OLD DOMINION UNIVERSITY

MASTER'S PROJECT REPORT

Deploying the Memento Damage Service: A Comprehensive Tool for Measuring and Analyzing Damage on Web Archives

---

Author:
Erika SIREGAR

Advisors:
Dr. Michele WEIGLE
Dr. Michael NELSON

A report submitted in fulfillment of the requirements for the degree of Master

in

Computer Science

2017

Declaration of Authorship

I, Erika SIREGAR, declare that this project report titled, "Deploying the Memento Damage Service: A Comprehensive Tool for Measuring and Analyzing Damage on Web Archives" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the project report is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

OLD DOMINION UNIVERSITY

Abstract

Computer Science

Master

Deploying the Memento Damage Service: A Comprehensive Tool for Measuring and Analyzing Damage on Web Archives

by Erika SIREGAR

Measuring memento quality has emerged as something that is important in the web-archiving related field. Brunelle has devised an algorithm of how to calculate the damage that happens on a memento by analyzing the impact of missing resource towards the appearance of a web page. In order for this tool could be used widely by a larger community, memento-damage should be 'packaged' as a service that is ready to use, perform the calculation in a short time, and distributable. This report will present the work that has been done to provide several ways of using memento-damage, which are available in 4 options: web service, REST API, library, and docker. The testing results show that memento-damage can be used to process a large number of mementos in a shorter time compared to that of the previous version. This tool also comes with a friendly user interface and CLI command to help improve the user experience.

Acknowledgements

I would like to thank Dr. Michele Weigle and Dr. Michael Nelson for advising and guiding me through this project. I would also like to thank Dr. Justin Brunelle for the knowledge transfer and helping me to understand the concept of damage calculation for web archives. Finally, I thank all the members of Web Science and Digital Library (WS-DL) lab for supporting and giving me feedback on this project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The rapid development of web technology and the increasing number of web documents created every day has made people more aware of the importance of archiving the web. This web archiving activity produces an output known as the memento. According to Van de Sompel et al. [15], memento is an archival record of a resource for a web page at time t. Memento is also described as a resource that encapsulates a prior state of the Original Resource [14]. These definitions imply that memento is something that we could use to go back in time and see how a page looked like in the past. In this report, we will use the terms URI-R to express a live web page and URI-M to indicate a memento.

As a representation of a web page in the past, memento consists of the same resources as that of the live web page with an addition of a toolbar from the web archival service to increase user experience. These resources include images, CSS, multimedias, javascript, iframes, and text. Figure 1.1 shows an example of a memento. Memento can be found on various web archiving service such as **archive.org**, **UK Web Archive**, and **archive.is**.

Memento plays an important role in web-archiving-related research since it provides a straightforward way to access the Web of the past as it is to access the current Web. Unfortunately, web archives do not always capture every resource on every page that they attempt to archive. This results in archived pages missing a portion of their embedded resources, which is considered as having a damage on the page. This damage can diminish user experience and satisfaction as they cannot get a complete information from the broken web page. But the definition of the *damage* itself is subject to user's perspective. It raised a question about how to objectively measured the damage of a memento that can correctly emulate user's (human) perception. Figure 1.2 illustrates the difference between a good page and a damage page, which is taken from **ODU's Computer Science website**.

1.2 Background

The basic idea of measuring memento damage is finding the ratio between the missing resources and all resources that supposed to be there. This idea is represented using the formula in

Equation 1.2:

$$total\_damage = \frac{missing\_resources}{total\_resources} \tag{1.1}$$

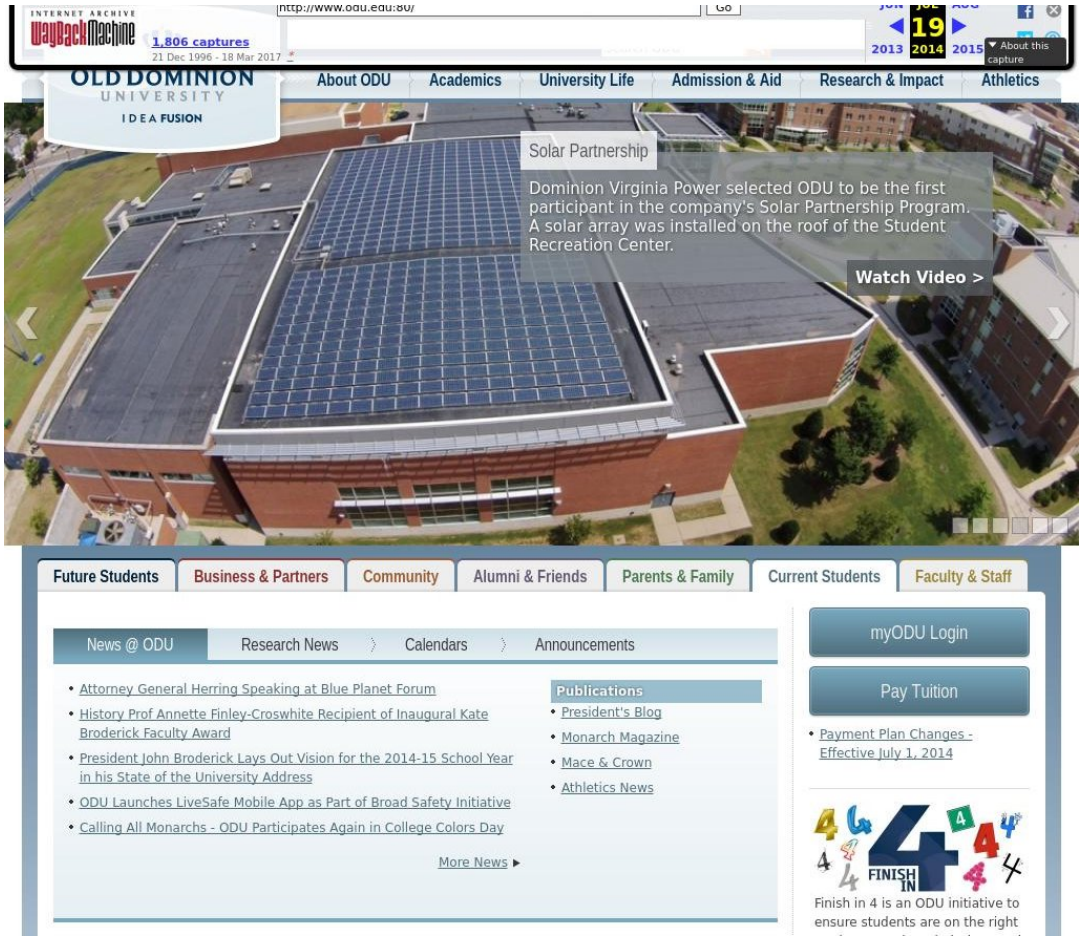


FIGURE 1.1: Example of a memento

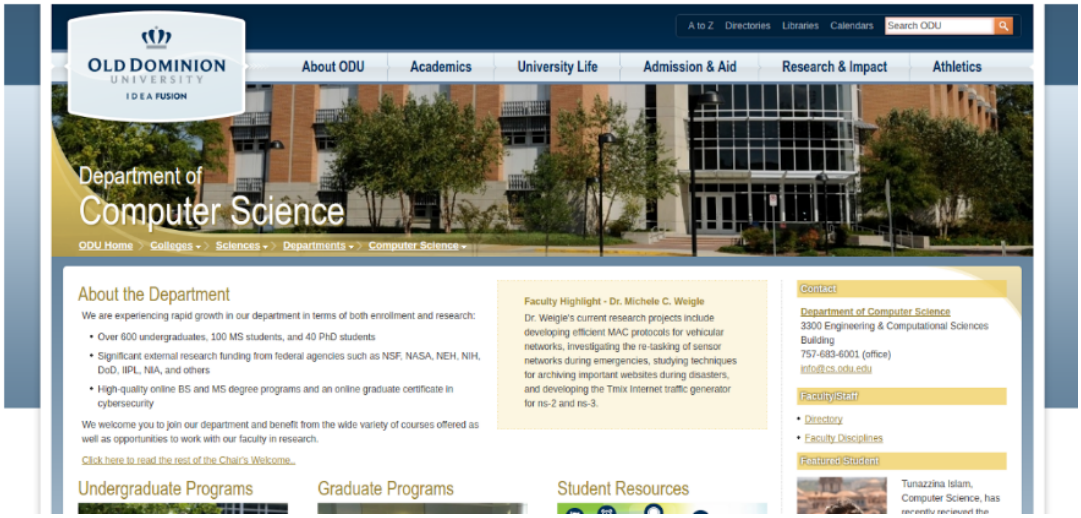
A web page could lose only 1 (one) image but caused a very high damage on the web page due to its large size and central position. On the other hand, a page might lose many of its resources but only have low damage score because those missing resources are less likely to be noticed by the user (unimportant) (Figure 1.3). In the cases like these, counting the number of missing resources do not always represent the real damage.

Brunelle [4] has devised a prototype for measuring the impact of missing embedded resources (the damage) on a web page. Basically, the idea is described using formula in Equation 1.2:

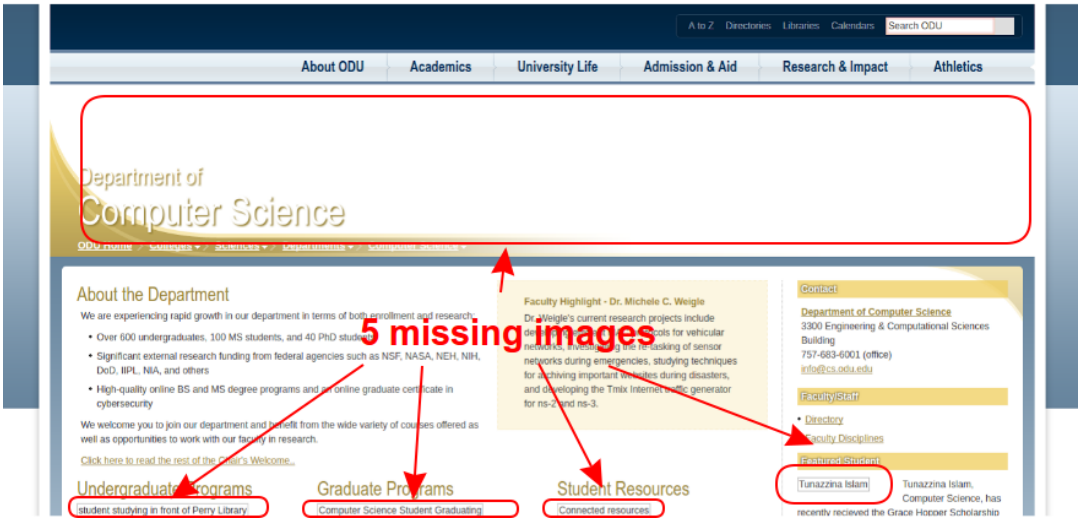
$$total\_damage = \frac{missing\_resources}{total\_resources} \tag{1.2}$$

In this prototype, all embedded resources owned by a web page are listed. Each resource is measured to obtain its actual and potential contributions to the total damage. *Total damage* is defined as a ratio between *actual damage* and *potential damage*. Actual damage represents the real damage that happens caused by a missing resource (`status_code` is 403 or 404), whereas potential damage denotes the damage that could happen if a resource was missing. This damage value ranges from 0 to 1, where a damage of 1 means the web page lost all of its embedded resources.

There are 5 (five) types of resources that are taken into account by this algorithm.



(A) ODU's website with no resource is missing



(B) ODU's website with 5 images are missing

FIGURE 1.2: Good page vs *damage* page

Each type of resources has different way of calculating the damage contribution (importance), which are:

1. Image
   Image importance is determined by image size and image position on the viewport. The bigger size the image has, the higher importance value it has. Furthermore, if the image crosses the viewport's central point, this image will also gain a *location importance* score.

2. *Cascading Style Sheets* (CSS)
   The importance of CSS is composed of *tag importance* and *ratio importance*, both are given a weight of 0.5, respectively. *Tag importance* represents the number of DOM elements that use certain *css rules*, while *ratio importance* reflects the ratio of *whiteguys* and *non-whiteguys* in a web page. *Whiteguys* is representation of pixels having same color with the background color. The page is divided into 3 (three) parts: left, center, and right. If more than $\frac{2}{3}$ of the whiteguys converge

on the right part of the page, the ratio importance will be equal to the ratio weight.

3. Multimedia
   Multimedia is treated in the same way as how we calculate the damage on images.

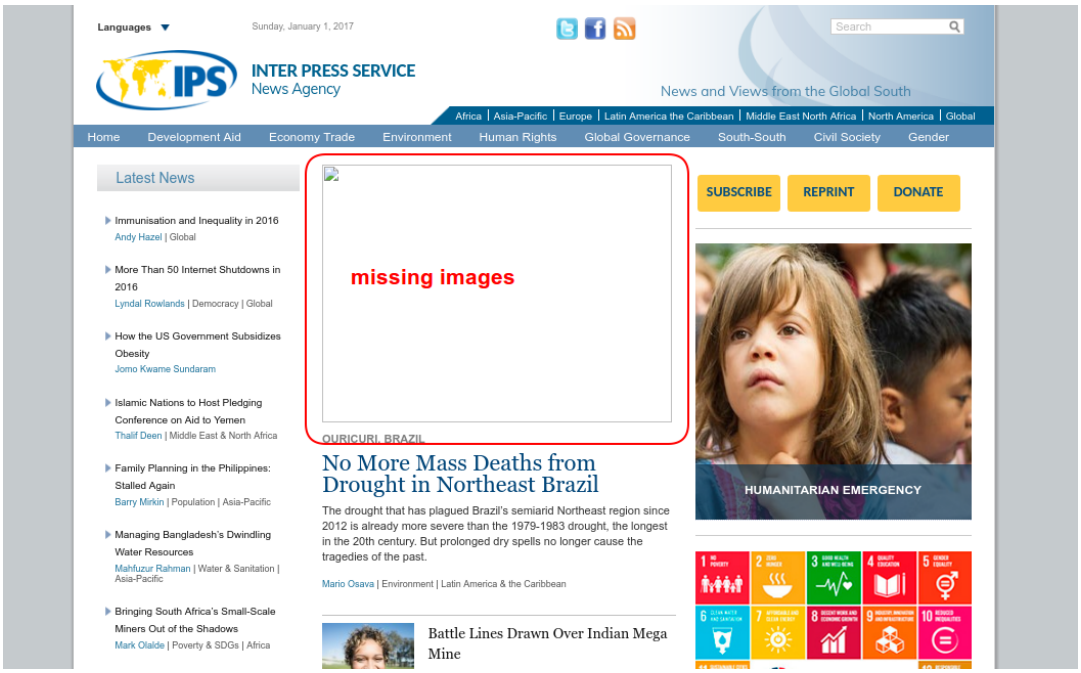
4. Javascript
   A missing javascript file has a damage value of 1, while the non-missing javascript file has a damage value of 0.

