UPPSALA
UNIVERSITET

Självständigt arbete i informationsteknologi
June 29, 2017

A web crawler to effectively find web shops built with a specific e-commerce plug-in

Malin Englund, Christian Gullberg,
Jesper Wiklund

Civilingenjörsprogrammet i informationsteknologi

Computer and Information Engineering Programme

UPPSALA
UNIVERSITET

Institutionen för informationsteknologi

Besöksadress:
ITC, Polacksbacken
Lägerhyddsvägen 2

Postadress:
Box 337
751 05 Uppsala

Hemsida:
http:/www.it.uu.se

Abstract

A web crawler to effectively find web shops built with a specific e-commerce plug-in

Malin Englund, Christian Gullberg, Jesper Wiklund

Nowadays online shopping has become very common. Being able to buy things online and get them sent to the door is something many people find convenient and appealing. With the demand comes the market and web shops have therefore become a popular place for companies to sell their items. Companies that want to sell their products to web shops can have a hard time finding potential customers in an efficient way. This project is an attempt to solve this problem, finding a large quantity of web shops with a specific e-commerce plug-in, in this case WooCommerce, in a short amount of time. The solution was to create a program with the purpose of searching the Internet locating web shops. The result of the search is stored in a database where the user can retrieve information, such as domain name and domain end, about the web shops found. The program attempts to find additional information, such as company name and revenue, for any Swedish web shops found. It was a success in the sense of efficiency but with room for improvement considering robustness and accuracy.

Extern handledare: Håkan Lundmark
Handledare: Virginia Grande and Björn Victor
Examinator: Björn Victor

Sammanfattning

Internetshopping är i dagens samhälle något väldigt vanligt. Att kunna köpa saker på internet och få det skickat till din dörr är något människor finner både bekvämt och lockande. Med efterfrågan kommer marknaden och internetbutiker har därför blivit ett populärt ställe att sälja sina varor på. Företag som vill sälja sina produkter till internetbutiker har problem med att hitta potentiella kunder på ett effektivt sätt. Det här projektet är ett försök till att lösa problemet genom att hitta många internetbutiker med specifierat e-handels plug-in, i detta fall WooCommerce, på kort tid. Lösningen var att skapa ett program med syftet att söka genom internet efter butiker. Resultatet av sökningen sparas i en databas där användaren kan hämta information, som domännamn och domänslut, om de hittade internetbutikerna. Programmet försöker att hitta mer information, som företagsnamn och omsättning, för alla svenska internetbutiker hittade. Projektet var framgångsrikt i termer av effektivitet men med möjlighet till förbättring i betraktande av robusthet och precision.

Contents

1 Introduction

The Internet is a big part of our lives. It gives us everything: from answers to simple questions, to the possibility to communicate with anyone across the world and online shopping. The online market grows each year [39] and with it the interest to sell online. In a research study by PostNord [40], it appears that customers have a large interest to research products online before buying them in a store. Hence, having a web shop combined with a physical store is of great importance when competing in today's market. With more companies looking to establish web shops the interest of marketing online grows as well.

The aim of this project is to simplify marketing to web shops for companies and sole traders. By creating a program that finds web shops, marketing becomes more efficient and convenient. The program searches through websites looking for web shops and storing them in a database. The user is then able to retrieve information about the web shops desired. With this information the user is able to find customers that are interested in the company's products.

2 Background

E-commerce is an important concept in this project and is the practice of advertising and selling online [6]. In Sweden e-commerce has grown linearly the last years, from a total revenue of 4,9 billions in year 2003 to 57,9 billions in year 2016 [39]. With a growing business area comes a demand for the technical aspects of e-commerce such as websites, plug-ins and payment solutions. A plug-in is software that can be added to other software (in the case of this project, websites) to improve functionality [29]. Companies that develop the technical aspects of e-commerce want to find customers by marketing their products. An important part of marketing is who the target group is [31] and this is why our project is needed.

When establishing a web shop many companies choose to use e-commerce plug-ins [10]. One of the most popular is WooCommerce [52], which is a open source plug-in. The purpose of WooCommerce is to create an easy way to open and maintain an online store by offering many functionalities such as scalability, store themes and selling statistics [52]. Globally WooCommerce is the most popular plug-in with 41 percent of the market of e-commerce plug-ins. The second most popular is Magento at 5 percent. The Swedish market is also highly dominated by WooCommerce at also 41 percent with the second most popular being Zen Cart at 9 percent [10]. In this project the Swedish mar-

ket is the main focus. For this purpose, the Swedish websites Eniro [20] and Allabolag [2] are used. Eniro is a service that provides information, such as contact information, about individuals and companies in Scandinavia, Finland and Poland. Allabolag is a website for retrieving information about companies based in Sweden.

An example of a company interested in marketing to web shops using WooCommerce is Crowderia, the external third party of this project. Crowderia is an IT company based in Uppsala, Sweden and Colombo, Sri Lanka. Crowderia is a team of 40 persons and develops web and mobile applications, games and UX/UI but also works with software testing and consultancy. Their interest in this project is to find new customers for their product Crowd logistics [16]. This product is a plug-in and works together with other plug-ins such as WooCommerce and Magento [32]. The purpose of Crowd logistics is to simplify the customer's business by improving the supply chain collaboration. In order to do this Crowd logistics provides many of functionalities in areas like product management, transport, warehouse, purchasing, web shop and mobile scanning. Regarding web shops Crowd logistics offers templates to configure already existing ones. In the other areas, apps are offered to simplify and automate different tasks.

