A Web Scraper For Forums

Navigation and text extraction methods

SHIDI ZHOU

MICHAEL PALMA
Abstract

Web forums are a popular way of exchanging information and discussing various topics. These websites usually have a special structure, divided into boards, threads and posts. Although the structure might be consistent across forums, the layout of each forum is different. The way a web forum presents the user posts is also very different from how a news website presents a single piece of information. All of this makes the navigation and extraction of text a hard task for web scrapers.

The focus of this thesis is the development of a web scraper specialized in forums. Three different methods for text extraction are implemented and tested before choosing the most appropriate method for the task. The methods are Word Count, Text-Detection Framework and Text-to-Tag Ratio. The handling of link duplicates is also considered and solved by implementing a multi-layer bloom filter. The thesis is conducted applying a qualitative methodology. The results indicate that the Text-to-Tag Ratio has the best overall performance and gives the most desirable result in web forums. Thus, this was the selected methods to keep on the final version of the web scraper.

Keywords: Data mining, Web Scraper, Java, Web forums, Text-extraction, Link Duplicates
Abstrakt

Webforum är ett populärt sätt att utbyta information och diskutera olika ämnen. Dessa webbplatser har vanligtvis en särskild struktur, uppdelad i startsidan, trådar och inlägg. Även om strukturen kan vara konsekvent bland olika forum är layouten av varje forum annorlunda. Det sätt på vilket ett webbforum presenterar användarinläggen är också väldigt annorlunda än hur en nyhet webbplats presenterar en enda informationsinlägg. Allt detta gör navigering och extrahering av text en svår uppgift för webbskrapor.


Nyckelord: Data mining, Web Scraper, Java, Web forums, Text-extraction, Link Duplicates
# Table of Contents

1 Introduction 1
   1.1 Background 1
   1.2 Problem 2
   1.3 Purpose 2
   1.4 Goal 2
      1.4.1 Benefits, Ethics & Sustainability 3
   1.5 Methodology & Methods 4
   1.6 Stakeholders 5
   1.7 Delimitations 5
   1.8 Outline 6

2 Web scraper components: Navigation & Text extraction 7
   2.1 Navigation & Link Duplicates 8
   2.2 Text Extraction 10
      2.2.1 Text Density 10
      2.2.2 Page rendering 10

3 Related Work 12
   3.1 Link Structure & Multi-Layer Bloom Filter 12
   3.2 Text Extraction Methods 13

4 Methods and Models 16
   4.1 Data Collection and Analysis 16
   4.2 Interviews and Results 16
   4.3 Quality Assurance 17
   4.4 Development Model 17

5 Work tools and structure 19
   5.1 Work Tools 19
      5.1.1 Maven 20
   5.2 Program structure 21
   5.3 System overview 21

6 Implementation 23
   6.1 Multi-Layer Bloom Filter 23
   6.2 Text Extraction 24

7 Performance Results 26
   7.1 Performance Test: Speed & Size 26
      7.1.1 Computer Performance 26
      7.1.2 Server Performance 28
   7.2 Performance Test: Accuracy 32

8 Conclusions and Future work 34
   8.1 Discussions 35
      8.1.1 Speed performance analysis 35
      8.1.2 Text extraction performance analysis 37
   8.2 Conclusion 37
   8.3 Future work 38

References 39

Appendix A 42

Appendix B 50
1 Introduction

Communication has become much easier since the appearance of Internet. For example, people can video chat, text over the internet and even share their opinion on web forums. Thus making forums a source of valuable information. Forums allow people with interest in similar ideas and topics engage in discussion. It allows the exchange of information to be quick and simple. Gathering the information available in forums has much potential in fields like data mining [1] and machine learning [2]. Many companies such as Cloudera [3], [4] work with the mining of data by using it or providing services to collect it.

1.1 Background

Many programs can navigate the web in search of information and data. The application known as a crawler can be defined as “a program that visits Web sites and reads their pages and other information in order to create entries for a search engine index” [5]. On the other hand, a scraper (a web scraper is a crawler used for extracting data from websites) normally consists of two components, one for navigating through the web-links and one for extracting text information.

There are four basic steps involved in the process of web crawling. 1) The crawler starts by navigating the pages of a website. 2) It continues indexing the words and content of the website. 3) It visits the web-links that are found in the website. 4) The crawler returns the set of web-links to navigate and retrieve the information from those web-links, then the process of scraping can start [6]. Text extraction can be described as using a computer program to iterate through a web page, copy the text data in it and store it somewhere (most of the time the data is stored locally). The data then is used for any purpose that the user intended to do.

The web is built up following a set of rules defined by web technologies e.g. HTML (HyperText Markup Language is the standard markup language for creating web pages). Yet the implementation and layout of each website have no restrictions in design. In particular cases, structures of commonly used sites like forums can be difficult for crawlers to navigate properly. As Pavkovic et al. [7] state, the logical structure of web forums are the same no matter the technology used to implement it. It starts with a website, which has a forum list. The forum list includes what is called forum threads and each thread has a minimum of one post.

Figure 1. Side by Side view of different text layout (left side news [8], right side forum [9])
Based on figure 1 above, it is clearly seen that the structure of forums makes the crawling difficult. Unlike a typical news website, a forum will have short text (usually user posts) throughout the entire web page instead of a single focused text area.

1.2 Problem

Because of the structure of web forums, it is challenging and inefficient for the standard web scrapers to navigate these websites and retrieve information. Forums usually have a lot of people express their opinion on posts related to a topic in what is called a thread. The number of posts in a thread can range from a few to a hundred, while the date of the creation of the thread and the date of the last post in the thread could also range from minutes to years apart. This will result in the creation of an outdated post, which is a post which contents are no longer valid or have already been solved.

Besides the issues with the outdated posts, the structure and layout of the forums available online are significantly different from each other. The websites are built up and defined by web technologies such as HTML but the implementation and layout of each website has no restrictions in design.

The problem statement is thus: How to implement a web scraper that can navigate in forums and gather user posts with a good performance?

1.3 Purpose

This thesis presents the development of a web scraper specialized in extracting information in forums. This thesis paper therefore includes the different web navigation (such as duplicate web-link handling), text extraction methods used and the reasoning behind the choices of the implementation. This paper will thus help readers to gain an insight on different methods in order to design a web scraper.

1.4 Goal

The goal of this degree project is to design and implement a web scraper specialized in forums. The reason behind the goal of this degree project is because there are no existing web scraper that works well in web forums that do not require constant supervision to execute. There are scrapers like Visual Web Ripper [10] or import.io [11] that need a person to select which areas of a webpage it wants to extract, this process has to be repeated every time a new website is being scraped. Other scrapers such as scrapy [12] are multipurpose for any website, no special adjustment to handle the peculiarities of web forums.

To achieve the main goal, the project has to fulfilled two requirements:

1. **Navigate through web forums**
   The crawler should be able to crawl through forums, given that it gets a list of web-links as input to start. Since forums have a different structure and layout from other websites likes news and social media. Therefore the crawler should be able to navigate forum threads and each of the pages that belong in a thread, before it moves on to the next thread.

2. **Extract user posts**
   While the crawler navigates through the forum pages, it should be able to extract the text from the user posts. These posts contain a great amount of additional data besides the post
itself, such as username, signature, date and more. The crawler should only gather the
data belonging to the post and date. Thereafter it can either be stored on a database or
processed while running if it is combined with an analyzing program.

1.4.1 Benefits, Ethics & Sustainability

The main benefactor of this degree project is any company or individual that wishes to use a
specialized forum scraper, since the program is designed and implemented to perform better
than non-specialized web scraper when used in forums. Furthermore the code is open source
and the thesis paper is available online. Therefore, everyone who has a need to use a
specialized forum web scraper can benefit from it.

The three pillars of sustainable development are social, environmental and economic aspects.
The social aspects that are considered in this project are related to ethics, while the aspects of
environmental and economical sustainability are closely connected in the development of the
software. A brief description of each aspects is defined as following by the ministry of
education of New Zealand [13]:

Social
“Social – this aspect acknowledges the need for equity within and between generations, and
within and between ethnic and social groups. It is inclusive of people's mental and physical
well-being and the cohesion of their communities based on a fair distribution of resources [13].”

Environmental
“Environmental – this aspect acknowledges the need to enhance and maintain the biophysical
systems that sustain all life on Earth. It includes the structure and function of natural ecosystems
and the interactions between them and people, and calls for guardianship of our environment
[13].”

Economic
“Economic – this aspect acknowledges the interactions of humans with the natural environment
in using resources to create goods and services which add value to their lives. It acknowledges
the resource use and waste disposal must occur within the capacity of our planet. It encourages a
fair trading system that equitably distributes benefits and costs. It further encourages innovation
and creativity in developments that lead to a sustainable future [13].”

The ethics encountered throughout the development of this system are mainly related to how
the data mining will be handled. Usually in forums users do not have their real names as
screen names. Nonetheless the act of storing and collecting the information people exchange
could concern users about privacy even if it is available publicly. Although the posted
information is public, collecting that information without the user knowing about it is
unethical.

There are many factors to consider about sustainable development regarding to Software
Engineering [14]. System production aspect and System usage are also aspects that should be
considered when discussing the sustainability of software. All of these aspects are considered
when evaluating the economic impact of the software.

A computer's central processing unit (CPU) is one of the major energy consuming components.
CPUs have defined energy states as well. It makes sense that if the CPU is not actively
processing information or performing computations, it should be consuming minimal energy. Typically the CPU has two states, active state and idle state (which consumes very little energy compared to the active state). From a sustainable point of view, the software should aim to keep the CPU in an idle state as long as possible. Thus, the performance of the software in relation of the CPU can be considered in the environmental aspects of sustainable development.

There are several software energy efficiency techniques. Among these are the computational efficiency, data efficiency and context awareness that suits this thesis work. All of these techniques are important for the environmental impact from the software. Computational efficiency can be defined as “getting the workload done quickly, with minimal energy consumption” [15]. In other words, the aim is to make the program finish computing as fast as possible, to allow the computer to rest and conserve energy. Typically this is achievable by trying to improve the time complexity of the algorithm and the multithreading methods.

