AD EXTRACTION TOOL

A tool which identifies and extracts the contents of both image and textual advertisements present in a given web-page. The main objective of this tool is to identify the ads being shown to the users and analyse the content of the advertisements. During a user study[1] conducted at Bell Labs, India it was found that the users were quite concerned about being shown advertisements with embarrassing content which included adult sites ad, dating and matrimony ads. We wanted to know that how many such advertisements were actually being shown to the users. Also we wanted to identify the common features among these embarrassing ads to build a filter list which could be used to block them.

Tools Used

In order to perform the extraction of ads, the following tools and algorithms were used:

- Selenium Web driver- It is a browser automation tool which automates web application for testing purposes and allows running automated scripts to perform various tasks. The Firefox web driver was used which automates the Firefox browser. Using the web driver, the web-page was loaded and was parsed to retrieve the advertisements present in that web-page. The advertisements which were inside iframes loaded using JavaScript which took some time to load and therefore slowed down the execution. (https://code.google.com/p/selenium/wiki/FirefoxDriver)
- Jsoup Parser- In order to parse the HTML content fetched by the Selenium web driver, Jsoup parser was used. It is a java library which parses the HTML content and allows extracting and manipulating data using DOM, jquery-like methods. (http://jsoup.org/)
- Aho-Corasick string matching algorithm was used for comparing the URLs with a list of ad-keywords to check if a given link was an advertisement or not.
- Also the Easylist's list of advertisement filters which is also used by many Ad-block plus users was used to identify links which were ads from all the links. Two lists were used for the same.
  - General advertising keyword list- It contains a list of general advertising keywords. It was used to see if a given link is an ad or not.
  - Third-party network list- It contains a list of thirdparties which are involved in delivering advertisements across the web. This list was used to check if the advertisement was from a third-party network or not.

[1] "Re-Examining User Concerns for Online Tracking and Advertising", SOUPS 2013, July 24–26, 2013, Newcastle, UK.
Lalit Agarwal, Nisheeth Shrivastava, Sharad Jaiswal, Saurabh Panjwani, Bell Labs Research, Bangalore, India

Extracting the advertisements

As there could be many links inside a HTML page, in order to identify the links which are ads, the URLs are compared with a list of ad-keywords to identify if a particular link is an ad or not. We used the list of filters from the Easylist subscription which is also used by most of the Adblock Plus users. In order to compare all the links with the list of filters, Aho-Corasick string matching algorithm is used. These are the steps involved:-

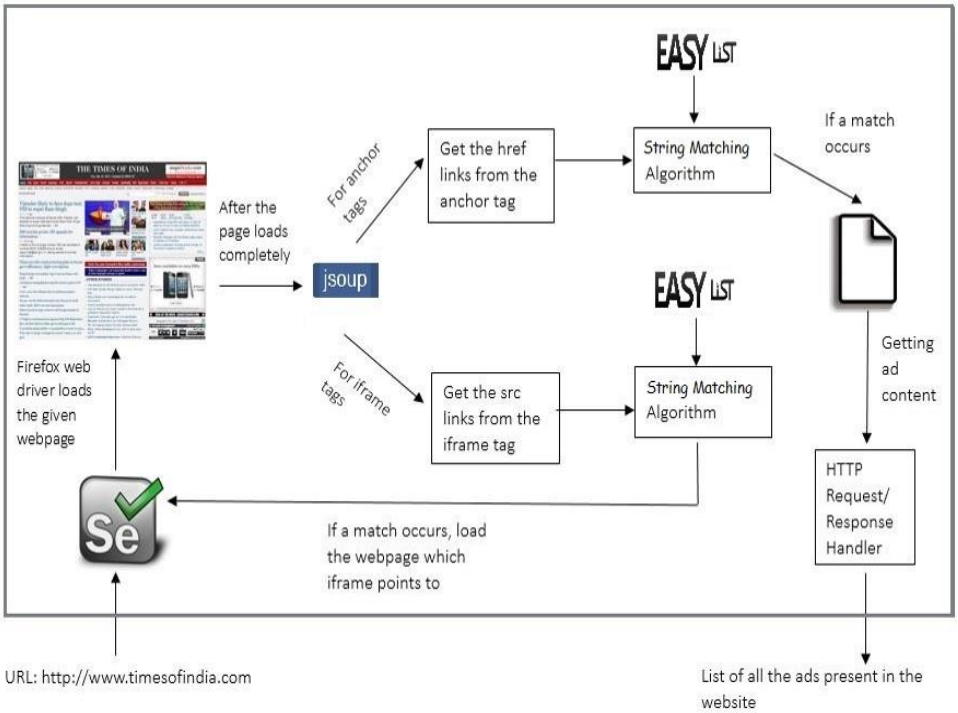
1) Using Selenium's Firefox web driver, the web-page from which the ads are to be extracted is fetched. The Selenium web driver automates the Firefox browser i.e. creates an instance of the browser and waits for the page to load and allows JavaScript to execute if required.

