**Evaluation of the performance of a web scraper within an error message based search engine application.**

Abstract:

* Make it up on the day

Introduction:

* An error message web app that’s able to perform a generic api call to stack overflow for related forum posts, a google web crawler that provides a list of urls that are appropriate for the documentation for individual languages and frameworks and finally a scraper that is able to extract information from other software development websites in order to provide more information to the developer.
* Brief mention on how search engines are important in everyday life
  + The difference to a web crawler?
* Why web crawlers are important
* Who uses web crawlers?
* Percentage of sites that use some form of web crawling
  + Stat / Source to back up claim
* What I will be making
* How I will test it

Literature review:

* Definition of a search engine
  + <https://research.aimultiple.com/big-data-stats/> - ctrl + F ‘IBM’, 90% of world data created in previous 2 years (2013)
* How the performance of a *BLANK* (Evaluation) :
  + Scrapers
  + Search engines - <https://www.accenture.com/us-en/blogs/search-and-content-analytics-blog/10-criteria-evaluating-search-engine>
* Ethical dilemmas
  + Legality

Research and Development Methodology (if applicable)

Design and implementation

* Simple time experiment, Commercial free scraper provided with list of URLs as well as my one an compare and contrast values.
* Crawl 3 error sites (Same in scalper) compare to either a commercial scraper or to this:
  + <https://www.researchgate.net/profile/Ashish-Kumar-Maurya/publication/276206236_URL_Ordering_based_Performance_Evaluation_of_Web_Crawler/links/555c088e08ae8f66f3ade28e/URL-Ordering-based-Performance-Evaluation-of-Web-Crawler.pdf>

Design / Data Collection

Data analysis

Results

Discussion

Conclusion

Reflection:

* How it can be improved

References

* Other masters project : <https://www.diva-portal.org/smash/get/diva2:1415998/FULLTEXT01.pdf>
  + Experiment = simple time and resource comparison of scraper for retrieving 1000 items in a single URL

Literature review:

The first search engine was designed as an internet searching tool within the university of Montreal

In 1991. “Archie”, sort for archives, was a script based program that would download all the files located on FTP sties and then created a database that once searchable via filenames. The year after, “Gopher” was created at the university of Minnesota which was described as “a distributed document search and retrieval network protocol” [2]. Meaning it was able to index plain text documents. This was shortly followed by “Veronica” and “jughead” which were able to search the Gopher index more effectively before the launch of the world wide web in 1993. The first web search engine released was “Wandex” which possessed the first web crawler known as “World Wide Web Wander”, created by Matthew Gray [4]. “World Wide Web Wander” was primarily used to keep track of the size of the growing platform rather than a traditional web crawler that we know today. [3] The first modern Day web crawler-based search engine was created in the same year, 1993. “JumpStation”, was the first engine that relied on a fully automated bot to index links available to be displayed to the user as a list of URLs that match some form of keyword criteria. “JumpStation” is considered the first search engine that behaves similarly to modern day crawler-based search engines.

In today’s day and age, modern search engines are able index billions and billions of web pages, images and video or audio files within mere seconds thanks to the evolution of web crawlers. In the very early days, website creators would participate in submitting their own indexing for pages they wish to be listed on a search engine and engine administrators would manually edit the links into a particular format to allow indexing. The first engine that indexed over 1million pages occurred in 1995 with the Pittsburgh's Carnegie Mellon Universities search tool “Lycos” that was able to supply page ranking for relevance as well as snippets of content from induvial sites. Modern day engines such as google utilise an entire network of crawlers in order to scour the billions of indexed pages on the world wide web. Early crawlers would typically be dealing with small data sets ranging from anywhere between a few kilobytes up to a few dozen megabytes. Whereas Google estimates that their entire search index contains over a few hundred billion web pages along side over 100 million gigabytes of content. [5]