Data efficiency can reduce the energy cost by minimizing data movement and delivers performance benefits [15]. In this thesis work, data efficiency can be achieved by implementing algorithms that minimize the data movement and make the program efficiently use the cache memory. Algorithms such as bloom filter [6] can allow the crawler to avoid duplicate web-links and links that are useless (e.g. Web link for a picture). This minimizes the unnecessary data movement. The program can also keep a small cache of websites (e.g. the HTML file) that takes a long time to load online otherwise.

When developing a program, software documentation is very important. It is also something important to consider as an engineer. Documentation improves on the quality of a software product. It also plays a significant role in software development and system maintenance [16], which has both social and economic impacts in sustainability.

1.5 Methodology & Methods

Quantitative research methodologies always produce numerical data that is analyzed using mathematical methods. Qualitative research methodologies involves words or language and it can also include pictures and observations [17].

Qualitative methods focus on understanding the meaning, opinion and behaviour to develop theories or computer systems and inventions. For this reason the qualitative methodologies generally use small data sets and the data collection continues until it is considered necessary. Additionally, there are two main research approaches to consider. These are the inductive and the deductive [18] research approaches and they are used for drawing conclusions and determining whether something is true or not.

The inductive approach [19] is based on formulating theories and propositions from observation and patterns. It is also used for developing artifacts and computer systems. The data is collected with qualitative methods most of the times. Thereafter the data is analyzed to understand the views of the phenomena.

The deductive approach [19] is based on determining if a theory is true or false by testing. The data is collected with quantitative methods and it will usually be large sets of data. The data is used for testing a hypothesis. The hypothesis has to be defined in a mathematical and logical way such that it can be measured and operated together with the collected data used for the testing and the expressing the expected result.
For this thesis a qualitative methodology was applied together with an inductive research approach. For the collection of data for this thesis, interviews were used. The interviews were semi-structured with the use of both opinion based and knowledge based type of questions. Most of the interviews were done face-to-face and one time through an online medium, Skype [20]. The interviews were analyzed using a grounded theory method. Each interview being a set of iterations alternating collection and analysis of data.

For drawing conclusions the research approach taken was of the inductive type. This means that the data collected was observed to find patterns and propositions on how to develop the system. Gathering the views of the requirements that are important in a scraper to be used in forums. Thus defining the goals and requirements of this project, this is covered in more detailed in chapter 4.

1.6 Stakeholders

Omilytics [21] is a big data analysis company targeting the financial market. The company focuses on analyzing publicly available data and extracting accurate information on world events and upcoming financial trends. The company believe that there lies many opportunities within this area and that financial algorithms will have an even bigger part in future trading. Omilytics [21] was founded in February 2016 and has been stationed at KTH Innovation since October 2016.

There are over 3 billion [22] people connected to the internet, creating a stream of information which is not fully explored. Omilytics [8] believes that the company can apply this on the financial market. Omilytics choose to look at media in the form of news and social media, but also data created due to the internet of things. By performing sentiment analysis and looking at statistical data, omilytics can recommend whether to buy, hold or sell a security (a security is a financial instrument that represents an ownership position in a publicly-traded corporation stock).

1.7 Delimitations

First of all, the programming language used for this degree project is java, even though the stakeholder's program uses python. This creates problem when two program needs to be run seamlessly as one but a solution to this is issues was addressed by saving the files locally. The reason for this choice is because the authors' lack of knowledge in python. The extra work that learning a new programming language adds to the project is not justified to switch to python instead of continuing with java as a mean to fulfill the requirements. Since a local file can be processed as input by a program in python.

Secondly, access to the stakeholder's code for their program has not been given. Therefore, the data generated by the scraper will be stored locally, for post processing by the stakeholder.

Also, some web forums such as Morningstar's web forum page [23] uses a database to store all the comments. These forums then use programming languages like php (stands for Hypertext Preprocessor, it is an HTML-embedded Web scripting language) to retrieve the comments from the database and display them to the users. This makes it difficult for the scraper to retrieve the text information from that forum, because it is not in the HTML file, rather loaded from a web server which the scraper does not have access to.
In addition, people in forums often tend to quote a previous user’s post in forum when answering their question or express their opinion for or against the view of the previous user. This quoting function thus results in duplicated posts. This problem can affect the accuracy of the text extraction when using extraction methods based on text density. However, the development of a post cleaning function to eliminate duplicated forum posts is discard (due to the shortage of time for implementation as well as the fact that this is not the main focus of this thesis).

Furthermore, since the text extraction component of the scraper will gather the text data from public websites, the security aspects are therefore ignored to a large extent (since this means that anyone can access this information online).

1.8 Outline

This paper first start with presenting the literature research in Chapter 2. This chapter presents the literature research, generally introduces different ideas and studies about the web crawlers and scrapers.

Chapter 3 presents related work.

Chapter 4 explains the research methodology and methods used in this degree project.

Chapter 5 gives general descriptions about the work process in this degree project. This includes the documentation about the work structure of the web crawler.

Chapter 6 describes the implementation of the Multi-Layer Bloom Filter and text extraction methods used in the project.

Chapter 7 presents the result. This includes the data collected when testing the scraper program.

In Chapter 8, discussion and conclusion is given based on the result. A discussion is given in detail about the work and its outcome and compares the performance of the different text extraction methods to each other. Finally, a final conclusion is given to the whole degree project and address the problems that have been solved, as well as the ones that have not been solved yet.
2 Web scraper components: Navigation & Text extraction

A web crawler is as defined before [5] as a program that visits links and gathers the information from these links to create entries for a search engine index. Usually collecting the relation between the links it visits, if there are any.

A web scraper is used for extracting data from websites. Although web scraping can be done manually by a software user, it usually refers to the automated processes implemented using a bot [24] or web crawler [5]. This is a form of copying, in which specific data is gathered and copied from the web, typically into a local file, database or spreadsheet, for later retrieval or analysis. From this scraper definition, it can be divided into two processes: one process that navigates through the web links (the web crawling component or navigation component), and another process that extracts text information from these links (text extraction component).

Web scraper overview

For link navigation, a web scraper takes in a list of web-links as input in order to start crawling (this list is usually referred as the seed list). The web crawling component will then start generating a list of web-links to be fetched (after filtering and normalization according to predefined configuration rules). The scraper then fetches or enqueues all the web-links of the fetch-list. The scraper will then download their content one by one, parse the content of the web-links and update the list of new links [6]. After that, the crawling component will put the new web-links back to the fetch-list, and the process iterates. These web crawling steps are illustrated in Figure 2.

![Figure 2. Web crawling steps (drawing by the authors)]

After the scraper returns a set of links which it will navigate and retrieve information from, it can prepare to start the text extraction process of the web scraper. Text extraction is a process that iterates through web-links and copies the text data available in it. This process will most
times store the text data wherever the program is designed to store it in (most of the time the data is stored locally). The data collected from these links can then be used for any purpose.

The issue that arises with crawling and extracting text from web forums come from the structure of forums. Forums starts with a website, that has a forum list. The forum list is formed by elements called forum threads, and each thread has a minimum of one post. Post can direct to other posts and to other threads, all of this makes the crawling difficult. Also, the layout and placement of the text in a forum compared to a website such as a news website makes it complicated to collect. A news website has usually only one area where the main content is found, compared to a forum where the content is distributed into user posts spread from the top of the page to the bottom of it, usually across several pages.

2.1 Navigation & Link Duplicates

One of the issues with web crawling is handling link duplicates. Although forums have many threads leading to unique and different links, there is a case in which it could collect many duplicates (the case being when the scraper is navigating a thread).

When a scraper is in a thread, it retrieves the web-links of all the page numbers available. For example, current page in figure 3 is on page 2 and the crawler component retrieves the web-links of pages 1, 3, 4 and 5. Then it will go to the next page (page 3) and it will retrieve the web-links of the pages 1, 2, 4 and 5. In this case the crawler has picked up the a duplicate of the web-links 4 and 5 that are not visited yet and the web-link for page 1 and 2 which are already visited. This could lead to a wasteful and endless loop of using large amounts of network bandwidth and hardware resources. Thus having a solid link duplicate prevention is necessary for the crawling to succeed in a web forum. There are a few methods to handle this problem as described by Kapoor et al. [6], two examples are brute force methods and bloom filter methods.

![Figure 3. Thread page navigation menu (generic forum)](image)

**Brute Force Method**

The easiest method to detect duplicates is to have a list of “visited” web-links. Once the crawler gets a new link, it will traverse through the “visited” list and see if it can find a match. This is the so called brute force method. There will be no error with duplicated links with this method [6], however it is a lot of work for the crawler and it is very inefficient regarding the space and time aspect. People have some improvements based on the brute force method, such as Ye Yun Ming and Yu Shui from Shanghai JiaoTong University [25], who developed a web crawler named Igloo [25] using Trie tree to store web links. Trie tree is a tree structure used to save a large amount of strings. It is a very space and time efficient way to store web links, compare to the original solution. Igloo stores two different kind of data, the string of web-link and the metadata of the web link. The string of web-links is stored as a path of nodes and metadata is stored in that last node. A simple visual representation is provided in figure 4.
Bloom filter method
Bloom filter, which was presented by Burton Howard Bloom [26], is a space-efficient data structure used to represent a set. It is used to test whether an element is a member of a set. A Bloom filter consists of two parts: A vector of N boolean values, initialized as false in the beginning. And K numbers of independent hash functions (H1,H2,H3...Hk) with the result range from 0 to N-1. For each of the element s in Set S, the boolean value in position H0(s),H1(s)...Hk(s) are set to true. To determine if an element is in the set, simply use the hash functions to calculate the positions in the Boolean Vector and see if all the values in the corresponding position are true. If yes, then it is in the set, vice versa. A simple visual representation is provided in figure 5.

