A general framework for scraping newspaper websites
Abstract

Data streaming nowadays is one of the most used approaches used by websites and applications to supply the end user with the latest articles and news. As a lot of news websites and companies are founded every day, such data centers must be flexible and it must be easy to introduce a new website to keep track of. The main goal of this project is to investigate two frameworks where implementing a robot for given website should take some acceptable amount of time. It is really challenging task, first of all it aims optimizing of a framework which means to put less efforts on something and have the same result and one another thing is that it will be used by professors and students at the end so quality and robustness play big role here. In order to overcome this challenge two different types of news websites were investigated and through this process the approximately time to implement a single robot was extracted. Having in mind the time spent to implement a single robot, the new frameworks were implemented with the goal to spend less time to implement a new web robot. The results are two general frameworks for two different types of websites, where implementing a robot does not take so much efforts and time. The implementation time of a new robot was reduced from 18 hours to approximately 4 hours.

Keywords: data streaming, news, articles, newspapers, web crawler, web site parsing, optimization, web robot, html, jsoup, selenium
Preface

I would like to thank my supervisor Jonas Lundberg who supported me during this project and teach me a lot of new things.
# Table of Contents

1. Introduction ................................. 6
   1.1 Background .................................. 6
   1.2 Previous Research ............................ 6
   1.3 Motivation .................................. 8
   1.4 Problem formulation .......................... 9
   1.5 Scope/Limitation .............................. 10
   1.6 Target Group ................................ 10
   1.7 Outline .................................... 11

2. Different Types of Newspaper Websites ........................................ 12
   2.1 Websites with Seeds ................................ 12
   2.2 Paging Websites ................................ 13
   2.3 Refined Problem Formulation ....................... 14

3. Method ........................................ 15
   3.1 Scientific Approach ............................ 15
   3.2 Method Description ............................. 15
   3.3 Reliability and Validity ........................ 16
   3.4 Ethical Considerations ......................... 17

4. Implementation ................................ 18
   4.2 Websites with Seeds – Algorithm .................. 19
   4.3 Websites with Seeds - Framework and SeedsSpecific interface .......... 21
      4.3.1 Class SeedsCrawler ......................... 21
      4.3.2 Interface SeedsSpecific ....................... 22
      4.3.3 Implementation Example ...................... 23
   4.4 Paging Websites – Algorithm ..................... 25
   4.5 Paging Websites – Framework and the PagingSpecific interface .......... 27
      4.5.1 Class PagingCrawler ......................... 28
      4.5.2 Abstract Class PagingSpecific ................ 29
      4.5.3 Implementation Example ...................... 29

5. Evaluation and Results ................................ 32
   5.1 Evaluation ................................... 32
      5.1.1 Robots evaluation .......................... 32
      5.1.2 Framework evaluation ........................ 32
   5.2 Results .................................... 33
      5.2.1 Framework test results for “websites with seeds” .......... 33
      5.2.2 Framework test results for “paging websites” ........... 34
5.2.3 Interesting data gathered during the tests 34

6 Discussion, Conclusion and Future Work 37
6.1 Discussion 37
6.2 Conclusion 37
6.3 Future Work 38

References 39
1 Introduction

This thesis project is just a part from one big system called “LNU Data Stream Center”. The following work is not focusing on the whole system because it is huge but instead the main goal during this thesis work was the web scraping and web crawling part of the system.

1.1 Background

Web crawling and web scraping are areas in the World Wide Web that are known for many years and have been evolved over time. At the beginning web crawlers were used only to collect statistics about the web but nowadays this concept is used even for finding vulnerabilities in specific website. The rapid evolution of the web affected directly the crawling process through the years and it is more complex to do it nowadays [4].

Web scraping on the other hand can be done without using web crawling, but the opposite way is not possible. Scraping at the beginning was mostly used by companies in order to get constantly competitor’s prices and keeping track of campaigns on the web [5]. Comparing to today in the early 2000s scraping was nothing more than just copying the whole content of a given web page, but nowadays scraping is much more than that. Web crawling is the process of browsing through a given website and extract all the fetching links starting from a specific list of URL’s [1] and web scraping is the process of collecting information from a web document [2]. Both of the techniques are used with different purposes and they are almost everywhere in the web, from search engines to web sites for finding the cheapest flight tickets. The term “robot” in this project is used to refer to an entity which combines scraping and crawling, as they usually are going hand by hand in such streaming systems. When it comes to the main goal of this work the “general framework” is a structure which will helps and guide later on the administrators of the data streaming center to build new robots within some acceptable time [3]. The general framework is a part from the LNU data stream center, and if we think about it as one "producer – consumer" scenario in this case the producer would be the framework. It is responsible for gathering the data and provide it to the consumer and this must be done continuously without any interruptions.

1.2 Previous Research

Web crawling and scraping have been discussed for many years and developed drastically through these years. A lot of research has been made but the difference between them is that, at the early 2000s researches have been focused more on the quality and correctness [6] of the information that is extracted rather than the performance, usability and maintainability.
Different researches aim different aspects of this huge area, but directly related projects to this topic has not been found, the closest one is done by three students: R. Penman, T. Baldwin and D. Martinez [7]. The research is more about the maintainability but there are a lot of similar approaches that has been used also in this project. The variety of web robots available on the internet is big, and about three of them have been analyzed with all the details, to check whether they are suitable candidates or not. Main requirements when choosing the candidates were the ease of use of the tool, secondly the correctness of the data and to see what it can do and what cannot. It was also checked about how close they are to the framework that is created during this thesis work because it is not available such framework that is 100% like the one implemented here.

