Evaluation of web scraping methods
Utvärdering av webbskrapningsmetoder

Different automation approaches regarding web scraping using desktop tools

KADDAY OUCIF
Evaluation of web scraping methods

Utvärdering av webbskrapnings metoder

Different automation approaches regarding web scraping using desktop tools

KADDAY OUCIF

Degree project in
Computer Engineering
First cycle, 15 credits
Supervisor at KTH: Jonas Wåhslén
Examiner: Ibrahim Orhan
TRITA-STH 2016:17

KTH
School of Technology and Health
136 40 Handen, Sverige
Sammanfattning

En hel del information kan bli funnen och extraherad i olika format från den semantiska webben med hjälp av webbskrapning, med många tekniker som uppkommit med tiden.

Den här rapporten är skriven med målet att utvärdera olika webbscrapnings metoder för att i sin tur utveckla en automatiserad, prestandasäker, enkelt implementerad och solid extraheringsprocess. Ett antal parametrar är definierade för att utvärdera och jämföra befintliga webbscrapningstekniker.

En matris av skrivbords verktyg är utforskade och två är valda för utvärdering. Utvärderingen inkluderar också tillväxagångssättet till att lära sig sätta upp olika webbscrapnings processer med så kallade agenter. Ett nummer av länkar blir skrapade efter data med och utan exekvering av JavaScript från websidorna.


Nyckelord
Webbscrapning, datautvinning, automatisering, semantiska webben, Business Intelligence, DOM parsing, HTML parsing, XPath
Abstract

A lot of information can be found and extracted from the semantic web in different forms through web scraping, with many techniques emerging throughout time.

This thesis is written with the objective to evaluate different web scraping methods in order to develop an automated, performance reliable, easy implemented and solid extraction process. A number of parameters are set to better evaluate and compare consisting techniques.

A matrix of desktop tools are examined and two were chosen for evaluation. The evaluation also includes the learning of setting up the scraping process with so called agents. A number of links gets scraped by using the presented techniques with and without executing JavaScript from the web sources.

Prototypes with the chosen techniques are presented with Content Grabber as a final solution. The result is a better understanding around the subject along with a cost-effective extraction process consisting of different techniques and methods, where a good understanding around the web sources structure facilitates the data collection. To sum it all up, the result is discussed and presented with regard to chosen parameters.

Keywords
Web scraping, data extraction, automation, semantic web, Business Intelligence, DOM parsing, HTML parsing, XPath
Foreword

I want to thank Booli for their hospitality and discussion around the subject. I also want to show gratitude to the company who gave me the opportunity to write this thesis. Lastly, I want to thank Jonas Wåhslén for his guidance throughout this project.
Innehållsförteckning

1 Introduction .......................................................................................................................... 1
  1.1 Problem objective ......................................................................................................... 1
  1.2 Goals ........................................................................................................................... 1
  1.3 Delimitations ................................................................................................................ 2
  1.4 The authors contribution to the thesis ........................................................................ 2

2 Theory and background .................................................................................................... 3
  2.1 Information about the project sponsor ........................................................................ 3
  2.2 The origins of web scraping ......................................................................................... 3
  2.3 Core knowledge of web extraction ................................................................................ 4
    2.3.1 Labeled ordered rooted trees ............................................................................... 4
    2.3.2 DOM and XPath ................................................................................................... 4
  2.4 The importance of wrappers ......................................................................................... 5
    2.4.1 Life-cycles and functions of a wrapper .................................................................. 5
  2.5 Approaches and techniques ......................................................................................... 6
    2.5.1 Most known techniques ....................................................................................... 6
    2.5.2 Automation techniques ........................................................................................ 7
  2.6 Data extraction systems ............................................................................................. 8
    2.6.1 Challenges ............................................................................................................. 8
    2.6.2 Wrappers in data extraction systems .................................................................... 9

3 Methods and results ........................................................................................................... 11
  3.1 Phase 1: Ways of dealing with the problem definition ................................................ 11
    3.1.1 Quality study: Filtering products and chosen parameters .................................... 11
    3.1.2 Possible tools to include in testing .................................................................... 12
  3.2 Phase 2: Evaluation of tools and testing .................................................................... 14
    3.2.1 Motives behind the filtering ............................................................................... 15
    3.2.2 Testing the solutions ........................................................................................... 15
      3.2.2.1 Prototype 1: Content Grabber ...................................................................... 15
        3.2.2.1.1 Extraction ............................................................................................... 15
        3.2.2.1.2 Automation ............................................................................................ 16
      3.2.2.2 Prototype 2: Winautomation ........................................................................ 16
        3.2.2.2.1 Extraction ............................................................................................... 16
        3.2.2.2.2 Automation ............................................................................................ 16
      3.2.2.3 Prototype 3: Copy and paste/Writing ............................................................ 16
    3.2.3 Results: Performance and accuracy test ................................................................. 17
1 Introduction

This thesis provides a solution for data extraction regarding web scraping and presents to you the different techniques used in this aspect. Different comparisons were also made between them with the help of the outlined parameters in this chapter.

A large percentage of the web consists of unstructured or semi-structured data. There is a lot of different services that displays data in different forms and with our society getting more and more digitalized throughout the years, obtaining data in various formats is a crucial way of providing different market needs.

The correct term to fetch data and present it in various formats is called web scraping and has risen to a lucrative business, making it possible to collect data and use it for everything from price comparisons to research. By obtaining the data needed, a service can be built offering data in different formats to customers.

1.1 Problem objective
Due to the growth of data sources for extraction as the business evolves, a performance-reliable, easy maintained and user-friendly framework is critical in maintaining market needs. Since the uprise of web scraping technologies, a variety of tools offering their services is available in the market offering similar results yet different approaches.

By studying different technologies and browsing for tools, an examination of different criteria was made:

- What datasets needs to be extracted?
- What tools suits the purpose?
- How do the techniques differ from each other?

The course of action was to set up parameters to outline test scenarios, making it possible to distinguish between possible solutions.

1.2 Goals
The company uses web extraction of data in multiple formats, by extracting information from HTA (Health Technology Assessment Reports) reports. The objective with the project was to evaluate the performance of different tools/technologies with regard to the following parameters:

- Performance evaluation/resource demand – how long does it take to obtain the data?
- Accuracy
  - Identifying data points – comparing hits between tools
  - Completeness of extraction – is data missing? If it does – how much is missing?
- Evaluate technologies and functionalities in relation to performance – what solutions are implemented to extract the data? The scraping techniques to get immersed in are:
  - HTML parsing
  - DOM parsing
  - More automatized solutions

A study about ease of use in programming and maintenance also took part. The project started off as a literature study, passing over to a quality study and quantity study.

1.3 Delimitations
The following delimitations arose:

1. The time-span for this project to follow through were short (10 weeks), considering a matrix of up to 5 tools was going to get evaluated – decreasing the options for the company to see the result for a variety of the tools.

2. A lot of these tools has trial versions of the software which makes the capability to perform tests insufficient without the full version. Some of these trial versions do have delimitations in functionality within themselves such as x hours to run the scraping agents.

