Web Scraping For GPU Price Research

SET11104

J. Burton | 40278490

40278490@live.napier.ac.uk

Keywords: *Web Scraping, Price Research, Automated Data Collection, Python*

Abstract – One quarter of internet traffic is made up of automated bots that scrape information from websites. Web scraping offers an alternative method of data collection for price research that circumvents the issues presented in data taken from official institutions, for example, costly fees and lack of insight to data collection methods. Each web scraper bot is unique and bespoke to its purpose, thus, there is no universal web scraper that can collect information from everywhere. Fortunately, the steps for designing and implementing a web scraper bot are standardised: choose info source, define data to be collected, locate its position on a page, collect the data with a programmed bot, automate the execution of the web scraper bot and analyse the scraped information. These steps have been utilised in this paper to collect information on graphics cards to demonstrate the use of web scraping as a data collection method for price research. The developed prototype succeeds in fulfilling this aim, however, still has some limitations such as holes in scraped data and only covering surface level analysis.

1 Introduction

Data analysis allows people to further understand the world around them regardless of application field. From house market prices predictions to sports team performance statistics its potential uses are endless. To precisely identify and verify patterns and trends, clean and accurate data must first be collected and then organised. Therefore, from the onset, data must be collected from the correct sources without tampering or influence.

This paper is interested in the analysis of online stores price data. Traditionally, price researchers rely on costly, third-party, official sources like transaction data or official price indices to gather usable data. Due to the lack of insight to the data collection process, it is necessary for the researcher to assume that this information is clean and reliable (Hillen, 2019). Manual data collection is another limitation of conventional practices as it is a very time-consuming process, especially when a high volume of data needs to be collected (DeVito et al., 2020). An alternative to these methods is through web scraping, the automated process of taking information from online sources, which allows for the mass collection of information without the high associated costs (Hillen, 2019; Macapinlac, 2019).

The aim of this paper is to provide an insight into the use of web scraping for the purpose of data collection for price research. In the next section, web scraping as a technology will be explored, along with its possible advantages and limitations. This will be followed by the description of steps to develop a practical application of a web scraper for data collection. Finally an insight to the limitations of the developed project along with future development suggestions will be presented.

2 Web Scraping

Web scraping is the process of collecting specific information from online sources through automated bots that impersonate a user (Hillen, 2019). While the idea of using web scraping for price research purposes is recent, the technology has been utilised to monitor a country's economic state[1]. Additionally, some research can only be completed through data collected by web scrapers or through slow, manual data collection methods (Xu et al., 2020).

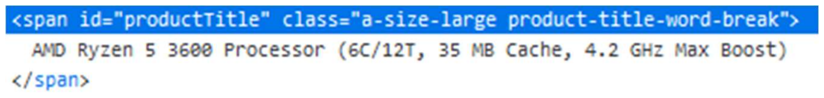
Automated bots account for approximately a quarter of all web traffic; mostly for commercial purposes (Macapinlac, 2019; Patel, 2018). The bots are used online for a variety of tasks including e-commerce, job boards and marketing. This is due to these fields requiring vast amounts of online data to drive their strategies and provide real-time information (Patel, 2018).

The procedure for creating and using a web scraper is as follows:

1) Choose Online Data Location – e.g. Amazon, eBay, NVIDIA, other online retailers.
2) Interpret data to be collected – e.g. Titles of items, prices, etc.
3) Collect information via bot – Usually written in R or Python.
4) Store information in database or spreadsheet.
5) Automate bot – Through Crontab on Linux, Task Scheduler on Windows etc.
6) Analyse Data – Visualisation through graphs etc.

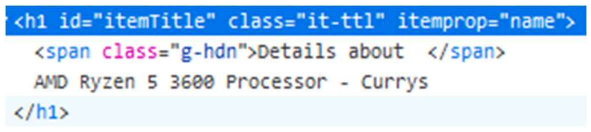
While the procedure for creating a web scraper remains the same for each application, the web scraper needs to be customised each time. This is because each online source refers to its information in a unique way, thus it is impossible to create a web scraper that can take all information from every available website. The difference in how information is referenced can be seen in the HTML for two different sites showing the same product. As seen in Figure 1 and Figure 2, the same information is referred to with both a different HTML tag (*span* and *h1*) and a different *id*.