Figure 5. Classic Bloom Filter (drawn by author)
Using a bloom filter will result in some false positives, but no false negative can occur. This means that there is always a chance that the link is a duplicate when the bloom filter returns positive, but when it returns negative, the link is definitely not a duplicate. Unlike the traditional brute force method, the required space does not matter with the size of an element, rather it depends on the size of the boolean vector. Therefore, a Bloom filter is suitable for space restriction situations with allowable errors.

The web crawler of the digital library Internet Archive [27] uses the original bloom filter [28] to detect duplicate web-links. But there are other variations that further improve the bloom filter for duplicate web-links detection, decreasing its false positive rate. Cen Zhiwang, Xu Jungang, Sun Jian in Graduate University of Chinese Academy of Science [29] presents a duplicated web-link detection method for web crawlers, which is based on Multi-Layer Bloom Filter algorithm.

2.2 Text Extraction

The two main approaches for extracting text: 1) calculating text density of the HTML source code 2) visually rendering the HTML source. The first approach is more suitable for systems running on a server where there is not a need for interacting with the program. The second approach usually expects to receive user input through a graphical user interface.

2.2.1 Text Density

Text density is done by identifying the clusters of text within a website to find the content. The interesting idea of this approach is that it focuses on clusters of text in a website instead of reading the HTML document and trying to identify all the component and the layout to find text. Thus making it effective across websites even when the layout or the design of the website changes. In the case of websites where the text information is spread into smaller sections like a forum, this approach can handle the extraction of such information with better results [30].

Combining regular expressions and text density is a method that Li discusses [30]. Using regular expressions to remove all the HTML tags according to labeling features of web pages, leaves the basic content left. Afterwards an improved density algorithm is applied to extract the text. This method is simple and quick. Density is a property that can be tweaked to suit the text extraction area of this thesis, in this case web forums. Text density approach can be more accurate in websites that has multiple text segments like forum posts. Three methods using a similar approach are explained in detail in the next chapter.

2.2.2 Page rendering

Another way of approach is visually rendering the HTML source. Zheng et al [31] proposed a template-independent information extraction approach to identify news articles based on visual consistency. This paper suggests to represent a page as a visual block tree and then derive a composite visual feature set that is stable in the news domain. And at last, using some machine learning algorithms to generate a wrapper to locate where the information is.

Yin and Lee [32] proposed a way to extract important information using both the rendering technique and PageRank algorithm [33]. It uses a ranking algorithm similar to PageRank, to rank the content objects within a web page. This allows the extraction of only important parts of web pages, for a cleaner view in mobile devices. Furthermore, they also discussed the
method in more extent in their paper [34], where they worked using a machine learning approach to classify each element in a web page into six functional categories: Content, Related Link, Navigation, Advertisement, Form and Other. This allows the extraction of only certain categories of content in a webpage to be delivered to a mobile device to fit a user's specific needs, or to facilitate web information processes like web mining or mobile search. Similar to the method they used with the PageRank algorithm [33].
3 Related Work

One of the issues with web crawling is handling link duplicates. When dealing with link duplicates for forums, a traditional approach (such as brute force method and classic bloom filter method) will no longer be sufficient because of the large amount of links a forum possess. Section 3.1 therefore provides an advanced method for handling link duplicates. There are several text extraction methods and approaches, but not all of them fits the requirement: to be able to run the method without constant supervision of a human when interacting with new websites. The approaches mentioned above such as page rendering approach in section 2.2.2 are not applicable to this scraper development. Section 3.2 thus presents the related text extraction methods that focuses on text density, which only cares about the amount of text in different areas of the website.

3.1 Link Structure & Multi-Layer Bloom Filter

Because of the layered characteristics of a web-link, if a certain substring $a_i$ (i ranges from 1 to L) in web-link $S = a_1/a_2...a_i$ is not in the set, then web-link $S$ is not in the set. The Multi-Layer Bloom filter [29] is based on this principle.

In a web-link $S = a_1/a_2/a_3/...a_i$, $a_i$ (I ranges from 1 to L) represent a unique layer in the Forum web's hierarchical structure. Therefore using a single bloom filter to execute duplicate web-link detection is not a wise decision since it does not make use of the special characteristics of a web-link. A Multi-Layer Bloom filter however, can have multiple layers (where each layer is a classic bloom filter) to store corresponding layer/subpage of the web-link.

When inserting a new web-link, the web-link is broken into a series of substrings ($a_1,a_2,a_3,...a_i$) and all of the substring $a_i$ (i ranges from 1 to L) is inserted into the $i_{th}$ layer bloom filter of the Multi-Layer Bloom Filter correspondly. When checking whether a web-link $S$ is in the set, check whether each $a_i$ (1 <= i <= L) in $S$ is in a Layer Bloom Filter or not. If a certain element $a_i$ is not in a layer bloom filter, web-link $S$ is not in the set, vice versa.

Similar to the classic bloom filter, Multi-Layer Bloom Filter also inevitably generates false positives. For example: given that a web-link $S1 = a_1/a_2/a_3/...a_i$ is inserted into the Multi-Layer Bloom Filter. $b$ is another subpage of the web-link $S2 = a_j/b/a_3/...a_i$ and $b$ is also stored in the 2nd Layer Bloom Filter. When querying about the web-link $S2 = a_j/b/a_3/...a_i$, even though $S2$ has not been visited before, the Multi-Layer Bloom Filter will answer that $S2$ has been visited before and treat it as a duplicated web-link. The source of this problem is simple, the Multi-Layer Bloom Filter divides the the web-link into a series of substrings and then uses a single classic bloom filter to represent each of the substrings. However, the Multi-Layer Bloom Filter does not have a data structure to store the entire web-link composition as a whole.

In order to solve this problem, an extra layer of Bloom Filter (Layer L+1) is added into the Multi-Layer Bloom Filter. This extra layer is used to store the information of the web-link from Layer 1 to Layer L. This new layer of Bloom filter also contains m bits. Whenever a new web-link is inserted, and XOR operation is used to all the bit positions in Level 1 to Level L and set the corresponding bit in the last Layer (Layer L+1) to 1. A visual representation of Multi-Layer Bloom Filter is provided in figure 6.
According to Cen al et al [29], the false positive probability of a classic Bloom filter is:

\[ f(k, m, n) = (1 - (1 - \frac{1}{m})^{kn})^k \approx (1 - e^{-kn/m})^k \]

Where \( k \) is the number of independent hash functions, \( m \) is the number of bits for the Bloom filter bit array, and \( n \) is the number of weblinks in total. It is very easy to see from the function below that the false positive probability of a classic Bloom filter is much larger than the false positive probability of Multi-Layer Bloom Filter:

\[ f_{MLBF}(k, m, n, L) = \prod_{i=1}^{L} f(k, m, n) = (f(k, m, n))^L \]

### 3.2 Text Extraction Methods

The text extraction of a multi-topic page, such as a forum thread, has to consider that the information is spread out throughout the page. The chosen methods involve generating an average or a threshold for the text density in different sections of the page. The reason for choosing the text density approach is also based on the fact that layouts and structures of the content in forums might vary. Thus any approach that focuses on the way the HTML and the website is constructed will not be able to work across many forums. Shifting the focus of the text extraction to be on the density of the text makes it more versatile and adaptable to different layouts.

The text extraction approach used for the development of the web scraper is the one using the text density of the document. From this approach, three methods were presented: the text-to-tag ratio [35], the text detection framework [36] and the word count [37] methods.
Text-to-tag ratio
Weniger al et. [35] proposed to compare the text-to-tag ratio of the HTML source. To start it is necessary to calculate the ratio of non-tag words to the tag words of each line in the document. These ratios are saved for each line, together with the content of the line itself. After the whole document has been processed, by adding all the ratios and dividing by the total number of lines, a mean ratio can be computed.

Next, the ratio of each paragraph can be compared against the mean ratio (the threshold), thus deciding if the corresponding content of the line will be extracted from the document. Only extract the document line if its ratio is above the mean. This shows an improvement in precision and space saving, over regular text extraction methods focused on parsing the structure of the HTML file. This precision and space saving provided by this method could be beneficial for a forum scraper.

Text detection framework
Wu [36] presents an extraction system that tries to detect the text framework of the website. The preprocessing work involves removing the header parts, as well as the content of all <style> and <script> tags. PHP and Java source code are also removed because they are not text content. If such tags were not filtered out at this step, then the Java or PHP code would be counted as words or tags (if the forum discussion is about programming languages, all the the code will be under a div tag called <code> as for example StackOverflow [38] works. This arrangement prevents the code that users posted themselves from executing, and makes the codes easy to extract since they are also considered as words). Then it uses Compound Text–Tag Difference (CTTD) formula

\[ H_i^{CTTD} = TC_i + \lambda \cdot LTC_i - T_i - P_i \]

Where \( TC \) represents the number of words in line \( i \), and \( LTC \) represents the number of link words (also called anchor words) in line \( i \). \( \lambda \) is a parameter used to weigh the impact of the link words(or anchor words) \( P \) is the number of link-break tag in line \( i \), and \( T \) is the number of all valid HTML tags for line \( i \).

Even though the CTTD can strengthen most content area in a web page, a smoothing process can extend the boundary information. There are no restriction for the smoothing algorithm, so the user may choose any smoothing algorithm they feel like will fit their situation. Since the situation often occurs in the boundary between texts, a smoothing will help recover the text lost in the gap.

Then, text block identification is applied to determine a set of content candidates. This helps greatly on determining whether a single value in the projection profile belongs to the content text line. An adaptive CTTD threshold is obtained via the formula:

\[ \theta_{CTTD} = CTTD_{\text{min}} + k \cdot (CTTD_{\text{max}} - CTTD_{\text{min}}) \]

Where \( CTTD_{\text{min}} \) is the non-zero CTTD score for all lines in the web page; \( CTTD_{\text{max}} \) is the maximum CTTD score of the given page. \( K \) is a parameter that controls the weight of the difference.