The first tool is the famous Octoparse. Although this tool is not 100% the same as the framework we have built, it is the closest one to it. Octoparse can extract the specific data from given URLs but the first problem with it is that its web based and there is no API integration, so it is not a good choice to automate a website to extract data. Some other problem here is that in the general framework that is the product of this thesis work the user should just provide the starting URL of the website and that’s it, but in Octoparse there are a lot of steps that must be followed just for the scraping process, and even more time must be spent for the crawling. Basically it is not a general solution, it requires a lot of details in order to get the correct data and of course it is paid one. The next tool that was analyzed is Web-Harvest which is desktop software for scraping and crawling. First thing that can be noticed in this tool is that it does not provide 7/24 service, so the user should manually press the run button each time in order to get some data. Positive side of this tool is that the scraping and parsing process is really fast, the user should not wait too long in order to get the data but to come to that point a lot of configurations are needed in order to extract the correct information. The last candidate was ParseHub, it is a web based tool but it provides some API so the web version is ignored. ParseHub provides some good API but it is something really similar to Jsoup, the library that was used during this thesis work. These kind of libraries helps the user just to get some specific content from a web page, for example if you want to make something that is working nonstop you should again design a framework, some algorithms are needed in order make it work properly. It just provides some small pieces of code that can be used in order to create a general framework and that’s why this tool actually cannot be compared with this project.

All the tools available cannot be compared with the general framework that is created during this project mainly because of their purposes. All the available tools are with general purpose, but this project is focused only in newspaper websites, all the decisions and algorithms are made with having on mind that this framework is for newspaper websites. In order to achieve something similar like this framework about three other available tools must
be combined because all of them lack some functionality and of course all of them are paid.

1.3 Motivation

This thesis project is just a part of one big system called LNU Data Stream Center, which aims constantly streaming information to the users interested in the system. These users are primarily students and teachers, who want to keep in touch with all the news published in specific web sites. The system mainly consists of two parts: Subscriber side and Publisher Side. Subscriber side is formed by the people using the system – the subscribers, they can subscribe to different channels that they are interested in, and for example an economics professor might be interested in a channel that represents a website for economic news. Through subscribing to the system this professor makes sure that he or she will receive continuously the articles and news published in the website. On the other hand, the Publisher Side consists of robots that are responsible for deriving information from web sites. A role of a robot is mainly focused on the scraping and crawling a given web site and then form the extracted information in a structured way so that it can be readable for the end user. Having in mind the previous example with the professor, in order to accomplish this demand a robot must be implemented in the system which will crawl, scrape the given website and then structure the information in a good way. This thesis work is focused exactly on this part about the robots. The framework as was mentioned is just a part from the LnuDSC, which aims helping the researchers in Linnaeus University. Through this whole system students and professors will be able to follow the news and the articles in area that is specified by them. There are two main benefits of this whole project: the first one is that it will save them a lot of time in searching the articles that they want and the second benefit is that the subscribers will have the articles or news as soon as they are published on the web. The results are two general frameworks for two different types of websites, where implementing a robot does not take so much efforts and time.
1.4 Problem formulation

Two types of newspaper websites are in the scope of this project, so first they must be analyzed carefully. For each type an algorithm will be formulated in order to parse and crawl and then implement four robots with this algorithm – two robots for the first type and two for the second.

All the robots must run for 24 hours without any interruption or exception. Each robot will be unique and it must monitor a website 24 hours a day and new information will be send to the server. One robot will be responsible only for one given channel and maybe with some potential sub channels. The next step is to generalize the two frameworks so that when implementing a new robot minor changes should be done in the program.

In the research papers that we have investigated, there was no results such as implementation time or correctness of the gathered data. Yes, there are general frameworks but there is no research made based on the goals that we had from the beginning, one of them was to explore how easy would be implementing a new robot. In the research papers that we have investigated, there was no results such as implementation time or correctness of the gathered data. Yes, there are general frameworks but there is no research made based on the goals that we had from the beginning, one of them was to explore how easy would be implementing a new robot. The number of available research papers in general for web scraping and crawling is limited and the main reason for that is a lot of people consider them as illegal but this is not true and is explained in chapter 3.4. When it comes to the second research question it is focused on the running the framework 24 hours per day. As it is running continuously the new articles should be derived and here it is possible to use different algorithms to do that. There could be different solutions to this problem based on the needs of the framework, for example our solution was to store only the links of the old articles and if there is some new link with article then the framework is extracting the article from it and passes to the already scraped...
links. Here it is also crucial to use the right data structures in order to get efficiently working framework.

More detailed problem formulation will be presented in section 2, once we have presented the two types of website that we are interested in. Generally, this project is focused on newspapers websites, in the figure above it is the first type from the right (www.dn.se for example).

<table>
<thead>
<tr>
<th>RQ1</th>
<th>How a general framework can be designed for robots that are scraping and crawling newspaper websites?</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQ2</td>
<td>For each instances of the frameworks, how can we identify every new article presented there?</td>
</tr>
</tbody>
</table>

The end of this project must result in two frameworks where someone who is responsible for the system can implement a new robot within some acceptable time. During the process of development of these robots time measurements will be made to know about how much time is needed to implement a new robot without the generalized frameworks and then compare the results extracted through using the generalized frameworks.

1.5 Scope/Limitation

In Section 2 will be presented certain types of newspaper websites that will limit the scope of this thesis work. The project will focus on only websites that are consisting of articles and websites that are not based on the RIA (Rich Internet Application) concept. The architecture of RIA websites are identical to the one of the desktop applications and there is extensive use of scripting languages and more interaction with data that is residing in a server [10]. Almost all of the content on the website is controlled by scripts and deriving information from such websites is beyond of the scope of this thesis project.

Another limitation is the evaluation of these frameworks should be done with people that do not have any previous knowledge about them, but in this case it was not possible to do so.