3. Since the more commercial tools cost money, a delimitation of tools to choose from are a fact.

4. Some of these tools has customized solutions with confidential information – making it hard to examine the techniques used. A limitation to choose from tools to evaluate arises.

1.4 The authors contribution to the thesis
This thesis is completely written and compiled by the author Kadday Oucif. Introductionary meetings with a colleague at the company at the early stage of prototype development with Content Grabber occurred. Jonas Wåhslén, the supervisor to this thesis, pointed the literature study in the right direction.
2 Theory and background

This chapter summarizes general information around web scraping for a wider knowledge about the subject. A review with ways of extracting data was made, different techniques and approaches were listed along with basic knowledge about consisting web site structures. A short presentation about different automation approaches around web scraping can also be found in this chapter.

2.1 Information about the project sponsor
This thesis refers to the project sponsor as “the company” since it is a non-public company. It is a consulting company located in Stockholm, Sweden operating in the MedTech (Medical Technology) area. It is focusing on medical devices and vitro-diagnostics in Europe. Since it offers various services in the MedTech scene, one of these are the ability to offer information about HTA (Health Assessment Reports) [1], therefore the need for a smooth and easy-going web scraping functionality.

2.2 The origins of web scraping
Web scraping is also referred to as web crawling or screen scraping, thus the correct term for what it is differs from source to source (a good comparison can be found at [2]). As stated in 1.1, web scraping is often referred to the technique of pulling data from various semi-structured or unstructured sites and storing it in a relational form. Several studies have been made comparing earlier web extraction tools [3]. By referring to the word tools in this thesis, the meaning relies on different desktop software solutions (with a focus on combined techniques and approaches) aiding in the extraction and storage of data.

Originated in the early 90’s, two techniques based on Information Extraction [4] were made; learning techniques/learning-based approaches and knowledge engineering techniques/rule-based approaches. Differing from modern data extraction tools, the human factor of the extraction part was a necessity - requiring cultivators to have programming skills and a knowledge of the domains involved. Later on, a more automated response came in the late 90’s with more developed wrapper-functionalities. A list of some of the consisting approaches can be seen in 2.5.

Before the forced upcoming of APIs (Application Programming Interface), scraping was an easier way for biomedical companies to retrieve information from the web [5] than the human copy and paste technique and was set as a standard to retrieve information. Now the tables have turned, with SOAP (Simple Object Access Protocol) and REST (Representational State Transfer) being the main software-architectural implementations, still not resulting in web services having a total control of data interoperability. This due to numerous facts such as; costs of implementing programming solutions or application servers lacking the desired functions to present data.
2.3 Core knowledge of web extraction
A large amount of customized approaches has been embraced with today's tool regarding web extraction, although there is a lot of common factors. This section lists core components to know about regarding web site structures and web scraping.

2.3.1 Labeled ordered rooted trees
A largely known web structure is HTML (Hyper Text Markup Language) [6], also referred to as “labeled ordered rooted trees”, consisting of hierarchical nested semi-structured data. Fetching data from HTML labels takes less effort than extracting it from other structured web data, such as software codes or email addresses – hence the development of tools that requires less human interaction.

2.3.2 DOM and XPath
DOM (Document Object Model) [7] [8, p.33-34] is a heterogeneous interface consisting of HTML tags making it possible to get, change, add or delete HTML elements (including navigation through hierarchical HTML structures). Considering the functions mentioned, it is also shortly referred to as a programming API for documents. Interesting algorithms have been developed using the VIPS algorithm with a good description of how the DOM tree can be used to extract information [9]. Figure 1 demonstrates the structure of a DOM tree in tags and visually in blocks [10].

![Figure 1 – Hierarchy of a DOM tree.](image)

XPath (XML Path Language) [11] [8, p. 79-98] is a syntax made to address one or a set of elements in a XML document [12], with the same functionality in HTML web pages, containing over 100 built-in functions. Figure 2 demonstrates what it could look like [4, p. 304].

![Figure 2 – XPath expressions choosing one (A) or multiple (B) items.](image)
Working with XPath, an effort is made to make the expressions as generic as possible (thus more robust), lowering the chances of breaking if a potential website change occurs. More about robust wrappers in the next paragraph (2.4).

2.4 The importance of wrappers
A challenge for web extractors is to adapt to the changes of various websites, by making the XPath extraction expression as generic as possible. This can be done with a variety of tree change mapping algorithms, used to compare DOM trees related to the documents. Finding the least cost-effective DOM tree will make the expression as adaptable as possible to website changes [4] [13, p. 141-143]. This is demonstrated very well in figure 3 [14, p. 1-2] where we can see that W1 works for the first (A) HTML page, although it would probably break if the website structure would change (B). There is never a 100% robustness assurance to the wrappers developed, although the data extraction becomes less human-demanding with it. Interesting efforts of trying to develop a robustness framework has been made throughout time [14]. Tools such as Connote [15] solved this problem with a highlighting change detection ability constructed with a machine learning approach (more about these different approaches can be found in 2.5).

![Figure 3 – A demonstration of wrapper robustness.](image)

**2.4.1 Life-cycles and functions of a wrapper**
Wrappers is commonly referred to as the extraction part of semi-structured or unstructured data, adapting algorithms to make the conversion of the data as accessible as possible. It is partly divided into a set of functions in chronological order.

![Figure 4 - The wrapper stages in chronological order (although generation and execution is seen as merged together in some instances).](image)
As seen in figure 4, wrapper generation is the first in order and usually requires a high level of human interaction. Nevertheless, today's desktop tools have come a long way with wrapper inductive solution perspectives. These inductive solutions have been studied thoroughly, with one of the pioneers regarding this being Kushmerick [16, 17] [4, p. 302]. Various sources of induction approached studies have been made [18] and are very hard to construct from scratch, since it requires trustworthy programming skills. In the end, this makes the manual part of the wrapper implementation less demanding and sets up a more user-friendly graphical response to the web extraction. By offering more of a non-programming interface, making the extraction a lot more manageable, this is what most extractors are looking for – an automated, user-friendly and easy maintained web extraction tool. There is a set of different approaches to this solution in this phase before the wrapper execution part takes place.

At last the maintenance, which is a hard part (especially from a wrapper inductive stance) to maintain. It is quite overseen due to the big focus on generation and a quest is set to make the extraction method as adaptable and automated as possible. Due to the web sources changing appearance, thus forcing an manual update on the wrapper generation, schemas were made to counteract the high human interaction with this problem such as in [4, p. 308-309].

2.5 Approaches and techniques
Maintaining the scraping functionality can drastically increase in complexity the more data there is to collect. A set of approaches are used or combined in web extraction systems and languages.

Different approaches have emerged throughout time with Machine learning algorithms taking a larger part into web system extraction solutions (an example of this approach can be found at [19] made by two students from Stanford). Another system that uses Machine Learning/AI is Diffbot [20]. It uses computer vision web page analyzing to scrape the web, although the web scraping part in most systems adapts to the different techniques of extracting data with Regular Expressions/various parsing methods using XPath (such as Content Grabber [21]). A list of common approaches are discussed in [4, p. 306-308], however, one thing to bear in mind is that a lot of these approaches are merged into different solutions within a desktop tool.