[1] http://www.thebillionpricesproject.com/



```
<span id="productTitle" class="a-size-large product-title-word-break">
  AMD Ryzen 5 3600 Processor (6C/12T, 35 MB Cache, 4.2 GHz Max Boost)
</span>
```

Figure 1 HTML from Amazon showing the name of the Product with the ID "productTitle".



```
<h1 id="itemTitle" class="it-ttl" itemprop="name">
  <span class="g-hdn">Details about  </span>
  AMD Ryzen 5 3600 Processor - Currys
</h1>
```

Figure 2 HTML from EBAY showing the name of the product with the ID "itemTitle".

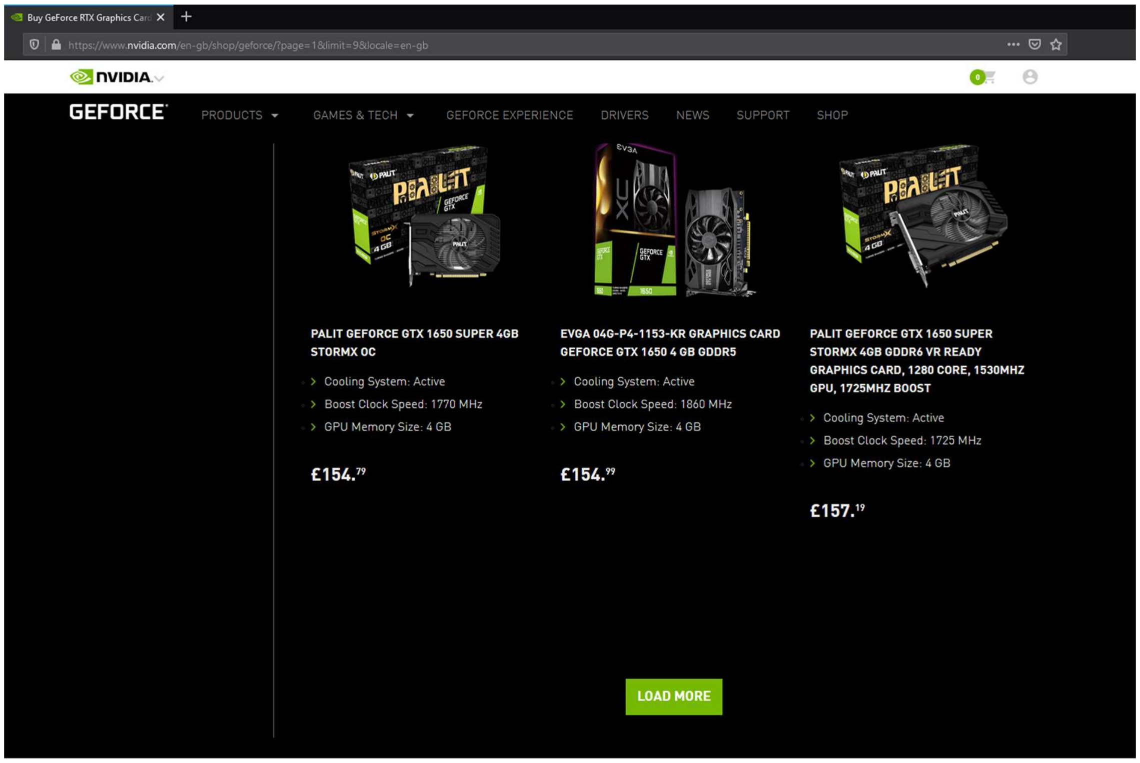
Web scraping offers many advantages over the traditional methods of data collection (Hillen, 2019):

- Low Cost – Reliable data is expensive, and the open-source nature of the web allows for access to data without cost. Furthermore, as hardware and maintenance costs are the only real investment for developing and running a web scraper the total cost of a web scraper is minimal.
- Real-time Sampling – Web Scrapers can run at any time, bypassing the wait time for a third-party to collect, clean and publish information. Furthermore, real-time sampling allows for analysis in greater detail by allowing researchers to see changes over a shorter period that may be missed in data with less sampling.
- Range – Web scrapers allow researchers to take information from many online sources, bypassing any predefined categories or definitions created by the institutions that provide traditional data sets and thus allow for the creation of data sets with customised categories.

