2.WEB SCRAPING AND ITS SCOPE

Web scraping is the process of collecting structured web data in an automated fashion. It’s also called web data extraction. Some of the [main use cases of web scraping](https://www.zyte.com/learn/what-is-web-scraping-used-for-learn/what-is-web-scraping-used-for/) include price monitoring, [price intelligence](https://www.zyte.com/learn/price-intelligence/), news monitoring, [lead generation](https://www.zyte.com/learn/lead-generation/), and [market research](https://www.zyte.com/learn/market-research/) among many others.

In general, web data extraction is used by people and businesses who want to make use of the vast amount of publicly available web data to make smarter decisions.

If you’ve ever copied and pasted information from a website, you’ve performed the same function as any web scraper, only on a microscopic, manual scale. Unlike the mundane, mind-numbing process of manually extracting data, web scraping uses intelligent automation to retrieve hundreds, millions, or even billions of data points from the internet’s seemingly endless frontier.

SCOPE

Web scraping allows you to acquire non-tabular or poorly structured data from websites and convert it into a usable, structured format, such as a .csv file or spreadsheet.

Scraping is about more than just acquiring data: it can also help you archive data and track changes to data online.

UNDERSTANDING THE PROJECT

**Web scraping**, **web harvesting**, or **web data extraction** is [data scraping](https://en.wikipedia.org/wiki/Data_scraping) used for [extracting data](https://en.wikipedia.org/wiki/Data_extraction) from [websites](https://en.wikipedia.org/wiki/Website). The web scraping software may directly access the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web) using the [Hypertext Transfer Protocol](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) or a web browser. While web scraping can be done manually by a software user, the term typically refers to automated processes implemented using a [bot](https://en.wikipedia.org/wiki/Internet_bot" \o "Internet bot) or [web crawler](https://en.wikipedia.org/wiki/Web_crawler). It is a form of copying in which specific data is gathered and copied from the web, typically into a central local [database](https://en.wikipedia.org/wiki/Database) or spreadsheet, for later [retrieval](https://en.wikipedia.org/wiki/Data_retrieval) or [analysis](https://en.wikipedia.org/wiki/Data_analysis).

Web scraping a web page involves fetching it and extracting from it. Fetching is the downloading of a page (which a browser does when a user views a page). Therefore, web crawling is a main component of web scraping, to fetch pages for later processing. Once fetched, then extraction can take place. The content of a page may be [parsed](https://en.wikipedia.org/wiki/Parsing), searched, reformatted, its data copied into a spreadsheet or loaded into a database. Web scrapers typically take something out of a page, to make use of it for another purpose somewhere else. An example would be to find and copy names and telephone numbers, or companies and their URLs, or e-mail addresses to a list (contact scraping).

Web scraping is used for [contact scraping](https://en.wikipedia.org/wiki/Contact_scraping), and as a component of applications used for [web indexing](https://en.wikipedia.org/wiki/Web_indexing), [web mining](https://en.wikipedia.org/wiki/Web_mining) and [data mining](https://en.wikipedia.org/wiki/Data_mining), online price change monitoring and [price comparison](https://en.wikipedia.org/wiki/Comparison_shopping_website), product review scraping (to watch the competition), gathering real estate listings, weather data monitoring, [website change detection](https://en.wikipedia.org/wiki/Change_detection_and_notification), research, tracking online presence and reputation, [web mashup](https://en.wikipedia.org/wiki/Web_mashup), and [web data integration](https://en.wikipedia.org/wiki/Web_data_integration).

[Web pages](https://en.wikipedia.org/wiki/Web_page) are built using text-based mark-up languages ([HTML](https://en.wikipedia.org/wiki/HTML) and [XHTML](https://en.wikipedia.org/wiki/XHTML)), and frequently contain a wealth of useful data in text form. However, most web pages are designed for human [end-users](https://en.wikipedia.org/wiki/End-user_(computer_science)) and not for ease of automated use. As a result, specialized tools and software have been developed to facilitate the scraping of web pages.