2.5.1 Most known techniques

- Copy and paste
- HTTP programming – establishing connection with static and dynamic websites [5, p. 789] [8, p. 101-146].
- Text Grepping – by using the programming languages Perl or Python, the functionality of the UNIX grep command can be used to extract data. This is often combined with regular expressions [22].
• HTML Parsing – Since HTML content based websites are a majority, parsing data is very common. A lot of APIs and frameworks is developed using various parsing methods \(^1\).

• DOM Parsing – used by embedding a web browser to retrieve dynamic content from a website. This works good for JavaScript or Ajax-heavy websites, as the browser reacts as a normal end-user would.

• Computer Vision web page analyzing [23].

There are many customized solutions of extracting data from the semantic web, yet these are the ones mostly mentioned in web scraping solutions during the research of this thesis. DOM parsing and HTML parsing techniques are often helped out using XPath functionalities.

2.5.2 Automation techniques
As stated, tools such as Diffbot, Connoteate, UiPath or Winautomation promotes an AI/Ma-chine Learning approach also called RPA (Robotic Process Automation), which basically interacts with the computer through “software robots” replacing the human interactive part. Other tools in this area is Automation Anywhere and Blue Prism. This is a revolutionary technique set to replace the human involvement in procedures thus leveling the automation of a set of processes.

Since this report concentrates on the web scraping methods, RPA is a step further in data processing. It is basically a technique that replaces the human labor in different office environment tasks, working between different systems.

RPA differs from automation (and is not really the same thing as a macro [24]) in that aspect that it interacts with the computer as a human would and learns tasks from a human perspective, all by using a combination of NLP [25, 26], Machine Learning, autonotics and machine vision [27].

Figure 5 is an example of what RPA is capable of, comparing it with other consisting technology [27, p. 304]. The red box is interesting looking from a data extraction aspect, since the “robots” in an AI instance can understand and read data as a human end-user would. A robotic process automation software that adapts the technology of screen scraping (by seeing the characters from the screen as an end-user would) is UiPath, although this is another technique that differs from web scraping.

\(^1\) A RSSParser (Rich Site Summary Parser) for a Currency calculator app (made in the Mobile applications course – H1033) can be found in 8.2 using the XmlPullParser class. This functionality parses information from the metatags in the RSS feed, demonstrating the functionality a parsing library possess.
2.6 Data extraction systems

As mentioned in 2.5.2, the desktop tools make life easier for a lot of companies by extracting, storing and scheduling data automatically with the help of running scraping “robots” or “agents”. By offering IDEs (Integrated Development Environment), paying a price for a web extraction system is beneficial for the company in the long end considering they have a lot of web sites to extract data from and especially with companies who do not have any programming skills. It is also easier than training or hiring personal with programming skills to construct such a system, with wrapper induction generated queries constructed without any programming involvement (XWRAP [28] being one of the early popular choices). The article “Web data extraction, application and techniques” [4] evaluates the different uses and development of web extraction systems whereas it is mentioned that the wrapper generation and execution of wrappers is highly automated in these instances.

2.6.1 Challenges

The challenges with web scraping has largely decreased with intelligent web extraction systems. Easier tools had a hard time scraping from websites that uses Ajax [29] or JavaScript [30], although a good workaround to this problem in modern scraping is the DOM parsing technique. Another one of the major challenges is the change-detection as mentioned in 2.3.1. Challenges arises depending of the scalability and what to extract. Factors such as if it is ordinary screen scraping or web scraping HTML code plays in. Some of these challenges summarized can be seen here [4, p. 302].
It is also very important to accept the privacy of the website by examining the conditions in the RES (Robots Exclusion Standard), also referred to as the websites robots.txt-file [5, p. 790]. A very good practical guide can be found in [31].

2.6.2 Wrappers in data extraction systems
As this thesis focuses on the different highly automated instances in web scraping technology, dealing with wrappers is not as challenging. By examining the paradigms and reading the information written in [4, p.311-312], we can see that most commercial extractors relies on a bunch of approaches making them powerful tools with high scalability. Machine learning algorithms is also widely used in automation tools (although not necessary for the web extraction part).
3 Methods and results

This chapter presents the different methods of the literature, quantity and quality study. It is divided in different phases, comparing different techniques and approaches to an automated web scraping solution. Results is presented by making prototypes, measuring the reliability and validity in each scenario and can be found analyzed in the next chapter.

A timetable consisting of 3 phases were defined to this thesis, with an urge to stay in the time-frame offered to work with this.

1. Phase 1: Literature study trying to strengthen the logic and techniques behind web scraping. After that, a quality study will take place to define a list of possible tools to evaluate.
2. Phase 2: Evaluation of different solutions in a quantity and quality study.

Automated extraction processes are outlined in this chapter by creating agents, although it usually takes some training to program the agents – a service that most companies offer with purchasing the product. The plan with the whole project process was to download data in different formats (including PDFs and text) from HTA reports and store them in a database. Later on, this data were supposed to be processed and uploaded to a website as a service.

Because of the time-limit and since the PDF extraction was solvable by using different middleware, a focus on only horizon scanning was made when testing the tools. A matrix of 30 links were scraped and evaluated with the help of set parameters.

3.1 Phase 1: Ways of dealing with the problem definition

By structuring the work in phases, as stated in chapter 1, a literature study was the starting point. By studying what has been made in the area before, how the different techniques and approaches differ from each other, a clearer picture was painted regarding the web extraction methods.

To get a better understanding around the subject, a visit to Booli was made for approximately one hour and a half. A discussion regarding different aspects of web scraping started while an insight in how their solution benefited them were presented.

3.1.1 Quality study: Filtering products and chosen parameters

A lot of products evaluated are commercial, so an access to them were not always a privilege. This makes the product matrix limited, as the testing was made on the product licenses offered.

Apart from looking for a tool possessing web scraping abilities, an affordable price (although this would be a smaller issue if the business evolves) and a set of parameters was set to evaluate:
1. **Time of extraction** – how long it takes to extract and process the data, since it takes a long time to do this by human labor.

2. **Scalability of the products** - is it possible for the tool to process a lot of work and commands?

3. **Ease of use** – is there any need for a technical background to use this tool? Since there isn’t anyone with a programming background in the company.

4. **Extraction completion** – is any data missing from the data extraction?

5. **Automation** – is it possible to start the extraction and be completely away from keyboard until the extraction ends?

Since the tools use similar methods to extract the information (as stated in 2.4), the two main points for investigation were:

a) Comparing different techniques:
   a. What time-difference there is of doing a complete data extraction with a tool than doing it by human effort. This will of course be too time-consuming if there is thousands of links to get data from, so we will limit the number of links to 30.
   b. What time-differences and how many data point-hits can be found by comparing different extraction techniques?
      i. Since a lot of tools use different XPath approaches, a comparison between HTML parsing and browser embedded/DOM parsing techniques will be examined.
      ii. When is it optimal to execute JavaScript from the target web source when performing the extraction? What’s the benefits of it?

b) Considering that some of the tools differ in automation approaches, using different functionalities, a comparison between solutions of triggering “robots” or “agents” in chronological order will be made:
   a. Using the in-built RPA functionalities of triggering agents.
   b. Constructing a batch-file, scheduling tasks with Task Scheduler (which is .NET framework queuing tasks onto threads [32]) that runs every agent.