5. Text
   Text importance is computed by calculating the number of words in a web page and divided by 1000.

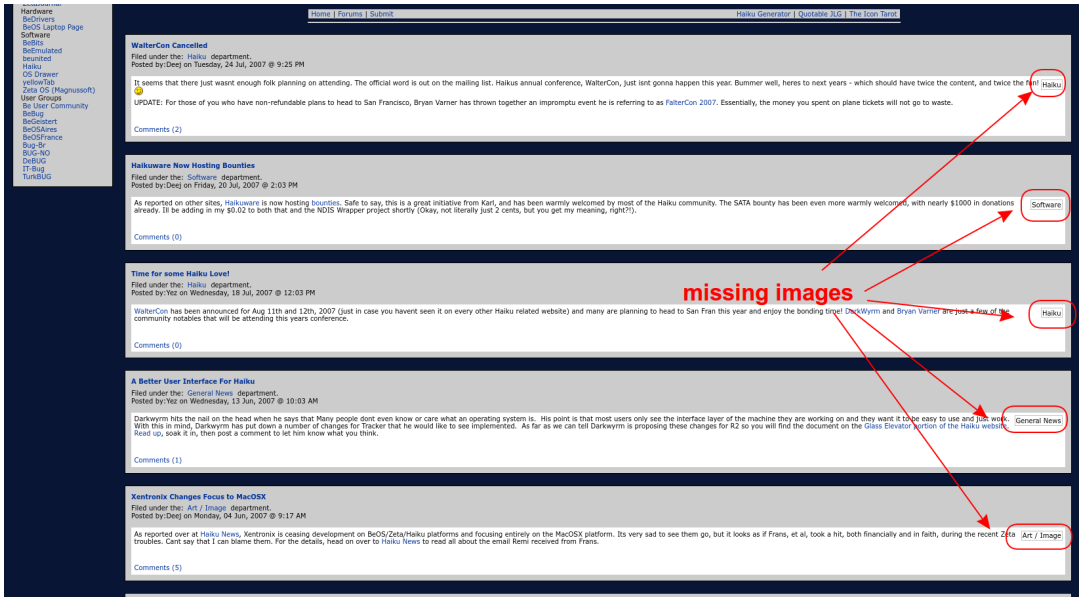
Although the testing result shows that this prototype successfully emulate human perception on a web page damage, it is not ready to be used by user, yet. There are several issues that need to be fixed to transform this prototype into a ready-to-use tool:

1. The damage calculation took a long time for 1 URI-M (1.5 hours for URI-M: https://web-beta.archive.org/web/20170131124832/odu.edu.
2. The installation process involves various complicated dependencies.
3. We found some errors in coding.
4. There is no detail that can give user an insight about what happens and what process is running behind the resulting damage value.
5. There is no error information which can cause an unaware user ended up waiting in infinity for a result that will never showed up.
6. It is not demonstrable and not shareable.
7. It is not handling redirection.

Therefore, we need to 'package' and deploy this prototype into a ready-to-use and distributable tool that can be used widely in various communities.



(A) A screenshot of http://web.archive.org/web/20170101170526/http://www.ipsnews.net/ that lost 1 central image, accessed on April 28, 2017



(B) A screenshot of http://web.archive.org/web/20071007215803/http://www.begroovy.com:80/ that lost 5 non-central images, accessed on April 28, 2017

FIGURE 1.3: The number of missing resources do not always represent the real damage

Chapter 2

Design

The Memento-Damage tool that we built comprises two main processes: the crawling (downloading the URI-M's resources) and computing the damage based on those downloaded resources. Figure 2.1 shows the flowchart of process overview in Memento-Damage.

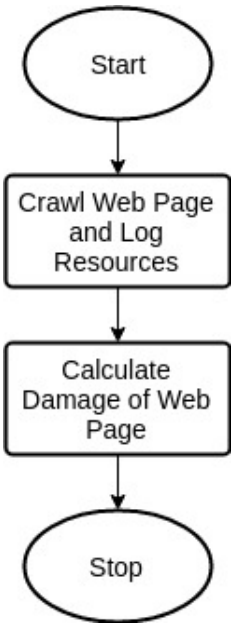


FIGURE 2.1: Memento Damage Process

2.1 Downloading the Resources of a URI-M (Crawling)

Crawling the resources does not mean that we phisically download the resources, but only downloading their network log. Figure 2.2 shows the flowchart of the crawling process in Memento-Damage. Downloading/crawling the URI-M resources begins with the user inputting their intended URI-M into the Memento-Damage. The tool will check the `content-type` of the URI-M and will only continue processing it if the `content-type` is `text/html`.

The tool will download all resources' network log of the input URI-M using `PhantomJS`, a headless WebKit scriptable with a JavaScript API [9]. `PhantomJS` is called from the inside of the python script `crawling` using *Thread* [6] and *popen* [7]. All non-ASCII characters will be decoded before these logs are written into output

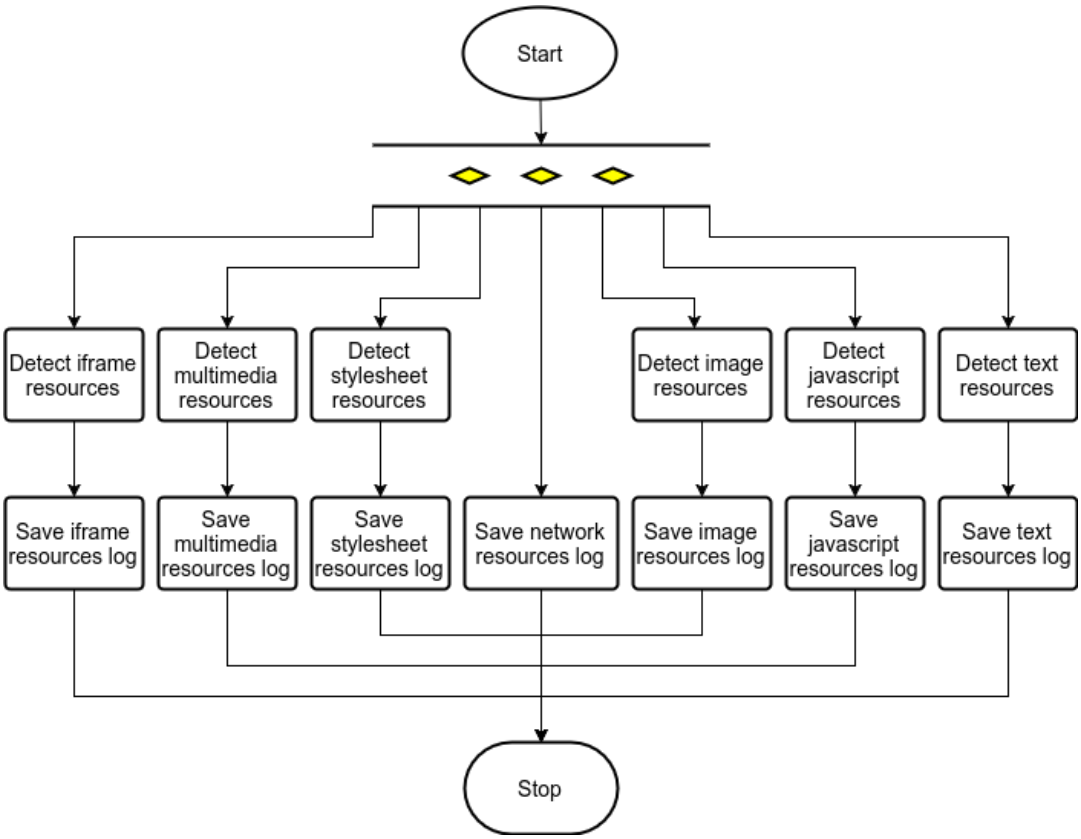


FIGURE 2.2: Overview flowchart of the crawling process

files as described in Figure 2.3 and Listing 2.1. The memento-damage tool creates various network logs according to the resources type:

1. **`image.log`**, contains the properties of image resources such as `top`, `left`, `width`, and `height`.
2. **`css.log`**, contains the properties of css resources such as `rules-tag` and the number of DOM element using certain rules-tag in the css files.
3. **`video.log`**, contains the properties of multimedia resources.
4. **`js.log`**, contains the properties of the javascript resources.
5. **`text.log`**, contains the properties of the text in the web page.
6. **`iframe.log`**, contains the properties of the iframe resources.

Moreover, this tool also generates **`network.log`** that contains the network log of all resources in a URI-M, **`app.log`** which contains the process log, **`source.html`** which is the html file of the URI-M, and **`screenshot.png`** which is the screenshot of the web page.

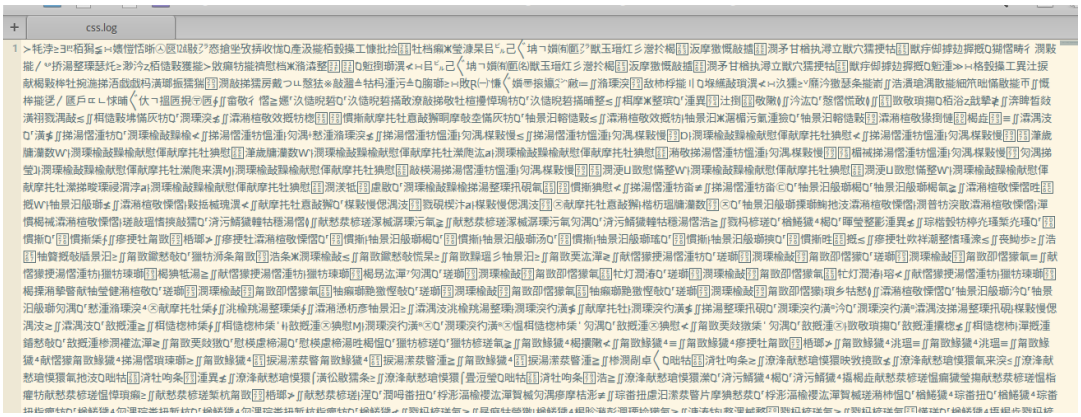


FIGURE 2.3: Example of Non-ASCII characters in the log

```
fs.write(resourceImageFile, unescape(encodeURIComponent(
    networkImagesValues.join('\n'))), "w");
fs.write(resourceVideoFile, unescape(encodeURIComponent(
    networkVideosValues.join('\n'))), "w");
fs.write(resourceCssFile, unescape(encodeURIComponent(networkCssValues.
    join('\n'))), "wb");
```

LISTING 2.1: Handling the non-ASCII characters appear in the log

2.2 The Damage Calculation Process

This subchapter will discuss the various process involved in measureing the damage on every resources.

2.2.1 Chasing Down Redirection

After crawling the resources, Memento-Damage will start calculating the damage value. In order to do this, Memento-Damage will read the log files that are previously generated. Memento-Damage will first read the **`network.log`** and examine the `status_code` of each resource. There are 3 conditions that could happen:

1. The resource is accessible (`status_code` = 200).
   For this kind of resources, there is no damage happens but we still compute the importance value of these resources to get the potential damage.

2. The resource is missing or not found (`status_code` = 403 or 404)
   For this kind of resources, the damage happens and the Memento-Damage will compute the effect of the missing resources on the web page (actual damage).

3. The resource is redirected (`status_code` = 301, 302, 307, or 308)
   For this kind of resources, we need to follow the redirection until reaching the final URI where the resource is either accessible (as mentioned in item 1) or missing (as mentioned in item 2).

Algorithm 1 SolvingRedirection

Input: $uri, network.log$
Output: $redirect\_urls$

```
t_uri = uri
while TRUE do
    append t_uri and its status_code to redirect_urls
    if status_code in [301, 302, 307, 308] then
        t_uri = header location of t_uri in network.log
    else
        BREAK
    end if
end while
```

When a URI is redirected, we will chase down the final URI by following the URI in the header `location` as depicted in Figure 2.5. For solving redirection, Memento-Damage works based on the rules in algorithm 1 and flowchart in Figure 2.4.

From the process in algorithm 1, we will obtain the complete redirection path for each redirected URI as well as its final `status_code`. Listing 2.2 shows an example of a complete redirection path for the URI-M: http://web.archive.org/web/20151002223158/http://www.slideshare.net/IanMilligan1. The redirect urls is formatted as a dictionary with 3 (three) keys, which are url, status_code, and content-type.

```
# original URI:
(http://web.archive.org/web/20151002223158/http://www.slideshare.net/
    IanMilligan1)

# the complete redirection path of the URI:
(u'http://web.archive.org/web/20151002223158if_/http://www.slideshare.net/
    ianmilligan1/slideshelf', 302, u'text/plain; charset=utf-8'),
(u'http://web.archive.org/web/20170331132121if_/http://www.slideshare.net/
    ianmilligan1/slideshelf', 302, u'text/plain; charset=utf-8'),
(u'http://web.archive.org/web/20170331132122if_/https://www.slideshare.net
    /ianmilligan1/slideshelf', 200, u'text/html; charset=utf-8')
```

LISTING 2.2: Example of a compete redirection path

2.2.2 Removing Blacklisted URIs

In a memento, there are several resources that actually not the part of the original web page such as the image and CSS that construct the navigation toolbar in the memento from the Wayback Machine [1]. These resources have to be removed to prevent the damage value from being overcalculated. Eliminating these resources are simply conducted by excluding all resources with any of these following URIs:

1. https://analytics.archive.org/
2. http://analytics.archive.org/
3. https://web.archive.org/static
4. http://web.archive.org/static

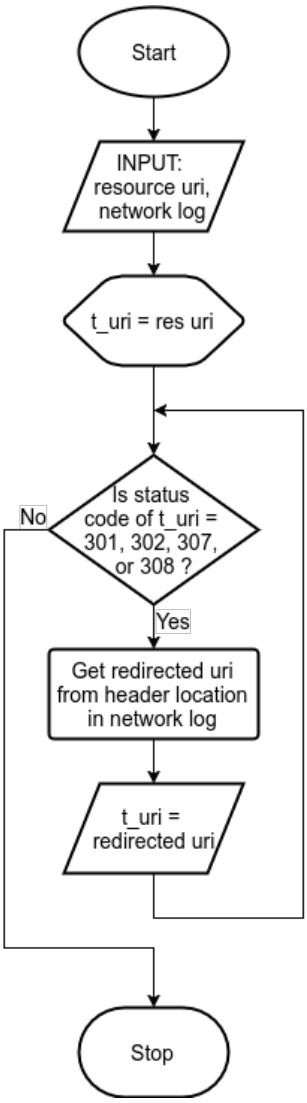


FIGURE 2.4: The redirection flowchart

Several other resources also have to be removed, which are the internal CSS (URI containing `[INTERNAL]`), base-64 image (URI containing the word `data:`), and resource's URI whose `headers Link` == <http://mementoweb.org/terms/donotnegotiate>; rel="type".

2.2.3 Removing hidden elements

Hidden elements are the elements that cannot be seen by the user. They have no effect towards the damages, hence should be excluded from the calculation. For this purpose, we have to detect the element's visibility using JQuery [8] as described in [16] and remove all resources whose visibility is *False*. Figure 2.6 presents the flowchart of removing hidden elements.