It’s difficult to pinpoint the origin of web scraping but the first popular library that allowed developers to harness the ability of web scraping was a Python library known as BeautifulSoup created in 2004. BeautifulSoup allowed developers to parse HTML elements an extract key information from any indexed web page. This had the potential to save developers hours of time when creating applications that wished to extract data from hundreds upon thousands of sites such as price monitoring on e-commerce sites, financial data aggregation for stock based application or for modern data crypto currency apps and finally news tracking applications that could congregate a large volume of material across the web onto one website. Soon after the release of BeautifulSoup web scraping began growing in popularity. Dedicated software began to be created such as Web Integration Platform V 6.0 by Stefan Andersen. Users were able to highlight key information within a web page and extract the data to an excel spreadsheet or supported database of their choosing for easier access and analysis. Web scraping is continuing to be utilised more and more as the World Wide Web appears to be growing endlessly and with the rapid development of artificial intelligence, the future would suggest a more predictable nature for web scraping that could allow more accessibility and freedom to data contained online.

* <https://d1wqtxts1xzle7.cloudfront.net/52774738/Literature_survey_for_crawling-with-cover-page-v2.pdf?Expires=1642985834&Signature=fwPu0qBAoep4NeVdiuhmHMrTsN9k7QPmhFfrFR-NjDA0YcvVu2tXgeWEqsm~Pi2JbjPUygReczDbiC-zNA~rkYvp9av6S4CiRMnyaHmkov8DrSG0wVwHJmE7~RAQIy1VD5FgXcDKHCQvAtqVwF74jMAJcIUe2JuiPy25DL5VmvFPdRw5xYYqg6ssazhSkBltlEW-0inp0rED07aNZYRL41YEdpkdbdYXpgzBr16mEQ1ZwMWW42WnWwe0xs05l2yJ7fxfV5BW3yP8FRD3M7fphg9qlvFhcIbDXzruHhUKlrCoFtz4S0X1uxGPTk-0YSF56d3XmbrJi7-dbi605j6xkw__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA>

Search engines:

A search engine is defined as a software system that is able to carry out textual based search queries that scour the world wide web in a systemic way following strict protocols in order to return a list of indexed URLs that are commonly sorted by the relevance associated with the search query provided. [Reference] Common search engines include Google, Yahoo and Bing are all characterised as Crawler-based search engines [Reference], that is they share common characteristics surrounding the crawling of websites in order to extract information for a search query. The other type of search engine is known as human-powered directories [Reference], these can be defined as performing searches based on organized subject categories. Human-powered directories utilise human engagement in order to filter and provide a condensed list of relevant results to a specific topic or subject, examples in real life include ApexKB and NowNow which was created by Amazon [reference].

The characteristics of a crawler-based search engine are able to be boiled down into a few main components:

* A crawler, also known as a spider, that is able to systematically browse through the web in order to perform some sort of indexing of potentially websites to be returned
* A database that is able to store the type of indexing performed by the web crawler
* A series of programs or algorithms that is able to assess / rank the sites returned from the web crawl that provide the most relevance to the search query
* A search interface that acts as the intermediary between the user and the database

These characteristics form the building blocks to almost all commercial search engines that exists today, most search engines providers thus have their own individual applications and systems that are able to perform / meet these characteristics. For example, Google utilises their own web crawler Known as Googlebot [reference], a database that is able to hold an incredible amount of large data known as Bigtable [reference], a site ranking algorithm tat is able to measure the importance based on the search query known as PageRank [reference] and finally a search interface such as google.com.

The ability to accurately rank and assess the quality of a web page relative to the keyword search is want distinguishes a good search engine from a bad one fundamentally. A large reason for Googles large success in the early days of their development was down to the page ranking algorithm known as PageRank. PageRank is defined as a link analysis algorithm, name after co-founder Larry Page, that assigns a numerical weight to each element of a hyperlink set in order to evaluate the importance / popularity of a web page. A higher PageRank value denotes that the web page has a higher volume of traffic of users in a given period of time as well more web pages within a given set that point to the specific web page as a hyperlink, known as “link popularity”.