Newer forms of web scraping involve monitoring data feeds from web servers. For example, [JSON](https://en.wikipedia.org/wiki/JSON) is commonly used as a transport storage mechanism between the client and the web server.

There are methods that some websites use to prevent web scraping, such as detecting and disallowing bots from crawling (viewing) their pages. In response, there are web scraping systems that rely on using techniques in [DOM](https://en.wikipedia.org/wiki/Document_Object_Model) parsing, [computer vision](https://en.wikipedia.org/wiki/Computer_vision) and [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing) to simulate human browsing to enable gathering web page content for offline parsing.

REASON FOR CHOOSING THIS PROJECT

Web scraping is integral to the process because it allows quick and efficient extraction of data in the form of news from different sources.

Such data can then be processed in order to glean insights as required. As a result, it also makes it possible to keep track of the brand and reputation of a company.

So, we chose web scraping.

THE CHALLEGE WE FACE WHEN WE USE WEB SCRAPING

**Bot access**

The most popular scraping internet data challenge. Websites can decide whether they will give bots access to clean data. Some sites forbid automatic data collection. The reasons for the ban can be completely different. If you come across a website that prohibits collection through its robots.txt, follow fair play principles and ask the site owner for permission to collect data. Otherwise, it is better to look for an alternative site with similar information.

**IP blocking**

IP blocking is one of the rare methods of dealing with parsers. But it is also the easiest way. Blocking is triggered when the server detects a large number of requests from the same IP address or when a search robot makes several parallel requests. There is also IP blocking via geolocation. This is when the site is protected from attempts to collect data from specific locations. The website will either ban the IP completely or limit its access.

**Captcha**

Also one of the rare and difficult types of scraping challenges. CAPTCHA allows distinguishing a person from a robot. Logical tasks or input of characters are displayed for verification, which humans solve quickly and robots do not. Indeed, many CAPTCHA solvers are now implemented in bots for continuous data collection, although it slows down the process a bit.

**Honeypot traps**

Website owners put honeypot traps on pages to catch parsers. Traps can be links that ordinary people can’t see, but parsers can. When a parser falls into a trap, the website can use the information it receives to block bots. Some of the traps have a CSS style of “display: none” or a masked color to match the background color of the page.

**Slow or unstable load speed**

Websites may be slow to load content or may not load at all when receiving a large number of access requests. In such a situation, you can refresh the page and wait for the site to recover. However, the parser will not know how to handle such a situation and data collection may be interrupted.

**Web page structure**

Another website challenge you have to face when scraping. Designers may have their design standards when creating web pages, so page structures will vary. Websites also undergo periodic changes to improve user interaction or add new features. This often results in structural changes to the web page itself. Web parsers are created with page code elements in mind, so these changes make the codes more complex, which affects how the parsers work.

And because they are customized to a specific page design, they won’t work for the updated page. Sometimes even a minor change requires a new parser configuration.

**Login requirement**

Sometimes you have to log in first to get information. After you send your login credentials, the browser adds a cookie value to several requests that run on other sites. That way, the website knows that you are the same person who just logged in earlier.

However, the login requirement is not a difficulty, but rather one of the stages of data collection. So when collecting data from websites, you need to make sure that cookies are sent with the requests.

**Real-time data scraping**

There are a myriad of instances where real-time data collection is important, such as price comparisons, inventory tracking, etc. Data can change instantly and generate huge revenue for companies. That’s why a parser needs to monitor sites and collect data around the clock. However, because parsers constantly monitor web pages, it always takes some time to query and provide data and any instability can lead to failures.

**Data from multiple sources**

Sometimes the information can be on different sources. Some of the data will be on the website, some will be in the mobile app, and some will be in PDF format. The scraper works in such a way that it collects all the information from one service. In such a case, it is difficult to collect and group the information, some of it may be missing altogether. Plus it takes a lot of time.

**Dynamic content**

A minor problem, but worth mentioning. Websites use AJAX to update dynamic web content. For example, delayed loading of images or infinite scrolling and displaying additional information by clicking a button when AJAX is invoked. This is a convenient way for users to view more data on websites, but not for parsers.