1.6 Target Group

The project will be interesting for developers that are working on web scraping and web crawling projects and it will rise another point of view in them that they maybe have not thought before. There are a lot of independent developers that are building such crawling and parsing tools, it will be interesting for them of course to see such a framework or even maybe make them to redesign theirs’.
1.7 Outline

This section will be followed by the chapter about Different types of newspaper website which is about the two types of website that were investigated during the thesis. The next section is Method which covers the methodology used during the project. Implementation chapter comes next which is about how all these work is implemented. Evaluation and Results part is directly after the description of the implementations, and it will show the results and the evaluation of the whole work. Last but not least is the section called Discussion, Conclusion and Future work, where the report is finished by discussions about this work, conclusions about the work done overall and what can be done in the future?
2. Different Types of Newspaper Websites

During the project, two different types of newspapers websites were under investigation. The main and first reason about choosing exactly those two types was that most of news websites are following one of these structures. The web has of course variety of types like RIA based websites and others, but because of the time limitations the focus was mainly on these.

2.1 Websites with Seeds

The first type that was chosen is the website with seeds. It can be said that this is the most common structure nowadays that can be seen in the newspapers websites. This kind of websites differs from the others with its structure representing the news. It has the so called “seeds” which are actually the main URLs for different kind of news categories, here are some examples: http://www.dn.se/ekonomi/, http://www.dn.se/sport/, http://www.dn.se/kultur-noje/. These links represent the different categories in the actual website, and each new article is published in one of these seeds. This was made because of the user experience, so that when the user enters the website the desired category can be easily found. In this type of websites all the articles are following strict rules in their URLs’ structure. Here is an example: http://www.dn.se/ekonomi/sveriges-basta-robotar-ska-utses/, here the seed is http://www.dn.se/ekonomi/ which means that the article is about economics and then is followed by an identifier about which article exactly will be opened in the browser.

![Figure 2.1 Structure of a typical website with seeds.](image)

Assumptions: In the websites with seeds category are all articles published in the following structure: http://seed+identifier for the article. From this the
assumption that each article is reachable from a given seed is made. All the articles belong to some category in the website, in a website for daily news these categories (seeds) could be Sport, Politics, and Weather and so on.

2.2 Paging Websites

The second type that was explored is the “paging website”. Here the news articles are nested inside a list and the list usually is long so that it cannot be represented in a single page and that’s why it consists of many pages. The rule about the seeds strict structure is not applicable here, a given link extracted from the news list can have different structure compared to the other ones. An example of such website is the http://www.svensknaringsliv.se/english/publications/, here the latest articles are represented in the first page, but if the user wants to read some older ones scrolling to the next page is needed. The list consists of all the articles that the website has published, and the length of this list differs from website to website, more active website will have a longer list with articles and much more pages to scroll of course.

Assumption: Websites from type paging are listing their articles, so that if the user wants to check some older ones, he or she must scroll to the next page. All the articles can be found within some range of pages and usually the older articles are left to the last pages. Another assumption here is that when moving from one page to another one the URL structure is slightly changed (only some numerical index is changed). Last but not least, starting from the first page all the articles can be reached and every article of course can be found in one of the pages.
2.3 Refined Problem Formulation

Two types of news websites were chosen, because of the main reason that most of the website available they are following one of these function, even the first type with the seeds is the most used one. The other type with the paging is also common but not that much as the seeds type. Mostly more scientifically oriented websites are using the paging structure so that the users can easily found the old articles and to provide them some chronological ordering of the articles. The first research question is about building a framework where implementing each new robot should be easy and not time consuming. As during the analyzing process of different newspaper websites, it is defined that two general frameworks should be developed for the most common two types of websites. Research question 2 aims to show that each of the frameworks must be able to identify all new articles from a given website.

RIA type websites are not included in this thesis, as they are really complex in structure and consist of a lot of technologies and frameworks. First of all understanding all these technologies will take some serious time, because there the variety is really big, they are different languages used like JavaScript, Ruby, Python and so on. Its varying from website to website which of the languages and technologies are used, so for each website firstly the language that it is implemented on must be learned and then analyze the whole web page in order to do some scraping and crawling for this web site. Second issue here is that this kind of websites they hide some of the content, for example the user can see the article in the browser but in the HTML document the link referring to that article is hidden, and it is not possible to extract this link programmatically.
3 Method

Quantitative methodology was the best candidate for the successfully implementation and realization of the project. The results at the end will be numerically represented as the implementation times of new robots will be measured. Together with these values, interesting statistics will be shown that were gathered during the robustness test of each robot, where they were supposed to run 24 hours without any problem and interruption. Measuring the implementation time of a new framework was the approach that we have used, and it was most suitable in our scenario. The goal was to develop a framework which is easy to use and to not spend too much time in adding a new robot and the best way to check whether this is done was by measuring the time for implementing a new robot. Research question 2 actually a part needed to answer research question 1, because if the second research question is not implemented in a proper way it will affect the results of the first research question. An example of that could be if the framework is not detecting all the new articles, then even the implementation time is really short, it will not match the goals of this project. Who would need incorrect framework for web crawling?

Interesting statistics will be shown about the average article length (in words) and number of extracted articles in 24 hours in the websites that have been scraped and crawled. The data about the average article length is interesting because some people are not interested in crawling and scraping websites with really descriptive news. This is so popular in mobile applications, most of the news are really short there and the reason for that is mostly the user experience. The second statistics are about the number of articles published in 24 hours. Reason to publish this data is some people want to know how overloaded will be their databases, because in applications that are using web crawling and scraping all the data is being recorded continuously.

3.1 Scientific Approach

Empirical research methodology has been using in the project and quantitative data was gathered from the empirical experiments. The data is about the implementation times of robots and some extra data was collected during these experiments like average article length and numbers of articles published in 24 hours.