The algorithm possesses both a complex version involving eigenvalues of matrices and an element known as a dampening factor, a constant that’s described as if a user browsing through links will eventually stop clicking, the probability of a user stopping at any given point is known as the dampening factor (usually = 0.85). The simplified algorithm is best described using a scenario where only 4 web pages exist. Modern day versions of PageRank apply a probability distribution of between 0 to 1 between the 4 pages, therefore the initial value of PageRank for the 4 pages is 0.25.

The formula to calculate the PageRank value is denoted below in figure 4. It describes the PageRank value of a web page being equal to the sum of the PageRank of all documents that possess an outbound link that points to the web page, divided by the number of outbound links in total.

An example scenario would be if we were to calculate the PageRank for web page A and all 3 other web pages had an outbound link that pointed to it. Page B had 2 outbound links, C only 1 to page A and D a link to all 3. The formula to calculate this can be scene in figure 5 which would equate to a PageRank value of 0.458 (assuming an initial PageRank of 0.25 for each page B though D).

A picture containing diagram

Description automatically generated

Diagram

Description automatically generatedFigure 4, formula to calculate PageRank of a web page with 3 nodes that possess an outbound link pointing to it. [6]

Figure 5, formula to calculate scenario problem [6]

Web Crawler:

A web crawler, often called a spider, was created in the 90s as a tool used by search engines or other applications for the purpose of indexing or copying information in order to perform fast and efficient searches. A web crawlers process usually begins with a list of appropriate URLs to index, known as seeds [reference]. A crawler will then crawl these seeds and depending on its function, indexing or archiving for example, will perform a number of actions such as downloading the HTML contents or constructing a graphical representation of the traversable URLS as well as extracting all the possible hyperlinks that are accessible on the individual web page with the purpose of storing these in the list of URLs for indexing in future. Most crawls will iterate through the list of URLs they intend to visit to ensure that no duplicates are re-entered, it is important that crawls act as efficient as possible so parsing through a web page that has already been crawled will not only take more time for the crawl to complete but will consume network and hardware resources as well. A simple architecture diagram for a web crawler can be seen below in figure 3.

Diagram

Description automatically generated

Figure 3, architecture diagram for a simple web crawler.

Crawl Policies:

The function of a web crawler is relatively simple as it repeats the same function(s) until all hyperlinks are indexed, however the behaviour of a crawler in the way it conducts these functions are a product of 4 main crawling policies, selection policy, re-visit policy, politeness policy and parallelization policy.

Selection policy:

The first policy is the selection policy. This policy defines a metric for the importance of a web page and creates an order of prioritization for the crawler to carry out its intended function, whether that be downloading or monitor for maintenance to ensure hyperlinks direct to the intended resource. This selection process is down multiple factors but the most important and highest weighted ones for the algorithm is the quality of the content within the web page, the popularity of the page, the number of visits it has had in a set time frame and the number of pages containing a hyperlink to the same endpoint and finally how relevant the contents of the URL itself matches the search parameter supplied.

Re-visit policy:

The 2nd policy that effects the behaviour of a web crawler is the re-visit policy. This short a short policy that tells the crawler how frequently to return to the current web page being index in order to check any editions that have been made to the web page. With crawls that can take anywhere from hours to a few weeks many pages during will have had edits, so it is important especially for crawls whose primary function is to download and maintain an accurate and update version of any page that may be indexed in the near future.

Vipul Sharma of Punjab university proposed an interesting analysis of combining the PageRank algorithm assignment on the relevance of a web page, with the Prato 80 – 20 principle [7] which states that 80% of effects comes from 20% of the causes. The idea is to better allocate a crawler’s resources during this revisitation policy as to better effectively and efficiently crawl web pages while minimizing resource consumption, such as power and bandwidth. They surmise that pages that possess a higher PageRank as well as a more recent last documented modification date should be crawled more frequently as they tend to receive a higher volume of traffic and be updated more regularly. The assumption that a web page crawled to frequently leads to wasted resources and one crawled too infrequently diminishes the quality of the search engine itself. By allocating web pages into collections denoted 20% and 80%, with the 20% collection containing web pages that are required to be crawled more frequently, they attempt to find a balance between the revisitation policy of popular and minor web pages to provide both a more accurate search result and a more efficient solution to the resource consumption problem. [8]