2.1 Technical background

In order to understand what this project is about, one needs to know what each part of the program does. The program consists of four parts: a web crawler using asyncio, a scraper, a Bloom filter and a database. This section will define and explain more about what these parts do.

2.1.1 Crawler

A web crawler, also known as a web spider [28], is a program that wanders around the Internet looking for websites [27]. The wandering is done by following hypertext links from page to page, recording information such as the hypertext links [9]. When wandering it can also collect other information of interest with the help of a web scraper, see section 2.1.3. A web crawler can be designed in different ways. In this project it was designed to add a few starting websites, also known as seed websites, to the queue list of the crawler. From the starting websites the crawler collects all hypertext links and adds them into the queue.

2.1.2 Asynchronous I/O

Asynchronous I/O is a form of I/O processing that permits other processing to continue while waiting for time consuming operations to finish. Knowing when to use asynchronous nonblocking operations can make an enormous difference in the response time of a system, when a program uses concurrent I/O calls [38]. Asyncio [47] is a library for asynchronous nonblocking I/O [25] and it was included into Python 3.4 in March 2014. Asyncio achieves its concurrency by using an event loop which determines when coroutines are executed. Coroutines can be seen as subroutines that can exit by calling other coroutines. The coroutine keeps its state so that the program may later return to the same point [30]. As seen in Figure 1, this makes it possible for a coroutine to stop executing, register a callback and leave control to another coroutine while waiting for I/O. A callback is executable code, in this case the coroutines, that is passed as an argument to other code. The event loop will poll the registered callbacks and once an I/O operation is complete the event loop will start executing it again and trigger a callback to finish the request. The coroutines will not do any work while they are waiting for I/O, and will execute only when there is actual work to be done. This means that a program can have a large number of concurrent I/O requests running in a single thread.

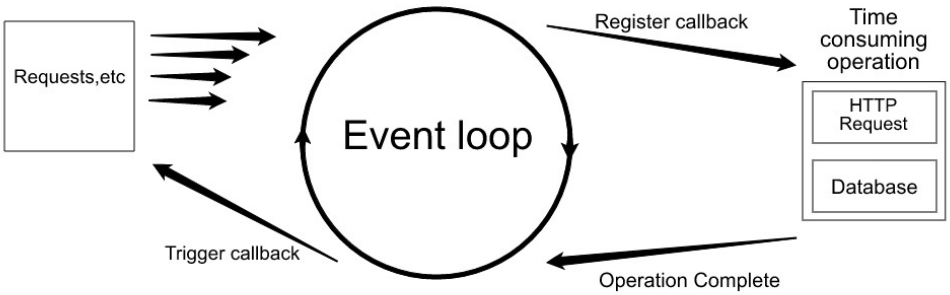

Figure 1: A figure showing how Asyncio's coroutines stop executing when a time consuming operation is started, and how the event loop restarts the execution after the operation is completed

2.1.3 Scraper

Web scraping means to collect information from an online source often using an automated tool [12]. This is done in two steps, firstly accessing the website and secondly

HTML parsing and extracting contents. The scraper accesses the website by establishing a connection through the HTTP protocol that coordinates the request. Once the HTML document is retrieved, the scraper can extract information. The extraction process is done by expression matching and sometimes additional logic. For this purpose, there are also HTML parsing libraries such as Beautiful Soup [48] and selector-based languages such as XPath [5][22].

2.1.4 Bloom filter

A Bloom filter is a data structure used to test if an element is part of a set [7]. The Bloom filter can be queried, asking if an element has been added to the set. False positives are possible while false negatives are not. In other words the query can return ”definitely not in set” or ”possibly in set”. Figure 2 shows a Bloom filter consisting of a list of boolean values and multiple hash functions that maps its input to the boolean values of the list. When the filter is empty all the boolean values are set to false. An element is added to the set by inserting the element into each of the hash functions and changing the mapped boolean values in the list to true. The possibility of collision, where different elements hashes maps to the same boolean values, makes it impossible to remove any element from the set once it has been inserted. Unlike most other data structures a loom filter can represent a set with an arbitrarily large number of elements. While it is always possible to add new elements to the set, the false positive rate will increase as more elements are added. The most extreme case would be when all the boolean values are set to true, which would make every query return a positive result.

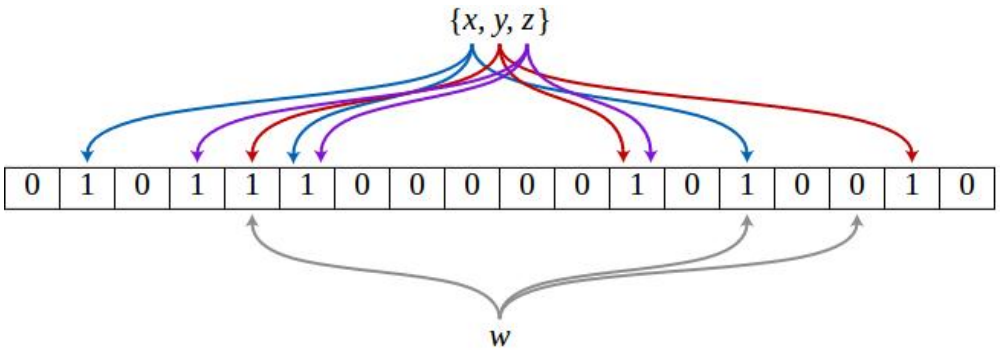

Figure 2: A Bloom filter using three different hash functions where the elements x,y and z has been added to the set. w represents a query to the filter [21].

There are two main reasons for using Bloom filters: space-efficiency and look up speed.