3.2 Method Description

The whole project overall is about implementation, experimenting and testing. Implementation part plays big role here, because the results and tests are highly dependent on the implementation. In order the main goal of this project be reached which in case is having two general frameworks where
implementing a new robot should take some acceptable time, implementation must be done extremely carefully. Implementation starts with building 4 separate robots – 2 for each type of website and evaluation of their performance which is followed by an analysis. The analysis includes studying the robots and identifying what they have in common, and what is website specific. Last step here is to design and evaluate a framework for each of the website types that makes use of what is common and highlights what is specific.

After all the necessary and core software pieces are implemented, to achieve the results, an evaluation process comes. This process is divided into two parts: robot evaluation and framework evaluation. The first part includes checking for the correctness of the extracted data from each robot, here the data is actually a JSON object which contains: title, content, published time of the article and from which URL it is extracted from. All these correctness verification are made manually through checking each extracted article and comparing with the content from the actual one in the website. Robustness test is also made for each robot, as one of the primary purposes of a robot is to be able to run without any interruption and error for really long time and in this part of the evaluation all the robots are ran for 24 hours. Second part of the evaluation is the framework evaluation, where the time for implementing a new robot using the framework is measured. For the whole evaluation is used empirical methodology and the obtained quantitative data was compared and analyzed.

3.3 Reliability and Validity

Ideal case – The ideal case of evaluating and testing this project would be quite different than the one used during the thesis. The goal here is to develop general frameworks so that can be used easily from other people with given some instructions. If such a project was done in a company, the best way to evaluate and test it would be through introducing the frameworks to developers who do not have previous knowledge about them and then measure the time needed to implement one new robot. The next step is to have another group of programmers solving the same problem without the use of the frameworks. The frameworks can be considered as a success if the group using the frameworks solves the problem significantly faster.

Current approach – The evaluation and testing in the thesis work was done only by me and my supervisor Jonas Lundberg. My supervisor’s role was to send me a website that I do not have any idea about at some specific time. I was acting like the person who is already instructed about the framework and was supposed to measure the time during the implementation
of a new robot for the chosen from my supervisor website. More details about our evaluation process is presented in section 5.1.

3.4 Ethical Considerations

There have been a lot of problems with companies that are doing the web scraping as a business and the laws about this vary from country to country. Back in 2000s a lot of companies were sued because they were scraping data from websites. Nowadays the same procedure is not followed unless some of the main rules about legally scraping a website are destroyed. Information about the laws in the United States were found during the research about the legal issues in web scraping and it is assumed that they are applied in every country. The first step that must be considered in the scraping process is that the scraping should not be in disruptive rate, which means the requests to the website should be made carefully with some reasonable time interval. The second point here is that the websites user agreement should be read and satisfied and last but not least the data extracted from the website should not be used with commercial purpose, for example the data extracted from there cannot be published in your own website. All these rules were strictly followed during the project and overall this thesis work does not break the rules about the scraping [11][12][13].
4 Implementation

The robot implementation is based on the producer – consumer pattern, both of the frameworks are using this approach. The producer in this case is the web crawler which is extracting articles and the consumer is just a mocked class that simulates consuming the produced JSON objects. The structures of the two frameworks are generally the same but they differ in content and usage, here is the general scenario for both of them:

![Figure 4.1 General scenario of a web robot](image)

The Main class is the starting point of the whole program, where a FakeConsumer and Crawler object are created. Here is an example code from the Main class:

```java
BlockingQueue<JsonObject> producer2consumer = new LinkedBlockingQueue<JsonObject>();
FakeJsonConsumer consumer = new FakeJsonConsumer(producer2consumer);
consumer.start();
SeedsSpecific specific = new TLSpecific();
SeedsCrawler producer = new SeedsCrawler(producer2consumer, specific);
producer.start();
```

The class starts with declaring BlockingQueue data structure which will store all the JSON object, which in this case are articles. Fake consumer is created straight after so that the whole process producer – consumer can be simulated, and after the creation the consumer is started and it is ready to consume new articles. Now comes time to the SeedsSpecific object which is a must in order to represent a website. The last part of the code shows the creation of SeedsCrawler object which will do most of the work in the program and it takes two parameters: The SeedsSpecific object and the BlockingQueue.

First of all, the project contains three packages that are used from both frameworks. Those packages contain classes and interfaces used with different purposes like: article creation, converting the article into JSON file
and implementation of fake consumer. In the following figure is illustrated the general packages, classes and the interface that are used from both of the frameworks. In the right side is displayed the shapes and their meanings for the rest of the report.

![Diagram of packages used from both frameworks](image)

**Figure 4.2** Packages used from both frameworks

4.2 **Websites with Seeds – Algorithm**

In this section the algorithm that was used for scraping and crawling websites with seeds is described. The algorithm generally consists of two parts: Initialization and Repetition.
The algorithm starts with the initialization part where a set data structure is initialized and filled with all the reachable links. After obtaining the available links the program checks all of these links whether some of them contain article or not, the idea is if the extracted link contains article, the article is derived from it. Sometimes errors occur in such situations, maybe the web
site is down for some reason or maybe the internet connection is slow so that is not possible to connect to the given URL. To handle such situations the algorithm in both of its parts uses an integer value that represents the maximum attempts that should be made to connect to the link. If such situation occurs in the initialization part the program sleeps some interval of time before the next trying, this is made in order to not cause the website some troubles. After the initialization part of the algorithm it is time for the endless loop – the repetition part. The robots are supposed to extract information continuously for without any interruption for years, and the repetition part is the place where the scraping and crawling process is done forever. It starts with some large sleeping interval for example in the robot implemented for the http://www.dn.se it was 5 minutes, because of the main reason that this whole loop will run forever so it must have some pauses between the iterations in order to not affect the website. After the sleeping part the algorithm again finds all the available links and compares them with the ones extracted in the initialization phase and if there is a link that does not match with the old ones, this link is added to newly initialized data structure from type HashSet. This data structure is used in order to store the links that are extracted during the iteration and it is passed as an argument to the method that goes through each of these links and checks whether it contains article or not, if it contains article, the article is extracted and saved. In both phases of the algorithm: in the initialization and in the repetition phase where a connection is trying to be made to some URL, the error handling technique described above is used in all of the situations, there is some limit that the program tries to connect to the problematic link.