* <https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.36.6087>

Politeness policy:

Another policy is a politeness policy. Due to the scale and automation of a web crawler, it is able to retrieve and index data incredible fast, a single web crawler alone can send 1000s of request per second to an individual site. Most crawlers work within a network as well, meaning multiple at any one time can be interacting with the same server, although accessing a different web address. A biproduct of this speed and automation is the vast consumption of network resources that are needed. This can have a crippling effect on a website or servers’ ability to process request and maintain visibility to other users.

* <https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=Robots+in+the+web%3A+threat+or+treat%3F+&btnG=> - M Koster threat to the web. Impact that crawlers can have if left to own devices

The solution that was devised to avoid this problem was presented by Martjin koster in 1994 known as the robot’s exclusion protocol [reference], Google then in proposed that this be made an official standard to the Internet engineering task force (IETF) [reference]. It’s a standard utilised by websites administrators to inform crawlers about which areas of a certain website should not be processed. A robots.txt file will exist utilising the root URL with a /robots.txt (E.g. <http://site.com/robots.txt>) extension that will contain strict instructions for what URL extensions the crawler is prohibited from accessing. This policy is usually the first to be executed during the crawl process. A common process for the crawler to adhere to is first perform a get request on the root URL with the extension of robots.txt to retrieve the text file, this is performed before any crawling is applied to the root URL. The file is then read for any commands that contain the disallow keyword with extensions present. The crawler will identify itself within the HTTP request using the header ‘user-agent’ and will then read the disallowed extensions related to that user-agent. An example of this can be seen in figure 1 for youtube.com. If this file doesn’t exist the crawler will just assume that the site has no limitations on what can and cannot be accessed.

Text

Description automatically generatedFigure 1, yotube.com robots.txt file for disallow URL extensions for any user-agent.

This disallow keyword meets the criteria for restricting access to pages the site administrators do not wish to be crawled due to either privacy or the irrelevance of the contents for a public facing search engine. However, this does not solve the issue of restricting the interval of connections made to one endpoint. The solution for this is use the crawl-delay command within the same txt file and the associated URL extension. The intention is for a crawler to read this delay, usually represented in seconds, and restrict itself from visiting the specified URL within the delay proposed. This is not apart of the standard presented to the IETF however, meaning that crawlers may interpret this delay command differently. An example of this can be seen below in figure 2, twitters robot.txt file that implements a 1 second delay for all crawlers traversing the site.

Figure 2, twitter.com/robots.txt, containing the delay command to limit reconnections for crawlers traversing the site.

It is possible however for crawlers that have malicious intentions, such as spam or DDOS attack and phishing attacks, to not follow the convention of restricting its access to the specified domains or adhere to the request delay. To avoid this, it is important for the security administrators to implement some sort of policy to recognize and restrict access to these malicious scripts to prevent any damage such has down time or exposer of sensitive information. In order to prevent these malicious crawls, it is important to identify common characteristics amongst these crawlers as well as prevent access once they have been detected. Many tools exist for this detection such as the PathMarker tool proposed by Shengye Wan et al [1] to detect or simply slow down the efficiency of malicious crawlers, based on their visitation path as well as their time features, to prevent them from downloading server based content.

* <https://www.researchgate.net/publication/310840290_MalCrawler_A_Crawler_for_Seeking_and_Crawling_Malicious_Websites>

Parallelization policy:

The final policy that crawlers will adhere to is the parallelization policy. Most crawlers that wish to index a large quantity of URLs and download their contents for indexing such as search engines, will execute in parallel with each other. This is known as a parallel crawler. [reference] This policy relates to maximising the download speed of the web page while avoiding repeated download of content within the same web address. If 2 crawlers in parallel retrieve the same URL during the indexing of a web page, it is important for the assignment of this new URL is only added to the stack once to avoid this download duplication.