However, web scraping also has limitations:

- Complexity – The creation of a web scraper can be relatively complex, with the researcher needing to manually search for the fields that are to be scraped within the sites HTML. This can be complicated through dynamic JavaScript elements that display information or prompt the user to *LOAD MORE* results on a page (Figure 3). Therefore, no two web scrapers are the same, as each scraper is bespoke to its function.

- Lack of historical data – One of the main advantages of traditional sources is the abundance of historical data, allowing researchers to create timelines and trends immediately after receiving data. This is a shortfall of web scraping as historical values only lead back to when the scraper script was initially executed.
- Over-abundance of information – As the access to information is seemingly limitless, there is a temptation to scrape all available information. This is further encouraged by web scraping's low cost, where researchers may be tempted to scrape all available information rather than just the information they need. Thus, researchers must have a clear scope of the information that might be required previous to implementing a web scraper (Hillen, 2019).


Buy GeForce RTX Graphics Card
https://www.nvidia.com/en-gb/shop/geforce/?page=1&limit=9&locale=en-gb
NVIDIA
GEFORCE
PRODUCTS
GAMES & TECH
GEFORCE EXPERIENCE
DRIVERS
NEWS
SUPPORT
SHOP
LOAD MORE

PALIT GEFORCE GTX 1650 SUPER 4GB STORMX OC
- Cooling System: Active
- Boost Clock Speed: 1770 MHz
- GPU Memory Size: 4 GB

£154.79

EVGA 04G-P4-1153-KR GRAPHICS CARD GEFORCE GTX 1650 4 GB GDDR5
- Cooling System: Active
- Boost Clock Speed: 1860 MHz
- GPU Memory Size: 4 GB

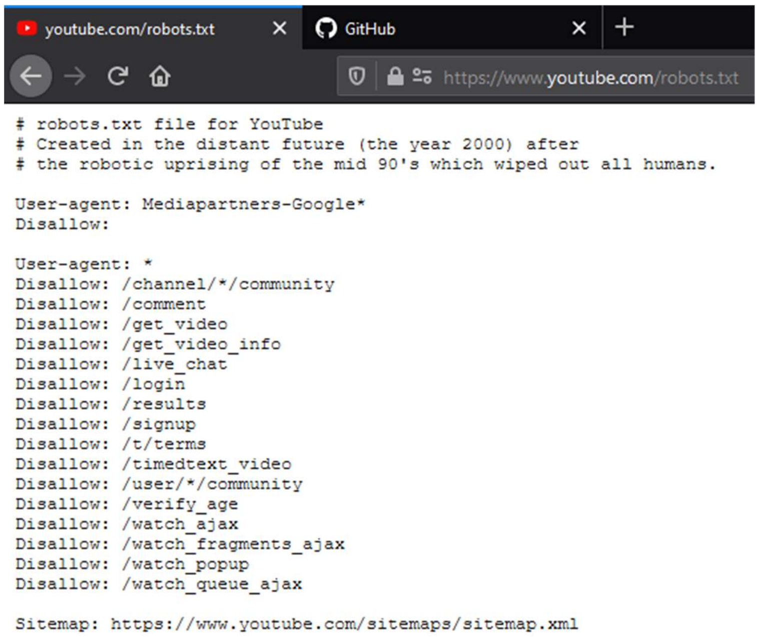
£154.99

PALIT GEFORCE GTX 1650 SUPER STORMX 4GB GDDR6 VR READY GRAPHICS CARD, 1280 CORE, 1530MHZ GPU, 1725MHZ BOOST
- Cooling System: Active
- Boost Clock Speed: 1725 MHz
- GPU Memory Size: 4 GB

£157.19

Figure 3 Screenshot of the NVIDIA web shop, showing a LOAD button to display more results.

In addition to these limitations, concerns surrounding the access of sensitive information through web scraping is still being debated both legally and ethically. This is because users are worried about others having access to potentially personal information (Hillen, 2019; Macapinlac, 2019). Furthermore, companies and website owners might not want people or bots accessing data from certain parts of their website. To counter this, most websites have a Robot Exclusions Protocol that limits where and what a bot can access on a site. This can be found by adding */robots.txt* to the URL of a website, for example, YouTube's Robots Exclusion Protocol can be found at: https://www.youtube.com/robots.txt (Figure 4). As it limits what and where a bot can interact on a website, it is important to fully understand this document as well as the information that is being collected before implementing a web scraper. This will allow the researcher to avoid any legal action that may be taken against them if they access the wrong information. Furthermore, many sites have access rights listed in their use policy document that states what a user or bot can access.