4.3 Websites with Seeds - Framework and SeedsSpecific interface

The framework for implementing robots for websites with seeds consists of one package which will be used in each implementation of a web robot, and one package for the robot that is going to be implemented. The package called lnudsc.robots.seeds is used every time when a new robot is going to be added to the program. It contains one class and one interface, the class is called SeedsCrawler.

4.3.1 Class SeedsCrawler

This class stores all of the logic and methods that are crucial in order to do properly the crawling and scraping of a website with seeds, methods residing there are used in traversing reachable links, building and saving articles. This class is implementing the previously described algorithm in Figure 4.3 and remains the same all the time without being affected from the
website that is going to be crawled and scraped, in order to be able to recognize which website is going to be crawled and scraped, the class takes an argument of type SeedsSpecific and a BlockingQueue data structure where the extracted articles will be passed.

4.3.2 Interface SeedsSpecific

The interface inside the package is called SeedsSpecific, which is used when implementing a class that will represent some specific website, it contains methods that are overridden by the implementing class. This interface is a must for the robot, because exactly through using this interface the website that is going to be crawled and scraped is identified. It contains all the necessary information that is needed to represent a website uniquely. Here is in more details the SeedsSpecific interface:

```java
public interface SeedsSpecific{
    boolean isRelevantSpecific(String url); // Checks if the link is relevant
    Set<String> getSeedSet(); // Returns the seed set
    int getMaxPageRetry(); // Returns the maximum page retry
    int getMaxSeedPageRetry(); // Returns the maximum seed page retry
    String getWebsite(); // Returns the website
    int getIterationSleepTime(); // Returns the iteration sleep time
}
```

Beginning with the first method in the interface it plays really important role in the integrity of the system, basically this method checks a given link passed as argument whether is relevant to one that we are looking for or not and the result is returned as boolean value. The content of this method as well as the content of the other methods differ from website to website. In-built Java Pattern class is used inside all of the implementations of the method, where the patterns that we are not interested in are described and the program checks given link whether it contains or not some of the patterns. For example this project is focused only in articles but not in raw data regarding economics, in the http://www.dn.se website there is part that contains economic data which is called fondlistor, to prevent the robot from crawling and scraping this part from the website, in the pattern is included the keyword that is representing the part that should be avoided. Example follows of the isRelevant method and the pattern usage:

```java
public boolean isRelevantSpecific(String url) {
    if (DN_FILTER.matcher(url).matches()) {
        return false;
    } else if (url.startsWith(http://)) {
        System.out.println("Web reference: " + url);
        return false;
    } return url.startsWith("http://www.dn.se/");
}
```

```java
private static final Pattern DN_FILTER = Pattern.compile(".*\/\/(rss|webb-tv)/\.*\/fondlistor\/(rss|print)/css/\|\|pages\|\).*");
```
The next method `getSeedSet` is used to get all the seeds initialized before the execution of the program as a set and it is used in the initialization part of the algorithm. Methods `getMaxPageRetry` and `getMaxSeedPageRetry` are used primarily for the error handling as they represent the number of attempts that should be made if some error occurs with the URLs. The `getWebSite` is just used to return the name of the website that the class implementing this interface represents. Finally method `getIterationSleepTime` is representing the pause between the iterations.

![Diagram](image)

**Figure 4.4** The package structure used in each new robot

### 4.3.3 Implementation Example

When it comes to the second package, its name and content differ from site to site but the overall structure remains the same. To present the idea better, the implementation of a robot for the `www.dn.se` website will be explained. The package name is called `lnudsc.robots.news.seeds.dn`, the `seeds` part indicates that the robot is scraping and crawling a web site from type “website with seeds” and the next part shows the name or in some cases the shortage of the website. There are two classes included in the package: DNSpecific and DNMain. The DNMain is where the application starts from, it creates FakeJsonConsumer object just for simulating the whole producer – consumer pattern, which is followed by the DNSpecific object. The DNSpecific object represents the website `www.dn.se` uniquely and contains all the starting seeds available: `www.dn.se`, `www.dn.se/ekonomi`,...
www.dn.se/sport, www.dn.se/kultur, www.dn.se/sthlm, www.dn.se/ledare, www.dn.se/motor. This class implements the SeedsSpecific interface that was mentioned before, and it is a must to have a class that implements exactly this interface for robots that are going to parse and crawl websites with seeds. Finally the main class must be implemented and it should look like this:

```java
public class DNMain{
    public static void main(String[] args){
        BlockingQueue<JSONObject> producer2consumer = new LinkedBlockingQueue<JSONObject>();
        FakeJsonConsumer consumer = new FakeJsonConsumer(producer2consumer);
        consumer.start();

        SeedsSpecific specific = new DNSpecific();
        SeedCrawler producer = new SeedCrawler(producer2consumer, specific);
        producer.start();
    }
}
```

The DNMain class starts with the initialization of the `BlockingQueue` and the fake consumer where the queue will act as a pipe between the consumer (in this case the created fake consumer) and the producer (the `SeedCrawler`), all the produced articles will be passed to that queue and will be read from there. Straight after comes the creation of DNSpecific object which is representing the website’s unique values. The producer here is the `SeedCrawler` which takes two parameters, a queue and a specific object. The specific object tells to the producer which site should be crawled and parsed with all its unique attributes, and as mentioned before the queue is used to indicate where the extracted articles should be passed, so that they are ready to be consumed.