Crawl limit:

Within the politeness policy for a good Samaritan web crawler whose main intention is to crawl through a site peacefully and does not intend on throttling bandwidth to other users trying to access a website a thus not degrading the experience, another parameter can be assigned to most crawler known as the crawl limit, also known as crawl budget. This is defined as the number of parallel continuous that will be accessing the resource simultaneously as well as the time to wait between future connections to the same endpoint (assuming a robot.txt file is not present). Depending on the crawler, this value may fluctuate depending on a number of factors such as response time for loading a resource as well as direct instruction on the number of connections or request time between fetches. Google for example monitors these using a statistic they define as “crawl health”, which is used to measure the response time of the server where the resource is located and allocating an appropriate number of crawl bots to minimize the impactfulness of the crawl.

* <https://developers.google.com/search/blog/2017/01/what-crawl-budget-means-for-googlebot>

How the performance of a web crawler is measured:

In order to be able to evaluate a crawler effectively, it’s important to understand the limiting factors that affect crawlers in general. As stated previously, crawlers that run in parallel will almost inevitably come across a previously crawled resource, in order for the maximum efficiency to be calculated effectively it’s important that the crawler is able to distinguish between pages that have been crawled previously in order to minimize overlap. Another limiting factor relates to the quality of the web pages provided I.E., the value calculated by the page rank is deemed sufficient enough to warrant the crawler performing its function such as downloading or indexing. The most limiting factor for most small scale analysis on the performance of a web crawler is down to the network and bandwidth restrictions on the machine running the crawler. Regardless of the efficiency of the crawler performing a particular function, if the bandwidth necessary to maintain a constant stream of web pages is insufficient and there is down time between crawls, the efficiency metric will be affected by this. [9]

Filippo Ricca and Paolo Tonella performed an easy to understand evaluation on a web crawlers’ performance. They describe how the behaviour of a crawler is down the policies mentioned previously. They state that “A crawler must carefully choose at each step which pages to visit next”, this statement is able to boil down the difference between a good crawler and a bad crawler in terms of behavior. A good crawler is able to traverse the next best web page in terms of usefulness to the topic at hand in the list of URLs found, whereas a worse crawler will simply crawl through in a more linear fashion or won’t be able to distinguish between higher priority web pages more effectively.

Within the same article they perform an evaluation on popular commercial crawlers by assessing their ability to meet certain features that would be indicative of a efficient crawler. These include completeness, the number of pages downloaded over an entire web application (stored in the web server). The latter can be a difficult value to calculate so they stipulate that this value is more of a comparison on the total web pages crawled compared to other crawlers being scrutinised. Another is the robustness, ability to function correctly when faced with both syntactical and scale based challenges. Download limiting, ability to limit the scope of the web application that is wished to be downloaded. And finally, the supported features available such as following the robot exclusion protocol, the user interface, ease of use and multithreading ability. This type of evaluation is very useful when presented with a multitude of variables that are difficult to control as well as the number of objects to be evaluated. The overall evaluation of the 7 crawlers presented is an excellent way of categorising particular crawlers based on the action required of them rather than just the hardware and network resources required to perform said action as well as the time taken to be completed. [10]

A more complex evaluation method was presented by Mohd Shoaib and Ashish K. Maurya of the Shri Ramswaroop Memorial University in India [11]. They expanded on an updated term frequency – Inverse document frequency (TF-IDF) algorithm presented by Shaojie Qiao et al of the School of Information, Science and Technology Southwest Jiaotong University [12]. Shoaib and Maurya incorporated the adapted TF-IDF algorithm into a performance analysis of a web crawlers selection policy by analysing the response time for a crawl that took place across 4 educational institute websites. They analysed the top URLs returned based on the updated similarity score. This similarity score is defined as a content score, a utilisation of the TF-IDF algorithm that is able to accurately reflect the importance of a term within a collection of documents, an adapted form of which is presented by Shaojie Qiao et al which expands on the already existing TF-IDF algorithm to incorporate a similarity measure to accurately assess the similarity of 2 documents, this will be expanded on in future. The structural score, an implementation of the SimRank algorithm that is able to effectively measure the similarity of two objects if the are related to “similar objects” based on link structure analysis [13], the formula for this sim rank is able to be seen in figure 6. And finally, the Combined similarity score that is presented below in Figure 7. K1 and K2 are constants that equal 0.7 and 0.3 respectfully.

