Computer Science Senior Project

Cristian Perez



Computer Science Department, Saint Martin’s University, Lacey WA

# Abstract/Summary

This paper will explore a computer science senior project designed with the primary objective to enable the detection of fraudulent activity in the Bing Search Ad Network. More specifically, to help identify advertiser accounts created by cybercriminals, that are used to conduct some type of scam or other activity that could put the end user at risk. The design required the consideration of tactics and procedures employed by the different types of scammers to avoid detection.

In order to obtain the most accurate results, the project should be designed to simulate a real user, conducting a search query and clicking the ad. It was concluded the straightforward way to accomplish this was with a crawler, but using a real browser. The crawler should also be able to scrape ad information, follow the ad to its final product webpage, take a screenshot, and finally save the html source code. This data can be use to detect fraudulent accounts, and in the longer term, be used to build models for machine learning detection.

# Introduction

Web requests vs browsers

Search advertising, also known under the terms: search-engine marketing, pay-per-click marketing, amongst others, is a business model in which search engines (e.g Google or Bing) incorporate ads into the Search Engine Results Page (SERP). The ads displayed are generally relevant to the search performed by the user. For example, if a user searches for the term “shoes”, the ads displayed should be for products or services associated with shoes. To determine what ads are displayed when a specific term is searched, a bidding system is used. Essentially the highest bidder gets his ads placed in the top spot, second highest in the second spot, and so forth.

The problem with search advertising, and online advertising in general, is that its business model can be exploited by cybercriminals to conduct fraudulent scams among other illicit activity. Essentially, a cybercriminal can create and advertiser account, place a high bid on a common search term, have his ad displayed in the top results, but instead when the ad is clicked – the user is redirected to a scam page. Search ad companies such as Bing and Google, implement proactive countermeasures to detect and prevent this type of ad to ever reaching the user. However, it is impossible to a hundred percent of fraudulent accounts in a proactive manner due to the volume of ads that are uploaded on a daily basis to the advertising ecosystem.

Until only a few months prior, I worked for the Bing Ads Traffic Quality team where one of my primary duties was identifying the fraudulent accounts that make it past our proactive detection system. The conclusion was reached that the best approach to detect these types of accounts is to write a script or program capable of simulating a real user experience. It should automate search queries, retrieve the ad elements from the SERP, follow to the ad’s landing page, and take a screenshot and download the source html code. This information would be reviewed by Traffic Quality Engineers and help determined whether an account is indeed fraudulent. The reason this design was chosen, is because it seemed logical that simulating the environment of the end user would produce accurate results, even at a small-scale level.

It is important to note that this goal of this project is to provide an additional tool in the Traffic Quality arsenal, but primarily to serve as a proof of concept for further development. The basic scope or functionality requirement for this project is as follows:

* User provides a list of terms to be used to perform search queries, this list is likely to be a compiled list of terms that have been associated with fraudulent advertisers at any given timeframe.
* Program performs a search query for every search term provided
* Program identifies the ad results and stores associated data
* Program crawls to ad landing page website, saves screenshot of landing page and html source code
* Results from scans are saved in the file system, no database involved

# Literature Review

The concept of web crawlers and web scrapers are not a new concept, they have been around for some time. The main reason for the use of web scrapers and crawlers is to collected information where an API alternative is not an option. As a result, most crawlers are designed for a specific purpose, and although they may contain similar features, it is unlikely that a scraper exists for our specific needs. For example, my design to crawl and collect ad information from Bing search results, is not likely to be useful in other scenarios.

On the other hand, there is various tools including frameworks designed specifically for crawling and or scraping content from the internet. For example, Scrapy (<https://scrapy.org/>), which is a perhaps the most powerful framework available for creating a web crawler, however, I opted not to use it in my design due to its high level of complexity, for a python beginner like myself. Furthermore, remaining alternatives to our project would have required commercial alternatives, which due to concerns with data security were not considered, but also it’s likely they wouldn’t have met our specific functionality needs.

# Methodology and Design

## Requirements

### Functional Requirements

**Inputs and Outputs**. The user provides two inputs, the browser client to be used for the scan, and the directory to the .txt file containing the list of search terms. Once these parameters are provided, the program proceeds to automatically perform a search query for every search term in the .txt file. The procedure is as follows:

1. The program initiates the browser selected by the user
2. Navigates to the Bing homepage
3. Performs the first query
4. Once SERP is loaded – Selects all ad data, including URL
5. The last 3 steps are repeated for every search term
6. Then program retrieves ad URL from URLs collected
7. Wait for page load. Takes screenshot, saves html source
8. Files are saved in File System. No Database implementation.

**Database Requirements.** The current version of the program does not have any database requirements – all information is stored in the file system. However, if the Traffic Quality team continues the development of this concept, it is likely that a database will be implemented into future versions to enhance data management and organization.

**Communications:** The program requires a network communication for proper functionality.

### Non-functional Requirements

* There were several non-functional constraints encouraged throughout the project’s evolvement. For one, I had never programmed in Python, nor had I programmed in any language for almost 2 years the nature of my role just didn’t require it. Learning all of the different concepts: scrapers, crawlers, parsing and storing data, python to the point where I felt ready start implementing was more time consuming and frustrating than anticipated.