### 3.1.2 Possible tools to include in testing

Since the more complex solutions support a variety of formats to extract data from, this is the most relevant technical abilities. Table 1 focuses on different system requirements and technical capabilities which do not include Android or iOS applications (as some tools offer services through applications as well such as Import.io).
<table>
<thead>
<tr>
<th>CONTENT GRABBER</th>
<th>WIN AUTOMATION</th>
<th>KOFAX KAPOW</th>
<th>UI-PATH</th>
<th>MOZENDA</th>
<th>IMPORT.IO</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>APPROACHES</strong></td>
<td>Web automation</td>
<td>RPA</td>
<td>RPA</td>
<td>Web automation</td>
<td>Web automation</td>
</tr>
<tr>
<td><strong>TECHNIQUES</strong></td>
<td>Web scraping</td>
<td>Web scraping</td>
<td>Web scraping, screen scraping</td>
<td>Web scraping, screen scraping</td>
<td>Web scraping</td>
</tr>
<tr>
<td><strong>OPERATING SYSTEMS</strong></td>
<td>Windows</td>
<td>Windows</td>
<td>Windows, Linux</td>
<td>Windows</td>
<td>Windows, OS X, Linux, Android, iOS</td>
</tr>
<tr>
<td><strong>OPEN SOURCE</strong></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>DATABASE OUTPUTS</strong></td>
<td>Oracle, SQL server, MySQL, OleDB</td>
<td>Any database</td>
<td>Any database</td>
<td>Any database</td>
<td>Any database(^2)</td>
</tr>
<tr>
<td><strong>COST</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

*Table 1 - A table of web scraping tools with their technical capabilities.*

One thing to bear in mind regarding the Operating System supports of every tool is that there is workaround-methods used by downloading emulators (like Oracle VM VirtualBox \(^5\)).

---

\(^2\) Mozenda can be used in any database that supports .CSV, .TSV or .XML formats by using the data exportation functions.

\(^3\) No formal database support, although there are custom solutions such as in: [http://support.import.io/knowledgebase/articles/511605-extractor-to-sql-database](http://support.import.io/knowledgebase/articles/511605-extractor-to-sql-database)

\(^4\) Makes money by selling large data to enterprises.

\(^5\) An emulator used in the course Operating systems – H1025.
3.2 Phase 2: Evaluation of tools and testing

Another filtering of the products had to be done for the testing to adapt with the project timeframe (considering a matrix of maximum 5 tools was set as the objective in the early phase of the project). These tools can be seen in table 2 and are also chosen based on functionality, since they were downloaded on a trial basis. Using the criteria from [4, p. 310], an examination if the systems fit the purpose along with specifics of the system was made. All of the products fulfilled the needs for a web scraping system (automation, data transformation and use of extracted data), with an addition of criteria made by the author (A-F). This was to better evaluate the tools based on what needs the project process requires.

- **Automation** – Localizing elements and access to web pages.
  - A: Ability to perform non-programming extraction wrappers.
  - B: In-built function to trigger multiple agents/robots without time scheduling.\(^6\)
  - C: Ability to work between systems using the agents/robots (RPA approach).
- **Data transformation** – Retrieving the data in various choosable formats.
  - D: Ability to pull data from a PDF (Portable Document File).\(^7\)
  - E: Ability to process data into relational form.
- **Use of extracted data** – The possibility of data to process in a managing system.
  - F: Database support.

<table>
<thead>
<tr>
<th></th>
<th>CONTENT GRABBER</th>
<th>WINAUTOMATION</th>
<th>KOFAX KAPOW</th>
<th>UiPATH</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AUTOMATION</strong></td>
<td>A</td>
<td>A, B, C</td>
<td>A, B, C</td>
<td>A, B, C</td>
</tr>
<tr>
<td><strong>DATA TRANSFORMATION</strong></td>
<td>D, E</td>
<td>D, E</td>
<td>D, E</td>
<td>D, E</td>
</tr>
<tr>
<td><strong>USE OF EXTRACTED DATA</strong></td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Table 2 - Evaluated tools.

---

\(^6\) The idea here was to trigger an agent at a specific time, causing a chronological chain reaction of agents to deploy in a streamline of extraction patterns. Whenever each agent is done with its job, the next one should be set for its extraction process. This cannot be mistaken for setting up agent triggers at different times (for example, one at 13.00 and one 14.00), because of the fact that each extraction process could vary. For instance, this could depend on server and performance reliabilities. Figure 6 demonstrates the web extraction pattern for each agent.

\(^7\) There are several 3rd party solutions for this, although an in-built function supporting PDF-extraction without any human involvement is desired when considering the ease of use.
3.2.1 Motives behind the filtering
The two tools that fell out of the matrix of tools to further evaluate from table 1 was Mozenda and Import.io. Import.io sells on-demand data and had different customized solutions for functions like PDF extraction and database support. This makes it less attractive looking at it from an ease of use standpoint. Mozenda had a limited trial, where not every function was offered when downloading the trial version, unlike UiPath and Winautomation that are RPA products offering full functionality for a limited set of time. Also, since Content Grabber is a web automation product along with Mozenda, it was interesting to see how RPA products works with the desired extraction process. Content Grabber and Winautomation were very cheap unlike products like Mozenda, which means that the consultation that comes along with the tool is in a price range desired. Although Kofax Kapow and UiPath has an expensive pricing as well, these were downloaded on a trial basis (with Kofax offering a 2 week trial license) with no restrictions in functionality, making them open to investigate for future work.

3.2.2 Testing the solutions
Prototypes to test the performance and accuracy of every tool was made. Figure 6 demonstrates what the extraction process looks like visually in a chronological order [33]. By setting up 30 agents/robots for every extraction process, using horizon-scanning\(^8\) only, scraping up to 30 links (which can be seen under 7.4.1), without pagination. The text will be saved in relational form (Excel/.CSV-files) and compared. Definitions around words like HITS and DATA POINTS can be found in the Appendix under Acronyms (7.1).

Note. The tests were made after checking the robots.txt. Some of the websites did not have a robots-file and some of them had very unclear directives, so these tests were made on the intention to respect the RES.

![Figure 6- A visual description of the web scraping process for each agent.](image)

3.2.2.1 Prototype 1: Content Grabber
30 agents were made with Content Grabbers trial version, scraping a variety of links. The links were scraped using three scenarios.

3.2.2.1.1 Extraction
1. **Optimized dynamic browser** - Extracting data using the Optimized dynamic browser (no execution of objects such as Flash).

---

\(^8\) Horizon-scanning is the term whereas only text from the actual web page is scanned without any clicking through links or downloading files. Only proper web text extraction.
2. **Static browser** - Ability to run the agents [34] with a “static browser”. The static browser disables any execution of JavaScript and may not be working on dynamic websites.