A picture containing shape

Description automatically generatedFigure 6, the SimRank algorithm. Supplies a value between 0 and 1 that denotes the similarity between 2 objects. If 2 objects are identical, a = b, the similarity is equal to 1. [13]



Figure 7, the result of step 5 is the content similarity, and the result of step 6 is the structural score. [11]

They went on to measure the crawling time required to extract the URLs across the educational institutes and save the contents of them. Along with the crawling time they also measured the ordering time needed to convert the URLs crawled into a structured list based on their calculated similarity score. Finally, they measured the precision of they crawl which they define as the ratio of the number of irrelevant web pages that contain no content related to the keyword search, in this case “student”, to the total number of records retrieved during the crawl. The performed the experiment using 2 different crawl limits, one set to 5 and one set to 10, and used the seed URL as the 4 institutes homepage and the keyword crawler. They go on to present a comparison of the top URLs from the updated selection policy algorithm as compared to a PageRank parameter based crawler as well as the crawling and ordering time of the 4 presented websites.

They overall execution and presentation of the study is an excellent foundation to how the performance of a crawler should be presented. Not only should the crawl and ordering time be presented for the 4 different websites, all could have a different architecture as well as varying page sizes which will drastically impact results. The comparison of 2 different ranking algorithms supplies an excellent perspective on the actual logic behind a crawler’s behaviour. Its all well and good for a crawler to be incredible fast in its crawl time a return in a timely manner, but if the content returned is not a the most appropriate order the overall efficiency of an application of a web scraper or the presentation of the data will be compromised in some shape or form.

The algorithm presented by Shaojie Qiao et al is an adapted page ranking algorithm based on the similarity measure which aims to bridge that gap between the link relations concept of crawlers being led along a tree within a web page to the actual contents contained within the web page and how that relates to the target information that the users is seeking. The algorithm is composed of 2 main components, a similarity metric that is able to quantify the likeness between two web pages and an implementation of an adapted PageRank algorithm that apply two distinct weightings to the title and body of a web page. This is similar to figure 7 where Shoaib and Maurya used the same weighting metric except applied it to the content similarity of a term within a collection of documents and its structural similarity principle of two object being similar if they are related to similar objects based on a link collection structure analysis where objects point to and are pointed at by many objects within the same collection.

The first component is what they define as a similarity metric and is combination of an adapted model for TF-IDF to find the likeness of the contents of a webpage. If a user wishes to find term *Ti* within a collection of documents *N*. The algorithm is as follows:

* Text

  Description automatically generated with medium confidenceTerm frequency scheme (TF-scheme). Simply put the number of times a keyword appears within a single document divided by the number of words within that document, figure 8.

Figure 8, formula for the term frequency scheme

* TF-IDF scheme. The log of the total number of web pages searched and the number of pages term *Ti* appears in, figure 9.

Figure 9, formula for the TF-IDF scheme.

* Text

  Description automatically generated with low confidenceAn adapted formula proposed by Salton and Buckley [14] that’s able to calculate the TF-IDF of an individual term. N+1 is applied to ensure that log value is always greater than 0 in case term *Ti* appeared in every document, Figure 10.

Figure 10, adaption of Salton Buckley formula .

* Text

  Description automatically generatedFinally for the first component, the finalised content similarity formula that is able to compute the similarity between 2 pages. If the weight of term *Ti* with a singular page X is denoted as:  then content similarity between page x and y is represented in figure 11.
* Figure 11, content similarity of term *Ti* between 2 pages.