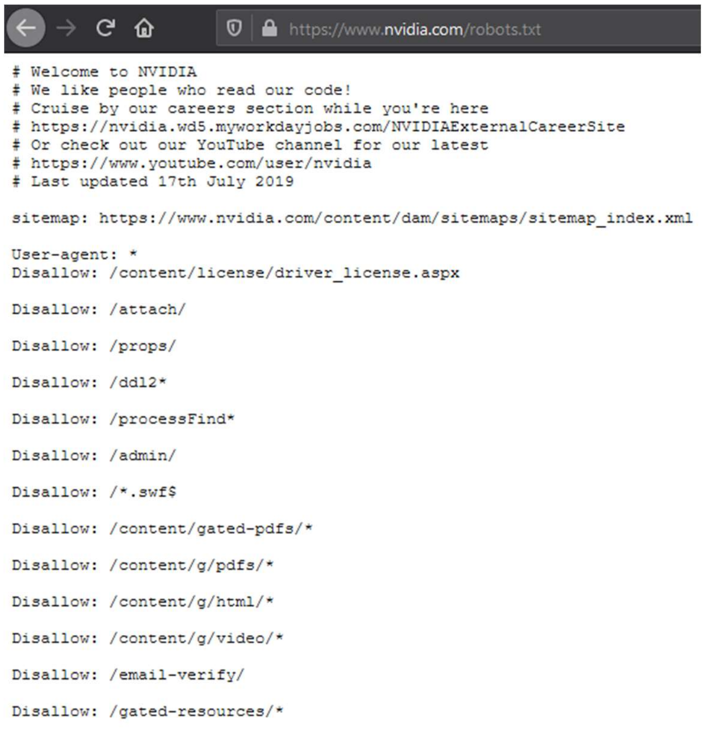
```
# robots.txt file for YouTube
# Created in the distant future (the year 2000) after
# the robotic uprising of the mid 90's which wiped out all humans.

User-agent: Mediapartners-Google*
Disallow:

User-agent: *
Disallow: /channel/*/community
Disallow: /comment
Disallow: /get_video
Disallow: /get_video_info
Disallow: /live_chat
Disallow: /login
Disallow: /results
Disallow: /signup
Disallow: /t/terms
Disallow: /timedtext_video
Disallow: /user/*/community
Disallow: /verify_age
Disallow: /watch_ajax
Disallow: /watch_fragments_ajax
Disallow: /watch_popup
Disallow: /watch_queue_ajax

Sitemap: https://www.youtube.com/sitemaps/sitemap.xml
```

Figure 4 Screenshot of youtube.com's robots.txt file or Robot Exclusion Protocol.

3 Components and Methods

There are two main components of data analysis: data collection and the analysis itself. As each web scraper is uniquely designed to collect specific data, the developed project will be split into two sub-projects: a web scraper for the collection of data and a simple web application for analysis and visualisation.

The developed project takes information on graphics cards (GPUs) from the NVIDIA web store and plots the individual prices over time. As mentioned previously, to ensure that the bot will be able to access the correct information on the website, the site's Robots Exclusion Protocol is examined, as seen in Figure 5.

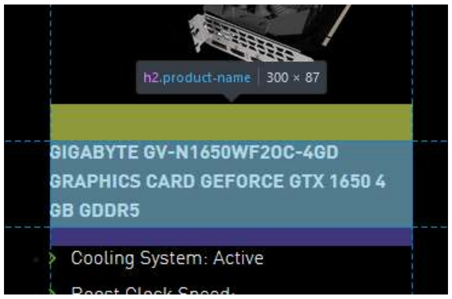


```
# Welcome to NVIDIA
# We like people who read our code!
# Cruise by our careers section while you're here
# https://nvidia.wd5.myworkdayjobs.com/NVIDIAExternalCareerSite
# Or check out our YouTube channel for our latest
# https://www.youtube.com/user/nvidia
# Last updated 17th July 2019

sitemap: https://www.nvidia.com/content/dam/sitemaps/sitemap_index.xml

User-agent: *
Disallow: /content/license/driver_license.aspx

Disallow: /attach/

Disallow: /props/

Disallow: /ddl2*

Disallow: /processFind*

Disallow: /admin/

Disallow: /*.swf$

Disallow: /content/gated-pdfs/*

Disallow: /content/g/pdfs/*

Disallow: /content/g/html/*

Disallow: /content/g/video/*

Disallow: /email-verify/

Disallow: /gated-resources/*
```

