio's File menu, select open (alternatively press `CTRL + O`). Point to the scripts folder, and open the file named `load.R`. It is crucial to run this script before running other ones, as it loads the data and necessary packages to the R environment.

For your convenience, we have created a pipe-separated file by appending all the `analytics.txt` files for the analyzed websites (located on the Desktop and named `FSE'14.csv`). After opening `load.R`, press `CTRL + ALT + R` to run it. From now on, you can open and run any of the scripts in the `R-scripts` folders similarly. Scripts whose name starts with `plot` are the ones we used for creating the plots in the evaluation section of the paper. The R script `statistical-model.R` makes the statistical model, as described in the paper, gives a summary of it, and displays its associated plots.

[3] `http://www.rstudio.com/`