3. **Static parser** - HTML parser/static parser that uses no browser, executes no JavaScript (just like the static browser) and just simply parses the metadata [35]. The difference between the static browser and the static parser is that the browser can handle iframes\(^9\), something that the static parser can not.

The browser used in the tool is a modified version of Internet Explorer. It is possible to read about these browser/parsing settings in [36, p. 186].

Every agent was set up using the Tree view function in the tool (which makes it possible to see the HTML structure of the page, this can be seen in figure 8 and 9). The tool makes it possible to visually click what data to extract and automatically generates the desired XPaths for the extraction.

*Pictures of the extraction process can be found under 7.4.2.1.*

### 3.2.2.1.2 Automation

A batch file with Taskviewer were set up to trigger the 30 agents after each other as the service only include web automation and this can be seen in figure 10. Results from every agent was presented in a relational table (Excel/.CSV).

*Pictures of the automation process can be found under 7.4.2.2.*

### 3.2.2.2 Prototype 2: Winautomation

30 agents were made with Winautomations evaluation version, scraping a variety of links.

#### 3.2.2.2.1 Extraction

Every agent was set up using Winautomations extraction function, which uses XPath and JavaScript. Figure 11 shows what it looks like when the extracted data is chosen, figure 12 shows the RPA functions for each extraction process and figure 13 shows a relational table of what the data will look like when extracted (an extraction preview).

*Pictures of the extraction process can be found under 7.4.3.1.*

#### 3.2.2.2 Automation

A job was made, structuring each job after each other in a chronological manner and this can be seen in figure 14.

*Pictures of the automation process can be found under 7.4.3.2.*

### 3.2.2.3 Prototype 3: Copy and paste/Writing

Information was extracted by copying, pasting and writing information from 30 different links to an Excel-file. This was done using a mouse and a laptop. Results will be presented in datasets as a relational table (Excel/.CSV).

---

\(^9\) Iframes are used to embed documents in the consisting HTML document. [33]
3.2.3 Results: Performance and accuracy test

Table 3 presents the different techniques tested for each weekday throughout a straight week. When comparing hits, a percentage with 2 decimals is counted with the last decimal rounded up to the closest number.

<table>
<thead>
<tr>
<th>WEEKDAY</th>
<th>TOOL</th>
<th>LINKS</th>
<th>AGENTS</th>
<th>RUNTIME</th>
<th>HITS</th>
<th>DATA POINTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monday</td>
<td>Winautomation</td>
<td>30</td>
<td>30</td>
<td>00:06:03</td>
<td>100%</td>
<td>4215/4215</td>
</tr>
<tr>
<td>Tuesday</td>
<td>Content Grabber – optimized dynamic browser</td>
<td>30</td>
<td>30</td>
<td>00:02:39</td>
<td>100%</td>
<td>4215/4215</td>
</tr>
<tr>
<td>Wednesday</td>
<td>Content Grabber – static browser</td>
<td>30</td>
<td>30</td>
<td>00:01:45</td>
<td>95.21%</td>
<td>4013/4215</td>
</tr>
<tr>
<td>Thursday</td>
<td>Content Grabber – static parser (HTML parser)</td>
<td>30</td>
<td>30</td>
<td>00:01:12</td>
<td>95.18%</td>
<td>4012/4215</td>
</tr>
<tr>
<td>Friday</td>
<td>Copy and paste</td>
<td>30</td>
<td>1&lt;sup&gt;10&lt;/sup&gt;</td>
<td>05:48:17</td>
<td>100%</td>
<td>4215/4215</td>
</tr>
</tbody>
</table>

*Table 3 - Results from the tests.*

<sup>10</sup> With the agent being the author, of course!
4 Analyze and discussion

This chapter starts off with an analysis of the different techniques used with a focus on the parameters chosen in the beginning of our methods section in 3.1.1. Later on, the current solution is presented followed up with occurring problems throughout the project. At last, the thesis subject is discussed in social, economic, environmental and ethical aspects.

4.1 Phase 3: Analyzing results; an inspection of set parameters

- **Extraction completion** – We can see in figure 17 at 7.4.5, as well as in table 4 (which is a numerical comparison between the hits) that the completion of each agent is much more reliable when executing JavaScript with a browser embedded technique, especially since a lot of websites presents dynamic content by using JavaScript in different forms.

The Optimized dynamic browser and Winautomation agents uses this technique apart from the static browser, that indeed uses a browser, yet disables all the JavaScript. There wasn’t a big difference between the static browser and the HTML parser when it comes to completion (although link number 30 did not work at all with the static methods). Nevertheless, we will see the benefits with these two in the time extraction parameter discussion.

<table>
<thead>
<tr>
<th>Winautomation</th>
<th>Optimized dynamic browser</th>
<th>Static browser</th>
<th>Static/HTML parser</th>
</tr>
</thead>
<tbody>
<tr>
<td>4215/4215</td>
<td>4215/4215</td>
<td>4013/4215</td>
<td>4012/4215</td>
</tr>
</tbody>
</table>

*Table 4 – Data point hits for every extraction process.*

- **Time of extraction** – There is no doubt that doing this by human labor is time consuming and very tiresome. By looking at figure 15 at 7.4.4, it is possible to see that links like number 13 and 7 (the links can be found under 7.4.1) took more than half an hour to collect data from instead of a few seconds. Although, by running the different agents, a pattern in time was discovered. The static techniques were much faster than the JavaScript-executing techniques. Even though no JavaScript were involved, a difference of 33 seconds could be noticed between the static browser and the static parser (see table 5). This could be crucial regarding the time efficiency as the HTA bodies increase. Another thing to bear in mind is that these tests were made without any pagination and this is another thing that could add to the time difference.
<table>
<thead>
<tr>
<th>Winautomation</th>
<th>Optimized dynamic browser</th>
<th>Static browser</th>
<th>Static/HTML parser</th>
<th>Copy and paste/Writing</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:06:03</td>
<td>00:02:39</td>
<td>00:01:45</td>
<td>00:01:12</td>
<td>05:48:17</td>
</tr>
</tbody>
</table>

*Table 5 - Time-span for each extraction process.*

- **Scalability and ease of use** – The scalability of every product holds up to the amount of agents set up. Almost every product looked upon could process a lot of data, so an important factor was the ease of use. Some of the systems were more complex (such as Kapow and UiPath) and were not included in the testing due to the time-limit. There were not enough time and consultation to review nor test these tools. A lot of RPA products have tons of functions to work with, which takes time and training to set up – making them hard to just dive in to. Although, all of the desktop tools are supposed to be easy to use with time, despite the level of technical ability a person possesses. With the right consultation and the right price, this could be beneficial in the end depending on what the extraction process requires (how many websites, what to extract, how to store the data et cetera). A review regarding the functionality, pros and cons of the products can be seen in 8.3.

- **Automation** - The two different automation approaches (the .bat file and the RPA functionality tested) is both functional in that aspect of running multiple agents in a chronological order without any human involvement. They were both easy to set up, although the Run Job-function (which can be seen under 6.3.2.) saves a bit more time and effort if there is a lot of agents to set up. Despite all that, these two products tested had no issues regarding the automation process needed for this project.