Figure 5 Screenshot of the Robots Exclusion Protocol from nvidia.com.

The */en-gb/shop/* URL needed for the scraper to collect the necessary information is not listed in this document, therefore, the URL can be scraped by an automated bot without consequence.

The next step is to identify the information that needs to be taken from the site, and this is found by examining the website's HTML. The browser[2] makes finding the name of elements easy by providing the option to display the element name when hovering over the relevant piece of information with the cursor, as can be seen in Figure 6. For example, from this we can tell that the name of the product is held in an h2 HTML tag and has the class *product-name.* This method was used to identify the relevant tags for collection: product name, product cost, type of cooling, clock speed and memory size.



h2.product-name | 300 × 87

GIGABYTE GV-N1650WF2OC-4GD GRAPHICS CARD GEFORCE GTX 1650 4 GB GDDR5

Cooling System: Active

Figure 6 HTML Tag listed above the relevant information, displaying tag type and class name. Site: NVIDIA Shop.

While there are complicated JavaScript elements on the NVIDIA shop page that allow the user to view more information (Figure 3), these can be bypassed through the URL. For example, when searching for the term *GTX 1650,* the URL: https://www.nvidia.com/en-gb/shop/geforce/?page=1&**limit=9**&locale=en-gb&search=GTX%201650 is used. In this URL, the argument *limit* (shown in bold) is passed to the site server, which indicates how many items to display on the page. Therefore, passing a value of 100 to the site could display 100 items rather than the 9 items by default. This method of increasing search space is utilised the developed web scraper to avoid programming interactions with the JavaScript elements.

Both the scraper and web application were written with Python as there is plenty of documentation on web scraping and application development with this language. Furthermore, there are specific libraries for the development of web scrapers and applications that query

[2] https://www.mozilla.org/en-GB/firefox/new/

websites for information along with libraries for data analysis. R could have equally been used for these reasons as it has been developed for data analysis purposes.

The Python libraries used for communicating with the web are *Selenium*[3] and *Beautiful Soup*[4].

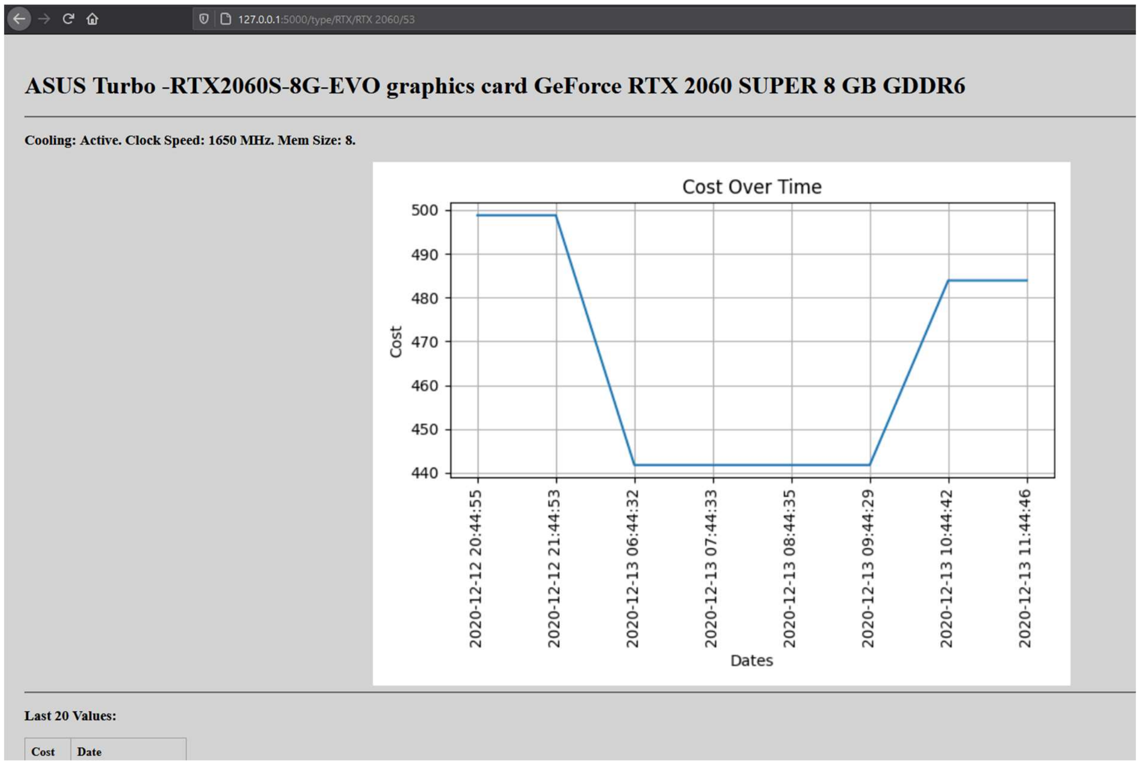
- Selenium WebDriver allows the user to open and navigate web pages through an automated browser that can either display on the user's screen or run in a headless configuration with no graphical interface. The browser driver chosen for this project was ChromeDriver for the Chrome browser, however, Firefox's geckodriver was used for some testing.
- Beautiful Soup is an HTML parser that allows the user to navigate HTML to collect data from individual HTML tags.