The second component of the Shaojie Qiao et al algorithm as mentioned previously is the adaption of the PageRank algorithm to provide a biased weight towards the title of a page rather than the contents. Identical to the Shoaib and Maurya weighting, a value of 0.7 is applied to the title and 0.3 to the content body of the web page. Shaojie Qiao et al argue that the two different contents have a different impact on the query result returned as well as the fact that title is the leading contributor to a web user interacting with a link as they haven’t had the opportunity to interact or consume the inner content as of yet.

* 3 main articles for working out similarity score discuss at end:
  + <file:///C:/Users/Compl/Downloads/Performance_Evaluation_of_Web_Crawler.pdf>
* IF LOW ON WORDS: Brief mention on this one for crawler detection
  + <file:///C:/Users/Compl/Downloads/Feature_evaluation_for_web_crawler_detec.pdf>

Web scrapers:

With the vast growth of the internet during the early 2000s, being able to accurately gather and maintain data in a timely manner was growing too cumbersome for a simple copy and paste job. Tools and libraries were being developed that allowed the extraction of key information from a vast number of websites to be harvested in a shorter period of time, much faster than a human being able to copy the information from one location to another. Thus, the web scraper was born.

A web scraper, also known as a harvester, is an automated process that usually involves the interaction of a web crawler whose intention is to extract key, unstructured data from a website usually in the form of interaction with HTML or DOM based elements and convert it into structured data. This data is usually extracted and saved to some sort of database or excel spreadsheet for use and analysis in the future. A common action for a system whose intention is to scrape information form specific web pages is to use a focused crawler whose seed URL will relate to a specific website or keyword for a topic. The crawler will begin to download the contents of these pages and extract other URLs and them to the queue of crawl items. Commonly some form of selection policy is applied on the list of URLs. This will allow the scraper to begin acting on sites deemed most sufficient to match the topic or search keywords in the hope of supplying the most accurate and up to date information as possible. The scraper will parse through the downloaded HTML elements to extract any data deemed necessary whilst saving or presenting them in a specific format to a user or database.

Web scraping is an incredible versatile technique for extracting data from web sources and is able to be tailored and modified depending on what resource needs to be accessed. Some common applications of web scraping include:

* Website editions detection
* price comparisons
* scrape job posting / message board
* Extracting business details from business directories such as a long chain of employee emails
* weather data monitoring

Multiple different techniques can be utilised in order to maximise the efficiency of data extracted thus minimizing the time for the scrape. The most common technique involves HTML parsing. A technique most efficiently used when web pages encapsulate data within some form of wrapper that possess some form of common temple between components. Another technique commonly used across scraping software and applications is DOM parsing. Embedded programs within web browser are able to retrieve content dynamically from the generated DOM trees formed from client side scripts.

Many commercial web scraping tools and software are available for purchase or even free use. Popular packages include Visual web reaper, web content extractor, UiPath, import IO and Web scraper. Visual web reaper and web content extractor are user friendly oriented web scraping software packages that allow for unstructured data extraction to either CSV files or a multitude of database software support such as MySQL or ODBC. UiPath is similar however attempts to replicate a user within a web browser more accurately and offer web service integration for other applications. Import IO is a free online software tool and maintains extracted data in the cloud rather than locally. Finally, web Scrapper is a browser extension based scraper and allows scheduling and parallel crawling similar to a search engine crawler in order to maintain data sets up to current format.