If a new robot is needed for some website with seeds the procedure will be like the following:

1. X – Abbreviation of the newspapers website
2. Create package lnudsc.robots.news.seeds.X
3. Create 2 classes – XMain and XSpecific
4. The XMain class will follow the same model as it was in the previous example and the XSpecific class must be differ with its content, the seeds for websiteX must be stored there.
4.4 Paging Websites – Algorithm

Unlike the Websites with seeds, here the algorithm has only one part and it is the repetition one, all the instructions described here in Figure 4.6 are repeating forever within an infinite loop.
Figure 4.6 Algorithm structure for websites from type paging
The approach for this kind of websites is different comparing to the ones with seeds. Here everything is based on CSS Selector. A CSS Selector is part from the rules of Cascading Style Sheets, where any content from the website can be selected through using CSS Selector, it contains pattern matching rules for example `select(#cars)` will result in output showing all the elements that has id cars. When it comes to the algorithm, it starts through visiting all the starting URLs that are initialized in the class called `PagingSpecific` which later on will be described in more details. The links there are stored in a `HashMap` data structure, where the key is a URL and the value is the CSS Selector that matches the links this page. During this process the algorithm also checks for each current link whether it contains a next page or not, and if it contains next page, recursively the actual method is called again. After extracting all the available links, it is time to scrape them. Each obtained link will be scraped if it is not marked as “scraped” or it is marked as “problematic”. Here is used an approach to handle the problematic links, for example if the connection times out or for some other reason is not possible to connect to the link, this link is marked as “problematic” so that can be checked in the next iteration again. When the URL is visited, the articles from it are extracted through using the CSS Selectors. This situation will be described in more details later on this section, basically in the `PagingSpecific` class is stored also the CSS Selectors for easily getting the article’s header, content and published time. Once the scraping process is done for given URL, the program marks is as “scraped” so that to not iterate over it again. Straight after a time interval comes where the program calms down for some minutes, to not affect the website’s server and then the whole process is done again as this is an infinite loop, all the time the steps described above are repeated.

4.5 Paging Websites – Framework and the PagingSpecific interface

Here the structure in general for the framework does not differ that much comparing to the framework for websites with seeds. The differences are mostly in content, but not that much in structure. For the websites from type “paging” the framework is wrapped by one package called `lnudsc.robot.paging`. It contains two classes which one of them is abstract. The first class is called `PagingCrawler`, it contains the all the methods that are crucial for crawling plus scraping paging websites and implements the algorithm described in Figure 4.6. Abstract class named `PagingSpecific` is used to represent a given website uniquely and stores all the information needed to do it.
4.5.1 Class PagingCrawler

`PagingCrawler` is responsible for both crawling and scraping given website. The algorithm described above is actually implemented inside this class and in order to do its job properly it receives a `PagingSpecific` and `BlockingQueue` objects as parameters. `PagingSpecific` here give information to the crawler object which website is going to be scraped and crawled. The `BlockingQueue` here again plays the role of a pipe where the extracted articles should be passed. Here is an example code from the crawler’s constructor:

```java
public class PagingCrawler extends Thread{
    PagingSpecific specific;
    private final BlockingQueue<JsonObject> jsonQueue;
    HashMap<String, String> obtainedLinks = new HashMap<String, String>();

    public PagingCrawler(PagingSpecific specific, BlockingQueue<JsonObject> jsonQueue){
        this.specific = specific;
        this.jsonQueue = jsonQueue;
    }
}
```

`PagingCrawler` is actually a thread and it contains the infinite loop, because as it was in the previous framework here the program will run also without any interruption for a long time and continuously supply the consumer with articles. The `HashMap` data structure is used in order to store the obtained links from the crawling process, it keeps the extracted link as key and the value here could be one these states: scraped, problematic or empty state – which means that this link has not been scrapped. Inside the `PagingCrawler`
the scraping and crawling process are based on CSS Selectors approach. For everything that is going to be derived from a website is used specific CSS Selector, and this selector can be easily obtained through the web browser with moving the mouse over the element that we are interested in and then with right click we can found the option called “copy CSS Selector”. Here also resides the methods that checks whether next button exists or not, so that crawler can turn to next page.

4.5.2 Abstract Class PagingSpecific

The previous text mentioned a lot about the CSS Selectors and the place where they reside is the PagingSpecific class. This class is used to describe the unique values for given website and all the specific information that is needed for the scraping and crawling a web page. Through extending this class it is a must to fulfil some requirements like initializing some attributes. This is the list of the attributes that must be specified for each new website that is going to be implemented in the program:

```java
public abstract class PagingSpecific {
    public String website;
    private int pageChangeRate;
    private String nextPageSelector;
    public HashMap<String, String> linksAndSelectors;
    private String titleSelector;
    private String contentSelector;
    private String dateAndTimeSelector;
}
```

Each class that is implementing PagingSpecific must provide these attributes, the abstract class of course contains some getter and setter methods but they are not shown in the example code here. Most of the variables declared are representing different types of selectors, for example selector for content, title, date and time that the article was published and so on. LinksAndSelectors data structure contains the links that the program should start the scraping from, these links are stored as keys, and on the other hand the value part is taken by the selectors. These selectors are identifying in which part of the stored links can be found sub links that are containing articles.