APPROACH FOR CHOOSEN PROBLEM

## How to avoid blocking

### Beware of honeypot traps

Before you start collecting data, make sure that the link is set to the CSS properties “display: none” or “visibility: hidden”. If a link has one of these properties, avoid it.

It is also advisable to only follow links from reliable sources. While this does not give a complete guarantee, it will still allow you to better judge the security of the sites.

### Use a headless browser

A headless browser includes all the features of a website display. Because of the lack of a graphical interface, a command-line utility is used to interact with the headless browser. They are more flexible, faster than real browsers.  And since there’s no overhead for any user interface, these browsers are suitable for automatically stress-testing and cleaning web pages. And when using such a browser, you do not need to load the entire site. It can load the HTML part and collect the data.

### Use captcha solving services

Captcha exists in many forms, but the point is the same – you need to solve a task to prove you’re human. And Captcha solvers automatically help you solve Captcha and improve your workflow. All you have to do is register, buy units, implement their Captcha submission API, and return the result as text.

There are two ways to solve such tasks. The first method is the Captcha solution services hire people to whom they send tasks to solve and forward the result. The second method is Optical Character Recognition (OCR). Artificial intelligence and Machine Learning determine Captcha content and its solution automatically.

### Use Proxy Servers

A proxy server is an intermediate server between a user and a website. It has its own IP address. When a user requests access to a website through a proxy server, the website sends and receives data to the IP address of the proxy server, which forwards it to the user. Web parsers use proxies to make the traffic look like normal user traffic.

So, to avoid blocking the parser on the site, you can buy IP address pools and distribute them at randomly scheduled intervals. Using proxies is the easiest way to distribute them. These programs route requests through different IP addresses, masking the real IP address.

### Detect website changes

Before collecting data, it is best to run a thorough website test to avoid problems. Detect any changes and program the parser so that it does not stop working in the changed site structure.

### Set a real User Agent

The User Agent (UA) is a string in the request header that specifies the browser and operating system for the web server. If your user agent does not belong to one of the major browsers, some sites will block its requests. To avoid this problem, you need to specify the UA used for the parser. It is better to use your browser’s user agent since browser behavior is more likely to match the expectations of the user agent if you don’t change it too often.

### Set random intervals in between your requests

More like an additional tip, not a solution to the problem. Do not overload the site with a large number of requests. It is better to make a time delay of a few seconds between requests. You can also save the URLs of the scanned pages so that you don’t have to come back to them again.

### Respect the Robots.txt

Also advice, but for those who browse websites before collecting data. The robots.txt file contains information about which pages the web parser can crawl and which pages it cannot crawl. Good bots respect these rules and follow scanning and data collection methods. Be sure to check robots.txt before parsing. If the site is completely blocked by bots, it’s best to leave the site.

### Don’t scrape during peak hours

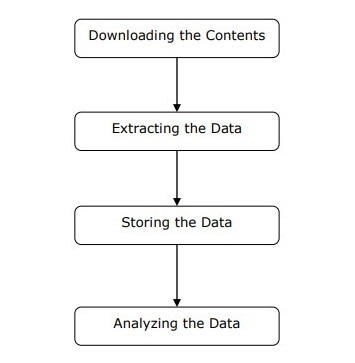
It is better to collect data from sites during off-peak periods, so as not to interfere with the site work. It also plays a big role for the parser itself, as it will significantly increase the speed of data collection.