### 4.2 A review around the techniques used

By analyzing set parameters, we can see that executing JavaScript with an embedded browser makes the extraction more reliable while it also makes it a bit slower. It is good to know the layout of the website where the extraction is done to gather knowledge of how the data is presented and if it works with a static parser/HTML parsing, then this is the most time benefiting technique. A good approach is to combine different agents when setting them up, executing JavaScript with browser embedded techniques/DOM parsing when needed (especially for dynamic websites) and static parsing methods (preferably HTML parsing) as a standard for the best result.

#### 4.2.1 Current solution and unused solutions

A lot of solutions were not as attractive since they costed a lot or were too complex to learn in such short time. A lot of meetings in different forms (through Skype or GotoMeeting) occurred with companies like Connoteate, Kapow and Winautomation.

Since the HTA bodies are quite static (the layout stays the same for a long time), apart from the links changing from time to time, there was not a big focus on the change detection when choosing the tools. Tools like Mozenda, UiPath and Kapow were also shortly downloaded...
and evaluated on a trial basis. Mozenda was really easy to use, although the trial version lacked in functionality. UiPath and Kapow was much more complex RPA tools with a lot of functionality. Another service that fell short on the timeline was Scrapinghub, that is built on the Scrapy framework.

There may have been better ways to solve the extraction with tools like UiPath or Kapow, however, since nothing was paid for and the time was poor for consultation, the chosen tool to go with was Content Grabber. Winautomation was very easy to use, yet since there was no possibility to disable JavaScript, the time performance of the scraping functionality decreased.

Content Grabber is very useful as a main solution since it has the functionality to shift between these different techniques thus making the extraction less time consuming. It is also easy to use since no coding is required and has everything needed for a web scraping tool.

![Diagram](image)

Figure 7 – Data processing scheme. Extracting, storing and distributing data to the website.

The current and final extraction process throughout this thesis can be seen in figure 7 and each part in the solution is described shortly.

**HTA Body** – is the starting url from which the data is going to be pulled.

**Web scraping tool** - is Content Grabber in our current solution. What it does is that it traverses through links, extracts information and downloads PDF files. PDF files are converted [37] to HTML files and extracted in that manner since Content Grabber only work with HTML content. 11

**Database** – in this solution is SQL Server.

---

11 Since a time-difference were noticed between the techniques, an urge to use the static methods for every agent if possible is set.
Alteryx\textsuperscript{12} – a data processing tool that puts the extracted data into workflows.

Website – the data processed in Alteryx is then distributed to the website.

4.2.2 The different aspects of web scraping in society
Web scraping is looked upon as a good solution for data extraction, especially with the lack of APIs in some parts of the web as discussed in chapter 2. Although, you have to be careful where you retrieve your data, because of the fact that not every source is welcoming your presence with a scraping tool.

As we talked about the RES before, from an ethical stance, you should always respect it and this is considered a norm on the semantic web since it has no legal repercussions (although the robots.txt file could be used in cases [38]). Since many site requests could affect the performance of a server, some websites try to block out the ability to retrieve data with an automated tool, although this may be one of many reasons since the content holder is the one setting the directives. Another common issue is the content stealing from different websites, where different cultivators scrape information sources and presents it as its own. This could be compared to taking the literature of a book and writing your own name on it instead of the authors. Very unethical and not appreciated, thus being another possible reason for websites to disallow any scraping functionality.

It is hard to connect this subject to any environmental topics, yet by seeing the comparison of the data extraction by human labor and using different tools, we could see that it saves a lot of time and effort by doing it with automated tools. Apart from a possible fee for the product, it is also much more beneficial from an economic stance since you do not have to hire too much staff to do the tasks. The downside with this is that the fee could cost a lot depending on your business needs, although there are cheaper (or free) alternatives available on the market. Apart from that, it also generates different lucrative businesses and opens up different services discussed in the next paragraph.

The art of web scraping has done a lot for society, providing data in different forms and offering us services like Pricerunner\textsuperscript{13} or Booli\textsuperscript{14}. This do not only facilitate our knowledge acquisition, it also opens up doors to different businesses providing different market needs and information sources. This thesis is built upon a project that is trying to extract data from HTA bodies, which is healthy for the humanitarian work and healthcare in society. There are clear indications and results that web scraping has helped society to improve as whole by just looking at the different services offered.

\textsuperscript{12} http://www.alteryx.com/
\textsuperscript{13} http://www.pricerunner.co.uk/
\textsuperscript{14} https://www.booli.se/
5 Conclusions

This chapter concludes the thesis; describing what has been done through the project process and summarizing the parameters in a short manner. Later on, important inputs around this work is given and a perspective of possible future research points around the area is proposed.

This thesis evaluated and tested different web scraping techniques with the objective to get a better understanding around desktop web scraping tools and the fundamentals of how they work. Parameters around the reliability and validity of each criterion; scalability, automation, time, ease of use and data point completion was set up before the project started.

Time - it is most time efficient to use less factors when extracting data by trying not to execute JavaScript from a web page if possible and preferably using a HTML parsing technique.

Ease of use - this thesis presents different desktop tools, where the easiest one to use are the ones with completed functions. Factors such as no coding and a point and click-interface make them very beginner friendly. Both the tools tested (Winautomation and Content Grabber) were the one easiest to dive in to.

Data point completion - the JavaScript enabled DOM parsing/browser embedded techniques were the ones most successful by looking at the results of the test.

Scalability – All of the tools evaluated could process a lot of web extraction sources.

Automation – Two automation approaches were represented in this thesis; one RPA function-based approach and a .bat-file running with Task Scheduler.

The most important lesson learned with this thesis has to be one thing; there is tons of solutions for a good web scraping functionality. A lot of these tools do the same thing and extracts the data in a timely manner, it is all a matter of taste. A recommendation before settling down with a solution is to try out, research and outline the most important parameters. Sometimes the completion is much more important than the extraction time (for example, if you have less links) and sometimes it’s the opposite. It is also very efficient to know the structure of the websites (for example, if they are dynamic or static) to make the web scraping process much more automated and reliable when setting up the agents.

Web scraping functionalities is usually one step of a data processing solution, which can be easier to handle with tools that support RPA, although RPA is not necessary for a web scraping solution at all. Other future questions would be how to make the RPA functionality work after the web extraction part; is there any need for an automated data processing solution in the near future as the links increase and how is it possible to process the data in a workflow before publishing it to the website without any human involvement? The PDF extraction could also be researched through a quantity study; is it more profitable to extract the documents with a data scraping solution (like UiPath’s PDF scraper) than it is to convert it to HTML content and then extract it?
6 Bibliography


7 Appendix

7.1 Acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Meaning of abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTA</td>
<td>Health Technology Assessment</td>
</tr>
<tr>
<td>SOAP</td>
<td>Simple Object Access Protocol</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>RES</td>
<td>Robots Exclusion Standard</td>
</tr>
<tr>
<td>RPA</td>
<td>Robotic Process Automation</td>
</tr>
<tr>
<td>PDF</td>
<td>Portable Document Format</td>
</tr>
<tr>
<td>RSS</td>
<td>Rich Site Summary</td>
</tr>
<tr>
<td>DOM</td>
<td>Document Object Model</td>
</tr>
<tr>
<td>XPath</td>
<td>XML Path Language</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>MedTech</td>
<td>Medical Technology</td>
</tr>
<tr>
<td>FAQ</td>
<td>Frequently Asked Questions</td>
</tr>
</tbody>
</table>

**Dataset** – A matrix of data, for instance, an excel table with data presented in different columns/rows.

**Data point** – A data point is only the desired data from each extraction, not the whole dataset. In this case, it is each word/sentence we are trying to extract from the web source through the metatags [39].
Pagination — When the robot/agent automatically changes the page, for example, by pressing the next button while constantly extracting the desired data from each page.