To filter noise, text verification determines whether a given text block can be included with content. Based on the content text block chrasistics, 3 rules has been established and have to be fulfilled by each text block, in order to be included and classified as a content text box. Those rules are:

1. The number of text in the text box should be more than threshold \( \theta_{TC} \).
2. The link density of a text box should be lower than threshold \( \theta_{den} \) in order to be classified as a content text box.
3. The number of line break tags of a text box should be more than \( \theta_b \), and its density should be more than \( \theta_{den} \).

**Word count**

Another similar method following the approach of text density, is analyzing the word count of the document. In the paper by Zhou et al. [37] the method of processing the HTML source as a paragraphed text string is explained. The idea is that by processing the HTML file as a string directly and extracting the main text content by only analyzing the word count of the paragraphs, the extraction is simplified and faster.

The first step involves removing all non-block HTML tags and afterwards replacing the block level tags (these are all the tags that are used for defining content) with line breaks. The next step splits the string into paragraphs, thus each line will represent a paragraph. Now each line can be counted by words, calculating a number for each paragraph. The average can then be calculated by adding the word count of each line and dividing it by the total number of lines, this is the threshold. Similar to the text-to-tag ratio method, after calculating the threshold, each line’s word count is compared against the threshold to decide whether it is extracted or not. In this case extracting the paragraph if its word count is above the threshold. This makes word count method a very simple implementation.
4 Methods and Models

To gather all the necessary information for a problem statement and a thesis purpose, it is necessary to apply methods and methodologies to collect, analyze and draw conclusions from data. There are two main methodologies [18], the quantitative (experiments and large data sets to reach conclusions) and qualitative (development in an interpretative manner on rather small data sets to create artifacts) methodologies. This thesis applies the second one, the qualitative research methodologies.

The data collection and analysis methods used for the implementation are described in the subsection 4.1. The specific data collection and analysis methods are described and discussed in subsection 4.2.

4.1 Data Collection and Analysis

There are several qualitative data collection methods, some of these are shared with the quantitative, such as the questionnaire, case study and observations. While interviews and language and text are unique to qualitative data collection. The purpose of each of these methods are described in brief by Hakansson [18] as follows:

*Questionnaire* collects data through questions, which are either quantifying data (closed, alternative questions) or qualifying data (open and reviewing questions).

*Case Study* is an in-depth analysis of a single or small number of participants. The case study data collection method is used with the case study research method.

*Observations, ethnography and participation*, observe behaviour with focus on situations (participation) and culture (ethnography).

*Interviews*, structured, semi-structured, and unstructured, give an deep understanding of a problem and capture participants’ point of view.

*Language and Text* are used for interpreting discourse and conversations and meanings in texts and documents.

The reason to use data analysis methods is to process the collected data. Thus modelling, cleaning and inspecting the collected data to use it to draw conclusions and support decisions and theories. Two methods that are suitable for qualitative data collecting methods are analytic induction and ground theory.

Analytic induction and Ground Theory are both iterative methods that work in rotation of collection and analysis of data. Thus alternating each process until a conclusion is determined. Analytic Induction stops with a hypothesis and Grounded Theory with a validated theory [18]. This is very similar to the process that is followed in reality when performing interviews.

4.2 Interviews and Results

The data collection for this project was done through interviews. Interviews can be of different types. In the case of this project, semi-structured interviews were used. Although most of the interviews were conducted face-to-face, a few were done online through Skype [20].
The reason for selecting the interviews as data collection method is because it gives a deeper understanding of the stakeholder’s thoughts regarding the web scraper and its functionalities. The interviews were semi-structured with the use of both opinion based and knowledge based type of questions [39]. The structured questions from the interviews are added to the appendix B. Only the questions are recorded, no answers were added to the documentation.

An inductive data analysis was applied to refine the requirements and highlight the important features the web scraper should have. The data analysis was done cooperatively with the data collection method from the interviews.

The nature of interviews allows for the possibility of analysing the data being collected on the moment. Thus allowing for generating a response to further deepen in the topic of the interview. Interviews makes the data analysis to be flexible and adapt the data collection based on the data being analyzed. Making the data collected and its analysis much richer and informative than the data obtained from other methods. Therefore the data collected was analysed during the same process and converted into requirements or refined the existing requirements and features of the system.

4.3 Quality Assurance

When doing research work for a thesis, it is important to be able to validate and verify the material gathered during the research, in a process called quality assurance [18]. The quality assurance process is important because it legitimizes the work done.

For a qualitative research that follows an inductive approach, the quality assurance process applied to the research must include: ethics, validity, dependability, confirmability and transferability. Each of these topics has an important role in the quality of the thesis [18].

*Ethics:* this covers the morals and principles in all the stages of the research work conducted. Including issues as privacy, protection, consent and confidentiality of any involved part in the research done.

*Validity:* this focuses on that the research follows the existing rules when conducted. Thus confirming that the results are correct presented.

*Dependability:* it makes sure that the conclusions are correct and reliable.

*Confirmability:* confirms that no bias or personal assessments have affected the results.

*Transferability:* it involves making detailed descriptions of the research, thus making it possible for other researchers to benefit from it as well.

All these topics are necessary to follow and apply to validate the thesis work.

4.4 Development Model

Software development models [40] are used to describe the sequence of phases for an entire lifetime of a product. This covers everything from the initial commercial idea until the final de-installation or disassembling of the product after its use.
There are many software development models, a few examples are the Waterfall model [41], The Cyclic model [41] and the Incremental model [41]. The Waterfall model describes a sequence of activities and corresponding artifacts, from the most general requirements through successively more detailed steps to implementation, after which the software is put into operation and maintained. The Cyclic model, in contrast to sequential processes (which a list of distinguished activities are done one after another like the Waterfall model), cyclical processes repeat themselves over and over. The goal is that each cycle brings the development closer to its successful completion. The Incremental model [42] is a process in which the functionality of the desired system is divided into small increments that are implemented and delivered one after another in quick succession. Each increment is chosen so that it expands on the previous one, and is small enough to produce quickly.

An Incremental model [42] was chosen for the implementation and development of this project work, based on the fact that it is flexible and less costly to change the scope and requirement during the process.

The system has two main requirements: 1) The ability to navigate different web forums and visit all the pages of each thread. 2) The ability to extract the posts of all the users in each of the pages of the forum, while keeping the extraction of any other unrelated data low. All the requirements were established during the interviews described in subsection 4.2.

For this model to work the goals have to be clear and the whole system needs to be defined before it can be broken down and built incrementally. The design stage of the system was done based on the information acquired from the literature research period. The information gathered during the data collection period from interviews with the stakeholders was also used during the design stage, by analyzing it to evaluate and refine the requirements of the system. From this stage the functionalities and the design choices made for the implementation of the crawler were outlined. This allowed for the progress of the implementation to be tracked easier.

The implementation stage was the development of the system. This included the integration of all the subsystems of the program into one single system. These subsystems were based on the two requirements previously mentioned.

Quantified measurements can prove difficult to assess due to the complexity of the issues facing when performing text extraction. In order to address these issues, a discussion with the stakeholders was held. This lead to a decision in what properties are important in a web scraper to examine in order to evaluate its performance. The product was tested using non-functional testing. The methods examines a set of properties of the product, such as accuracy, speed and file size generated. The performance test was done by multiple executions on different computers. There are no other freely available web scrapers to compare the performance of this project against similar products.
5 Work tools and structure

This section talks in detail about the general idea about the structure of the forum crawler as well as the tools to use for this degree project. Subsection 5.1 describes the tools selected for the work and used. Subsection 5.2 describes the program structure. Subsection 5.3 gives an overview of the whole scraper program.

5.1 Work Tools

Since the programming language was already decided to be java by authors, the choice of framework is therefore also limited to java frameworks. A framework can be defined as: “a real or conceptual structure intended to serve as a support or guide for the building of something that expands the structure into something useful.” [43] The framework greatly helps with the efficiency of the software development process. A brief research about existing frameworks for web crawlers has been done. Among the frameworks, four of the them stood out from the rest: Apache Nutch, Heritrix, Crawler4j and Jsoup. In the end, Jsoup was chosen to be the framework to use, due to its dedicated website with better navigation, examples, Application programming interface (API) documentation and much more, making it really easy to get started with more in depth. The requirements met by each framework is presented in table 1. In the following text, all four frameworks are presented, along with the detailed reason of selecting Jsoup.

Table 1. Characteristics table of frameworks based on project requirements

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Nutch</td>
</tr>
<tr>
<td>Distributed implementation</td>
<td>Yes</td>
</tr>
<tr>
<td>Updated documentation</td>
<td>No</td>
</tr>
<tr>
<td>Setup difficulty</td>
<td>Hard</td>
</tr>
<tr>
<td>Extensible/modular</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Jsoup</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Heritrix</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Crawler4j</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
</tr>
</tbody>
</table>

Apache Nutch

Apache Nutch [44] is a highly extensible and scalable open source web crawler software project. It is coded entirely in the Java programming language, has a highly modular architecture, allowing developers to create plug-ins for media-type parsing, data retrieval, querying and clustering. Nutch has a complete library for distributed mode available for its users to use. It is therefore praised to be the tool to use when it comes to making a distributed crawler. However, it is also very hard to implement and install for the server, and the configuration time is long and cumbersome. Also, Nutch’s documentation is outdated, same goes for the tutorials. Furthermore, even though Hadoop is a solid technology, it has an operational overhead. Many other core, like AWS (Amazon web service) for example, has many of the distributed fundamentals exposed as a service already, so taking advantage of those kind of core is better than running on Hadoop [45]. Since Omilitycs [8] main program is running on an amazon server as well, decision has been made internally to not to use Nutch [44].
Hetirix

Heritrix [46] is the Internet Archive's open-source, extensible, web-scale, archival-quality web crawler project. It is written in Java and has a modular architecture as well. It has some advantages: first of all, there is no network overhead for Heritrix to function. Secondly, it does not need any installation, user are only required to include the Heritrix jar file to his/her project. Finally, Heritrix runs on a single Java Virtual Machine (JVM). This means Heritrix can limit the issues such as configuration management, tracking how well the crawler is performing and rate limiting. However, It also has some down sides. To start, Heritrix architecture is complex and cumbersome, and it is hard to find help when running into errors. Also, Heritrix architecture is more monolithic and not designed to add parsers and extensibility. This means that Heritrix takes in an entire HTML file and stores it without filter out any text at all. Therefore Heritrix is also not an ideal tool for this project either [46].