The space-efficiency of a Bloom filter is achieved by not storing the elements. Only the values mapped by the hash functions are stored. This means that only a small number of boolean values are needed for each element. If the chosen hash functions have good uniform distributions then less than ten boolean values are needed per element to achieve a false positive rate that is less than one percent. The most common way of representing the list of booleans is by using a bit array, which requires slightly more than one byte per element independent of the size of the element.

As for the speed of a Bloom filter, querying the filter consists only of hashing an element and verifying if all the mapped values are true. As most Bloom filters are represented by arrays the complexity of a query is $\theta(1)$, making it faster than other commonly used data structures such as linked lists(θ(n)) and a balanced binary trees(θ(log n)) [14]. While it would be possible to achieve the same speed using other forms of hash tables they would require much more memory since they need to store the elements.

2.1.5 Database-management system

A database-management system is defined as a collection of interrelated data and a set of programs to access those data. The collection of data, usually referred to as the database, contains information relevant to a user [50]. Database-systems are used in many applications such as airlines, universities, banking, credit card transactions, human recourses and much more. An example of how to modify data in a database for universities is to add new students, instructors and courses, register students for courses and assign grades to students. Accessing data is done by queries, a request for information from a database [4]. Databases are not just for enterprises and organizations. Databases are for anyone to create and use. There are several options of database management systems online to download such as mySQL [33] and Oracle [36].

3 Purpose, aims and motivation

Marketing to web shops can be very time consuming, especially finding web shops fulfilling specific requirements such as e-commerce plug-in. The focus of this project is to find web shops using the e-commerce plug-in WooCommerce. To find out what kind of plug-in a web shop is built with can be difficult for someone not familiar with HTML. The purpose of the program is to help companies and sole traders with marketing to web shops in an efficient and convenient way. For example the user is able to retrieve Swedish web shops based on revenue, which helps the user to find appropriate

customers. The program is also beneficial for Crowderia, in their search for web shops, which could be interested in their product Crowd logistics.

The aim for this project is to create a platform independent web crawler to fill a database with a large amount of web shops based on the plug-in WooCommerce. Efficiency in the crawling process is an important aspect. The user is able to retrieve information from the database using queries. The key feature in the database is that the user is able to retrieve web shops based on their: revenue, company name and top-level domain.

To our knowledge, there is no other program with the same functionality as ours. Similar programs are able to find web shops based on specific plug-ins. Our program differs in the way that the user is also able to choose companies from Sweden based on their revenue, which could be very useful when looking for a specific target group. Using our program would help make marketing more convenient and efficient. Otherwise the search for web shops would need to be done manually, which would require more labour. This means companies using this program will save both time and money.

An ethical problem could be that the program could be used with bad intentions. An example of that is that the user could use the program for mass spamming or scamming of certain companies. Unfortunately, this issue is not resolved in this project.

3.1 Delimitations

The program is only able to find companies in Sweden since it uses Allabolag [2] to get the revenue for all the companies found in the crawling process. Since Allabolag only has information about Swedish companies, this is a delimitation of the program. Another delimitation is that the program is not able to find web shops using any plug-in but only WooCommerce.

4 Related work

In section 4.1 some commercial projects are described, concerning web crawling and web scraping, related to this project such as: Google, Builtwith and Bing. There are many open source projects regarding crawling and scraping. Many of these projects are similar. An example and discussion about them is described in section 4.2.

4.1 Commercial projects

Builtwith [11] is probably the work most related to our project. Providing information about the structure and frameworks of a website, competitive analysis and business tool providing e-commerce data are some of the things Builtwith does. Some of the functionality is not free to use. There is a monthly or yearly fee to get access to more data. At Builtwith it is possible to search for any e-commerce plug-in, to retrieve web shops information such as telephone number, email and address. This is similar to our project but differs in that the user is able to select companies based on their revenue and it is free to use.

Search engines are one of the most common and widely known type of program that use a web crawler in combination with scrapers. Based on a query the search engine produces a list of pages it finds related to the query [9]. A search engine tries to find reliable pages based on number of other pages that refers to them. This results in the most popular pages being listed on the first page. Google, Yahoo! and Baidu are currently the most used search engines [1].

Google has its own web crawler called Googlebot [23]. The Googlebot uses an algorithm to determine which sites to crawl, how often and how many pages to fetch from each site. It starts with a list of URL's provided by a previous crawl process or by the webmaster.

4.2 Open source projects

There are many open source projects regarding web crawling and web scraping. Many frameworks have the same core functionality, crawling the Internet to find specified information. What kind of information the framework retrieves is not specified, but instead up to the user to decide. A possible reason for there being so many similar frameworks is that there is at least one for every major programming language.

Examples of open source web crawling frameworks are: Nutch written in Java, Aspseek in C++, Ebot in Erland and Scrapy [49] in Python. Scrapy was originally designed for web scraping but it can currently be used for both HTML parsing and web crawling. Scrapy is related to the project since the framework is intended to help with similar problems, namely web crawling and information gathering. However, Scrapy is a general purpose framework while this project solves a more specific task.

5 Method

Based on the group's previous programming experience, the choice of programming language for this project was between Java and Python. Both programming languages are platform independent [37] [45]. The main difference between the two programming languages is that Python has built-in high level data types and dynamic typing. This generally results in Python programs being three to five times shorter than equivalent Java programs but Java on the other hand has a faster executing time [46]. In this project the difference in executing time is not very relevant since a vast majority of the executing time of the program consist of waiting for HTTP requests. Therefore, Python was chosen as the main programming language for this project.