The scraper takes a list of search terms from a text document in the project's data folder and creates custom URLs from these values that are passed to the Web Driver so that the scraper can collect the appropriate information. For example, if the value *GTX 1660* were present in the text file, the scraper would then construct and collect information from the following URL: https://www.nvidia.com/en-gb/shop/geforce/?page=1&limit=9&locale=en-gb&search=GTX%201660&sorting=fg. It then stores the scraped data into an SQLite3 database. The database contains a table of GPUs and a table of costs related to each GPU in a one-to-many relationship. The scraped data is filtered to produce only allow products that are GPUs through a list of keywords that are associated with things like *storage* or *Ryzen* as the GPU information that is wanted does not have any *storage* and *Ryzen* is AMD's CPU line that are not GPUs. These variables would be included in laptops and prebuilt computers which are outside the scope of the scraper

[3] https://www.selenium.dev/
[4] https://www.crummy.com/software/BeautifulSoup/bs4/doc/

As the scraper has been developed in a Linux environment, the script is automated using *crontab* with the scraper running every hour. The sampling period would likely be daily or every 12 hours in a practical setting, however, hourly readings allowed for fast verification of the execution of the script.

Finally, the data has been visualised through a simple Flask[5] web application as can be seen in Figure 7, where a page for a specific GPU model and data set is displayed, showing that the price of this specific model of GPU changed twice over the sample period.


127.0.0.1:5000/type/RTX/RTX 2060/53

ASUS Turbo -RTX2060S-8G-EVO graphics card GeForce RTX 2060 SUPER 8 GB GDDR6

Cooling: Active. Clock Speed: 1650 MHz. Mem Size: 8.

Last 20 Values:

Cost	Date

Figure 7 Screenshot showing individual GPU model page in final Web App

A URL hierarchy has been created that allows the user to easily navigate between different GPU datasets and pages as can be seen in Figure 8. Starting from the homepage, the user can choose between RTX and GTX graphics cards, from here the user can choose a model of GPU,

[5] https://flask.palletsprojects.com/en/1.1.x/

where they will be presented with a table of different manufacturers versions of the GPU model, as seen in Figure 9.