2) Once the page loads completely, the HTML content of the page is fetched from the browser and is parsed using Jsoup to look for all the links present in the webpage.

3) For ads outside iframes, once the page completely loads, we parse the HTML content of the page using Jsoup to look for anchor tags. For each anchor tag, it compares the href link in that anchor tag with the list of advertisement filters to see if that link is an ad or not. If it is an ad, it stores the link in a file.

4) For ads inside iframes, once the page loads the content of iframes, the web driver identifies all the iframes in the HTML page and compares the source links of the iframes to the advertisement filter list using the AhoCorasick string matching algorithm. If the string match occurs, the iframe content is parsed to look for anchor tags and then the links in these anchor tags are compared with a list of ad filters. If the string match occurs again, it is considered to be an ad and the link is stored in a file.

5) In order to get the content of these ads, we process HTTP get requests on the links collected and extract the meta content, title of the landing web-pages from the HTTP response to identify the content of these advertisements.



Methodology

We ran the ad-extraction for around 5,000 web pages and looked for advertisements in these web-pages. The data used for this part included the web browser history of the participants which were collected during the user study. We were able to get around 50,000 URLs from 28 users. We randomly selected 3 sets of URLs from these 50,000 URLs each containing 500, 2500 and 5,000 URLs respectively and ran our test on these URLs to look for embarrassing ads. Multithreading was used to run five Selenium Firefox driver instances simultaneously for faster execution. The tests resulted in an excel file containing the following information for each ad- Ad-Title- The ad title displayed for text ads

- Ad-Content- The ad content displayed for text ads
- Ad-Display URL- The ad URL displayed for text ads
- Ad-Src URL- Source URL of the ad, extracted from the HTML code of the ad
- LPTitle- The title of the landing page of the ad
- LPURL- The URL of the landing page of the ad
- URL- The URL of the web-page on which this ad was present
- ImgSrc- Source address of the image, if it was an image ad

Apart from the excel file we also downloaded the HTML pages and images of these ads for doing further analysis. The content of these excel files were scanned to identify embarrassing textual ads and a separate list of such ads was created using the excel files. For image ads, we individually browsed through the images which we downloaded to see if they could be embarrassing. This resulted in a list of embarrassing ads present in those pages which we used to analyze and build a filter for blocking embarrassing and sensitive ads.

Results

We identified embarrassing/sensitive ads from the ad data we collected by running the ads-extractor tool. We considered an ad sensitive if it belonged to any of the following categories- dating, matrimony, nightwear and adult sites. We were able to extract 4014 text and 2321 image ads out of which, we found 73 (2%) embarrassing text ads and 51 (2%) embarrassing image ads. The majority of the embarrassing ads were either dating ads or matrimony ads. We also looked at their source URLs, landing page content to build a list of filters to block such ads.

Limitations of the tool

Limited tools are available on the internet which automates a browser and allows iframes to load using JavaScript. We tested some headless browsers tools but they failed to give appropriate results. Due to the absence of any another alternative, we had to use Selenium's Firefox web driver. Since each instance of the web driver opened a new Firefox window, it slowed down the process when working with multiple URLs.

Also since targeted ads are usually loaded using JavaScript, the Firefox driver waited for the entire page to load and then only parsed the HTML page which added to the delay. Also the tool failed to

extract flash ads because since they are embedded objects, it is not easy to identify if they contain an ad or not.

Acknowledgements

I would like to thank Dr. Saurabh Panjwani, Dr. Sharad Jaiswal and Dr. Nisheeth Shrivastava (Bell Labs, India) for their constant feedback. The tool would not have been possible without their guidance and support.

References

[1] F. Roesner, T. Kohno, and D. Wetherall. Detecting and Defending Against Third-Party Tracking on the Web. In Proc. of NSDI, 2012

[2] J. Mayer and J. Mitchell. Third-party web tracking: Policy and technology. In Proc. of IEEE Symposium on Security and Privacy, 2012.

[3] B. Ur, P. G. Leon, L. F. Cranor, R. Shay, and Y. Wang. Smart, useful, scary, creepy: Perceptions of online behavioral advertising. In Proc. SOUPS, 2012.