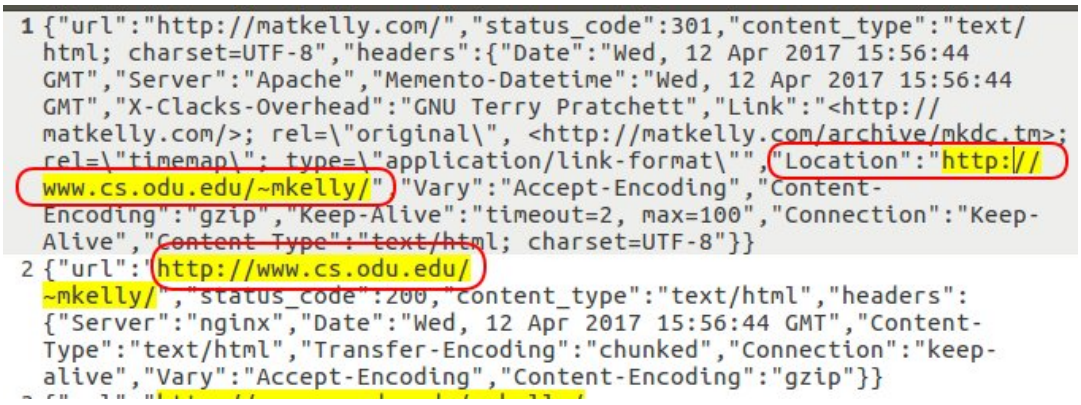


FIGURE 2.5: chase 302

2.2.4 Solving Overlapping Resources

Some URIs have resources that are overlapped with each other. Figure 2.7 shows an example of overlapping frames on URI-M: `http://web.archive.org/web/20151002223158/http://www.slideshare.net/IanMilligan1` (accessed on April 1st, 2017). These overlapped resources need to be treated differently to prevent the damage value from being double counted.

To solve this problem, we created a concept of *rectangle* → ('Rectangle', 'xmin ymin xmax ymax'). We view the overlapping resources as rectangles and calculate the size of the intersection area. The size of one of the overlapped resource will be reduced by the intersection size, while the other overlapped resource will get the whole full size. Figure 2.8 and Figure 2.9 give the illustration of the rectangle concept.

2.2.5 Calculating Damage on Every Resource

After the calculation processes finished, the Memento-Damage tool will create a file named **`result.json`**, which contains the results of the damage calculation. Figure 2.10 depicts the flowchart of the damage analysis process.

2.3 The Interface Design

After designing the processes, the next step to do is designing how the user will use the Memento-Damage tool. For this purpose, we design 2 types of service to enable user utilizing the Memento-Damage tool: the online service and the local service. The local service is created in a *Command-Line Interface* (CLI) format. The design includes the input and output design. There are four mandatory items that must be presented in the output:

1. The total damage value
2. The detail of every resource that constructs the URI-M.
3. Information about time spent for doing the calculation.
4. Process log and error messages if a problem happened.

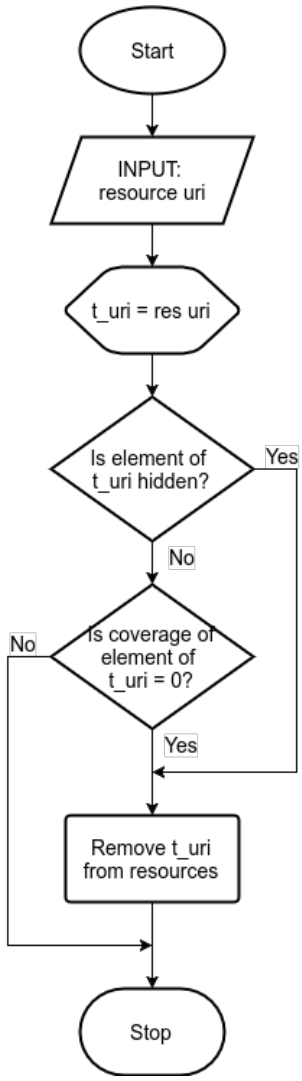


FIGURE 2.6: Removing hidden elements

5. Information about to which URI the damage value belong (for CLI version only).

2.3.1 Online Service

The online service relies on the service provides by Old Dominion University (ODU) server. This service runs on a resource-limited server, thus not recommended for calculating damages on a large number of requests.

Website

Website interface provides the easiest way to use the Memento-Damage tool. User can obtain the damage value as simple as open the website, input the URI-M, and hit the *check* button. The designed website has 2 pages, *home* and *result*. *Home* page provides some examples of URI-M, a *textbox* for inputting the URI-M, and the resulting damage value. *Result* page provides the details of every resource that contributes to

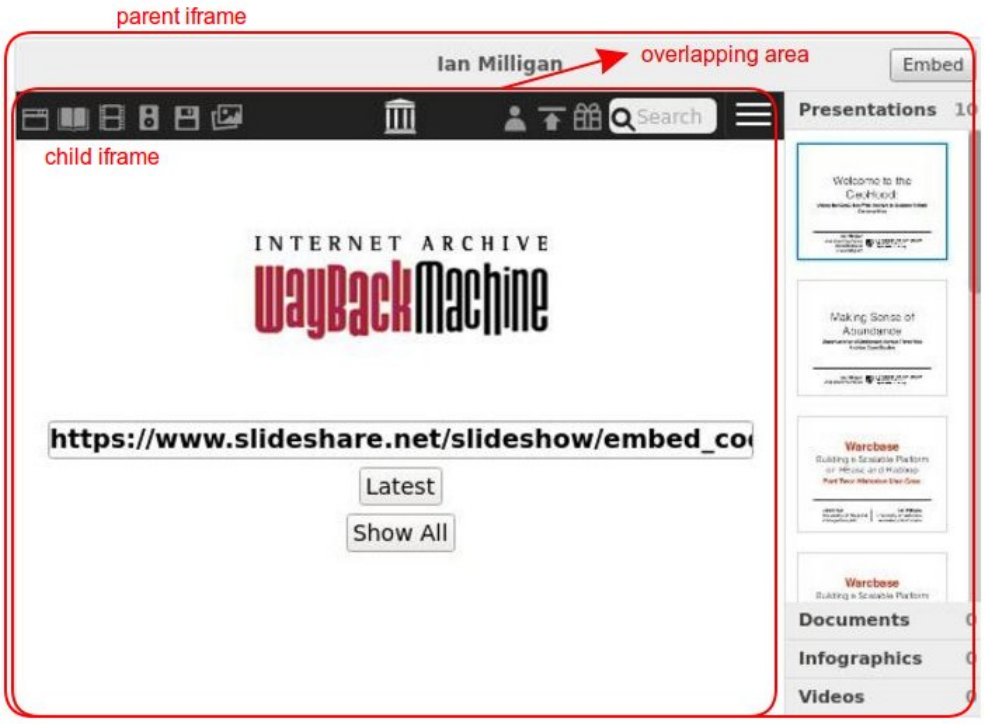


FIGURE 2.7: Overlapping iframes found on `http://web.archive.org/web/20151002223158/http://www.slideshare.net/IanMilligan1`

the total damage value. The website also provides *FAQ* and *Help* pages to help the users understand and get some insights about how to use this tool.

REST API

The REST API option is provided to support machine to machine communication. User can use HTTP Client and send request to calculate the damage of the intended URI. Using the REST API, user can send the request in a batch format.

2.3.2 Local Service

The local service is designed to handle damage calculation on a large number of requests. Using the local service, user can reproduce the Memento-Damage on their own machines and create as many Memento-Damage instances as they need. The local service is made availabe as a Docker's image and Python's library.

Docker

Docker is a software container platform, in which everything required to make a piece of software run is packaged into isolated containers. This makes for efficient, lightweight, self-contained systems and guarantees that software will always run the same, regardless of where it's deployed [5]. In order to be able to use Docker, users need to be familiar with some basic Docker commands such as `pull`, `run`,

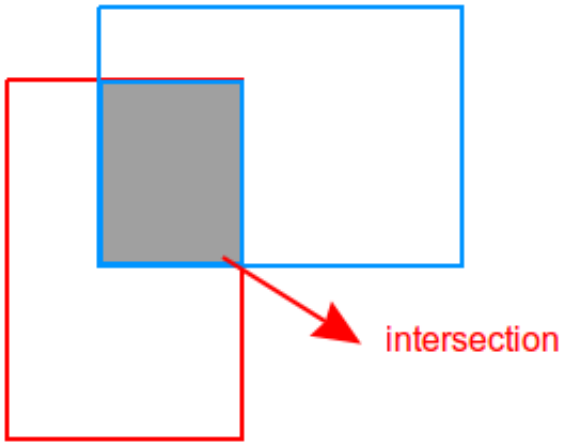


FIGURE 2.8: Intersection Illustration

```
def rectangle_intersection_area(a, b):
    dx = min(a.xmax, b.xmax) - max(a.xmin, b.xmin)
    dy = min(a.ymax, b.ymax) - max(a.ymin, b.ymin)
    if (dx >= 0) and (dy >= 0):
        return dx * dy
```

FIGURE 2.9: Rectangle Intersection

`start`, and `exec`. The complete command list are available on `https://docs.docker.com/edge/engine/reference/commandline/docker/`.

Docker is a way to make the Memento-Damage tool distributable. Using docker, user can run the Memento-Damage tool in any machine without have to install the prerequiste softwares (`Python` and `PhantomJs`). Every single instance of Memento-Damage Docker image already contains the functionality of both the online service as well as the local service (CLI).

Library

The library option is the simplified version of the Memento-Damage Docker image. It is suitable for users who already have `Python` and `PhantomJs` installed on their machines and do not quite familiar with Docker. In terms of size, the library has a smaller size than that of Docker image which has a size of 1 GB. It means that the downloading process will be faster. Similar with the Docker version, the library version also provides both the online and local service.

2.4 Logging Level

We use the logging levels provided by Python logging library [1]. Table 2.1 shows 6 (six) logging levels provided in Python. The logging functions are named after the

[1] https://docs.python.org/2/library/logging.html

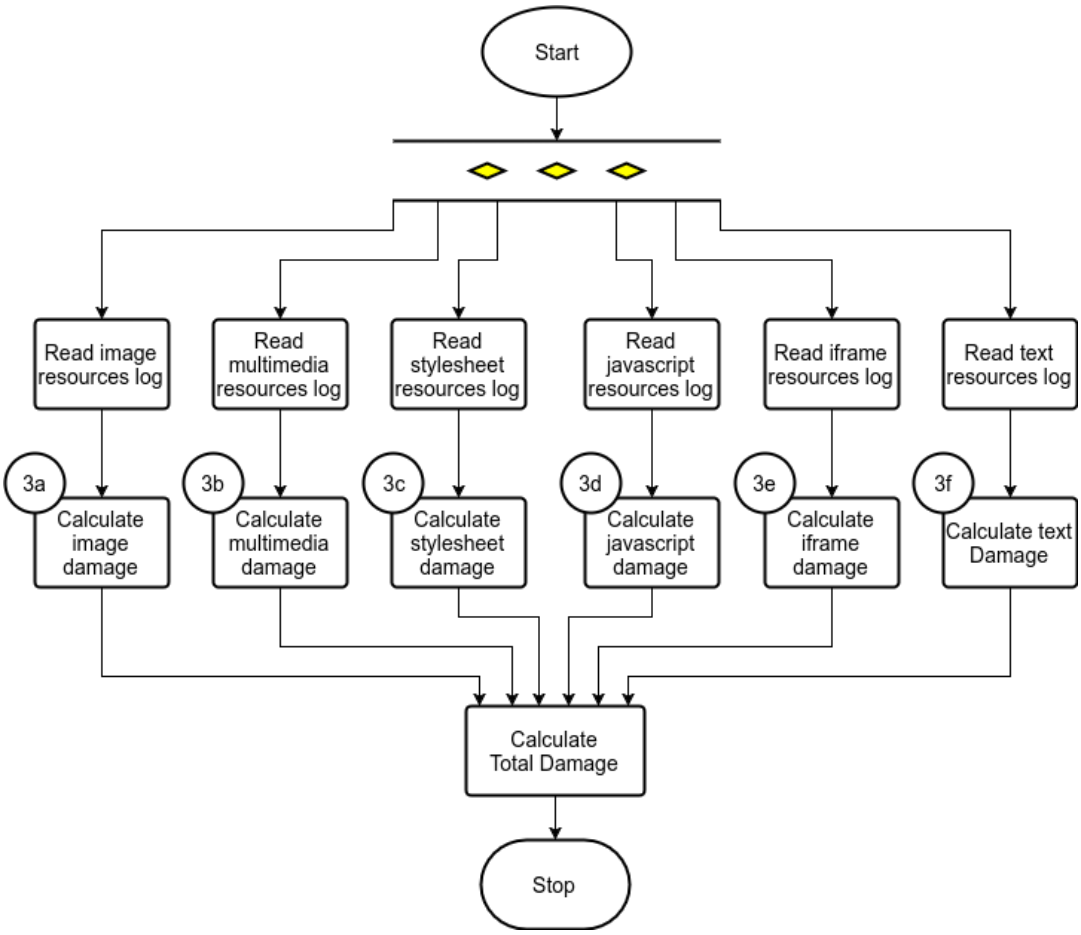


FIGURE 2.10: Overview flowchart of the damage analysis process

level or severity of the events they are used to track [12]. In this project, we only focus on the 3 (three) logging levels, which are `DEBUG`, `INFO`, and `ERROR`. `DEBUG` provides detail information of the process that is currently running, `INFO` gives a confirmation that things are working as expected, and `ERROR` provides an information if a problem happens. Figure 2.11 shows an example of `DEBUG` and `INFO` in Memento-Damage.

No	Level	Numeric value
1	CRITICAL	50
2	ERROR	40
3	WARNING	30
4	INFO	20
5	DEBUG	10
6	NOTSET	0

TABLE 2.1: Logging Levels

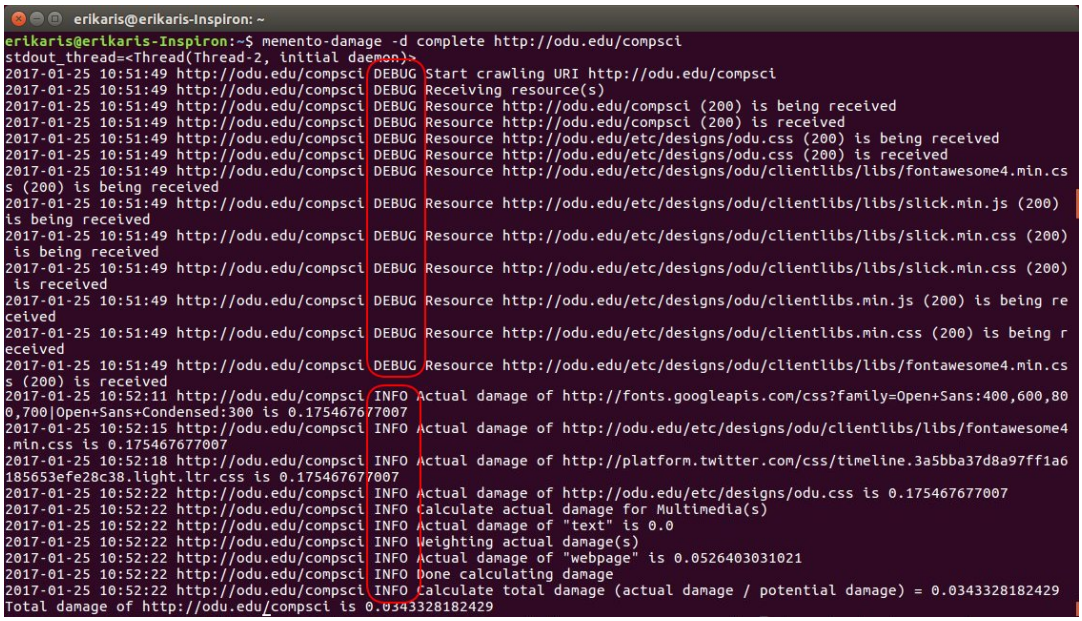


FIGURE 2.11: Example of DEBUG and INFO in Memento-Damage

2.5 Analyzing and Calculating Damage on Every Resource

2.5.1 Image, Multimedia, and Iframe

Image, multimedia, and iframe resources are handled in the same way. Memento-Damage gets the list of image, multimedia, and iframe resources from the log files and obtains their properties such as *width*, *height*, *top*, and *left*. Then, Memento-Damage will find the matched URI from network resources downloaded by `PhantomJs` to obtain its `status_code`. Based on these properties, the size and location importance will be computed as illustrated in Figure 2.12 and Figure 2.14.