### Use Cases

The primary and likely only actors and users will be the Engineers from the Bing Ads Traffic Quality Team. They were will the only ones interacting with the program. Their interaction with the system will be to provide the list of terms of terms to be queried, and analyze the results.

* **Users**: The primary and likely only users of this system will be members of the Bing Ads Traffic Quality Team. These are individuals with sufficient technical expertise to analyze the underlying code for the following scenarios: a bug is encountered and needs to be fixed, code needs to be enhanced or improved, or new features need to be added. However, once the program deemed to be sufficiently stable/reliable it is possible that supporting vendor teams will make use of it.
* **User and System Interaction:** The user interaction with the program is as follows:
  1. User executes the python program
  2. Program prompts the user to select the web client (browser) to use for scan
  3. User selects web client
  4. The program prompts the user for the text file containing terms to be searched
  5. Program begins executing queries (if no issues are encountered)
  6. Upon completion, user reviews the results of scan for suspicious advertisers

## User Interface

The user interface of the system is a shell prompt. This can be observed in Appendix A.

## Database

My program design does not incorporate a database; data is stored in the file system.

# Implementation

## How to implement Design

To implement this design, or a crawler design in general, basic programming concepts are needed. In addition, it is important to get familiarized with the network request libraries of the language you choose for your design. In my case I chose python, but crawling designs are language agnostic, for me python was just a preference. One more important thing to understand or have knowledge of when designing a crawler or scraper is CSS/XPATH selectors.

### Development methods

My initial development method was an iterative approach. I chose to take this approach because it seems the easiest for my limited programming skills. The initial code contained more for loops than you can imagine, and it was extremely difficult to follow – but it ran. Once I was okay with the program’s functionality, I proceeded to refactor the code into functions using a top-down design. It good some debugging at times but I was able to refactor the original code into separate functions that made the code much easier to follow and understand.

### Tools

* **Selenium**: Selenium is a framework designed for the automation of web browsers. It is primarily used to automate testing of web applications, but can be used for practically any purpose involving browser automation including mobile browsers. Versions of Selenium are available in the following programming languages: Java, C#, Python, Ruby, PHP, Perl and JavaScript.
* **Selenium WebDriver:** The Selenium WebDriver is the component of the Selenium framework that integrates directly with the web browser to provide the automated instructions. Each web browser has its own unique WebDriver; the corresponding driver must be used in order for Selenium to function properly. For example, a WebDriver for the Chrome browser cannot be used to automate tasks in Internet Explorer.

### Programming Language

* **Python:** Python was selected as the programming language for this project for the following reasons:
  + Python a simple and straightforward language, its syntax is clean and easy to read compared to other languages with more explicit syntax.
  + Despite being generally considered a programming language for beginners, Python is extremely powerful - thanks to the high number of libraries available, including libraries and frameworks specifically useful for this project.
  + Python is hardware agnostic and can be run in any environment.

## Solved Problems Encountered

* Selecting correct html elements

### Development Method

### Test Plan

Due to the nature of my project, I was only able to conduct limited functionality tests to ensure program functionality. I was not able to run more prolonged tests to gage the programs detection capability for the following reasons:

* The Traffic Quality team contains the search terms relevant to current fraud modes of operation
* Crawling ads from my home network will cost the advertiser, even though the click in invalid
* It’s against Bing’s Policy

### Present Test Results

The Traffic Quality team began conducting daily scans about to weeks ago, and have successfully identified and closed multiple fraudulent accounts. Nature and number of accounts were not disclosed.

## Completion Process

|  |  |
| --- | --- |
| Week 1 | Research Methodology |
| Week 2 | Research Tools and Programming |
| Week 3 | Learn Python |
| Week 4 | Learn Python |
| Week 5 | Learn Selenium Concepts |
| Week 6 | Learn Crawler and Scraping Concepts |
| Week 7 | Design |
| Week 8 | Design |
| Week 9 | Start coding |
| Week 10 | Implement Crawl Feature |
| Week 11 | Implement Scraping Feature |
| Week 12 | Implement Storing Feature |
| Week 13 | Complete working Protype |
| Week 14 | Refactor Code and Share with Team |
| Week 15 | Begin Documentation |
| Week 16 | Complete Documentation and Report |

# References

Wlosik, M. (2018, April 09). What is Search Advertising and How Does it Work? - Clearcode Blog. Retrieved May 6, 2019, from <https://clearcode.cc/blog/what-is-search-advertising/>

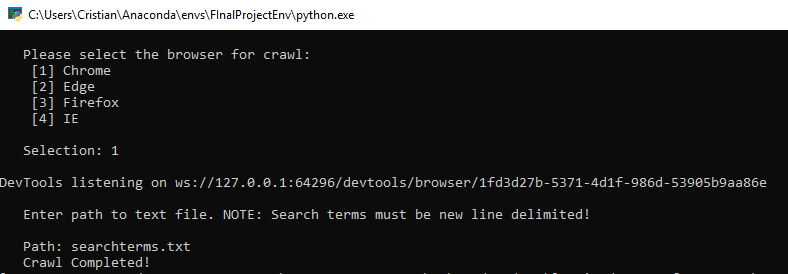
In text Citation (Wlosik, 2018)

# Appendices

Appendix A

Program Walkthrough

User Interface



Results