4.5.3 Implementation Example

Each new robot from this type that is going to be implemented is stored in a separate package. To explain the idea better example implementation will be shown for the Swedish newspaper website www.nyheteridag.se. The first step is to create a package where the whole implementation will reside. The package for this website is called lnudsc.robots.news.paging.ni, where
“ni” is the abbreviation of the name of the newspaper. In this example implementation and in the previous one, the point was that only specific object should be created when implementing a new web robot, here again the first class that is implemented is the NISpecific class. It is extending the PagingSpecific abstract class so that can represent the www.nyheteridag.se. When it comes to the content of the NISpecific it stores all the links that the program will start from like: www.nyheteridag.se/category/politik, www.nyheteridag.se/category/ekonomi and so on. These are the starting points and the program will start the crawling part from them. Together with the starting URLs the CSS Selectors for the links and for the other attributes like the content, title, published time. Here is an example code from the initialization part within the NISpecific class:

```java
private int pageChangeRate = 1;
private String nextButtonSelector = "#visaflerLink-ctl00_MainContent_newsFrontDaysList_ctl04_ctl00";
public HashMap<String, Strings> linksAndSelectors = new HashMap<>();
String website = "http://www.nyheteridag.se/";
private String titleSelector = "h1";
private String contentSelector = "#content-area p";
private String dateAndTimeSelector = "#post-area > p";
```

PageChangeRate is the offset that is used when changing from one page to another and this is different for each website, some websites they have changing rate 10 for example if the URL for the first page ends with 10 the second one will end with 20. As from the name can be guessed the nextButtonSelector identifies the next button in each link and usually the selector is the same for all the links that are going to be scrapped. Last step here is to build a main class from where the application can be started, for this example it is called NIMain. Here again first the fake consumer is created so that the program can simulate the producer – consumer scenario, and finally the PagingSpecific object is created and passed as parameter in the creation of the PagingCrawler object, and of course a BlockingQueue is needed in order to store the derived articles. In the end the whole NIMain class is looking like that:

```java
public class NIMain{
    public static void main(String [] args){
        BlockingQueue<JsonObject> producer2consumer = new LinkedBlockingQueue<JsonObject>();
        FakeJsonConsumer consumer = new FakeJsonConsumer(producer2consumer);
        consumer.start();

        PagingSpecific specific = new NISpecific();
        PagingCrawler producer = new PagingCrawler(specific, producer2consumer);
        producer.start();
    }
}
```
Figure 4.8 Implementation structure of the www.nyheteridag.se web site
5 Evaluation and Results

5.1 Evaluation

The evaluation consists of two parts, as the whole project can be separated into robots and frameworks in this section both of them are evaluated. This process is described in more details in the following sub sections.

5.1.1 Robots evaluation

The whole process starts here with one of the most important goals of this project – checking about correctness of the obtained data. Articles gathered from some specific website should be accurate and the full content of course must be presented. In order to make sure that the robots are extracting the correct data, the articles that were obtained has been checked manually one by one. During this verification period not only the article is self is checked but also the other attributes that form the JSON object that is going to be passed for consuming are checked, for example: the publishing time, title of the article and the URL from which it was extracted. Here is also checked manually whether the newly published articles are extracted in order to satisfy Research Question 2. The process cover checking the articles published in the website with ones that the framework obtained whether they match and if there are some articles missing or the articles obtained are more than the newly published. Once this whole evaluation is done it was sure that the project is on the right way, but this is just the beginning. Next step here was to test the robots for robustness. These robots in the future when the whole LNU Data Stream Center is ready will run for years without any interruption that it was really crucial to do such evaluation now. All the implemented robots were tested for robustness through running them for 24 hours without any interruption. Passing this test successfully showed that the robots are doing their job well and of course the error handling in their implementations is in a good level. Last but not least the speed of the robots is crucial here. An evaluation of it was also made through checking the time intervals that the robots visit the links. Controlling the speed of the robot is really important from ethics perspective, because if it tries blindly to connect to each given URL, and repeat it without any pauses this will cause problems to the website and of course to the person who is responsible for the robot.

5.1.2 Framework evaluation

The purpose of the frameworks from the beginning was to make the implementation process of a new robot easier and not time consuming work.
After implementing both of the frameworks three tests were made in order to prove that they are suitable. For the first framework which was handling the websites from type “seeds” two tests were implemented, and for the second type websites “paging” one test were realized. The procedure was as the following: at before appointed time a mail was received with the website that is going to be tested the framework, here the requirement was to test the framework with website that has not been crawled and scraped during the project. For achieving correct and accurate results, every step that has been taken during the tests were noted, of course with the starting and ending time of each step. The results can be seen in section 5.2. This was the main part of the evaluation, when it comes to details the maintainability of the framework was also a thing to consider. During the design process of the frameworks primary goal was to have good Object Oriented Design and easy maintainability of the frameworks. The skeleton of them are really easy to understand and if something is going to be changed it is not hard to understand which part of the code is doing what. Last factor here was to make it really easy implementing a new robot for given type and with minimum changes. This resulted in two frameworks where only one class should be added if a new robot is going to be implemented to the system and these are the classes described in previous sections, for seeds websites – SeedSpecific and for paging ones - PagingSpecific.