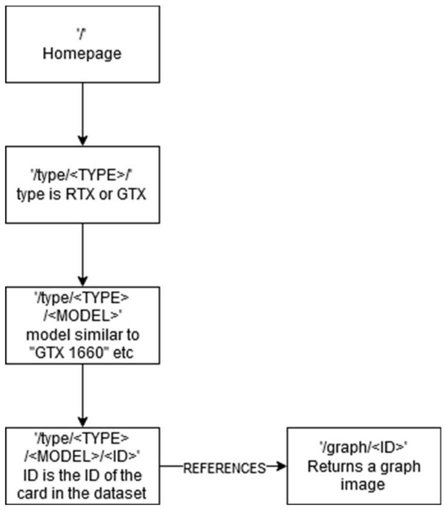
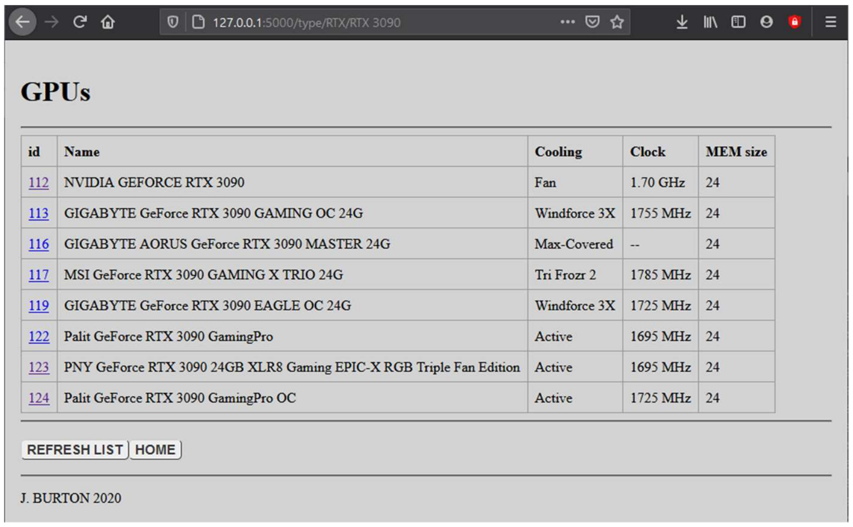


Figure 8 URL Hierarchy of web application.



GPUs

id	Name	Cooling	Clock	MEM size
112	NVIDIA GEFORCE RTX 3090	Fan	1.70 GHz	24
113	GIGABYTE GeForce RTX 3090 GAMING OC 24G	Windforce 3X	1755 MHz	24
116	GIGABYTE AORUS GeForce RTX 3090 MASTER 24G	Max-Covered	--	24
117	MSI GeForce RTX 3090 GAMING X TRIO 24G	Tri Frozr 2	1785 MHz	24
119	GIGABYTE GeForce RTX 3090 EAGLE OC 24G	Windforce 3X	1725 MHz	24
122	Palit GeForce RTX 3090 GamingPro	Active	1695 MHz	24
123	PNY GeForce RTX 3090 24GB XLR8 Gaming EPIC-X RGB Triple Fan Edition	Active	1695 MHz	24
124	Palit GeForce RTX 3090 GamingPro OC	Active	1725 MHz	24

J. BURTON 2020

Figure 9 Screenshot of the RTX 3090 model page.

4 Limitations and Future Work

While this application effectively demonstrates the practical application of web scraping for GPU data collection for the purpose of data analysis, however, there are several limitations to the developed product:

- Holes in data – one problem with the web scraper is that it does not reliably collect all information available. This leads to inconsistent graphs and records and causes some GPUs to have 2 samples listed, while others have 20 samples listed etc. This may be due to execution of the scraper on a Raspberry Pi Zero W and not a more powerful machine that can execute the script faster and more reliably. Based on this assumption, it could be valuable to compare the data collected on a more powerful system to determine if the lack of processing power is what is causing the scraper to miss information. However, the difference in sample count could be explained by the product being removed from the web store, thus, the scraper would have collected the correct amount of data.
- Lack of historical information – as the scraper was repeatedly run over the course of a weekend, there is a lack of historical price information on all GPUs and thus it is difficult to demonstrate that the web application can accurately plot the change in price over a longer period. This is further shown in the hourly sampling rate which is not always long enough to monitor a change in price for a GPU. However, Figure 7 shows the reduction in price for a product over 3 hours, suggesting that had the scraper been run daily, this small sale period would have remained unseen proving that as research as shown a higher sampling rate could provide a more accurate insight to the data (Hillen, 2019).
- Lack of range – the scraper only looks at NVIDIA GPUs on the NVIDIA website, meaning that all the information collected comes from one source and thus does not give an accurate depiction of the average cost of each model. Therefore, a future improvement on this project could be the integration of other data sources (GPU sellers)

such as Amazon[6], eBay[7] or Newegg[8] to find a mean value for the cost of each model of GPU. Furthermore, scraping information from competitor companies such as AMD and using the scraped data to make a comparison to the NVIDIA GPUs on costs for similar specifications would allow the researcher to identify better value for money products.