Image

Specifically for image resources, Memento-Damage excludes `base-64` images from the damage calculation since it is not an embedded resources and its `status_code` is always 200. Thus, not removing this images will cause a 'null' damage value as described in Figure 2.13.

Iframe

Dealing with iframe is quite tricky. Although it is treated like an image, it still needs some customization. First, by default, crawling process cannot access content inside of iframe. This problem becomes more complicated when this iframe contains another iframe. Therefore, we need to create a script that has the ability to switch from main frame to iframe. Second, the location properties of the resources inside of iframe are calculated relative to the iframe position, not to the main frame position. It could lead to a wrong damage calculation. Third, most of the iframe implementation pertains to a cross-domain problem and browser will not allow accessing resources from iframes of different domains [2]. Hence we need to customize the script accordingly to overcome all of these problems.

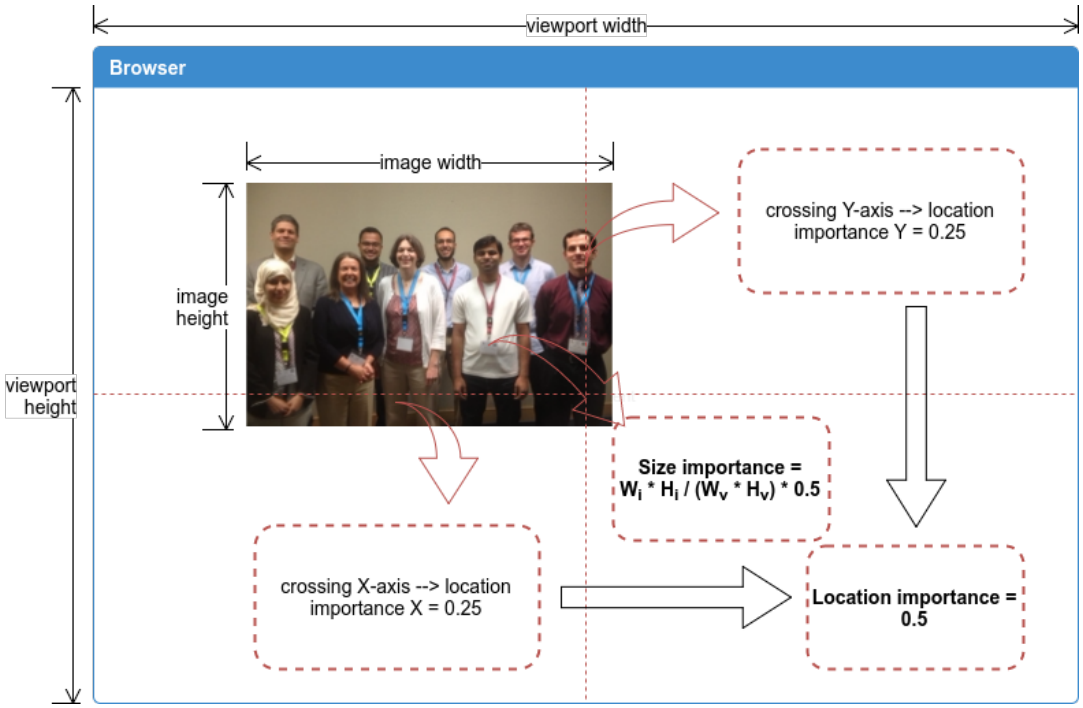
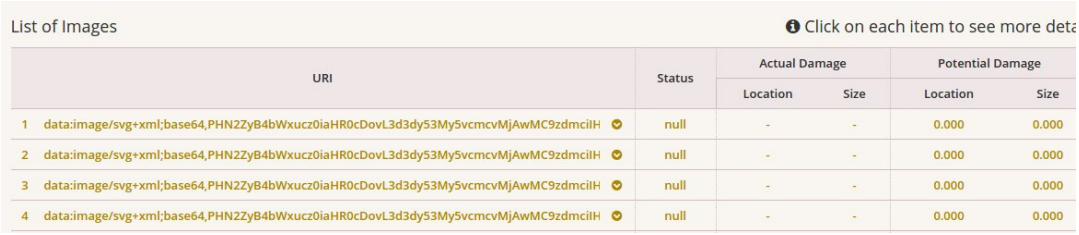


FIGURE 2.12: Size and location importance for image resources


List of Images — Click on each item to see more deta

	URI	Status	Actual Damage		Potential Damage	
			Location	Size	Location	Size
1		null	-	-	0.000	0.000
2		null	-	-	0.000	0.000
3		null	-	-	0.000	0.000
4		null	-	-	0.000	0.000

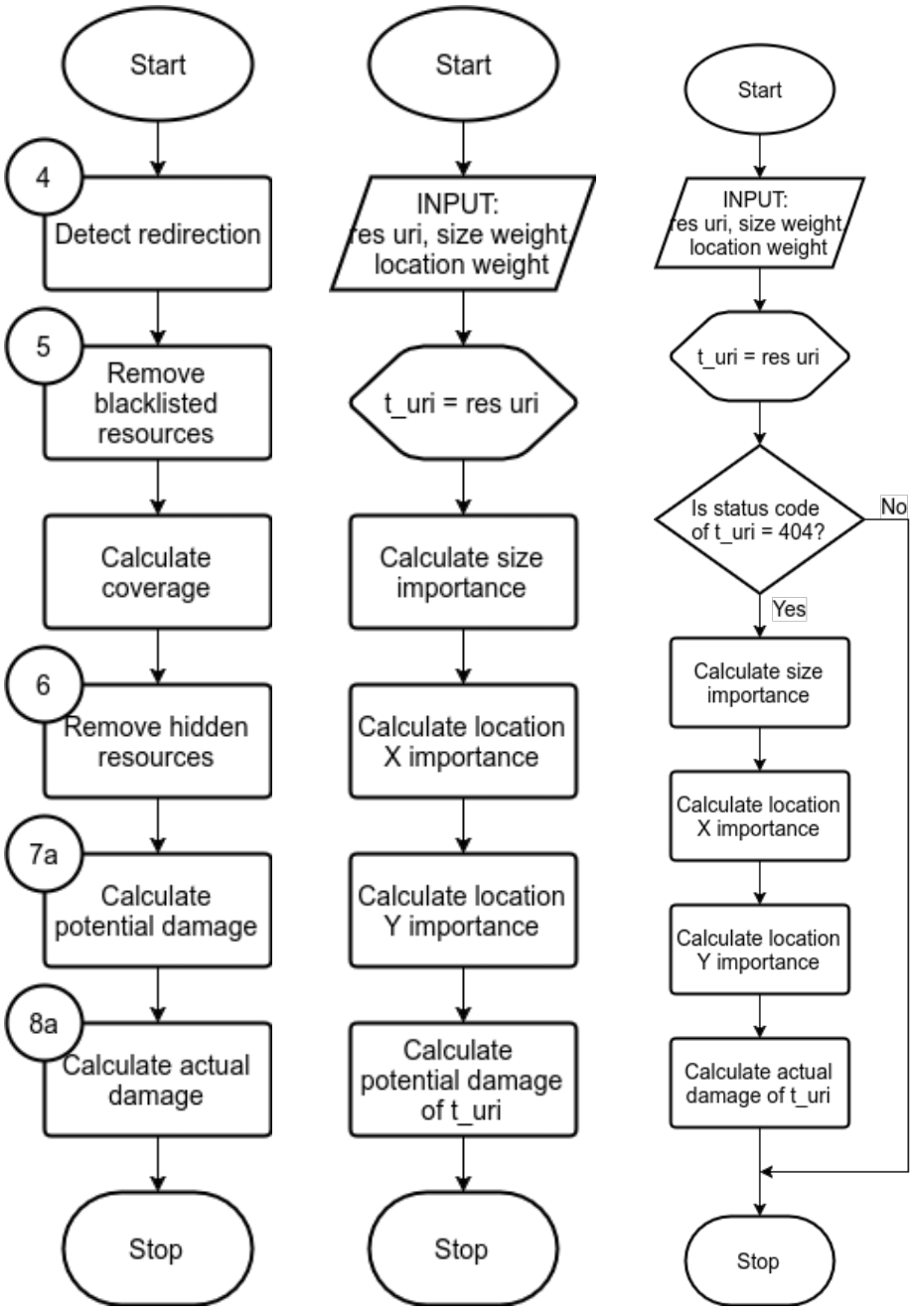
FIGURE 2.13: Base-64 image caused a [null] value

2.5.2 CSS

The CSS damage composed of tag importance and ratio importance. An improvement that we did for tag ratio calculation is modifying the script to handle complex selector text such as: *btn-outline-inverse btn-lg*. Internal CSSes are not deemed as embedded resources, hence they have to be excluded from the damage analysis. Figure 2.15 presents the flowchart of CSS damage calculation.

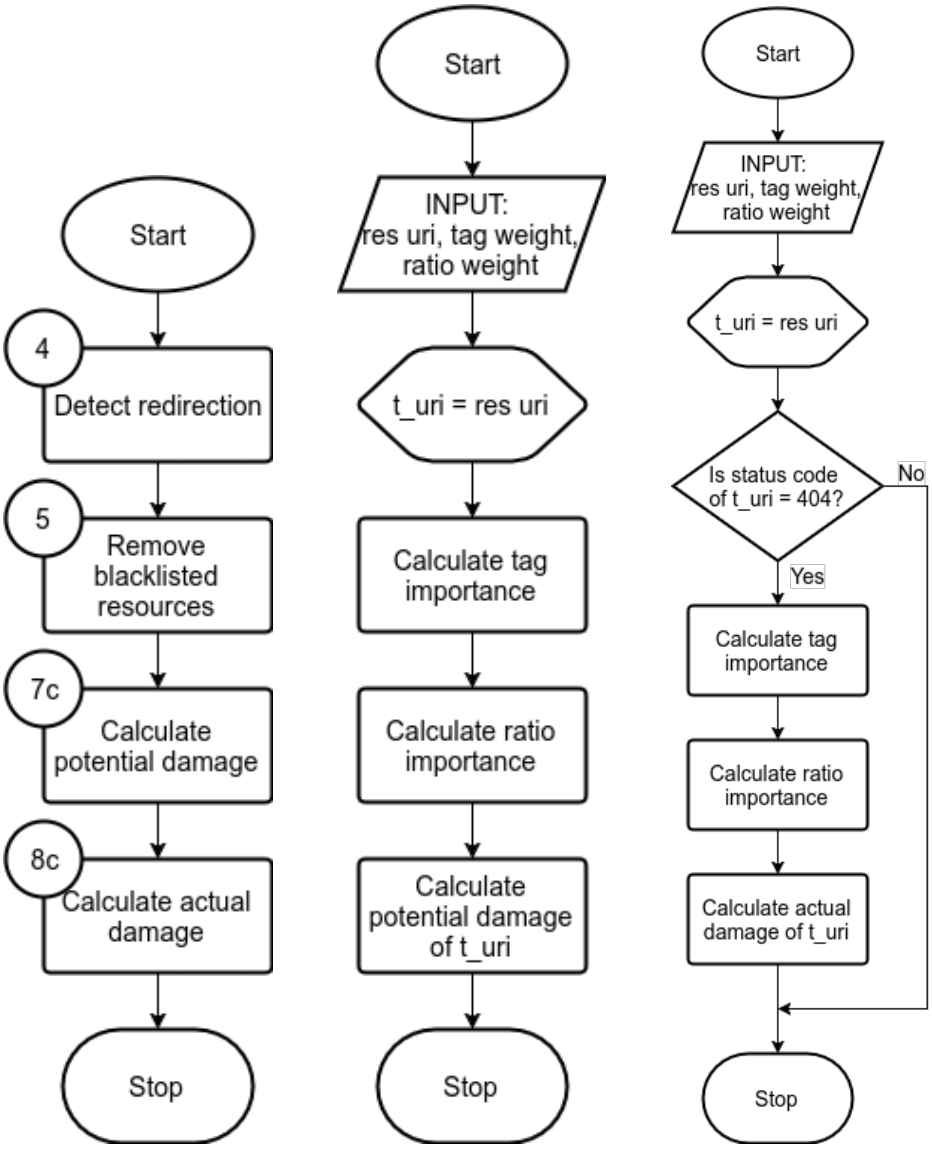
2.5.3 Text

Text damage is computed by calculating the number of words in a web page and divided it by 1000. Text damage only contributes to potential damage so that if a text-contained page lost all of its embedded resources, the damage would never be 100%.



(A) *Image Damage Overview Flowchart* (B) *Image Potential Damage Flowchart* (C) *Image Actual Damage Flowchart*

FIGURE 2.14: The flowchart of Image Damage



(A) CSS Damage Overview Flowchart (B) CSS Potential Damage Flowchart (C) CSS Actual Damage Flowchart

FIGURE 2.15: The flowchart of CSS Damage

2.5.4 Javascript

Measuring the damage caused by missing javascript is still being something that is obscure and complicated. We still do not have a stable and robust algorithm for calculating javascript damage. One solution that we can implement now is to give a damage value of 1 if the javascript is missing and 0 if it is accessible.

Chapter 3

Implementation

The designs described in chapter 2 are implemented in various ways. Crawling process is implemented using `javascript` language and executed by `PhantomJs`. On the other hand, the damage analysis design is implemented using `Python`. We chose `Python` because it is a simple and straightforward programming language and provides many useful libraries. For the online service, the web interface is developed using `Python` and `HTML5`. Furthermore, we also use `Docker` for developing the local service.

The Memento-Damage tool is developed under the environment as follows:

- Ubuntu 16.04 LTS 64-bit operating system.
- Memory 11.6 GiB.
- Processor Intel® Core™ i7-6500U CPU @ 2.50GHz × 4.
- Harddisk 500 GiB.
- Python 2.7.12
- PhantomJs 2.1.1
- Docker 1.12.3
- HTML5 and CSS3 for website interface.

The complete script of the implementation of Memento-Damage tool is available on `GitHub`: `https://github.com/erikaris/web-memento-damage`.