We chose to build the web crawler with asynchronous I/O calls using the Python library Asyncio [47]. For the web crawler to be efficient, it needs to visit a high number of web pages within a short period of time. This means many HTTP requests need to be handled concurrently. Since HTTP requests includes long waiting times, using asynchronous I/O calls in the program was the optimal choice. Another common way of achieving concurrency is by using threads. In the case of this project threads would be a worse choice because asynchronous I/O is more efficient in distributing CPU time. There are alternatives to achieve asynchronous I/O but Asyncio was chosen because it is included in the Python library.

When choosing hash functions for the Bloom filter the distribution and speed of the functions was considered. It is important for the hash functions to have as uniform distribution as possible to avoid collisions, which could cause false positives. There are numerous hash functions that provide good distributions. We chose to not use a cryptological hash function, such as SHA-256, because they are usually slow [17]. However, Google's Farmhash [44] is designed to give a uniform distribution while being much faster than cryptologic hash functions. Therefore, Farmhash was our final choice.

A Bloom filter also includes an array of boolean values. For this purpose, the bit array library [43] was chosen because of its space efficiency.

MySQL [35] was the choice for the database because it is one of the most commonly used database systems [18] and the group has previous experience with it. Since it is one of the most commonly used database system it is therefore more likely for the user to be familiar with the system. This is important because the user has to retrieve the results manually using SQL queries. It is also cross-platform compatible [34] which was important, since the aim of the project was for the program to be platform independent. Other popular database systems, for example Microsoft SQL Server, can only be used on a single platform.

6 System structure

The system consists of four parts: a crawler, Bloom filters, scrapers and a database. The system is started by the user giving it seed websites as starting points. The crawler then works together with the Bloom filters and one of the scrapers to search through the Internet for web shops with the e-commerce plug-in WooCommerce. The result of the search is stored in a database and updated by a second scraper. The user is then able to interact with the database using queries.

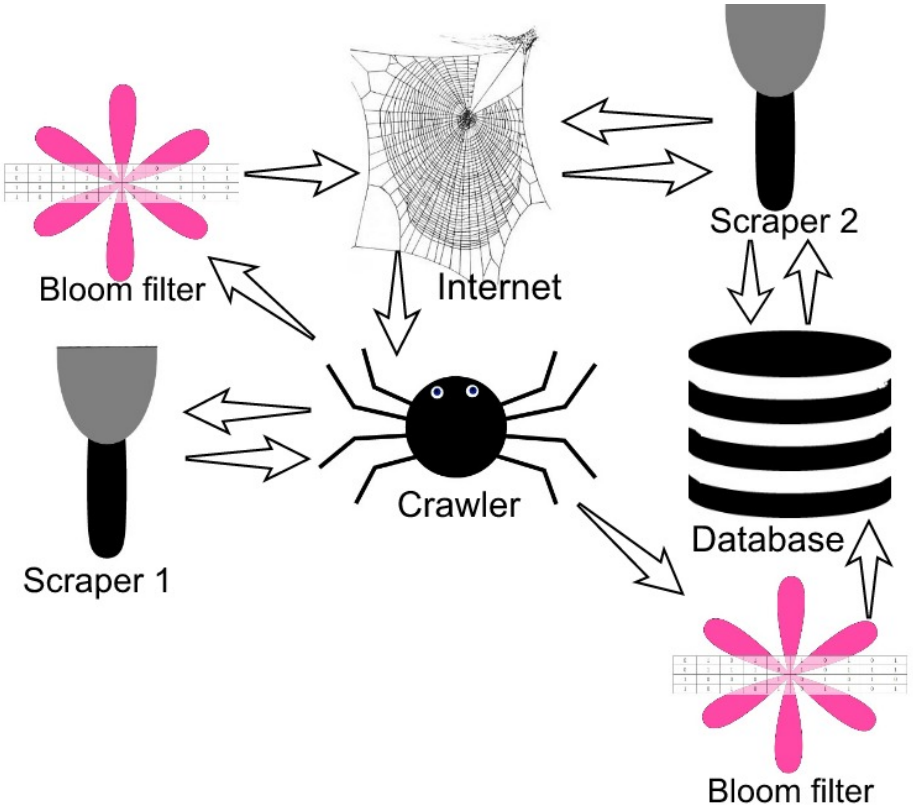

Figure 3: The system structure, how the parts of the system interact. Image of the web from Wikimedia commons [3]

6.1 Crawler

The crawler is the main part of the system, interacting with almost every part of the system. Firstly, with the help of a Bloom filter, preventing the crawler from visiting

the same website twice, it gathers information from websites using HTTP requests. Secondly, the crawler uses a scraper to find out what type of plug-in the website uses. Thirdly, it stores the website in the database using MySQLdb [19] if the website uses the specified plug-in.

6.2 Bloom filters

The system includes two Bloom filters, both of them communicating with the crawler. A Bloom filter consists of two main parts: an array of boolean values and hash functions. To represent the array of boolean values the bit array library [43] was used to create an array of bits. For the hash function part of the filter, two hash functions were created by giving two different seeds, i.e. initialization values, to Google's Farmhash [44] function.

6.3 Scrapers

There are two scrapers in this program: a WooCommerce [52] scraper and an Allabolag [2] scraper. The WooCommerce scraper, in Figure 3 called scraper 1, is provided with Beautiful Soup [48] objects by the crawler and identifies if the web shop uses WooCommerce. The Allabolag scraper, in Figure 3 called scraper 2, retrieves domain name and top-level domain from the database. With this information the scraper looks for company name and revenue on Allabolag and updates the database.