Crawler4j

Crawler4j [47] is an open source web crawler for Java which provides a simple interface for crawling the Web. Using it, you can setup a multi-threaded web crawler in a few minutes. It already has a distributed mode, so it is easy to assign the number of workers. Also no extra installation is needed, simply download the code from github. However, Crawler4j has very little documentation, just a couple of small tutorials. It will therefore take some time to completely understand. Crawler4j uses a lot of memory due to the excessive use of global variables and unnecessary duplication occurs in collection.

Jsoup

Jsoup [48] is an open-source Java library of methods designed to extract and manipulate data stored in HTML documents. It provides very convenient methods to extract and manipulate the data. Jsoup is easy to set up and start running. Its documentation is also up to date and include relevant tutorials for usage [48]. Since jsoup is just a java library, the user can avoid unnecessary additional overheads (e.g. overhead from hadoop) by writing the program with the help of jsoup framework, and implement additional library (e.g. java.util.concurrent) to achieve the creation of a distributed web scraper.

Out of the four selected web crawler frameworks described above, two do meet our requirements and stood out: Crawler4j and Jsoup. Crawler4j is very simple and straightforward to set up, the same goes for Jsoup. The difference is mainly the presentation of the available documentation. Crawler4j is available on Github [49] with only a short description of how to set it up and use the basic functionalities. On the other hand, Jsoup has a recently updated website with multiple examples, a well documented API, making it really easy to get started with more in depth.

5.1.1 Maven

Maven [50] was used to help managing the package dependencies easily. Maven is a build management tool that can be used for building and managing any Java-based project. Maven will download all the libraries that it recognizes you using, automatically. It was decided that the Jsoup framework [48] would be used for this project and the program will be released in a server. In this case maven handles the downloading of the dependencies instead of doing it manually, making the dependencies more manageable.
5.2 Program structure

The crawler can be broken down to three modules: The Worker, The Main and The Bloom Filter. Therefore in this section each of these three module is given a brief discription before the general idea is introduced in section 5.3.

The Worker class

The worker takes in a web-link as an argument and has three main functions: Link extraction, Text extraction and Link filtration. Link extraction function scrape through the web page from the given link, and extract web-links inside the page. These links are stored and later send to the Link filtration function to process. The text extraction function however, is focused on a different task: To extract user posts in the web page and store them in a csv file. Link filtration function takes in the web-links extracted by the Link extraction function and get rid of unnecessary or unwanted web-links. When the Link filtration function is done filtering the list of web-links, it then start to check the remaining web-links with the Multi-Layer Bloom Filter to get rid of web-links that has already been visited before. The remaining web-links are added to the bag_of_tasks queue for later processing. The worker class is designed to be able to function in a thread. Therefore user can create numbers of workers to be active at the same time.

The Main class

The main class consists of three parts: Multi-Layer Bloom Filter Bit array, signal handler and the user configuration. Multi-Layer Bloom Filter Bit array is a space efficient data structure to detect duplicate web-links. This is done in order to avoid the worker to get unnecessary work (which both waste time and resources). The signal handler will feed the Worker a web-link from the bag_of_tasks queue one at a time as long as there are still web-links in this bag_of_tasks queue. When a worker is done with the current web-link, it will ask the main class for a new web-link (from the bag_of_tasks queue) to work with. The user configuration is simply a section where the user can change the parameters for the program (e.g. Multilayer Bloom Filter Size, Number of independent hash functions, and Number of Workers).

The Bloom Filter class

The job of the Bloom Filter class is simple: when the Bloom Filter is given a web-link from the worker class, it start to check whether the web-link has been visited before. The Bloom Filter ensures that no workers will get the web-link that has already been visited and prevent the bag_of_tasks queue to have duplicated web-links.

5.3 System overview

In the project folder, there is a csv file called “seed-list”. The user should modify the file by adding the web-links of the forums that the scraper should visit. Once done, the user will need to go to the Main.java file (located in src/main/java/com.scraper.app/), open and put in the changes to the user configuration section (otherwise the crawler will run in a default mode, 4 cores). This is the section where the user can modify the number of workers, the Bloom filter array size and number of hash function. These settings will affect the execution time as well as the probability of false positives for the bloom filter.
As illustrated by the figure 7 above, the program will start by parsing the “seed-list” csv file, and store the content to an ArrayList. The worker will take the first web-link from the bag_of_tasks queue (a bag_of_task queue is implemented using a data structure called concurrent linked queue, this data structure allows multiple threads to be able to complete its current operation regardless of the state of other threads accessing the queue. This also executes insertion and removal operation in constant time.), and process it (It also removes the web-link it process from the bag_of_tasks queue). Typically the web-links in the beginning are the index pages of the website, therefore the worker can extract a lot of links. Those link are then filtered by the worker with the help of Multi-Layer Bloom Filter. The Multi-Layer Bloom Filter takes web-links given by the worker, and start checking if those links has been visited before. If the link has been visited before, it will be discarded. If the link had not been visited before, it will be inserted into the bag_of_tasks queue. The worker thus takes the web-links from the bag_of_tasks queue one at a time (Note: when a web-link is taken by the worker, it is also deleted from the bag_of_tasks queue). When a worker is done with the given link, it will send a csv file to the database and then send the filtered web-links to the bag_of_task queue. It will take another web-link from the bag_of_tasks queue and start processing it. Note that the worker will only take the first web-link (the first element from the front of the queue) in the bag_of_tasks queue. Finally, if the all the workers have finished processing the tasks, are in an idle state and the bag_of_tasks queue is empty (this indicates that all the web-links have been visited in the given forums), the program terminates.
6 Implementation

This section presents the implementation of the link duplicate handling methods in more detail, as well as the text extraction methods that the web scraper uses. Subsection 6.1 describes how the multi-layer bloom filter was implemented. Subsection 6.2 describes how each of the different text extraction methods were implemented.

6.1 Multi-Layer Bloom Filter

A website usually has a main web page and a series of child subpage, the child subpage can have a series of child subpage and so on... This form a hierarchical structure, resulting the web-links to take the form: “http://a1/a2/a3...aL” or “https://a1/a2/a3...aL”. The “/” in the web-links indicates the start of a new subpage since it act as a separator between subpages. These subpages are called layers, since each of them indicates a different path [29].

The crawler takes out the web link protocol identifier (http or https) out from the link since they are common among all web-links. And the identifier doesn’t contain interesting information for the crawler. So the Main class parse the web-link by remove the string “http://” and “https://” first, and then split the link by the separator “/” and store each substring into a String Array individually. In this case, the duplicated web-link detection (Detection of a long web-link, such as “http://a1/a2/a3/a4...aL”) is transformed into a detection of duplicated strings (e.g. a1, a2, a3...aL).

The scraper first get the string of web-link and convert it to an array of substrings, using the “link_to_layers” function. The program then uses the Multi-Layer Bloom Filter functions to insert the item into the set using “bloom_filter_insert” function, or query if the item is in the set using “bloom_filter_query” function (Note: All the functions mentioned above are in the Bloom Filter Class).

The Bloom Filter bit array in the Multi-Layer Bloom Filter is set to contain 28755176 bits, and the Multi-Layer Bloom Filter contains 15 layers in total with 20 independent hash function. This means that if the program visits 1 million web-links in total, the false positive probability will be 1.0007418 × 10^-90, base on the MLBF formula from chapter 3.

The “bloom_filter_insert” function calculate k hash address (using k independent hash functions) of each substring and set corresponding bit to 1. It then performing XOR operation among the bit-positions in Level 1 to Level L and set the corresponding bit in the last Layer (Layer L+1) to 1.

The “bloom_filter_query” function takes in a web-link string and return a boolean value to indicate whether the link has been visited before. The function checks each substring with its corresponding layer bloom filter by using the k independent hash functions and check the corresponding bit. If the any bit has not been set to one, the function returns false, indicate that the web-link has not been visited before, else it returns true. At last it performs the XOR operation and check if all the corresponding bits in the L+1 Layer has been set to 1. If the any bit has not been set to one, the function returns false, else it returns true.
6.2 Text Extraction

To implement the text extracting methods, the tools provided by the Jsoup framework were used. These tools are library functions for manipulating the HTML code that the crawler is working with when navigating websites. This includes functions such as selecting or removing part of the document by matching HTML tags and more. In addition, the oldest timestamp and newest timestamp in each web page is extracted and printed to the beginning of the text document. This gives the user a brief understanding on whether the posts inside the web page are outdated.

Word Count
The initial step to implement the word count method was to use the functions of Jsoup made for manipulation of the HTML code. Specifically, the selection and removal functions to clean up the HTML document from any non-block HTML tags (block tags are used for defining content). In the non-block tags, script, style, header and a few more tags are included. After eliminating these tags, the HTML document is parsed into a string. Once in a string, the remaining block tags are replaced with new line indicators, thus breaking the string into paragraphs at these points.

At the next stage, the scraper calls a count word function to count the words of each of the paragraphs. For this crawler, the word count is used as key identifier and the line itself is the value. Therefore each line with the same value are appended to the existing one with the same key. After getting the word count of each line, a total word count average is calculated by adding the total amount of words and dividing by the total number of lines. The average is used as a threshold. Thereafter by comparing the word count of each line against the total average, the crawler estimates which paragraphs has content that is of interest (i.e a user post) or is uninteresting (i.e part of advertisement or similars). Thus, extracting the content of interest without caring about the rest.