3.1 Website

According to the design in section 2.3.1, the website consist of 2 (two) main pages: the home page and the result page. These pages are implemented in a responsive display (flexible to the screen size). The *home* page provides the examples of some mementos and a *textbox* to input the URI-M (Figure 3.1). The result page shows the calculation result of a certain URI-M as can be seen in Figure 3.2. This page also provides the damage details for every resource, with the items are displayed in a expandable-collapsable table format. This website can be accessed on `http://memento-damage.cs.odu.edu/`.

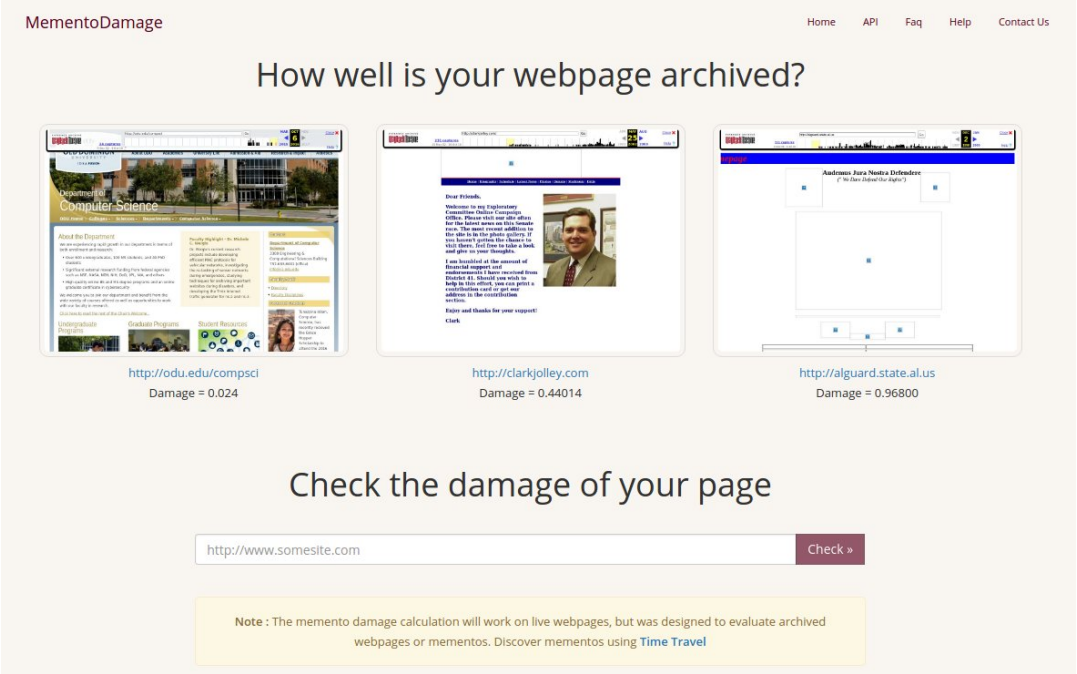


FIGURE 3.1: The *home* page design

3.2 REST API

REST API facilitates damage calculation from any HTTP Client (e.g. web browser, curl, etc) and give output in JSON format. This enables user to do further analysis with the resulting output. User can create a script and utilize the REST API to calculate damage on few number of URIs (e.g. 5). The default REST API usage for memento damage is:
`http://memento-damage.cs.odu.edu/api/damage/[the input URI-M]`.
Figure 3.3 and Figure 3.4 show examples of using Memento-Damage REST API with `Curl` and `Python`.

3.3 Docker

In order to use Memento-Damage Docker, user must first download and install Docker from the official Docker website [1]. We use `Ubuntu-LXDE` [11] to set the desktop environment so that the rendering result from `PhantomJS` will be similar to that of resulting from a real browser. Another reason for choosing `Ubuntu-LXDE` is because LXDE is lightweight and ideal for computer with low specifications [10].

The user can start using Memento-Damage by following these steps:

1. Pull the docker image
   `docker pull erikaris/memento-damage`

2. Run the docker image. Command example:
   `docker run --name memento-damage-wsdl02 3eef3f61e1e8`

[1] `https://docs.docker.com/engine/installation/`

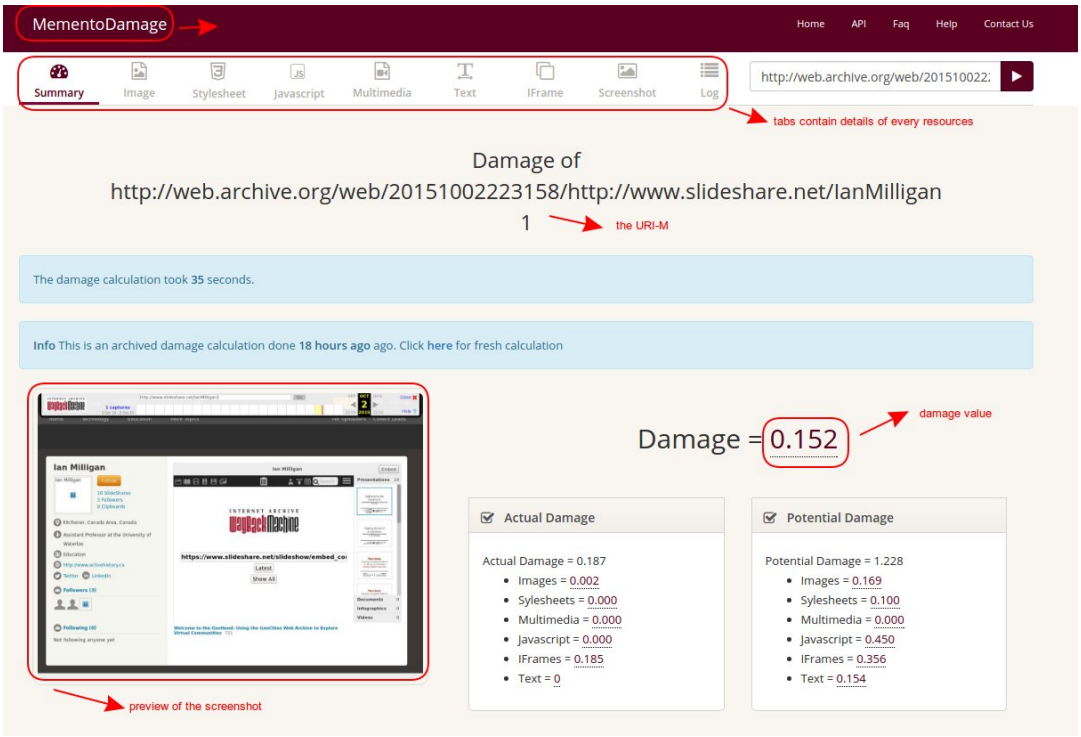


FIGURE 3.2: The calculation page

```
curl http://memento-damage.cs.odu.edu/api/damage/http://odu.edu/compsci
```

FIGURE 3.3: Example of using Memento-Damage REST API with Curl

User can also specifies how they want to use the Docker such as defining the output folder and the port number.

3. Start utilizing the Memento-Damage tool using command:
   `docker exec -it memento-damage bash`

4. Execute Memento-Damage using various commands provided.

3.4 Library

Using library is relatively similar to using docker. The installation process is simpler and faster than the docker version. But user has to ensure that all the requirements (phantomjs 2.xx and python 2.7) are installed on their machines.The latest library version can be downloaded from `https://github.com/erikaris/web-memento-damage/tree/library`. Install the library using command: `sudo pip install memento-damage-x.x.x.tar.gz`.

```
import requests
resp = requests.get('http://memento-damage.cs.odu.edu/api/damage/http://odu.edu/compsci')
print resp.json()
```

FIGURE 3.4: Example of using Memento-Damage REST API with Python

3.5 Command Line Interface (CLI)

Using the Docker or the Library version gives the user an ability to interact with Memento-Damage using Command Line Interface (CLI), where input is given through `stdin` and the output is displayed in `stdout`. Listing 3.1 gives the list of CLI options provided by Memento-Damage.

```
Usage: memento-damage [options] <URI>

Options:
-h, --help            show this help message and exit
-o OUTPUT_DIR, --output-dir=OUTPUT_DIR
output directory (optional)
-O, --overwrite       overwrite existing output directory
-m MODE, --mode=MODE  output mode: "simple" or "json" [default: simple]
-d DEBUG, --debug=DEBUG
debug mode: "simple" or "complete" [default: none]
-L, --redirect        follow url redirection
```

LISTING 3.1: CLI options for Memento-Damage

Explanation about the options:

1. *-d DEBUG* → shows the debugging processes that happen during the damage calculation as described in section 2.4. There are 2 (two) values available: simple and complete. The *simple* mode only shows the logging process on the level INFO (Figure 3.5), while the *complete* mode shows the logging process on both level INFO and DEBUG (Figure 3.6). Default value for this option is *none*

```
erikaris@erikaris-Inspiron:~$ memento-damage -d simple http://odu.edu/compsci
stdout_thread=<Thread(Thread-2, initial daemon)>
2017-01-25 10:45:39 http://odu.edu/compsci INFO Start crawling URI http://odu.edu/compsci
2017-01-25 10:45:39 http://odu.edu/compsci INFO Receiving resource(s)
2017-01-25 10:45:41 http://odu.edu/compsci INFO Page is loaded
2017-01-25 10:45:43 http://odu.edu/compsci INFO Page is loaded
2017-01-25 10:45:46 http://odu.edu/compsci INFO Saving HTML source --> creating /tmp/tmpbhmvZX/source.html
2017-01-25 10:45:46 http://odu.edu/compsci INFO Processing images --> creating /tmp/tmpbhmvZX/image.log
2017-01-25 10:45:46 http://odu.edu/compsci INFO Processing videos --> creating /tmp/tmpbhmvZX/video.log
2017-01-25 10:45:47 http://odu.edu/compsci INFO Processing stylesheets --> creating /tmp/tmpbhmvZX/css.log
2017-01-25 10:45:47 http://odu.edu/compsci INFO Processing screenshot --> creating /tmp/tmpbhmvZX/screenshot.png
2017-01-25 10:45:47 http://odu.edu/compsci INFO Crawl finished in 7873 miliseconds
2017-01-25 10:45:47 http://odu.edu/compsci INFO Remove blacklisted URIS
2017-01-25 10:45:47 http://odu.edu/compsci INFO Blacklisted URIS: https://analytics.archive.org/, [INTERNAL]
2017-01-25 10:45:47 http://odu.edu/compsci INFO Resolve URI redirection
2017-01-25 10:45:47 http://odu.edu/compsci INFO Calculate percentage coverage
2017-01-25 10:45:47 http://odu.edu/compsci INFO Find missing URIS
2017-01-25 10:45:47 http://odu.edu/compsci INFO Start calculating damage...
2017-01-25 10:45:47 http://odu.edu/compsci INFO Calculating potential damage
2017-01-25 10:45:47 http://odu.edu/compsci INFO Calculate potential damage for Image(s)
2017-01-25 10:45:47 http://odu.edu/compsci INFO Potential damage of https://pbs.twimg.com/media/C1fiX8PWgAAoTnj.png:small is 0.2909285
71429
```

FIGURE 3.5: Option *-d simple*

2. *-m mode* → define the output format displayed on terminal: simple or json. Simple format just straightforwardly prints the damage value on the terminal as illustrated in Figure 3.7, while *json* option displays the output in a JSON format as can be seen in Figure 3.8. The default value for *mode* is *simple*.

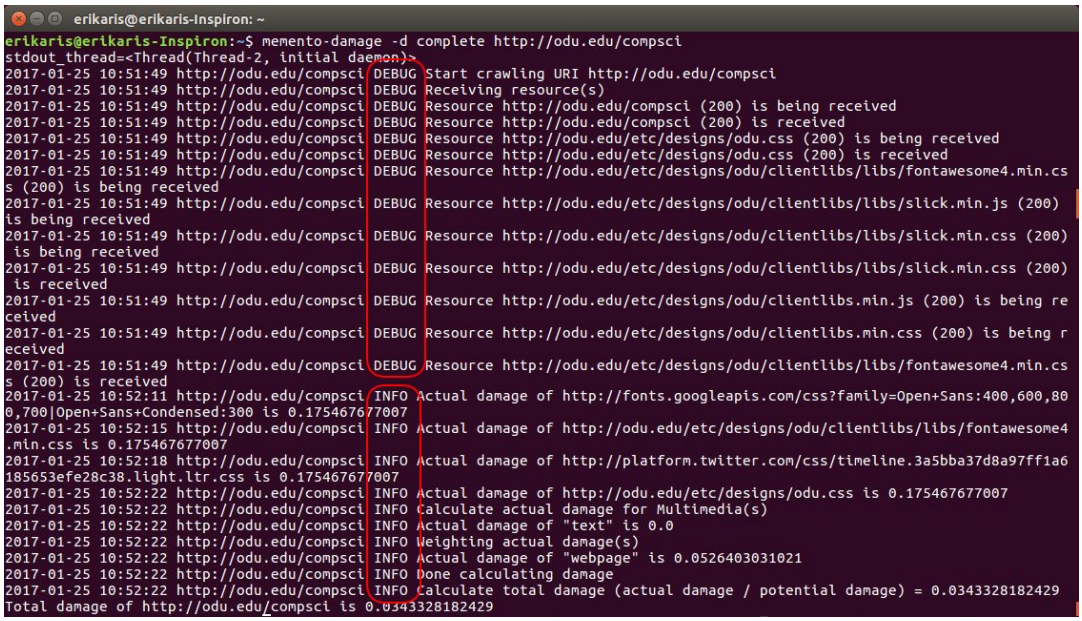


FIGURE 3.6: Option *-d complete*

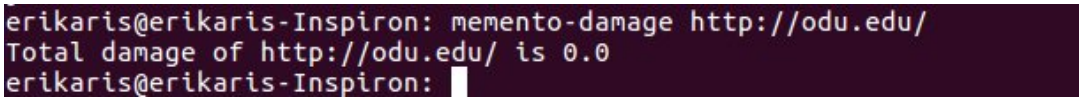


FIGURE 3.7: Option *-m simple*

3. *-o output_dir* → specify the directory to store the output. By default, the log output will be removed after the calculation process finished.

4. *-O overwrite* → Overwrite the previous damage calculation of the same URI-M.

5. *-L Redirection* → follow the redirection if the `status_code` of the input URI is 302. Figure 3.9 shows the difference between Memento-Damage's output with and without following redirection (Figure 3.9b and Figure 3.9a, respectively).

3.6 Output Format

1. **`network.log`** → contains a key-value pair which represent the respond for the header request.

2. **`result.json`** → a file contains the final result of the memento-damage analysis. The content of this file is written in `json` format and consists of several key-values pair as can be seen in Figure 3.10.

3. **`image.log`** → the content structure is shown in Listing 3.2 and Figure 3.11.