Hits — A hit is when the desired data point gets extracted and stored.

7.2 Currency calculator application: RSSParser class
This is a class pulling data from a RSS feed constructed in Java. 15

```java
public class RSSParser {

    public static final String TITLE = "title";
    public static final String DESCRIPTION = "description";
    public static final String LINK = "link";
    public static final String ITEM = "item";
    public static final String PUBDATE = "pubDate";
    public static final String CUBE = "Cube";
    public static final String CUBE2 = "time";

    private XmlPullParser parser;
    private ArrayList<String> itemList;
    private ArrayList<ParserType> psetTypeList = new ArrayList<ParserType>();

    public RSSParser() throws XmlPullParserException {
        parser = XmlPullParserFactory.newInstance().newPullParser();
    }

    public ArrayList<ParserType> getParserType(){
        ArrayList<ParserType> psetTypeList = psetTypeList;
        ParserType psty = new ParserType();
        psty.setCurrency("EUR");
        psty.setRate("1");
        psetTypeList.add(psty);
        return psetTypeList;
    }

    public void parse(InputStream xmlStream, ArrayList<String> itemList) throws Exception {
        parser.setInput(xmlStream, null);
        this.itemList = itemList;
        int parseEvent = parser.getAttributeValue(0, "exchangeRate");

        while(parseEvent != XmlPullParser.END_DOCUMENT) {
            switch(parseEvent) {
                case XmlPullParser.START_DOCUMENT:
                    //
                
            }
        }
    }
}
```

15 Link that the data is pulled from: [http://www.ecb.europa.eu/stats/eurofxref/eurofxref-daily.xml](http://www.ecb.europa.eu/stats/eurofxref/eurofxref-daily.xml)
break;
case XmlPullParser.END_DOCUMENT:
    //
    break;
case XmlPullParser.START_TAG:
    String tagName = parser.getName();
    ParserType pstype = new ParserType();

    if(tagName.equalsIgnoreCase(CUBE)) {
        parseEvent = parser.getEventType();
        if(parseEvent == XmlPullParser.START_TAG){
            if(parser.getAttributeCount() == 1){
                pstype.setTime(parser.getAttributeValue(null, "time"));
                Log.i("bra","Parser:" + parser.getAttributeValue(null, "time"));
            } else if (parser.getAttributeCount() == 2){
                pstype.setCurrency(parser.getAttributeValue(null, "currency"));
                pstype.setRate(parser.getAttributeValue(null, "rate"));
            }
        }
    }

    pstypeList.add(pstype);
}
break;
case XmlPullParser.END_TAG:

break;
case XmlPullParser.TEXT:
    //
    break;
default:
    //
}
parseEvent = parser.next();
}
7.3 Evaluated tools
These are the tools that were evaluated and tested. The evaluation starts with general information about the product following with different pros and cons found with the products.

7.3.1 Solution1: Content Grabber
- Made by Sequentum who also made the Visual Web Ripper\textsuperscript{16}.
- Techniques: DOM parsing/HTML parsing with XPath and Regular expressions (with a possibility to make static extractions). [40]

7.3.1.1 Pros
- Content Grabber seemingly does the job, scaling numerous web sites in a timely manner.
- Ability to shift between different modes using static techniques. This could improve the performance scalability as shown in chapter 3.
- Affordable.
- Fast customer service.
- API.
- Good documentation.
- Specializes in web scraping.

7.3.1.2 Cons
- “Semi-automated” in the automation part outside web scraping, considering a batch file is made to trigger the different agents/robots. This is the current solution in the company. It works, although it could get done in a smoother way (preferably by using some in-built functions).
- Has a problem getting around protected PDFs.
- Only works with HTML content.

7.3.2 Solution 2: Winautomation 5.0
- An automation product using RPA.
- Available in 2 editions.
  - Professional edition – which includes the Job Compiler making it possible to execute jobs in multiple computers.
- Techniques: DOM parsing/HTML parsing with XPath and Regular Expressions. Although the data is only parsed with the help of a browser (DOM parsing) when testing the tool.

7.3.2.1 Pros
- Very easy to use, barely any technical skills needed.
- Affordable.
- Fast customer-service.
- Help-forum.
- Works with variables – making it easy to store data and perform different actions.

\textsuperscript{16} \url{http://www.sequentum.com/}
• RPA product – ability to work between systems.

7.3.2.2 Cons
• Lacking in documentation.
• No debug-system.
• Lacks in options:
  o Not possible to disable JavaScript execution.
  o Not very visual – lacks presentations of generated XPaths or tree views of web site source. Although, it has an extraction preview (figure 13, can be found in 7.4.3.1).
7.4 Testing
This chapter presents the different scenarios regarding the testing of the project.