- Lack of analysis – throughout the application, there is only one type of analysis that takes place: plotting cost over time. While this demonstrates that web scraping CAN be a valid data collection method for data analysis, it does not fully utilise the data that has been collected. Perhaps a future iteration could use this timeline information as a predictor for the sale of specific GPU models or all models. This would provide a similar function to already available services[9]; but it would be highly specific to NVIDIA GPUs and reliant on web scraping for data collection whereas other services may use traditional sources or manual collection. Furthermore, the application could benefit from fully utilising the other data that has been collected, for example:
    - Plotting counts of types of GPU model e.g., Memory size, memory type, cooling type etc. Furthermore, narrowing the categories within the data would allow for better comparison within models i.e., two models with the same amount of memory may have vastly different prices.
    - Plotting a spread of prices would allow users to compare against the current price and determine whether the price for the product is a good deal or not.
- Unfriendly user-interface – A design fault of this project was the lack of friendly user interface. While aspects such as a structured URL hierarchy and clearly labelled navigation buttons improve user-friendliness, plain graphs and tables are not nice to look at and can be difficult to understand. Therefore, utilising some form of reactive

[6] https://www.amazon.co.uk/
[7] https://www.ebay.co.uk/
[8] https://www.newegg.com/
[9] https://www.moneysavingexpert.com/deals/christmas-deals-predictor/#week-07-December-2020

graph that displays the same information in an interactable manner may increase information accessibility.

- Sampling rate is not precise – The web scraper script takes time to run, with some operations taking longer than others, meaning that the time sample is taken is not exactly an hour apart. Furthermore, other programs or services running on the machine may slow the operations speed further. The effect of this is increased again by the lack of processing power on the Raspberry Pi platform where the scraper was developed, thus repeating the suggestion that a more powerful system should be used to run the script.

5 Conclusion

To conclude, data analysis is an incredibly valuable method of understanding things around us and this process relies on accurate and clean data, therefore, proper data collection is essential. Web scraping provides a cheaper alternative to the costly, traditional methods of data collection, through the careful selection and automated collection of data by bots. While web scraping allows a user to bypass many of the issues that occur in traditional data collection methods, it is not without limitations. One of the biggest limitations is the lack of historical data, meaning that data only goes back to the first execution of the web scraper, preventing in-depth analysis until multiple sampling periods have passed.

There is no *general use* web scraper, however, most follow the same standardised steps. These have been used to develop a web scraper and application to fulfil the aim of collecting information for GPU price analysis. The created web scraper demonstrated its value as an applicable data collection source for data analysis. The developed web scraper automatically takes information on GPUs from the NVIDIA web store at set periods and visualises the data by plotting the price over time for each product in the web application. The developed application succeeds in demonstrating the aim despite the level of analysis being relatively low. However, the application follows the general limitations present in all web scrapers such as a lack of historical data that may discourage others from developing web scrapers for this purpose.

6 References

DeVito, N. J., Richards, G. C., & Inglesby, P. (2020). How we learnt to stop worrying and love web scraping. *Nature*, *585*(7826), 621–622. https://doi.org/10.1038/d41586-020-02558-0

Hillen, J. (2019). Web scraping for food price research. *British Food Journal*, *121*(12), 3350–3361. https://doi.org/10.1108/BFJ-02-2019-0081

Macapinlac, T. (2019). *The Legality of Web Scraping : A Proposal*.

Patel, H. (2018). *How Web Scraping is Transforming the World with its Applications*. Towards Data Science. https://towardsdatascience.com/https-medium-com-hiren787-patel-web-scraping-applications-a6f370d316f4

Xu, Q., Cai, M., & MacKey, T. K. (2020). The illegal wildlife digital market: An analysis of Chinese wildlife marketing and sale on Facebook. *Environmental Conservation*, *47*(3), 206–212. https://doi.org/10.1017/S0376892920000235