4. **`iframe.log`** (Listing 3.3 and Figure 3.12). The content in Figure 3.12) is read into an array `iframe_logs` with the content shown in Listing 3.4.

```
"total_damage": 0.14685302055818122,
"weight": {
    "text": 0.09999999999999998,
    "image": 0.29999999999999993,
    "multimedia": 0.5,
    "js": 0.05,
    "iframe": 0.29999999999999993,
    "css": 0.05
},
"calculation_time": 18,
"text": {
    "words_per_image": 1000,
    "num_words": 374
},
"uri": "http://web.archive.org/web/20030201161950/http://clarkjolley.com",
"timer": {
    "request_time": 1493781576.353206,
    "response_time": 1493781594.672518
},
"jses": [],
"actual_damage": {
    "text": 0.0,
    "image": 0.07500471313947875,
    "multimedia": 0.0,
    "js": 0.0,
    "iframe": 0.0,
    "total": 0.07500471313947875,
    "css": 0.0
},
"potential_damage": {
    "text": 0.03739999999999999,
    "image": 0.4233468191964283,
    "multimedia": 0.0,
```


FIGURE 3.8: Option *-m json*


```
1 {"rectangles": [{"height":206, "left":813, "top":5, "width":2
    06}], "url":"http://www.cs.odu.edu/~mkelly/mofro.png", "
    visible":true, "viewport_size":[1024,777]}
```


LISTING 3.2: Image.logs

```
root@c21ce3979912:/app/cache# memento-damage http://matkelly.com
2017-05-01 20:49:15 http://matkelly.com ERROR Qt: XKEYBOARD extension not present on the X
 server.
2017-05-01 20:49:16 http://matkelly.com ERROR Status code 301
2017-05-01 20:49:16 http://matkelly.com ERROR Protocol "" is unknown
2017-05-01 20:49:16 http://matkelly.com ERROR Status code 301
2017-05-01 20:49:16 http://matkelly.com ERROR Page is moved permanently (Status code 301)
Error in processing http://matkelly.com: Page is moved permanently (Status code 301)
```

(A) CLI output without following the redirection

```
root@c21ce3979912:/app/cache# memento-damage -L http://matkelly.com
2017-05-01 20:49:33 http://matkelly.com ERROR Qt: XKEYBOARD extension not present on the X
 server.
Total damage of http://matkelly.com is 0.0
```

(B) CLI output with following the redirection

FIGURE 3.9: The CLI output without and with following the redirection


```
{
    "csses": [...],
    "redirect_uris": [],
    "iframes": [],
    "total_damage": 0.0,
    "weight": {"text": 0.09999999999999998...},
    "calculation_time": 13,
    "text": [...],
    "uri": "http://matkelly.com",
    "timer": {"request_time": 1491998203.706438...},
    "jses": [...],
    "actual_damage": {"text": 0.0...},
    "potential_damage": {"text": 3.5058871334277204...},
    "error": false,
    "images": [...],
    "message": "Calculation is finished in 13 seconds",
    "multimedias": [],
    "is_archive": false
}
```


FIGURE 3.10: result json


```
{"rectangles":
      [{"height":206,
        "left":813,
        "top":5,
        "width":206}],
"url":"http://www.cs.odu.edu/~mkelly/mofro.png",
"visible":true,
"viewport_size":[1024,777]}
```


FIGURE 3.11: Image Logs

```
{"url":"http://web.archive.org/web/20151002223158if_/http://
   www.slideshare.net/IanMilligan1/slideshelf", "status_code"
   :null, "content_type":null, "headers":{}, "height":480, "
   left":360, "parent":"", "top":120, "visible":true, "width"
   :618, "viewport_size":[1024,777]}
```

LISTING 3.3: iframe.logs

FIGURE 3.12: Iframe Logs

```
[
a:  {"url":"http://web.archive.org/web/20151002223158if_/http
   ://www.slideshare.net/IanMilligan1/slideshelf", "
   status_code":null, "content_type":null, "headers":{}, "
   height":480, "left":360, "parent":"", "top":120, "visible"
   :true, "width":618, "viewport_size":[1024,777], "
   redirect_urls": [("url": xxx, "status_code": xxx, "content
   -type": xxx), (), ()], "coverage": xxx,  "important":
   False},
b: {"url":"http://web.archive.org/web/20151002223158if_/http:
   //www.slideshare.net/IanMilligan1/slideshelf", "
   status_code":null, "content_type":null, "headers":{}, "
   height":480, "left":360, "parent":"", "top":120, "visible"
   :true, "width":618, "viewport_size":[1024,777], "
   redirect_urls": [("url": xxx, "status_code": xxx, "content
   -type": xxx), (), ()], "coverage": xxx, "important": False
   }
]
```

LISTING 3.4: Array Iframe Logs

Chapter 4

Testing and Evaluation

4.1 Problem found during testing

Testing is the most important and critical part of this research project. During testing, we found so many bugs and errors that have to be fixed, including bugs related to *headers* identification, URI without scheme, OSError, IOError, and case-sensitve problem.

1. **Limitation from Internet Archive**.
   Internet archive has a limitation related to the number of concurrent requests that can be sent at a time from the same IP address. We encountered an abundant number of *connection refused* issues as depicted in Figure 4.1. It caused the testing process took quite a long time for crawling the resources of each URI. The current testing send 10 concurrent requests at a time.

FIGURE 4.1: 'Connection Refused' problem with internet archive

2. **Handling lazy load**.
   Lazy load caused some of the images are not shown on the screenshot as illustrated in Figure 4.2.

3. **Problem with Webrecorder**
   Webrecorder uses two iframes that are from different domain. A browser will not allow a cross-domain frame access, which makes Memento-Damage failed to load the resources from the top frame. To solve this problem, we need to reformat the webrecorder URI as described in Listing 4.1. Figure 4.3 and Figure 4.4 show the difference between before reformatting and after reformatting.

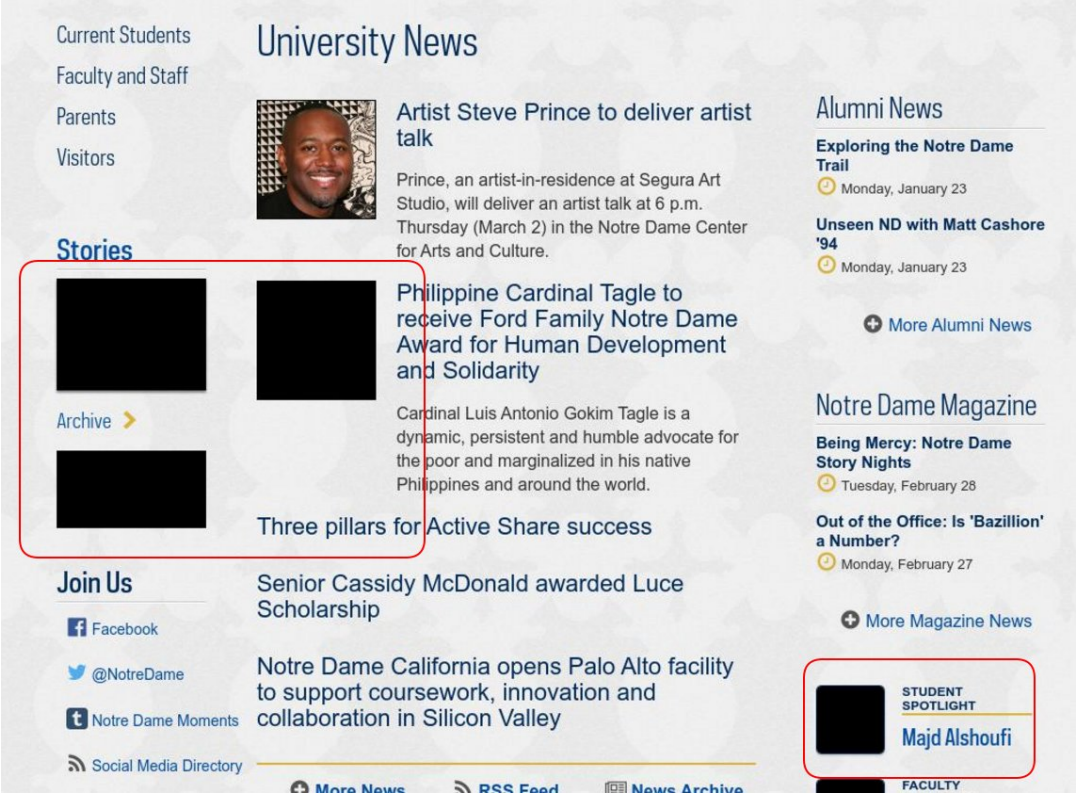
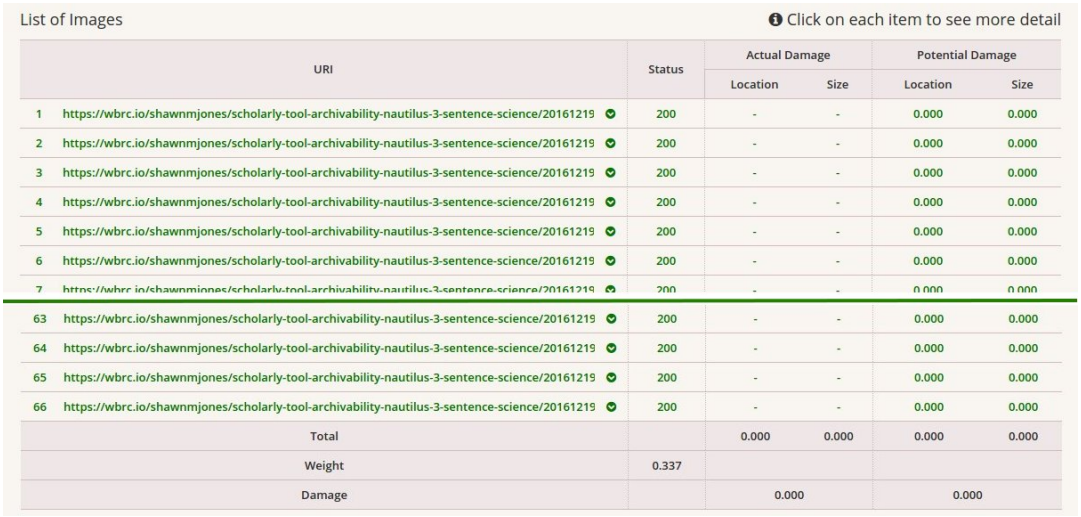


FIGURE 4.2: Lazy load caused some images are not displayed on the screenshot

4. **Problem with no-end-backslash URI-Ms**
   Issue with slash at the end of the URI. From the URI-Ms in the list, there are some URI-Ms that behave weirdly (show different displays) when given or not given a backslash (/) at the end of the URI-M. The list of URI-Ms with backslash problem is shown in Table 4.1.

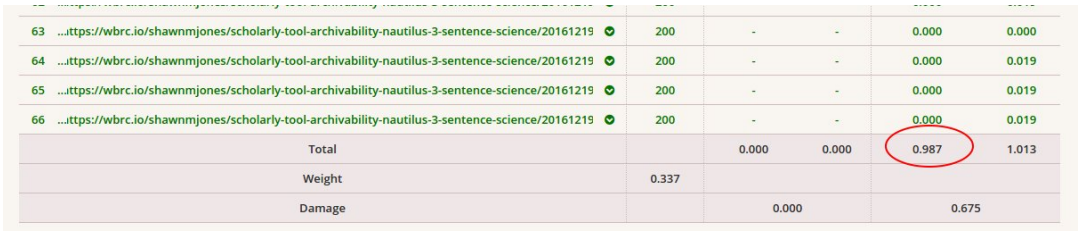
5. **A mismatch between URI and** `header location`
   Solving the redirection on a URI u is conducted by finding the next URI in **`network.log`** that match u's `header location`. But sometimes, a mismatch could happen because the `header location` is written differently. For example, in Figure 4.5 we see that the `header location` is encoded while there is no URI in **`network.log`** encoded. It means, our attempt to find a matching URI for the `header location` will be failing. Another cause of mismatch is the use of capital letters in `header location`. To anticipate these problems, we decode and convert the `header location` into a lowercase format before matching it with the URI.



List of Images — Click on each item to see more detail

	URI	Status	Actual Damage		Potential Damage	
			Location	Size	Location	Size
1	https://wbrc.io/shawnmjones/scholarly-tool-archivability-nautilus-3-sentence-science/20161219	200	-	-	0.000	0.000
2	https://wbrc.io/shawnmjones/scholarly-tool-archivability-nautilus-3-sentence-science/20161219	200	-	-	0.000	0.000
3	https://wbrc.io/shawnmjones/scholarly-tool-archivability-nautilus-3-sentence-science/20161219	200	-	-	0.000	0.000
4	https://wbrc.io/shawnmjones/scholarly-tool-archivability-nautilus-3-sentence-science/20161219	200	-	-	0.000	0.000
5	https://wbrc.io/shawnmjones/scholarly-tool-archivability-nautilus-3-sentence-science/20161219	200	-	-	0.000	0.000
6	https://wbrc.io/shawnmjones/scholarly-tool-archivability-nautilus-3-sentence-science/20161219	200	-	-	0.000	0.000
7	https://wbrc.io/shawnmjones/scholarly-tool-archivability-nautilus-3-sentence-science/20161219	200	-	-	0.000	0.000
63	https://wbrc.io/shawnmjones/scholarly-tool-archivability-nautilus-3-sentence-science/20161219	200	-	-	0.000	0.000
64	https://wbrc.io/shawnmjones/scholarly-tool-archivability-nautilus-3-sentence-science/20161219	200	-	-	0.000	0.000
65	https://wbrc.io/shawnmjones/scholarly-tool-archivability-nautilus-3-sentence-science/20161219	200	-	-	0.000	0.000
66	https://wbrc.io/shawnmjones/scholarly-tool-archivability-nautilus-3-sentence-science/20161219	200	-	-	0.000	0.000
	Total		0.000	0.000	0.000	0.000
	Weight	0.337				
	Damage		0.000		0.000	

FIGURE 4.3: Before Webrecorder Reformat



	URI	Status	Actual Damage		Potential Damage	
			Location	Size	Location	Size
63	...ttps://wbrc.io/shawnmjones/scholarly-tool-archivability-nautilus-3-sentence-science/20161219	200	-	-	0.000	0.000
64	...ttps://wbrc.io/shawnmjones/scholarly-tool-archivability-nautilus-3-sentence-science/20161219	200	-	-	0.000	0.019
65	...ttps://wbrc.io/shawnmjones/scholarly-tool-archivability-nautilus-3-sentence-science/20161219	200	-	-	0.000	0.019
66	...ttps://wbrc.io/shawnmjones/scholarly-tool-archivability-nautilus-3-sentence-science/20161219	200	-	-	0.000	0.019
	Total		0.000	0.000	0.987	1.013
	Weight	0.337				
	Damage		0.000		0.675	

FIGURE 4.4: After Webrecorder Reformat

```
erikaris@erikaris-Inspiron:~$ curl -I 'https://wbrc.io/min/b=assets/site/js&f=li
b/jquery.easing-1.3.js,lib/jquery.touchswipe-1.6.2.min.js,lib/jquery.nicescroll-
3.1.9.min.js,orig_script.js,script.js,search.js'
HTTP/1.1 302 Found
Server: nginx/1.10.0 (Ubuntu)
Date: Wed, 12 Apr 2017 16:49:54 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 0
Connection: keep-alive
Location: https://webrecorder.io/_set_session?path=/min/b%3Dassets/site/js%26f%3
Dlib/jquery.easing-1.3.js%2Clib/jquery.touchswipe-1.6.2.min.js%2Clib/jquery.nice
scroll-3.1.9.min.js%2Corig_script.js%2Cscript.js%2Csearch.js
X-Proxy-Cache: BYPASS
Strict-Transport-Security: max-age=31536000
```

FIGURE 4.5: encoded redirected urim

```
#!/usr/bin/python
def reformat_uri_webrecorder_io(self):
if self.uri.startswith('https://webrecorder.io') or self.uri.startswith('
    http://webrecorder.io'):
wrc_comps = self.uri.replace('https://webrecorder.io/', '').split('/')
user, project, date = wrc_comps[:3]
wrc_url = '/'.join(wrc_comps[3:])
new_uri = 'https://wbrc.io/{}/{}/{}id_/{}'.format(user, project, date,
    wrc_url)
self.uri = new_uri
```

LISTING 4.1: Webrecorder Reformat

4.2 Testing with Small Samples

For this testing, we run damage analysis for a small number of URIs, using both the Docker version and the library version of Memento-Damage. The testing are conducted on 3 (three) different machines:

1. A Dell laptop with specification: Intel® Core™ i7-6500U CPU @ 2.50GHz × 4 , Memory 11.6 GiB, and Ubuntu 16.04 LTS 64-bit.
2. A WSDL02 machine, with specification: Intel(R) Xeon(R) CPU E5-2660 v2 @ 2.20GHz, Memory 4 GiB
3. Amazon web service

The testing result show that for the same URI, there are slight differences between the damage values output by those 3 (three). It happens because those 3 different machines produces slight different screensthot. To avoid this environment-dependent problem, we recommend to use the Docker version of Memento-Damage that guarantees the URI will always be run on the same machine's environment.