5.2 Results

5.2.1 Framework test results for “websites with seeds”

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:00</td>
<td>Test started</td>
</tr>
<tr>
<td>8:00-8:20</td>
<td>Analyzing the web site and walkthrough</td>
</tr>
<tr>
<td>8:20-8:42</td>
<td>Finding and extracting the seeds</td>
</tr>
<tr>
<td>8:45-9:38</td>
<td>Implementing BBCSpecific and Main</td>
</tr>
<tr>
<td>9:38-12:10</td>
<td>Testing and verification</td>
</tr>
</tbody>
</table>

Table 5.1 Test results for website www.bbc.com

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>11:00</td>
<td>Test started</td>
</tr>
<tr>
<td>11:00-11:15</td>
<td>Analyzing the web site and walkthrough</td>
</tr>
<tr>
<td>11:15-11:40</td>
<td>Finding and extracting the seeds</td>
</tr>
<tr>
<td>11:45-12:27</td>
<td>Implementing BloombergSpecific and Main</td>
</tr>
<tr>
<td>12:27-14:10</td>
<td>Testing and verification</td>
</tr>
</tbody>
</table>

Table 5.2 Test results for website www.bloomberg.com
Summary: It takes 4 hours and 10 minutes for handling the BBC and 3 hours and 10 minutes for Bloomberg. It is noticeable that most of this time is taken by the testing and verification process. These websites as well as the other that are used during the project contain a lot of articles, and it is time consuming to make sure that the gathered data is correct as this process is done manually. An average time of 3 hours and 30 minutes is needed to implement a new newspaper robot.

5.2.2 Framework test results for “paging websites”

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:00</td>
<td>Test started</td>
</tr>
<tr>
<td>8:00-8:33</td>
<td>Analyzing the web site and walkthrough and identifying page changing rate</td>
</tr>
<tr>
<td>8:33-9:45</td>
<td>Finding the CSS selectors for links, title, content, published time and next button</td>
</tr>
<tr>
<td>9:45-10:18</td>
<td>Implementing, NISpecific and NIMain</td>
</tr>
<tr>
<td>10:18-12:30</td>
<td>Testing and verification</td>
</tr>
</tbody>
</table>

Table 5.3 Test results for website www.nyheteridag.se

Summary: Here the implementation takes 4 hours and 30 minutes and again most of the time is taken by the testing and verification part. For this kind of websites analyzing the website part takes longer time compared with websites with seeds, because here the CSS selectors should be identified which actually means the HTML code must be checked. An average time of 4 hours 30 minutes is needed for implementing a new newspaper robot from this type.

5.2.3 Interesting data gathered during the tests

During the tests some interested data was collected about the different websites and their content. Here are some of them:
All the tests that are done have been tested for robustness which means running 24 hours without any interruption. During this time the robots collected all the available articles and check continuously for new ones. As from Figure 5.1 can be seen, the website with most articles is www.bbc.com, and the reason for that is BBC is widely known newspaper website and it includes news from big variety of categories. The website with least articles is the www.svensknaringsliv.se, it does not have too much articles because it is only about news about a business federation in Sweden.

Another interesting data gathered is about the length of the articles derived from some websites as shown in Figure 5.2. The longest articles are published in the website www.svensknaringsliv.com, it contains more
detailed information and there the average length of an article is about 600 words. In Figure 5.1 was shown that this websites has the least amount of articles in total but however the longest articles are published there. Here the website www.gsmarena.com contains the shortest articles, because it is a website for news from the phone industry and the news are really short there.
6 Discussion, Conclusion and Future Work

6.1 Discussion

Both of the research questions are answered now, so there is a general framework for implementing a new web robot and it is clear how each of these frameworks identify a new article. It was not possible to find any research related exactly to the research questions that were under investigation during this thesis work. The crawling method used for example in the seed framework is a well-known approach in general but all its implementations differ from program to program. Most of the available algorithms on the internet, they do not provide really good accuracy. There are libraries of course also available but the problem with them is that they are not flexible and of course not accurate, for example the design of the whole library is not well structure and understandable. A research made in Ottawa University [4] divides the website exactly to the same categories as in this thesis work, and they also end up with the idea that crawling RIA websites are totally different from the regular ones. One another thing that suits exactly with their research again is that it was not planned to end up with the same categories of web sites as they did in the research paper from University of Ottawa but through analyzing all the elements, implementations and trying to generalize everything this thesis work ended up almost with the same results as they did when it comes to categorizing the websites for crawling and scraping.

6.2 Conclusion

Conclusion for RQ1: The implementation section clearly show how the frameworks are built, and in the Results section is shown that implantation time for a new web robot is not too long about 3 hours and 50 minutes. As there are about two more categories of websites, for the time given for this thesis work the focus was only on the most common types – websites with seeds and paging ones. The frameworks are as general as possible and with some documentation everyone can use them. In order to use them is not expected to have some deep web knowledge, basically it is enough to know what HTML element, CSS Selector and URL is. Last but not least it should be paid attention here that in the results for implementing a new web robot, most of the time is spent in verification of the articles and analyzing the web page but not in the coding part.
Conclusion for RQ2: It is shown in the implementation part, where the algorithms for each types of websites are described that all the time the programs check the links obtained until now. For the websites with seeds implementation, the program stores the scraped links separately, and is checked each time when an URL with article is identified whether this URL is already stored or not. Almost the same approach is used for the second type of websites, but there the links are saved in a HashMap data structure. Each link extracted contains a value, it can be empty value – which shows that the link is new and not scraped, scraped – indicates that is already scraped and problematic - which shows that the link is problematic and because of some reason the program could not connect to it. The links marked as problematic are crawled again in the next iteration. Both approaches make sure that once the article is obtained from a link it will be not obtained again and only the new articles will be stored during the infinite loops in both of the frameworks.

6.3 Future Work

If the time given for this thesis work was longer, the first thing that should be done is to make one more framework but for RIA based websites. Scraping and crawling these websites differ from the other ones and some serious amount of time is needed in order to come up with some general framework. Last but not least it would be really good if the tests done during this project are done by two different kind of programmers. The first group should not be aware of the frameworks and then try to implement a web robot for given website. The second group will use the frameworks and try to do the same as the first group. In both of the tests again the time must be measured so that to be able to compare what time does it take implementing web robot with and without using the frameworks.
References