* Reference for scraper techniques: <http://ir.kdu.ac.lk/bitstream/handle/345/1051/com-059.pdf?sequence=1&isAllowed=y> :

Evaluation of a scraper:

* <https://www.ripublication.com/gjpam17/gjpamv13n2_43.pdf>

Legal issues:

Within the topic of web scraping there’s a boundary that exists that constitutes fair use and illegal activities constituted by both the European union and the United States. Within the United States, there are several legal complications that can arise from unsolicited web scraping. The most common issues are the scraping of copyrighted material. Both the US and European union govern that the scraping under commercial terms can be in breach of intellectual policy however it’s a grey area within the EU if its for scraping for personal use and whether the unwanted extraction of copyrighted material is deemed intellectual theft. The clearest cut example of illegal activity comes under the US Computer Fraud and Abuse Act. Any intent to disrupt or delay

RE WRITE THIS

* *HiQ Labs v. LinkedIn*, 2019. The Ninth Circuit Court of Appeals ruled that [scraping](https://en.wikipedia.org/wiki/Web_scraping) a public website without the approval of the website's owner isn't a violation of the CFAA.[[47]](https://en.wikipedia.org/wiki/Computer_Fraud_and_Abuse_Act#cite_note-47) A Supreme Court appeal is pending.[[48]](https://en.wikipedia.org/wiki/Computer_Fraud_and_Abuse_Act#cite_note-48)

References

[1] - <https://cybersecurity.springeropen.com/articles/10.1186/s42400-019-0023-1#Sec36>

[2] & [3] - <https://dash.harvard.edu/bitstream/handle/1/12964432/REGULATING%20SEARCH%20ENGINES.pdf?isAllowed=y&sequence=1> (Use some more reference from within here)

[4] Matthew Gray wrote the first Crawler, the World Wide Web Wanderer, which was used from 1993 to 1996 [Dr Rajender Nath, Khyati Chopra, “Web Crawlers: Taxonomy, Issues & Challenges”, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 4, April 2013]

[5] - <https://www.google.com/search/howsearchworks/how-search-works/organizing-information/>

Crawler Policy reference:

* <http://www.csjournals.com/IJCSC/PDF2-1/Article_49.pdf> [Selection and re visit reference]
* <https://www.researchgate.net/profile/Filippo-Ricca/publication/220135806_Web_crawlers_compared/links/54894a710cf268d28f090a14/Web-crawlers-compared.pdf> [Selection and re visit reference]

[6] - <https://www.geeksforgeeks.org/page-rank-algorithm-implementation/>

[7] - Hadrien Bullot and S K Gupta. A Data-Mining Approach for Optimizing Performance of an Incremental Crawler.

[8] - <https://www.researchgate.net/publication/273901588_A_Hybrid_Revisit_Policy_For_Web_Search>

Proceedings of the IEEE/WIC International Conference

on Web Intelligence (WI’03)

[9] – References for the entire first paragraph for the limiting factor of crawlers:

* All from <file:///C:/Users/Compl/Downloads/Performance_Evaluation_of_Web_Crawler.pdf>
  + [1] Bhaskar Reddy,Kethi Reddy,” Improving efficiency of web crawler algorithm using parametric variations“ Ph.d thesis submitted in June 2010 at Thapar University India.
  + [3] Shaojie Qiao ,Tianni Li, Jiangtao Qiu,” SimRank: A Page Rank Approach based on Similarity Measure “ 2010 IEEE.
  + [5] Hongzhi Guo, Qingcai Chen, Xiaolong Wang, Zhiyong Wang, Yonghui Wu,” STRank: A SiteRank Algorithm using Semantic Relevance and Time Frequency “Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics .

[10] - <https://www.researchgate.net/profile/Filippo-Ricca/publication/220135806_Web_crawlers_compared/links/54894a710cf268d28f090a14/Web-crawlers-compared.pdf>

[11] - <https://www.researchgate.net/profile/Ashish-Kumar-Maurya/publication/276206236_URL_Ordering_based_Performance_Evaluation_of_Web_Crawler/links/555c088e08ae8f66f3ade28e/URL-Ordering-based-Performance-Evaluation-of-Web-Crawler.pdf>

[12] - <https://booksc.org/ireader/22194284>

[13] - <http://ilpubs.stanford.edu:8090/508/1/2001-41.pdf>

[14] - <http://www.ict.nsc.ru/jspui/bitstream/ICT/1231/1/solton-1-29-03.pdf>