4.3 Correlation Test with the Previous Work

4.4 Testing with Large Samples

4.4.1 Dataset

The testing with large samples is actually a rehash of what have been done in the previous work [4] because we want to see the changes that happen during this past three years. Thus, we use the same URI list as that of the previous work. Both testing code and the list of URIs are available on github [1]. The testing is conducted using the Docker version of Memento-Damage on *WSDL02* machine. The process for each URI is run in parallel using several workers as illustrated in the flowchart in Figure 4.6.

Before being used, the URI list is checked using simple script to remove any duplicate URIs. The duplication check shows that 333 out of 1861 URIs are duplicated. The distinct version of the list contains 1528 URIs, which is also available on github[2].

[1] https://raw.githubusercontent.com/erikaris/web-memento-damage/testing/testing/uris.txt
[2] https://raw.githubusercontent.com/erikaris/web-memento-damage/testing/testing/uris_distinct.txt

No in the list	URI-M
4	https://web.archive.org/web/19970126104450/http://www.ascc.edu
5	https://web.archive.org/web/20000301044612/http://www.dps.state.al.us
6	https://web.archive.org/web/19990125103021/http://www.sarpc.org
8	https://web.archive.org/web/20000621021817/http://www.blackstarnews.com
9	https://web.archive.org/web/19990125094845/http://www.dot.state.al.us
10	https://web.archive.org/web/20020525205220/http://clarkjolley.com
14	https://web.archive.org/web/19970310194443/http://www.aptv.org
15	https://web.archive.org/web/19961101000501/http://cmt.com
18	https://web.archive.org/web/19981206004215/http://www.legislature.state.al.us
19	https://web.archive.org/web/19961023021028/http://www.arborweb.com
23	https://web.archive.org/web/20000817094530/http://shpda.state.al.us
25	https://web.archive.org/web/20000817010647/http://www.tudorhistory.org
27	https://web.archive.org/web/19981206182418/http://www.mtg.unimelb.edu.au
28	https://web.archive.org/web/20000817094530/http://psychology.state.al.us
31	https://web.archive.org/web/20030201194753/http://www.hblb.state.al.us
48	https://web.archive.org/web/19961023021028/http://www.arborweb.com
62	https://web.archive.org/web/20010429022932/http://www.alalinc.net/jic
63	https://web.archive.org/web/20010201144800/http://www.williamsf1.com

TABLE 4.1: List of URI-Ms that give different display with and without backslash at the end of the URI-M

Using this distinct URIs, we download the timemap for each URI to get all of its URI-Ms. There are a total of 108,511[3] URI-Ms downloaded, which is almost three times as many as that of the previous work (45,341). This downloaded mementos as well as their resources are stored locally in Docker's `'/app/cache'` folder for further analysis. In this testing, The total size of the final output is 32 GB.

The crawling process is very slow since there is a limitation from internet archive in the number of concurrent requests that we can send at a time. This problem is solved by defining a specific user-agent 'memento-damage research ODU <@Web-SciDL>' when sending requests to server. But we still get a limitation of sending 10 (ten) concurrent requests at a time. Despite this permission from Internet Archive, the 'connection refused' problem still happens several times. But at least, the testing progress becomes a bit faster than before.

The testing was initially runs with 4 concurrent requests. Assuming that 1 URI-M takes 30 seconds to process 1 (one) URI (on average), then 108.511 URIMs need (108,511/4)*(0.5 mins)/60 = 226.06 hours = 9.41 days. But in reality, apart from the 'connection refused' problem, we faced so many other issues that caused the testing takes a longer time than expected. Some of these issues are caused by testing script itself, but some other are caused by non-script error such as memory overload, 'too many open files' problem, and limited disk quota problem.

The testing steps using the Docker are as follows:

1. Pull the docker image using command `docker pull erikaris/memento-damage`

[3]https://raw.githubusercontent.com/erikaris/web-memento-damage/testing/testing/output_urim_distinct.txt

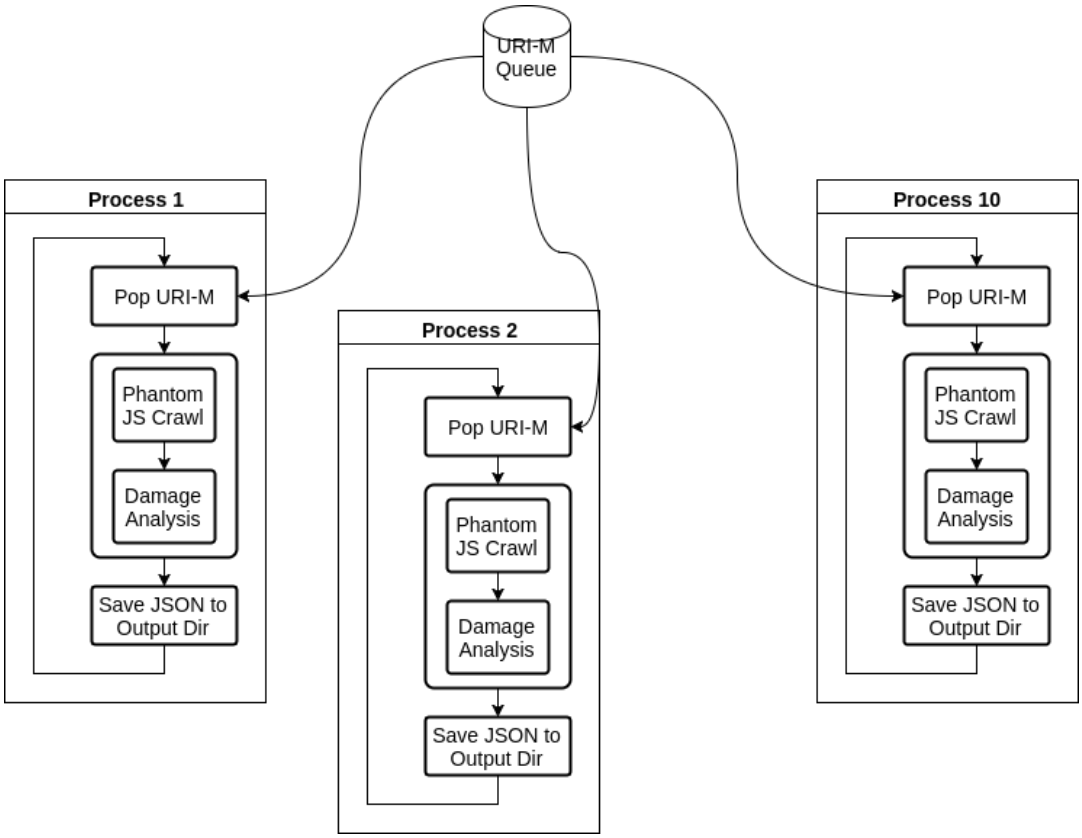


FIGURE 4.6: Testing workers

2. Create and run the container Figure 4.7 for the docker image using command:
   `docker run -itPd -m 2g -e PORT=8888 -v /data/esiregar:/app/cache --net=host --name=memento-damage-test erikaris/memento-damage:latest /app/entrypoint.sh`

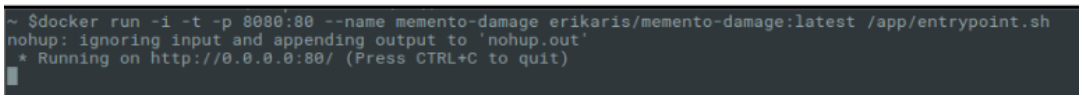


FIGURE 4.7: A successfully running Memento-Damage's Docker container

3. execute any command that we want to run using:
   `docker exec -it memento-damage bash`.

Command for testing: `nohup ~/python/bin/python memento_damage_test_2.py uris_distinct.txt output_urim_distinct.txt output_dir &`. The testing code is available on github [4].

[4] https://raw.githubusercontent.com/erikaris/web-memento-damage/testing/testing/memento_damage_test.py

4.5 Result

The testing takes a total time of 1 (one) month, from March 6, 2017 to April 26, 2017. This long duration of testing happens due to the numerous of issues that we face during the testing. The result of the testing are visualized as 1 (one) line graph, 5 (five) bar charts, and 1 (one) area chart. The statistical summary of the test result can be seen in Listing 4.2.

```
Summary of Data
=========================================
Total URI-M: 108511
Number of URI-M are successfully processed: 80580
Number of URI-M has json-format error: 3
Number of URI-M are failed to process: 27928
Number of URI-M has not processed yet: 0
```

LISTING 4.2: The statistical summary of the testing data

Figure 4.8 shows the number of average missing embedded resources per memento per year. From Figure 4.8, we can see that, in general, the damage value (Dm) is directly proportional to the percentage of missing resources (Mm). From the year 1996 to 2005, Dm and Mm share a similar pattern. In some years, the Dm value is higher than Mm value which means the resources that are missing during those years are important. The most interesting case happens in 2011 where the Dm value is significantly higher than Mm. It means, although on the average the URIMs in 2011 only lost 4% of their resources, but these losses actually caused 4 (four) times higher damages than the number showed by Mm. On the other hand, in 2008, 2010, 2013, and 2017, Dm value is lower than Mm, which implies those missing resources are less important. This result gives us insight that Dm can well represent the real damages that happen on a URIM by taking the importance of each resource into account.

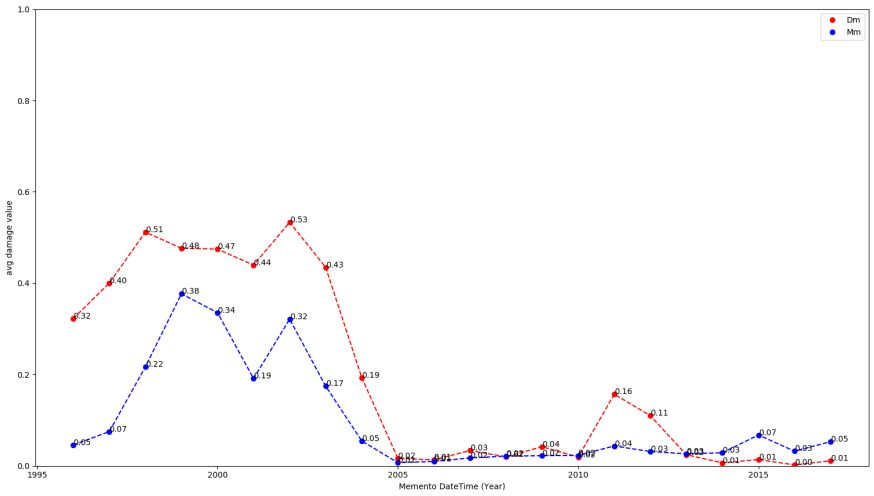


FIGURE 4.8: the average embedded resources missed per memento per year

Furthermore, from Figure 4.9 we can see that around 40% (40,000) of the URIMs included in this testing do not lose any of their resources at all. Meanwhile, roughly 25% of the URIMs only lose 1 (one) or 2 (two) resources . These facts show that more than 50% of the tested URIMs are in a good condition (not damaged).

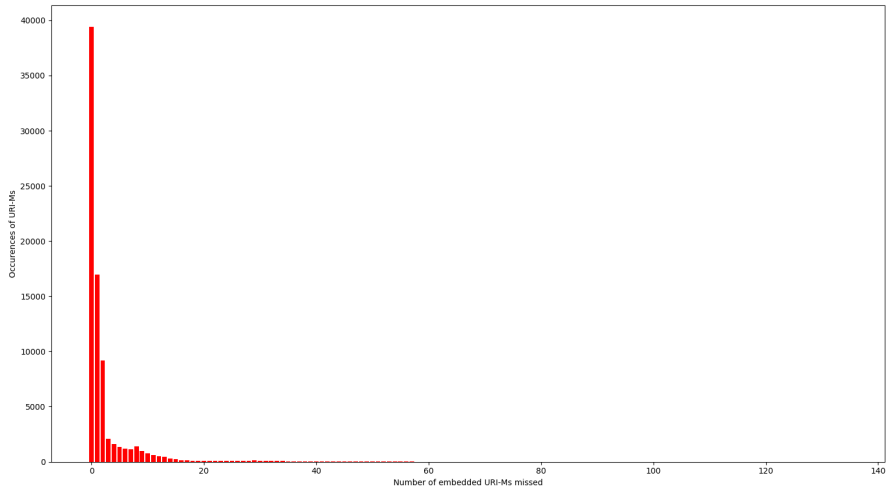


FIGURE 4.9: The average number of missing embedded resources per uri-m

From Figure 4.10 we can see that the number of resources dereferenced by each URI varied between 0 to 260 resources. Interestingly, there are over 3,000 URIMs that dereference 0 resources. But by cross-checking Figure 4.10 and Figure 4.13, we can see that there are around 1,300 URIMs that has no resource at all. This indicates the URIMs that are not successfully dereferenced any of their resources (Figure 4.10) are the URIMs which do not have any resources at all. Therefore, we can claim that almost all of the tested URIMs successfully dereferenced at least one of their resources.

Figure 4.12 shows the number of missing resources grouped by the MIME types. From Figure 4.12, we can see that most of the missing image cases happen on the URIMs from before 2005, with the largest loss rate occurs in 2002 (roughly 18 images are missing per memento). There is no multimedia resources and only very few CSSes (less than 1 CSS in a year) that are missing from the tested URIMs. Meanwhile, it appears that most of the missing javascripts are from the URIMs created after 2005. It might due to the fact that most of the URIs created before 2005 did not implement the javascript technology.

From Figure 4.13, we can see the number of tested URIMs grouped by their archived year. Most of the tested URIMs are created after 2010 with the largest URIMs are from 2016. The number of URIMs in 2014 declines significantly in 2015. It might be caused by the change that happened in Internet Archive regarding their archiving activities.