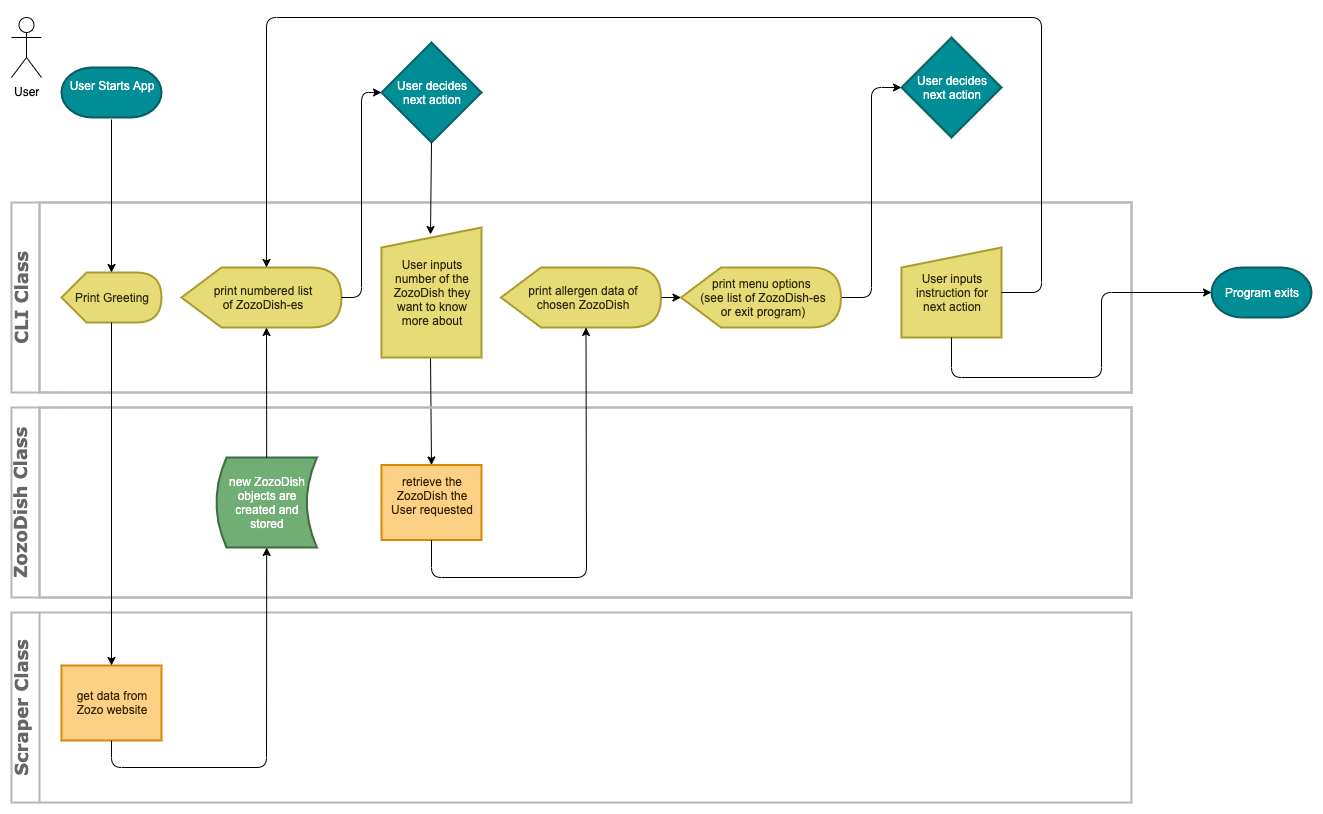
## Dealing with neural network-based site protection

Some sites are very concerned about their data and use different protections based on neural networks. These protections use any data about the user that can be obtained from the browser. This is how a neural network determines whether a user is a real or a bot.

In a nutshell, we solve such problems by analyzing the script that collects user data to understand what kind of information the neural network gathers. We determine the mechanisms for identifying the automated software. Then we develop a script to replace the data. After the data is replaced the access to the site appears. Now you can collect any information you want.

DIAGRAM/FLOW CHART FOR THE PROJECT





CODING LANGUAGE

We have used python language used in this project.

HOSTING LANGUAGE

EXTERNAL TOOLS

TEAMS ABILITY TO IMPLEMENT WINNING SOLUTION

Background of team members:

Major Expertise of team members:

Roles and responsibilities of team members:

Previous projects undertaken:

Team strengths:

Team achievements:

Personal motivation:   
  
  
 I want to be an entrepreneur. By now i just want to shine in cybersecurity and in the IT field.

And by this project I have a better vision about cybersecurity through this project.