6.4 Database

The user mostly interacts with this part of the system, being able to retrieve information using queries. The database is locally stored on the computer which runs the program. The crawler stores the web shops found in the crawling process in the database. The database consist of a table containing: web shop ID, company name, revenue of the company, domain name and top-level domain, as seen in Figure 4. Null values are shown when company name or revenue was not found. The revenue is shown in Swedish crowns and ID is automatically incremented.

28	Folkets Möbler...	folketsmobler	se	3297
29	NULL	groomify	se	NULL
30	NULL	dermashopper	se	NULL
31	NULL	cubot	se	NULL
32	NULL	ecoland	se	NULL
33	NULL	hockeytime	se	NULL
34	ITsale Sweden...	itsale	se	3762
35	Eko-print Swe...	eko-print	se	2578
36	Adrecord AB	adrecord	com	15220
37	Jumpking Stud...	jumpking	se	7558
38	NULL	frontierhockeysw...	se	NULL
39	Lernberger St...	lernbergerstafsing	com	18312
40	NULL	jollitime	com	NULL
41	NULL	creativemakeup	se	NULL
42	NULL	imtheartist	se	NULL
43	NULL	inlandet	se	NULL
44	NULL	larm-online	se	NULL
45	NULL	hushallstianst	nu	NULL
46	Klimathuset i G...	klimathuset	net	5748

Figure 4: An example of a database showing the columns: ID, company name, domain name, top-level domain and revenue of the company after a crawling and scraping process.

7 Algorithms

This section describes the crawling and scraping algorithms, giving a step by step description on how the algorithms work and how they are implemented.

7.1 Crawling algorithm

This subsection describes the algorithm for the web crawler. The crawler is given starting web pages by the user, which it stores in a queue. When the program is started it creates a number of coroutines, chosen by the user. When a coroutine is executed it removes a web page from the queue. The web page Bloom filter is then used to deter-

mine if the web page has been visited before. If it has not been visited before a HTTP request is sent. Once the HTTP requests is received the Beautiful Soup library is used to extract information. If the crawler is able to find a WooCommerce plug-in then the web domain is added to the SQL database, unless the web domain has already been added. The crawler then collects all URL's from the web page source code and adds them into the queue. When all the URL's have been added, the coroutine starts over and tries to remove a new web page from the queue. Coroutines are stopped when they wait for I/O requests and the event loop only starts executing them again once they have received data from their requests. The web crawler keeps searching through web pages until a predefined number, chosen by the user, of web pages have been added to the queue.

7.2 Scraping algorithms

This subsection describes the algorithms for the Allabolag [2] scraper and the WooCommerce scraper. The purpose of the Allabolag scraper is to find company name and revenue. The scraper is given domain name and the top-level domain from the database. With this information the scraper does a search on Eniro [20] for web shops. The first result is picked and the organization number, an identification number for Swedish companies given by the Swedish government, is extracted. With the organization number the scraper can easily look for revenue and company name on Allabolag since organization number is a part of the URL. However, if no organization number can be found on Eniro, another route is taken. The scraper searches with domain name on Allabolag and enters the first company link found in the search. To verify this is the right company, the scraper does a Google search with the company name. The first result's URL is matched to domain name and top-level domain retrieved from the database. If this is a match the revenue and company name are collected with help of the Beautiful Soup library [48] from the company's site on Allabolag.

The WooCommerce scraper identifies if a web page is built with WooCommerce plugins or not. It looks through all Javascripts on the web page to see if the phrase "WooCommerce" is included.

8 Requirements and evaluation methods

The requirements are developed in order for the program to be useful from a functional point of view. Three parts of the program: the crawler, the Bloom filters and a scraper, are tested individually with their own requirements. The requirements from our external

party, Crowderia, were that the program should be able to retrieve some web shops with information. With no other specific requirement we created our own requirements that we thought would make the program beneficial for other companies.

8.1 Crawler

The crawler needs to be able to visit many websites in as short time as possible. Since I/O requests usually takes a long time the crawler has to be able to do many request concurrently. We estimate that the crawler needs to be able to visit more than 100 web pages per minute to be useful when searching for web shops. After researching web crawlers, we found a program called Botify [8]. Botify is a widely used commercial product that works with marketing analytics and uses a web crawler that is able to search more than 200 pages per minute. Reaching half of Botify's speed, 100 web pages per minute, was estimated to be a reasonable requirement. Doing it manually, waiting for HTTP request and searching in the source code, would approximately result in 2-4 web pages per minute, when tested by us. This would mean our crawler would be 33 times faster than doing it manually.

To evaluate this requirement, the crawler will be run for two minutes at a time, starting from different websites each time. The amount of visited web pages is measured and an average of visited web pages per minute is calculated.

8.2 Bloom filter

In order for the crawler to work efficiently, it needs to know which web pages it has already visited. It is the purpose of the Bloom filter to keep track of the visited web pages. The Bloom filter needs to give a low percentage of false positives even when a large amount of web pages has been visited. The requirement for the Bloom filter is that it should have a false positive rate of less than one percent while it has less than one million elements in the set. A false positive rate of less than one percent means less than one out of a 100 websites will be lost, which was well above what Crowderia expected. When marketing it is not crucial to find all web shops, rather finding many in a short amount of time. This is why the time we gain on using this type of Bloom filter will result in more web shops in the end thanks to the efficiency of the filter. Practically this means that if we run the program for 10 minutes and it searches through 100 websites per minute we will visit 1000 websites. Out of those 1000 websites we might miss 10 websites, but that does not even mean that they are web shops.

The Bloom filter is tested by inserting a large amount of different pseudo random strings into the Bloom filter. The amount of collisions is then measured.