Text-to-Tag Ratio
The text-to-tag ratio method does not delete the HTML tags, instead it counts them and also counts the non-tag words of each line of the source code. By counting the amount of words that are tags and the ones that are not, a ratio can be calculated for each line. The ratio is simply the number of non-tag count divided by the tag count.

The next step is to calculate the average ratio of the whole document. By calculating the average ratio of the whole document it is possible to find a threshold. To normalize the results the total ratio is divided by $2 \ast r + 1$. The radius is a user set variable used to refine the results. For the scraper implemented for this project, the radius was set to 25 as this gave the best result during the testing of the implementation.

After generating the normalized document ratio, the threshold is compared to the ratio of each line. This is similar to the last step of the word count, where if the compared value is greater than the threshold, it is extracted. If it is lower, it is ignored.

Text-detection framework
A third method was implemented to compare with the two previous methods. The text detection framework presented by Wu [36].
First stage of this method is the preprocessing stage. In this preprocessing stage, Jsoup is used to remove all the header content as well as the <style> and <script> tags. Comments of the code is also removed since they do not provide valid information that is relevant (not required in order to find text content such as user posts) to the scraper. PHP source code is also removed since they are not text content.

After filtering the non-content text parts out, the program moves into the next stage: text detection stage. This estimates if a line of text belongs to a block of content or not, thus detecting which parts of the document to extract. In this text detection stage, a compound text-tag difference formula (\( H_{CTTD} = TC_i + \lambda \cdot LTC_i + P_i - TG_i \), for detailed information, see section CTTD-TDF) was applied to calculate a score for each line of the document. Since CTTD isolates the link text count and line break count during the calculation, a more accurate score for each line is obtained. Functions such as countWord, countLinkWord, countTags and countTagP are created to gather data and help calculating the CTTD score for each line. The countWord function takes in the whole line as a string and calculates the TC score, which is the number of words in that line. The countLinkWord function does a similar job as the countWord function, but instead this countLinkWord function calculates the number of anchor words in that line (LTC). The countTag function calculate the number of tags in the given line, and countTagP only counts the number of <p> or <br> tag in the specified line. 

When the CTTD score has been calculated for each line in the HTML file, a classic Gaussian kernel is implemented as the function to smooth the resulting CTTD scores:

\[
H'_{CTTD} = \frac{1}{10}(CTTD_{i-2} + 2 \cdot CTTD_{i-1} + 4 \cdot CTTD_i + 2 \cdot CTTD_{i+1} + CTTD_{i+2})
\]

After the smoothing stage, the content boxes’ border line can be easily defined. The program then uses all the obtained CTTD scores to calculate an adaptive threshold. This threshold is then used to help determine whether a textbox can be constructed. In this stage, lines that have high CTTD score than the threshold and are grouped together will be put together and construct a text box.

After all the candidate text boxes are constructed, the program will then move to the final stage: text verification stage. In this stage, all the constructed text boxes will be processed one by one, to determine whether the textbox contains text content. The program will thus only keep the textbox that have useful content (i.e user posts).
7 Performance Results

In this chapter the final implementation of the project is presented. The result of the Multi-Layer Bloom Filter is briefly mentioned below. Follow by an intro for the results of different text extraction methods. In section 7.1, the performance result of the running speed and file size extracted for both Computer and Server are presented. In section 7.2 the resulting text extracted by different methods are presented. The parts of the raw tables with data is included in Appendix B.

The Multi-Layer Bloom Filter’s job is to detect and prevent duplicated web-links to get picked up by the scraper program. During the entire performance test period, the scraper program did not alert that it has been given a web-link that has been visited before. Hence the 100% of non-false negative rate.

As for the three different text extraction methods, all of the text extraction methods have a similar approach but the small differences makes them behave differently in the scenario of extracting text from a multi-topic webpage such as a forum. The best way to test the different methods and decide which method works better for forums was to implement them and compare them as part of the crawler. The performance test has been divided into two parts:

1. The speed and size test measures the number of web pages extracted in 5 mins, and the average file size for each web page. This is presented in detail in section 7.1.
2. The Accuracy test shows the actual text extracted by each method with each other and the original web pages. This is presented in section 7.2

7.1 Performance Test: Speed & Size

To measure the speed and the size of the generated data of our web scraper, It was decided to perform some testing by executing the program multiple times on different computers. It was also decided to record the number of files created and the total size of it all, such that it would be easy to divide it by the execution time. Thus creating an average of the amount of files the scraper creates over time and also the average size of the generated data per file.

7.1.1 Computer Performance

First, the testing was performed on two laptops. In this case two laptops were used to execute each of the three methods for 10 rounds. Each round was set to terminate after 5 minutes. At the end of each round, the amount of files created was recorded together with the total size of all the generated files.

Text-Detection Framework

The Text-Detection Framework had a minimum of 1761 files generated with a minimum of 17MB of space used. And a maximum of 3246 files generated with a maximum 23MB of space used. The Text-Detection Framework also had an average of 2263 files generated per round. Since a round is 5 minutes and each extracted file is equivalent to an extracted a page, it can be said that the scraper extracts 452 pages per minute. The average size of a file in this case is 0.008MB and approximately 3.6MB per minute. The Text-Detection Framework results are illustrated in figure 8.
Text-To-Tag Ratio

The Text-To-Tag Ratio method had a minimum of 1767 files generated with a minimum of 8MB of space used. And a maximum of 2777 files generated with a maximum 19MB of space used. The Text-Tag Ratio also had an average of 2510 files generated per round. Since a round is 5 minutes and each extracted file is equivalent to an extracted a page, it can be said that the scraper extracts 502 pages per minute. The average size of a file in this case is $6.75298804 \times 10^{-3}$ MB and approximately 3.39 MB per minute. The results are illustrated in figure 9.

Figure 8. Total files & total size of generated data, Text-detection Framework (generated by author)

Figure 9. Total files & total size of generated data, Text-to-Tag Ratio (generated by author)
Word Count
The Word Count method had a minimum of 1000 files generated with a minimum of 7.3MB of space used. And a maximum of 3315 files generated with a maximum 28MB of space used. The Text-detection framework also had an average of 2684 files generated per round. Since a round is 5 minutes and each extracted file is equivalent to an extracted a page, it can be said that the scraper extracts 537 pages per minute. The average size of a file in this case is $8.00432263 \times 10^{-3}$ MB and approximately 4.3 MB per minute. The results are illustrated in figure 10.

![Total Files vs. Total size (MB) - WC](image)

Figure 10. Total files & total size of generated data, Word Count (generated by author)

7.1.2 Server Performance
The testing was also performed on two server. In this case two identical KTH Servers [51] (Colombiana and Avril) were used. The servers have the following specification presented in table 2:

<table>
<thead>
<tr>
<th>Brand &amp; Model</th>
<th>CPU</th>
<th>RAM</th>
<th>HDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>HP Proliant DL 385 G7</td>
<td>2 x AMD Operton</td>
<td>82GB</td>
<td>2TB</td>
</tr>
<tr>
<td></td>
<td>6172 2.1 GHz</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(12 Cores/CPU)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each of the three methods was executed for 50 rounds. Each round was set to terminate after 5 minutes. At the end of each round, the amount of files created was recorded together with the total size of all the generated files. The results after a 50 total rounds from the two servers is presented for each method. For a better analysis on performance, each of the methods were run again after changing the amount of cores the program would use. Thus recording the data from the testing done using 4, 8 and 16 cores. The results from running the scraper on the server while only using 4 cores is similar to the one running on the laptops presented earlier, therefore it was decided to be skipped from being presented in the server test data. Instead, the server test was focus on presenting the results of the performance on 8 and 16 cores.
Text-Detection Framework

The Text-Detection Framework had a minimum of 3869 files generated with a minimum of 21MB of space used. And a maximum of 5165 files generated with a maximum 51MB of space used, when utilizing 8 cores. The Text-detection framework also had an average of 4607 files generated per round. Since a round is 5 minutes and each extracted file is equivalent to an extracted a page, it can be said that the scraper extracts 921 pages per minute. The average size of a file in this case is 0.006MB and approximately 5.5MB per minute.

When the Text-Detection Framework was utilizing 16 cores, the results were a minimum of 5951 files generated with a minimum of 32MB of space used. A maximum of 8523 files generated with a maximum 116MB of space used instead. The number of average files generated per round was 6446. The rate of page extraction per minute was 1290 and with a rate of data generated at 9MB per minute. The Text-Detection Framework results are presented in figure 11 and 12.

Figure 11. Total files & total size of generated data, Text-detection Framework (generated by author)

Figure 12. Total files & total size of generated data, Text-detection Framework (generated by author)
Text-To-Tag Ratio

The Text-To-Tag Ratio had a minimum of 3517 files generated with a minimum of 18MB of space used. And a maximum of 8865 files generated with a maximum 35MB of space used, when utilizing 8 cores. The Text-To-Tag Ratio also had an average of 3517 files generated per round. Since a round is 5 minutes and each extracted file is equivalent to an extracted a page, it can be said that the scraper extracts 703 pages per minute. The average size of a file in this case is 0.006MB and approximately 4.22MB per minute.

When the Text-To-Tag Ratio method was utilizing 16 cores, the results were a minimum of 5415 files generated with a minimum of 12MB of space used. A maximum of 7088 files generated with a maximum 23MB of space used instead. The number of average files generated per round was 6241. The rate of page extraction per minute was 1248 and with a rate of data generated at 3.24MB per minute. The Text-To-Tag Ratio results are presented in figure 13 and 14.