7.4.1 Links used for each agent

<table>
<thead>
<tr>
<th>Number</th>
<th>Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><a href="http://www.aliexpress.com/category/5090301/mobile-phones.html?spm=2114.11010108.103.2,SJ0Ckr">http://www.aliexpress.com/category/5090301/mobile-phones.html?spm=2114.11010108.103.2,SJ0Ckr</a></td>
</tr>
<tr>
<td>2</td>
<td><a href="http://babynames.net/all/african">http://babynames.net/all/african</a></td>
</tr>
<tr>
<td>3</td>
<td><a href="http://babynames.net/all/arabic">http://babynames.net/all/arabic</a></td>
</tr>
<tr>
<td>4</td>
<td><a href="http://existenz.se/">http://existenz.se/</a></td>
</tr>
<tr>
<td>5</td>
<td><a href="https://www.euroscan.org/search?search=&amp;specialty=&amp;agency=&amp;tech-stage-dev=&amp;tech-stage-progress=&amp;date=&amp;date-updated">https://www.euroscan.org/search?search=&amp;specialty=&amp;agency=&amp;tech-stage-dev=&amp;tech-stage-progress=&amp;date=&amp;date-updated</a></td>
</tr>
<tr>
<td>6</td>
<td><a href="http://examples.winautomation.com/view_people">http://examples.winautomation.com/view_people</a></td>
</tr>
<tr>
<td>7</td>
<td><a href="http://www.fotbollskanalen.se/?tab=senaste">http://www.fotbollskanalen.se/?tab=senaste</a></td>
</tr>
<tr>
<td>8</td>
<td><a href="http://www.futhead.com/16/players/?bin_platform=ps">http://www.futhead.com/16/players/?bin_platform=ps</a></td>
</tr>
<tr>
<td>9</td>
<td><a href="http://www.has-sante.fr/portail/jcms/c_6056/fr/recherche-avancee?expression=exact&amp;expression=exact&amp;text=Saisir+vos+mots+cl%C3%A9s&amp;text=Saisir+vos+mots+cl%C3%A9s&amp;liaison_word=and&amp;searchOn=full-Text&amp;catMode=or&amp;dateMiseEnLigne=indexDateFrom&amp;dateDebut=&amp;dateFin=&amp;types=opinions%2Fgenerated_AVISMedicament&amp;search_antidot=&amp;portlet=e_39085&amp;sort=pdate&amp;replies=50">http://www.has-sante.fr/portail/jcms/c_6056/fr/recherche-avancee?expression=exact&amp;expression=exact&amp;text=Saisir+vos+mots+cl%C3%A9s&amp;text=Saisir+vos+mots+cl%C3%A9s&amp;liaison_word=and&amp;searchOn=full-Text&amp;catMode=or&amp;dateMiseEnLigne=indexDateFrom&amp;dateDebut=&amp;dateFin=&amp;types=opinions%2Fgenerated_AVISMedicament&amp;search_antidot=&amp;portlet=e_39085&amp;sort=pdate&amp;replies=50</a></td>
</tr>
<tr>
<td>11</td>
<td><a href="http://www.hsrc.ie/search/?search=&amp;specialty=&amp;year=&amp;sortf=date-desc#devices">http://www.hsrc.ie/search/?search=&amp;specialty=&amp;year=&amp;sortf=date-desc#devices</a></td>
</tr>
<tr>
<td>12</td>
<td><a href="http://www.hsrc.ie/search/?search=&amp;specialty=&amp;year=&amp;sortf=date-desc#diagnostics">http://www.hsrc.ie/search/?search=&amp;specialty=&amp;year=&amp;sortf=date-desc#diagnostics</a></td>
</tr>
<tr>
<td>13</td>
<td><a href="http://www.hsrc.ie/search/?search=&amp;specialty=&amp;year=&amp;sortf=date-desc#drugs">http://www.hsrc.ie/search/?search=&amp;specialty=&amp;year=&amp;sortf=date-desc#drugs</a></td>
</tr>
</tbody>
</table>
http://www.imdb.com/list/slist050274118/

http://kth-primo.hosted.exlibrisgroup.com/primo_library/libweb/action/search.do?fn=search&ct=search&initialSearch=true&mode=Basic&tab=default_tab&index=101&dum=true&srt=rank&vid=46KTH_VU1&frbg=&vl%28freeText0%29=web+scraping&scp.scps=scope%3A%28%28%24%29%29%2Cscope%3A%28%28%24%29%29%2Cscope%3A%28%28%24%29%29%2Cprimo_central_multiple_fe

http://www.nice.org.uk/guidance/published?type=jpg

http://www.pricerunner.se/cl/1/Mobiltelefoner

http://www.pricerunner.se/cl/1422/PlayStation-4-spel

https://www.scottishmedicines.org.uk/SMC_Advice/Advice_Directory/SMC_Advice_Directory?ds=Y&searchtext=Enter+keywords%3A&category=&submissionType=&from-Date=From%3A&toDate=To%3A&acceptedForUseCheck=Y&acceptedForRestrictedUseCheck=Y&notRecommendedForUseCheck=Y&withdrawnCheck=Y&supersededCheck=Y

https://biblioteket.stockholm.se/sok?freetext=Astrid%20lindgren

https://biblioteket.stockholm.se/sok?freetext=august%20strindberg

https://thepiratebay.se/top/400

https://thepiratebay.se/top/200

http://eurorivals.net/top-scorers.html


http://www.webhallen.com/se-sv/spel/playstation_4/action/

http://examples.winautomation.com/examples/view_people/2

http://examples.winautomation.com/examples/view_people/3

http://examples.winautomation.com/examples/view_people/4

http://2015.top100.winespectator.com/lists/
7.4.2 Content Grabber testing

This section presents the extraction and automation-process of Content Grabber visually.

7.4.2.1 Extraction

The generated XPath expression and a selection count can be notified in the bottom left corner.

![Figure 8 - Tree view functionality.](image1)

![Figure 9 - Tree view functionality.](image2)
7.4.2.2 Automation

![Batch script execution]

Figure 10 - Agentrunner.bat

This figure demonstrates the construction of the Agentrunner.bat file. Log_level is a command line that determines the log level detail (how much the log should document) and log_to_file saves the log file to a file instead of a database.
7.4.3 WinAutomation testing
This section presents the extraction and automation-process of WinAutomation visually.

7.4.3.1 Extraction

*Figure 11 - Extracting data with WinAutomation*

*Figure 12 - Highlighting data with WinAutomation*
<table>
<thead>
<tr>
<th>Value #1</th>
<th>Value #2</th>
<th>Value #3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avelumab treatment of non-small cell lung cancer metastatic stage</td>
<td>ICET</td>
<td>Date Added: 17/01/2016</td>
</tr>
<tr>
<td>Cytovir CMV for cytomegalovirus infection in patients undergoing allogeneic haematopoietic stem cell transplantation</td>
<td>NIHR-HSRIC</td>
<td>Date Added: 06/01/2016</td>
</tr>
<tr>
<td>Lenalidomide for the treatment of multiple myeloma</td>
<td>ICET</td>
<td>Date Added: 29/12/2015</td>
</tr>
<tr>
<td>Obinutuzumab for the treatment of chronic lymphatic leukemia</td>
<td>ICET</td>
<td>Date Added: 27/12/2015</td>
</tr>
<tr>
<td>Etrolizumab</td>
<td>ICET</td>
<td>Date Added: 27/12/2015</td>
</tr>
<tr>
<td>Afiblercept solution (Eylea) for visual impairment due to choroidal neovascularisation secondary to pathological myopia</td>
<td>NIHR-HSRIC</td>
<td>Date Added: 15/10/2015</td>
</tr>
<tr>
<td>Vincristine liposomal (Marqibo) for relapsed or refractory acute lymphoblastic leukaemia - third line</td>
<td>NIHR-HSRIC</td>
<td>Date Added: 15/10/2015</td>
</tr>
</tbody>
</table>

*Figure 13 - Extraction preview in Winautomation*
7.4.3.2 Automation

Figure 14 - Jobrunner method in WinAutomation. An automation scheme for the different agents that runs the different jobs in chronological order using the Run Job function.
7.4.4 Time comparison
The y-axis represents the time-frame (in seconds) for every scraped link and are represented in numbers on the x-axis. The URLs for each link can be found at 7.4.1.

Figure 15 - The estimated time to do this with human labor (time is presented in seconds)

Figure 16 - Time comparison between different techniques
7.4.5 Completion comparison
The y-axis represents the number of data points and the links (7.4.1) are represented in numbers on the x-axis. The blue bar is shared between Winautomation and the Optimized dynamic browser in Content Grabber, since they had a 100% extraction completion. CB is an abbreviation of Content Grabber.

Figure 17 - Completion comparison between tools