8.3 Scraper

The requirement for the scraping algorithm is that it should be able to identify the company behind Swedish web shops with over 70 percent accuracy. Crowderia had a requirement of 20 percent accuracy, a number they estimated to be enough for the program to be useful to them. We consider this number being too low, since we want other companies to find our program useful. However, we do not expect to be able to identify all the companies since there is no database containing all that information. Therefore, we have to combine different algorithms of identifying web shops. Since finding revenue is what makes our project different from other similar projects, the accuracy of the scraper should be a reasonably high number. Furthermore, being able to automatically identify 70 percent of a large quantity compared to doing it manually, yields a high number of identified companies and revenues in a short amount of time.

The evaluation is done by using the scraper on a database containing 79 Swedish web shops using WooCommerce [52] found by the crawler from Prisjakt [41].

9 Evaluation results

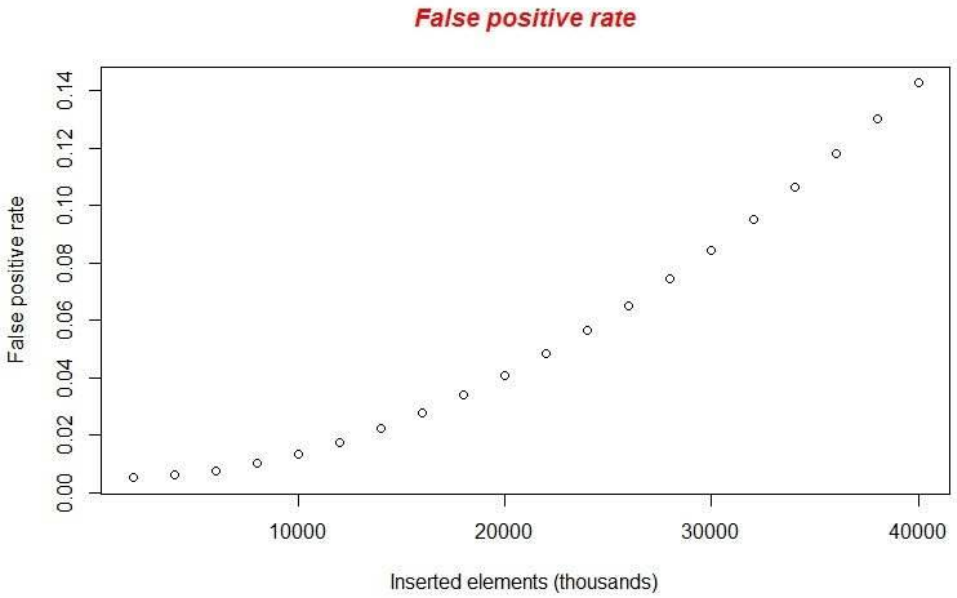

Figure 5: A graph showing false positive rate in relation to inserted items.

As seen in Figure 5 the false positive rate was below one percent when the set contained one million elements. The false positive rate did not increase above one percent until approximately five million inserted elements.

id	company_name	domain_name	domain_end	revenue
27	Yav Brand and Design Agency AB	crazydaisy	se	290
73	Vision It Varberg AB	visageonline	se	4183
8	Svegard Data AB	knivshop	se	2942
5	Strålin Foto Aktiebolag	fotoabild	se	5058
72	Snö & Tö AB	snoochto	se	2847
53	New Bubbleroom Sweden AB	naturligarum	se	124366
11	NAE Group Sweden AB	mdshop	se	30155
17	MILJÖGÅRDENS MÖBLER Aktiebolag	miljogarden	se	75713
45	Lidingö Hushållstiänst Sven Roslund Aktiebolag	hushallstianst	nu	17890
47	Letro Sport Aktiebolag	headlamp	se	8767
14	IC Distribution Sweden AB	poolteam	se	49331
77	Emanco Direktservice AB	texstyle	se	20516
69	Ellos AB	smartphonestore	nu	1834858
67	DiSara Coaching AB	sovdags	se	438
43	Card Collector Entertainer BOCHO Kommanditbolag	inlandet	se	NULL
79	BUY WATCHONLINE Sweden AB	watchonline	se	474
21	Avtramp AB	boardlife	se	3588
75	Vishram AB	vishram	se	NULL
76	Svdsec Bevakning AB	svdsec	se	23671
65	Svensk Villavärme AB	svenskvillavarme	se	7811
4	Styleport AB	styleport	se	NULL
63	Puurly AB	puurly	com	193
1	Pause Ljud & Bild Norrlandsgatan AB	pauseljudbild	com	15760
15	PASSION HOME AB	passionhome	se	4848
52	Maskinoruppen i Ängelholm Aktiebolag	maskinoruppen	com	231503
39	Lernberger Stafsing AB	lernbergerstafsing	com	18312
46	Klimathuset i Göteborg AB	klimathuset	net	5748
37	Jumpking Studsmattor Sweden AB	jumpking	se	7558
34	ITsale Sweden AB	itsale	se	3762
28	Folkets Möbler Järfälla AB	folketsmobler	se	3297
35	Eko-print Sweden AB	eko-print	se	2578
22	Bonti of Sweden Aktiebolag	bonti	se	27570
16	Aktiebolaget Sandbergs Järn	sandbergsjarn	se	11610
68	Adtraction Marketing AB	adtraction	com	124961
36	Adrecord AB	adrecord	com	15220

Figure 6: The picture shows all the companies where company name could be found by the scraper in the database containing Swedish web shops using WooCommerce.

The scraper was able to identify 44 percent, 35 out of 79, of the web shops. Of those 35 companies, 32 companies had a revenue registered on Allabolag [2]. This yields 41 percent accuracy for finding revenue.