Figure 4.14 shows the number of total resources in each URIMs and their missing resources. The *x axis* represents each URIM sorted from the URIM that has the most resources to the URIM that has the least resources, while the *y axis* represents the

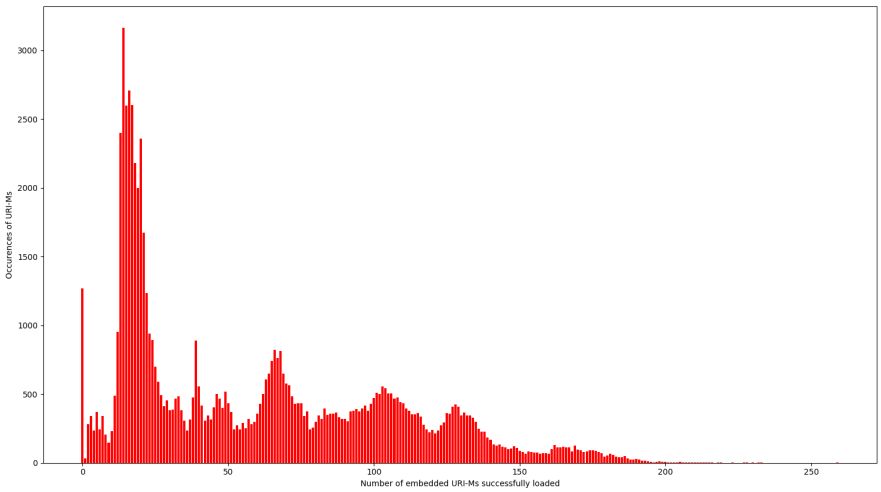


FIGURE 4.10: The average number of successfully dereferenced embedded resources per uri-m

number of resources owned by each URIM. From Figure 4.14, we can see that none of the URIMs being tested has complete resources. It means, the URIM lost at least one of its embedded resources.

4.5.1 Comparison with the previous version

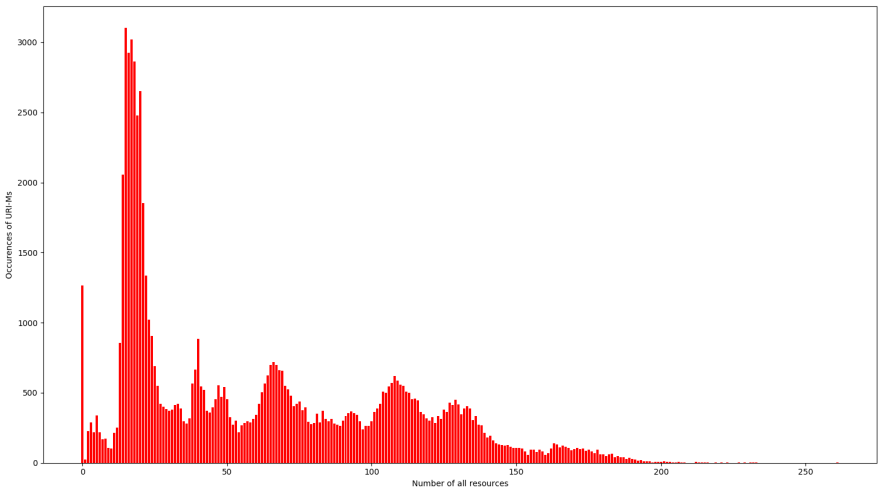


FIGURE 4.11: Distribution of the number of all resources per URI-M

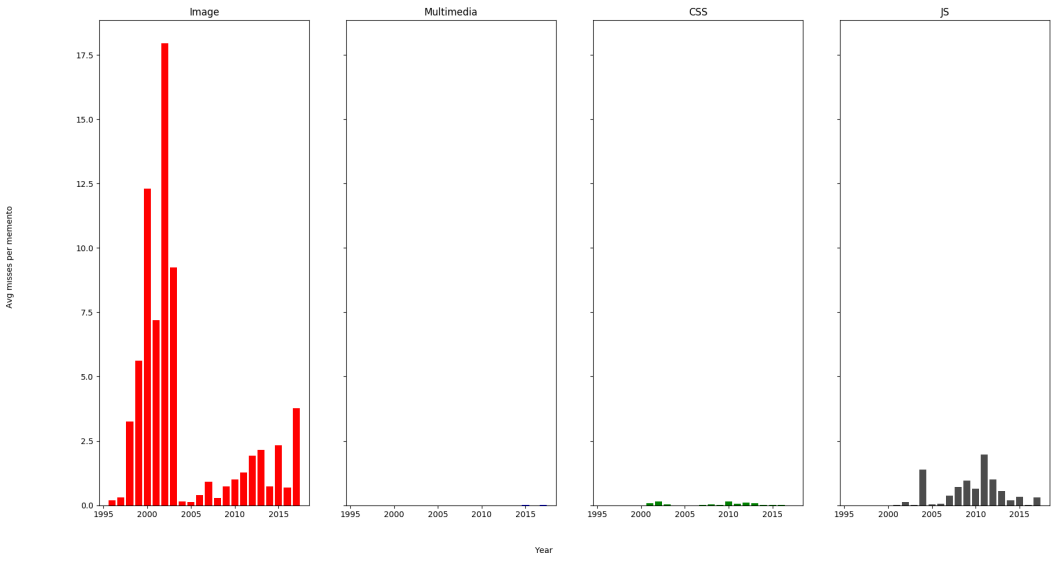


FIGURE 4.12: Number of missed embedded resources per memento per year and MIME type

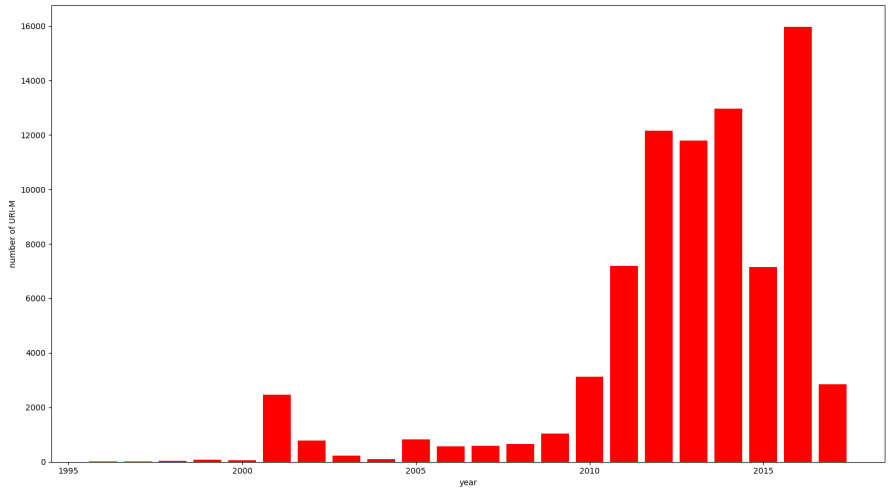


FIGURE 4.13: The number of mementos per year

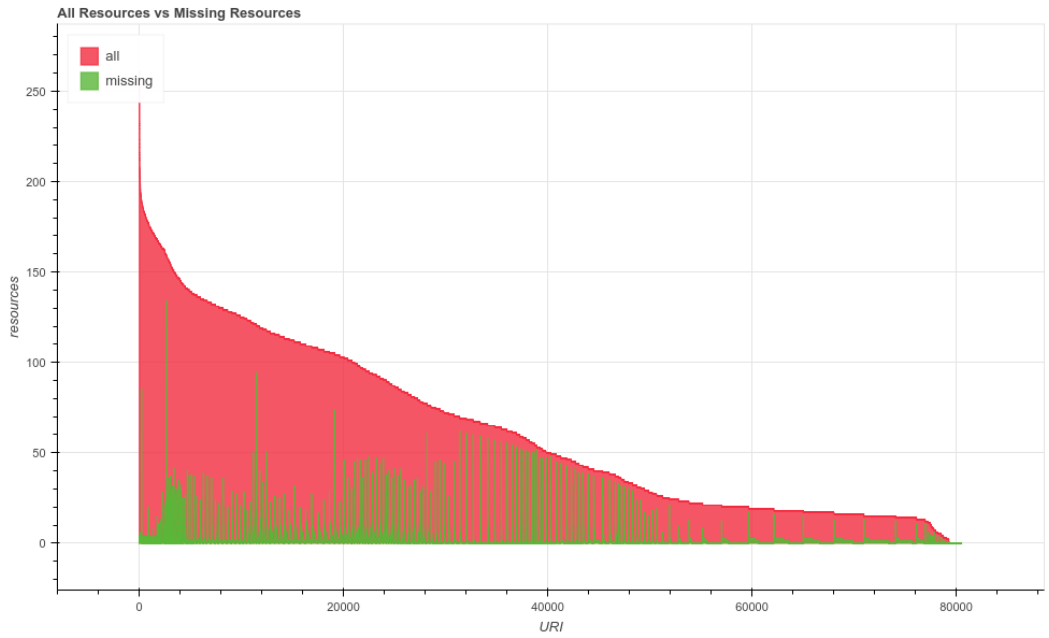


FIGURE 4.14: Comparison of All Resources vs Missing Resources

No	Items	Previous	Current
1	Programming Language	Perl	Python, HTML
2	Interface	CLI	Website, REST API, Docker, Library, CLI
3	Execution	Single instance	Multiple instance (batch)
4	Output format	Single output format: stdout	Various output format: stdout, plain text, JSON format
5	Installation	No integrated installation. All of the software dependencies must be installed separately.	Integrated installation. Single command to install the Memento-Damage tool and all of its dependencies.
6	Resources from the Wayback Machine	Included in the damage calculation	Excluded in damage calculation
7	Bugs and typos	There are several critical bugs and typos that severely affect the result of the damage calculation	Fixed
8	Redirection	Redirected URI resources are not chased down	Follow the redirected URI resources until we get the final status code (200, 403, 404).
9	Internal CSS	Included in the damage calculation	Excluded in the damage calculation.
10	Resource detection	Use native javascript and file extensions	Use JQuery

TABLE 4.2: Comparison between the previous and current verson of Memento-Damage tool

Chapter 5

Conclusion

5.1 Future Work

There are several challenges and issues that remain unasnwered in this research project. These issues need to be solved to improve the performance of Memento-Damage. We put these challenges as future work, which are:

1. Need a more stable algorithm for measuring damage on iframe
   (a) if iframe is missing
   (b) if iframe is accessible but the resources inside of it are missing.
   (c) cross-domain iframe.
2. A more elegant solution for lazy load problem.
3. Handling image without tag <img>
   This problem is mostly related to the images that act as a background such the one on
   `https://web.archive.org/web/19981206004215/http://www.legislature.state.al.us`.
   Losing this image will cause a huge damage on the page since it has a large size and it also crosses both vertical and horizontal center.
4. Handling [xhr] resources.
   For now, the tool cannot catch the damage value caused by a 404 multiget-xhr.
   The example of this problem can be found on the URI:
   `https://bizcontent.silk.co/explore/table/collection/website/column/publisher/column/change-in-traffic-jul-dec-2015/pin/filter/equals/publisher/Mediaquest%20Corp`.
5. Handling the *not in archive* problem.
   Example: `http://wayback.archive-it.org/8130/20161215231900/https://www.mendeley.com/profiles/helen-palmer/`.
   "Not in Archive. The page you requested has not been archived in Archive-It".
6. Handling the inconsistency in screenshot rendering between library and docker version.
   It happens because PhantomJS generates a slightly different screenshot when run on different environment. The screenshot depends on the web-rendering conducted by the OS installed on the machine. Different OS has different theme which will affect the page display as well as the screenshot produced by PhantomJS.

7. More stable algorithm for measuring javascript damage.
8. Prospective integration with other Web Science and Digital LIbrary (WS-DL) lab's products, such as WAIL, Memgator, Thumbnail summarization, and Web archive for storytelling.
9. Use other headlesss web browser as a replacement for PhantomJs [13] such as headless Chrome [3].

5.2 Conclusion

In this research, we have improved the calculation method for measuring the damage on a memento (URIM) based on the prototype from the previous version [4]. We succesfully deployed the memento damage tool into 4 (four) interfaces: Websites, REST API, Docker, and Library. This Memento-Damage is a comprehensive tool that not only presents the damage value, but also provides the detail of every resources as well as various output formats to support further data analysis. This tool also has underwent a series of testing both using small samples and large samples. The testing result using small samples show that the current Memento-Damage tool works faster in calculation the damage than the previous version. The testing using large data samples shows that Memento-Damage can handle an enormous number of data (108,511 URIMs). Improvements are needed to make the Memento-Damage tool better and increase user experience.

Bibliography

[1] Internet Archive. *Wayback Machine*. http://web.archive.org/. [Online; accessed 18-April-2017].

[2] Henri Benoit. *Cross-document communication with iframes*. https://benohead.com/cross-document-communication-with-iframes/. [Online; accessed 18-February-2016].

[3] Eric Bidelman. *Getting Started with Headless Chrome*. https://developers.google.com/web/updates/2017/04/headless-chrome. [Online; accessed 18-April-2017].

[4] Justin F. Brunelle et al. "Not all mementos are created equal: measuring the impact of missing resources". In: *International Journal on Digital Libraries* 16.3 (2015), pp. 283–301. ISSN: 1432-1300. DOI: 10.1007/s00799-015-0150-6.

[5] Inc. Docker. *What is Docker*. https://www.docker.com/what-docker. [Online; accessed 14-March-2016].

[6] Python Software Foundation. *16.2. threading — Higher-level threading interface*. https://docs.python.org/2/library/threading.html. [Online; accessed 24-October-2016].

[7] Python Software Foundation. *17.1. subprocess — Subprocess management*. https://docs.python.org/2/library/subprocess.html. [Online; accessed 14-March-2016].

[8] The jQuery Foundation. *jQuery*. https://jquery.com//. [Online; accessed 18-April-2017].

[9] Ariya Hidayat. *PhantomJS*. http://phantomjs.org/. [Online; accessed 18-February-2017].

[10] Chris Hoffman. *How to Install the Lightweight LXDE Desktop on Ubuntu*. https://www.howtogeek.com/107368/how-to-install-the-lightweight-lxde-desktop-on-ubuntu/.

[11] *LXDE*. https://help.ubuntu.com/community/LXDE.

[12] Vinay Sajip. *Logging HOWTO*. https://docs.python.org/3/howto/logging.html. [Online; accessed 18-April-2017].

[13] Vitaly Slobodin. *[Announcement] Stepping down as maintainer*. https://groups.google.com/forum/!topic/phantomjs/9aI5d-LDuNE. [Online; accessed 18-April-2017].

[14] Herbert Van de Sompel, Michael Nelson, and Robert Sanderson. *HTTP Framework for Time-Based Access to Resource States – Memento*. http://mementoweb.org/guide/rfc. [Online; accessed 4-April-2016].

[15] Herbert Van de Sompel et al. "Memento: Time Travel for the Web". In: *CoRR* abs/0911.1112 (2009). URL: http://arxiv.org/abs/0911.1112.

[16] Tsvetomir Tsonev. *How do I check if an element is hidden in jQuery?* `http://stackoverflow.com/questions/178325/how-do-i-check-if-an-element-is-hidden-in-jquery/178450`. [Online; accessed 16-March-2016].