![Figure 13. Total files & total size of generated data, Text-to-Tag Ratio 8 cores (generated by author)](image1)

![Figure 14. Total files & total size of generated data, Text-to-Tag Ratio on 16 cores (generated by author)](image2)
Word Count
The *Word Count* had a minimum of 3543 files generated with a minimum of 24MB of space used. And a maximum of 6013 files generated with a maximum 46MB of space used, when utilizing 8 cores. The *Word Count* also had an average of 4608 files generated per round. Since a round is 5 minutes and each extracted file is equivalent to an extracted a page, it can be said that the scraper extracts 922 pages per minute. The average size of a file in this case is 0.006MB and approximately 5.5MB per minute.

When the *Word Count* method was utilizing 16 cores, the results were a minimum of 4896 files generated with a minimum of 14MB of space used. And a maximum of 8465 files generated with a maximum 61MB of space used instead. The number of average files generated per round was 7850. The rate of page extraction per minute was 1570 and with a rate of data generated at 3.14MB per minute. The *Word Count* results are presented in figure 15 and 16.

![Total Files vs. Total size (MB) - WC8](image1.png)

Figure 15. Total files & total size of generated data, Word Count on 8 cores (generated by author)

![Total Files vs. Total size (MB) - WC16](image2.png)

Figure 16. Total files & total size of generated data, Word Count on 16 cores (generated by author)
7.2 Performance Test: Accuracy

Accuracy test was performed by randomly selecting five different web pages and then executing the three different text extraction method on these pages. Then comparing the result with each other, as well as the original web page. The text extracted by each of these three methods is compared to the original web page with the help of an online article comparison tool Copyscape [52] to get a percentage of the overall text extracted from the web page. One example of the text extraction done is presented in the figure 17 below, for more detailed information of this and other examples of accuracy test, please see Appendix A.

Figure 17. Extract of data extracted by each method and original page. Upper left: Original; right: word count; Lower left: text-to-tag ratio; right, text-detection framework (generated by author)
Text detection framework
Text detection framework method extracted 46% of the text from the example page of elitetrader [9] website. In total 562 words are extracted for the example page. In those 562 words that are extracted, all of them are user posts and no junk (such as advertisement, or short personal quote) were extracted. However, a lot of one liner comments from the user are also discarded along with the advertisements, mainly because the comment doesn't have a lot of words in them. Such examples in this page are: “normally i would keep scanning but im pretty focused on AMD now, bought a couple more k at .45”, “If you are bored of the stock market learn cryptocurrencies. Bitcoin has already been one of the largest asset bubbles of all time and it is not over yet.” In total, 6 user posts are under the CTTD thus treated as junk text and therefore discarded.

Text-tag Ratio
Text-tag ratio method extracted 57% of the text from the example page of elitetrader [9] website. In total 699 words are extracted for the example page. In those 699 words that are extracted, all of them are user posts and no junk (such as advertisement, or short personal quote) were extracted. In this method, all the one liner comments are also recognized and therefore extracted.

Word count
Word count method extracted 60% of the text from the example page of elitetrader [9] website. In total 748 words are extracted for the example page. Unfortunately, in those 748 words that are extracted, only 39% of them are user posts. The rest of the text extracted are either junk (such as advertisement, or short personal quote) or information that is website specific (this includes small text on how to log in or information regarding the forum). In this method, all the one liner comments are also recognized and extracted, along with useless information such as “Separate names with a comma.”, “ET IS FREE BECAUSE OF THE FINANCIAL SUPPORT FROM THESE COMPANIES”, “Get the #1 trading app in the App Store” or “(You must log in or sign up to reply here.)”.

33
8 Conclusions and Future work

Subsection 8.1 introduces a discussion of the results presented earlier in Chapter 7. Analysing the performance of the web scraper in two main aspects, speed and accuracy, in subsections 8.1.1 and 8.1.2 respectively. Subsection 8.2 presents the conclusions of the thesis work. The last subsection of this thesis, 8.3 explains the author's’ thought on future work and possible improvements for the web scraper.

Reflection

This thesis present the development of a specialized web scraper for forums. The goal of this degree project was to design and implement a web scraper specialized in forums. The goal was fulfilled during this thesis and documented from design to implementation. A presentation and evaluation of the web scrapers performance in forums was done in order to support the fulfilment of the goal and to verify that the scraper works. The user of the web scraper will benefit from having a specialized web scraper to mine data from forums effectively, since currently there is a lack of availability for scrapers in this area. Any general scraper will extract a great amount of text besides the user posts, thus being slower and consuming more space with each extraction.

By fulfilling the goal, the problem statement was solved with the work documented in this thesis. All the choices and reasoning for how the web scraper should be implemented are detailed, this ties in with the purpose, which was to help the readers gain insight on the different methods to consider for a web scraper specialized in forums.

The related work introduced different ideas and methods to each of the processes of web scraper (navigation and text extraction). As a contribution to the related work, the different text extraction methods implemented were compared and tested as part of a web scraper. This means that their performance and their usability has been tested for web forums and this information can be used for future work in the area of web forums, web scrapers and text extraction as well.

Quality assurance

This thesis work was conducted following the processes of quality assurance for qualitative methodology. These processes are as introduced in Chapter 3, the following: Ethics, Validity, Dependability, Confirmability and Transferability. To assure the quality of this thesis, these processes were taking into consideration and followed in a strict manner.

- Ethics, to assure the integrity and privacy of any research work done for the thesis in which the stakeholders [ref] were involved, legal contract was signed by the authors.
- Validity, to make the work valid, all the methodologies and rules were applied to the thesis.
- Dependability, to ensure the correctness of the conclusion, the results were checked to be reliable. This check was done by performing the testing multiple times before drawn any conclusions from them.
- Confirmability, by presenting all the raw data and using multiple testing units (laptops and servers), any bias was avoided.
- Transferability was achieved by properly document in detail every important step done throughout this thesis work.

By following all the processes of assurance, the quality of this thesis is validated.
8.1 Discussions

Discussions includes the positive effects and the drawbacks of each text extraction method. The discussion is divided into two parts: the overall performance of the web scraper for all three methods in Speed and Text Extraction Accuracy. Speed performance is analysed in section 8.1.1 and Text Extraction Accuracy is analysed in section 8.1.2. The information is discussed and analysed in detail for all the results, to prepare the conclusions drawn in section 8.2.

8.1.1 Speed performance analysis

Analyzing the results from chapter 6, each of those 3 methods clearly show a speed up when increasing the number of cores being used. The Text-Detection Framework method clearly shows an increase in speed when using more cores. For the Text-Detection Framework the speed performance increased from 2975 pages in 5 minutes on average to 6446 in 5 minutes on average by increasing from 4 to 16 cores as seen in figure 18. This means from an estimation of 595 pages per minutes with 4 cores, the speed is just above double the amount of pages per minute with 1290 pages with the use of 16 cores. This method has the highest file size generated, which means that it extract the most data per page as well. Thus the rate of the generated data will be higher than the other methods.

Figure 18. Speed up performance per core in web pages crawled, Text-Detection Framework (generated by author)

Word count method increased in a very positive trend which each additional core used. In this specific case the speed performance increased from 2912 pages in 5 minutes on average to 7850 in 5 minutes on average by increasing from 4 to 16 cores as seen in figure 19. This means from an estimation of 582 pages per minutes with 4 cores, the speed can be almost triplicated to 1570 pages per minute with the use of 16 cores. This clearly shows the benefits of running the scraper in parallel using Java’s multithreaded library. Also by having the lowest average of file size per web page extracted (3.14MB, from Chapter 6), it makes the Word count method the fastest crawler with minimal data generated.
The Text-to-Tag Ratio method has a really stable increase in performance compared to the Word Count method. The speed performance of the Text-to-Tag Ratio increased from 2621 pages in 5 minutes on average to 6241 in 5 minutes on average by increasing from 4 to 16 cores as seen in figure 20. This means from an estimation of 524 pages per minutes with 4 cores, the speed can be said to have increased slightly above twice to 1248 pages per minute with the use of 16 cores. Although there is a benefit in the use of multithreaded for this method, it is not as substantial as the one for Word Count. By having an average of file size per web page extracted of 3.24MB. It makes the Text-to-Tag Ratio method almost as efficient in data generated as the Word Count but by having a lower rate of extraction if falls behind of it when it comes to speed.
8.1.2 Text extraction performance analysis

Based on the result from chapter 5, each of those three methods displays a unique character. The Text detection framework method gives a clear section for each user post extracted, as well as a timestamp for each section to indicate the time this user post is created. Text detection framework does not extract any junk (such as advertisement, or short personal quote) or information that is website specific (this includes small text on how to log in or information regarding the forum). However, all the short user post (e.g. one liner comment) are also considered junk and therefore not extracted.

The Word count method at first seems to extract a lot of text from web pages. However, when we start to analyse the text extracted by word count method, we discovered that some of the text extracted are not what we desired. Around 50% of the text extracted are just junk information (such as advertisement, or short personal quote) or information that is website specific (this includes small text on how to log in or information regarding the forum). On the other hand, short user posts (e.g. one liner comment) are also extracted in the word count method. All of this suggest that the word count method does not seem to filter out the junk information that well.

The Text-tag ratio method has a similar amount of text extracted compare to the word count method, but smaller file size than the text-detection framework method. All the user posts, including the short posts, are extracted. All the text extracted by Text-tag ratio method, was only from user posts. All the junk information (such as advertisement, or short personal quote) or information that is website specific (this includes small text on how to log in or information regarding the forum) for the web page are excluded in the filtering process.

8.2 Conclusion

In terms of speed performance, Text detection framework method and Text-tag ratio method have similar behaviour in performance when the number of cores increases. The Text-tag ratio method seems to be slightly slower than Text Detection Framework method. Word count method however, shows a significant increase in speed performance when increasing the number of active cores to 16. This is most likely because the Word Count methods only relies on the word count of the document, which can be performed quickly. Judging from the graph in the Discussion section, all three methods all seems to have great potential in multi-threading and do not seem to hit their limit yet. In the results section, each of the methods have highs and lows that stand out from the average results in speed. Due to the web crawling of the Jsoup of each web-link being non-deterministic, it is difficult to analyze these peaks. A last thought on this observation is that the testing time is limited to rounds of 5 minutes. This could possibly be a part of the program having to run for a longer period of time until being completely stable in its overall performance. This does not have a greater impact on the over goal of the project, thus they are not discussed in further detail.