The crawler was run ten times for two minutes at a time and it visited 1354 pages on average. In other words, 677 pages per minute which is well above the required 100 pages per minute.

10 Result and discussion

The scraper test showed that the scraper was able to identify the revenue of 41 percent of the 79 web shops. Since we had a requirement of 70 percent this was under our expectations. The original design was to use whois [51] to identify the company owning a web shop. Unfortunately, the search result on whois concerning Swedish top-level domains, such as .se and .nu, is copyrighted by iiS, which does not allow large or automatic data gathering [26]. We were not able to find any substitute for whois, which could provide the same kind of information. Therefore, information needed to be gathered from multiple other sources.

We created two algorithms, described in section 7.2, for this purpose. The reason for not being able to reach 70 percent accuracy was mainly because of two reasons. One being that the first algorithm, that uses the Eniro web shop search, was only able to find web shops registered by Eniro. The other reason being that the second algorithm, that uses the domain name to search on Allabolag, was not able to find companies where the company name differs greatly from the domain name. This depends mostly on how the search algorithm on Allabolag is designed. Other reasons could be that companies are inactive or in their startup phase, which makes it harder to find information about them.

The crawler was on average able to visit 677 web pages per minute, which was well above our expectations. We believe that we got an unexpectedly good result because of our choice to implement the HTTP requests using asynchronous IO. While there was problems with coroutines getting stuck at certain web pages, it did not have a big impact since the crawler uses many of coroutines. However, this would pose a problem if the crawler is intended to run for an extended period of time. The exact period of time is dependent on: computer hardware, chosen number of coroutines, Internet connection, and websites visited.

The Bloom filter performed according to our expectations and was able to achieve a false positive rate of less than one percent when containing up to five million elements, as seen in Figure 5.

11 Conclusion

The purpose of the project is to make marketing more efficient for companies and sole traders by streamlining the search for new customers who use a certain type of web shop plug-in called WooCommerce. This is achieved by creating a crawler that searches the

Internet for web shops built with the WooCommerce plug-in and storing what is found in a database. A scraper is then used to identify the companies behind said web shops as well as their revenues.

The result was a working prototype that is able to crawl websites with a speed of 677 web pages per minute, store all the web shops found using WooCommerce and find information, such as company name and revenue, for almost half of the Swedish web shops. To make the prototype ready for commercial deployment, improvements to robustness and accuracy are necessary. This would enable it to run for a longer period of time and improve the ability to identify companies that run Swedish WooCommerce web shops. However, it can be of use for Crowderia or other companies as it still fulfills its main purpose.

12 Future work

During this project we mainly focused on a solution for finding company name and revenue for Swedish web shops. The program could be expanded to find these attributes for web shops owned by companies based in other countries. An example of how to implement this is by scraping a corresponding site to Allabolag [2] called Company check [13] for the UK.

The program could be expanded to find more plug-ins than just WooCommerce. This could be done by adding more scrapers looking for other plug-ins or by expanding the already existing scraper.

In the current version of the program the algorithm for finding information such as company name and revenue is not sufficient. More algorithms need to be combined with the two already implemented ones to get a higher accuracy.

The scraper collecting information from Allabolag [2] is currently not running simultaneously with the crawler. The scraper is working directly with the database updating information. A better way would be if the crawler uses the scraper when a web shop is found and directly stores all information into the database. For this expansion, the asyncio library [47] used by the crawler needs to be implemented in the scraper as well.

To make the program user-friendly a possible expansion is to implement an application so the user does not have to use SQL queries to extract information. An application is needed if the database would be global since a global database is vulnerable to SQL injections. An advantage of having a global database is that it would require less storage

on the local computer.

References

[1] Alexa *The top 500 sites on the web* http://www.alexa.com/topsites (12 May 2017)

[2] *Allabolag* http://www.allabolag.se/ (13 May 2017)

[3] Bassett L. (6 February 2006) https://commons.wikimedia.org/wiki/File:Zygiella_web.jpg (12 May 2017)

[4] Beal V. *webopedia* http://www.webopedia.com/TERM/Q/query.html (28 April 2017)

[5] Berglund, A. Boag, S. Chamberlin, D. Fernández, M. Kay, M. Robie J. and Siméon, J. (14 December 2010) *XML Path Language (XPath) 2.0* http://www.w3.org/TR/xpath20/ (12 May 2017)

[6] Black J., Hashimzade N. and Myles G.(2017) *A Dictionary of Economics* http://www.oxfordreference.com.ezproxy.its.uu.se/view/10.1093/acref/9780198759430.001.0001/acref-9780198759430-e-928?rskey=XBcqm5&result=2 (21 April 2017)

[7] Bloom, H.B. (7 July 1970) *Space/time trade-offs in hashing coding with allowable errors* https://dl.acm.org/citation.cfm?doid=362686.362692 (15 May 2017)

[8] *Botify* https://www.botify.com/about/ (29 June 2017)

[9] Britannica ACADEMIC *search engine* http://academic.eb.com.ezproxy.its.uu.se/levels/collegiate/article/search-engine/396602 (9 April 2017)

[10] Built With (15 May 2017) *Ecommerce Usage Statistics* https://trends.builtwith.com/shop (16 May 2017)

[11] *Built With* https://builtwith.com/ (08 April 2017)

[12] Chandler D. and Munday R. (2016) *Dictionary of Social Media* http://www.oxfordreference.com.ezproxy.its.uu.se/view/10.1093/acref/9780191803093.001.0001/acref-9780191803093-e-1255# (11 May 2017)

[13] *Company Check* https://companycheck.co.uk/ (14 May 2017)

[14] Cormen, T.H. Leiserson, C.E. and Rivest, R.(20 August 2009) *Introduction to algorithms*, (third ed), MIT Press, Cambridge, Massachusetts.

[15] *Crowderia* http://crowderia.com/ (08 April 2017)

[16] *Crowd logistics* http://crowdlogistics.com/ (12 May 2017)

[17] Dang, Q. (August 2012) *Recommendation for Applications Using Approved Hash Algorithms* http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-107r1.pdf (15 May 2017)

[18] DB-engines (May 2017) *DB-engines ranking* https://db-engines.com/en/ranking (14 May 2017)

[19] Dustman A. *MySQLdb User's Guide* http://mysql-python.sourceforge.net/MySQLdb.html (13 May 2017)

[20] *Eniro* https://www.eniro.se/ (13 May 2017)

[21] Eppstein D. (15 August 2007) https://commons.wikimedia.org/wiki/File:Bloom_filter.svg (12 May 2017)

[22] Glez-Pena, D. (9 January 2014) *Web scraping technologies in an API world* https://doi-org.ezproxy.its.uu.se/10.1093/bib/bbt026 (11 May 2017)

[23] Google *Googlebot* https://support.google.com/webmasters/answer/182072?hl=en (12 May 2017)

[24] Guo, D. Wu, J. Chen, H. Yuan, Y. and Luo, X. (27 February 2009) *The Dynamic Bloom Filters* http://ieeexplore.ieee.org.ezproxy.its.uu.se/document/4796196/?reload=true (11 May 2017)

[25] Hastings L. (17 October 2012) *PEP 429 – Python 3.4 Release Schedule* https://www.python.org/dev/peps/pep-0429/ (13 May 2017)

[26] iiS *Regler och beskrivning av whois* https://www.iis.se/domaner/free/regler-och-beskrivning-av-whois/ (15 May 2017)

[27] Ince D.(2013) *A Dictionary of the Internet* http://www.oxfordreference.com.ezproxy.its.uu.se/view/10.1093/acref/9780191744150.001.0001/acref-9780191744150-e-3024# (21 April 2017)

[28] Ince D.(2013) *A Dictionary of the Internet*
http://www.oxfordreference.com.ezproxy.its.uu.se/view/10.1093/acref/9780191744150.001.0001/acref-9780191744150-e-735?rskey=zxruZO&result=1
(21 April 2017)

[29] Ince D.(2013) *A Dictionary of the Internet*
http://www.oxfordreference.com.ezproxy.its.uu.se/view/10.1093/acref/9780191744150.001.0001/acref-9780191744150-e-2557?rskey=RRaWVP&result=1
(21 April 2017)

[30] Knuth, D. (1997) *Fundamental Algorithms. The Art of Computer Programming 1*, (third ed), Addison-Wesley, Boston, page 193–200.

[31] Kotler, P. Armstrong, G. and Parment, A. (2011) *Principles of marketing*, (Swedish ed), Pearson Education Limited, Harlow, England, page 15.

[32] *Magento* https://magento.com/ (19 April 2017)

[33] *MySQL* https://www.mysql.com/ (09 April 2017)

[34] MySQL *Cross-platform compatibility*
https://dev.mysql.com/doc/mysql-enterprise-backup/4.0/en/platforms.compatibility.html
(7 June 2017)

[35] MySQL *MySQL 5.7 Reference Manual*
https://dev.mysql.com/doc/refman/5.7/en/ (17 May 2017)

[36] *Oracle* https://www.oracle.com/index.html (09 April 2017)

[37] Oracle *Java™ Platform Overview*
https://docs.oracle.com/javase/8/docs/technotes/guides/index.html (7 June 2017)

[38] Palach, J. (25 June 2014) *Parallel Programming with Python* Packt Publishing, Birmingham, UK.

[39] PostNord i samarbete med Svensk Digital Handel och HUI Research (2017) *E-barometern 2016 årsrapport* http://pages.postnord.com/rs/184-XFT-949/images/e-barometern-arsrapport-2016.pdf (24 April 2017)

[40] PostNord (2014) *E-commerce in the nordics 2014*
http://www.postnord.com/globalassets/global/english/document/publications/2014/e-commerce-in-the-nordics-2014.pdf (6 May 2017)

[41] *Prisjakt* https://www.prisjakt.nu/ (14 May 2017)

[42] *PyCharm* https://www.jetbrains.com/pycharm/?fromMenu (8 May 2017)

[43] Python *bitarray 0.8.1* https://pypi.python.org/pypi/bitarray/ (15 May 2017)

[44] Python *pyfarmhash 0.2.1* https://pypi.python.org/pypi/pyfarmhash/0.2.1 (13 May 2017)

[45] Python *Python success stories* https://www.python.org/about/success/tribon/ (7 June 2017)

[46] Python (1997) *Comparing Python to Other Languages* https://www.python.org/doc/essays/comparisons/ (8 May 2017)

[47] Python (7 April 2017) *18.5. asyncio — Asynchronous I/O, event loop, coroutines and tasks* https://docs.python.org/3/library/asyncio.html (12 May 2017)

[48] Richardson L. *Beautiful Soup Documentation* https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (29 April 2017)

[49] *Scrapy* https://scrapy.org/ (12 May 2017)

[50] Silberschatz, A. Korth, H.F. and Sudarshan, S. (2006) *Database System Concepts*, (fifth ed), McGraw Hill, Avenue of the Americas, New York, page 1-3.

[51] Whois *WHOIS Lookup* https://www.whois.com/whois/ (14 May 2017) l

[52] *WooCommerce* https://woocommerce.com/ (08 April 2017)