In term of text extraction performance, Word count method gives all the text from user posts as well as unwanted advertisements or information that is website specific. An additional information filtering technique may and should be used to improve the end result. Text detection framework method seems to extract only the text from user posts, but at the same time it gets rid of all the small posts (e.g. one liner comments) as well. Therefore it might not be optimal to use text-detection framework method since those small posts are still interesting to
extract. Text-tag ratio method clearly dominates in text extraction performance since it extracts all the user posts from the web page, regardless of their size.

Overall, the method that performed the best was the Text-tag Ratio method. The speed performance showed increased benefit of the multithreading similar to the Text-Detection Framework. While it generates a lot less data, similar to the Word Count, this could be an issue but after comparing the data extracted from each method it showed really good results. The Accuracy test showed that although the Text-tag Ratio method is similar in speed to Word Count, the accuracy of the text is almost pure user posts while keeping the data much lower than the Text-Detection Framework because it filters out junk (i.e any text that is not a user post) very well. As a balanced option in between the other two methods, the Text-tag Ratio easily shows results that are a great match for a scraper focused on web forums.

### 8.3 Future work

The future work is based on problems or ideas that appeared during the implementation of the web scraper and also from the limitations of this project degree.

One problem with forums is that people sometimes like to quote a previous user’s post in forum. Thus quoting results in duplicated posts. This issue has not been resolved by any of the three method. Therefore the first step of the future work is to develop a post cleaning function can be developed to eliminate duplicated forum posts.

Another idea is to try to make the scraper adaptive. This mean that the scraper should recognize the forum specific structure, and remember where in this structure can the scraper get user posts. This means that the web scraper has the ability to identify a relationship from where the text is extracted in the HTML code and reuse this information in other pages of the same forum, thus reducing the extraction process down to a simple matching of tags. The scraper should also recognize the web site is a forum in general: when the scraper get a link to another forum that has not been specified before, it will recognize it and add this link to the “seed-list” csv file.

Another future addition could be adding a topic specific search function for the scraper, so the scraper can go and get only the data about one topic (e.g. trending topic).

Also, even though the current scraper can extract the oldest and newest timestamp for each web page, which gives a rough understanding on whether the user posts in this web page is outdated. It would be more convenient to determine and give each user posts a timestamp. This additional feature will thus greatly help people to get time sensitive data.

Furthermore, for some web forums such as Morningstar’s web forum page [23] uses a database to store all the comments. These forums then use programming languages like javascript to load the comments from a web server( which the scraper does not load when it parses the HTML page source). This makes it incapable of extracting any user posts in these types of forums. Further research can be done to find alternative ways to render the javascript when parsing the HTML page source. Although there are alternatives that can handle these websites, the text density approach has its benefits in performance. Thus a deeper analysis can be carried on in the future about this topic.
References


[17] (c) Copyright Skillsyouneed Com 2011-, “Quantitative and Qualitative Research Methods | SkillsYouNeed.” [Online]. Available:


Appendix A

Example 1:

Text Detection Framework

Mon Jun 19 17:57:56 CEST 2017
Thu May 25 17:57:56 CEST 2017

Ive been trading for a long time, been through the highest highs, and the lowest lows, but have always been able to make a really good living when i need too. Biggest problem im having right now is that its just too easy to walk into work a couple days a week make a few k a month and just keep getting by, ive just lost ambition.

When i started trading our firm posted the top 5 traders out of 2000+, every day, and i used to fight incredibly hard to be in that top 5 every single day, id make it atleast 1 or 2 out of 5 every single week.

Then got burned by a broker once who stole all my money, had multiple systems blow up in my face, it just takes a toll on you. These days i just dont give a shit, theres no competition, nothing, im just drifting, the difference in the PNL numbers is embarassing,

I really want to get back to where i was when i started trading where i was dedicated to trading every single day, and grinding for every dollar, dedicated to “winning” like Charlie Sheen, so im thinking if i start a journal i might feel more committed.

My trading is pretty scattered in terms of what i look for, but id have to say if im looking for anything its gap ups/parabolics when it comes to stocks, and my entries are mostly based on support/resistance, then Volume/VWAP, but bottom line is Price action ”Trumps” all.

Anyways, if you want to join the ”super terrific happy hour” and stare at me while i spin an umbrella please feel free, hopefully we can lose a fortune together.

First tip: Buy Enron.

My trading is pretty scattered in terms of what i look for, but id have to say if im looking for anything its gap ups/parabolics when it comes to stocks, and my entries are mostly based on support/resistance, then Volume/VWAP, but bottom line is Price action ”Trumps” all.

Anyways, if you want to join the ”super terrific happy hour” and stare at me while i spin an umbrella please feel free, hopefully we can lose a fortune together.

First tip: Buy Enron.

&nbsp;
So lets start scanning right away in the morning, something that catches my eye is PLCE, that thing gapped up yesterday on earnings and got crushed while the market was rocking up, my thoughts yesterday was any move through 110 its going to 105, there will probably be upgrades this morning with banks looking to defend it, so look to short in too that cause there is just no way anyone is buying that piece of shit the highs after that bar yesterday, especially with how much it diverged from the sandp, i think that was the only stock in the market that closed on its lows yesterday.

Last edited:
May 19, 2017 at 6:23 AM

#2 &nbsp; &nbsp; May 19, 2017 at 6:07 AM
ASNA, i dont usually like trading gap downs but the volume bar is massive, there was big time rotation yesterday. i will pay attention too it if it looks like it is holding over 2, i could easily see a push to 2.50 and it will be easy to take size.

Last edited:
May 19, 2017 at 6:11 AM

#3 &nbsp; &nbsp; May 19, 2017 at 6:11 AM
GLYC it seems insane, but i know tons of people were shorting that thing from 9 if it gaps up i might think about buying it because its probably going to turn into a squeeze towards 15, normally i would never take this kind of trade but it seems like such an obvious trap. Dont short a Gap up on this under any scenario, you might win, but you are just asking for trouble.

Last edited:
May 19, 2017 at 6:25 AM

Word Count
NVDA/AMD both bouncing, I really like the look of AMD, buying AMD gap up might be one of my top plays today but i got to wait till closer till the open. Just took a couple thousand AMD long from 11.48 i guess with commissions thats my top play i will look to build into that over 11.50 or sell it if it cracks down

Terms and Rules
Ive been trading for a long time, been through the highest highs, and the lowest lows, but have always been able to make a really good living when i need too. Biggest problem im having right now is that its just too easy to walk into work a couple days a week make a few k a month and just keep getting by, ive just lost ambition.
I know i probably shouldn't have to say this but DO NOT BUY/SHORT these symbols im just openly going through my process, i will short list this too 5-6 stocks then eventually just pick 1-2

Trade Logging & Analysis
Separate names with a comma.

Futures Trading Platforms and Order Routing
Forums &gt; General Topics &gt; Journals &gt;
The Super Terrific Happy Hour | Elite Trader
Get the #1 trading app in the App Store

GLYC it seems insane, but i know tons of people were shorting that thing from 9 if it gaps up i might think about buying it because its probably going to turn into a squeeze towards 15, normally i would never take this kind of trade but it seems like such an obvious trap. Dont short a Gap up on this under any scenario, you might win, but you are just asking for trouble.

(Don't reply unless you are logged in or feel really comfortable.)

ET IS FREE BECAUSE OF THE FINANCIAL SUPPORT FROM THESE COMPANIES:
When I started trading our firm posted the top 5 traders out of 2000+, every day, and i used to fight incredibly hard to be in that top 5 every single day, id make it at least 1 or 2 out of 5 every single week.

So lets start scanning right away in the morning, something that catches my eye is PLCE, that thing gapped up yesterday on earnings and got crushed while the market was rocking up, my thoughts yesterday was any move through 110 its going to 105, there will probably be upgrades this morning with banks looking to defend it, so look to short in too that cause there is just no way anyone is buying that piece of shit the highs after that bar yesterday, especially with how much it diverged from the sandp, i think that was the only stock in the market that closed on its lows yesterday.

Discussion in 'Journals' started by Max E., May 19, 2017 at 5:21 AM.

I really want to get back to where i was when i started trading where i was dedicated to trading every single day, and grinding for every dollar, dedicated to "winning" like Charlie Sheen, so im thinking if i start a journal i might feel more committed.

My trading is pretty scattered in terms of what i look for, but id have to say if im looking for anything its gap ups/parabolics when it comes to stocks, and my entries are mostly based on support/resistance, then Volume/VWAP, but bottom line is Price action "Trumps" all.

Then got burned by a broker once who stole all my money, had multiple systems blow up in my face, it just takes a toll on you. These days i just dont give a shit, there's no competition, nothing, im just drifting, the difference in the PNL numbers is embarassing, ASNA, i dont usually like trading gap downs but the volume bar is massive, there was big time rotation yesterday, i will pay attention too it if it looks like it is holding over 2, i could easily see a push to 2.50 and it will be easy to take size.

normally i would keep scanning but im pretty focused on AMD now, bought a couple more k at .45

You are using an out of date browser. It may not display this or other websites correctly. You should upgrade or use an alternative browser.

If you are bored of the stock market learn crypto currencies. Bitcoin has already been one of the largest asset bubbles of all time and it is not over yet.

Text-tag ratio

I've been trading for a long time, been through the highest highs, and the lowest lows, but have always been able to make a really good living when i need too. Biggest problem im having
right now is that its just too easy to walk into work a couple days a week make a few k a month and just keep getting by, ive just lost ambition.

When I started trading our firm posted the top 5 traders out of 2000+, every day, and i used to fight incredibly hard to be in that top 5 every single day, id make it atleast 1 or 2 out of 5 every single week.

Then got burned by a broker once who stole all my money, had multiple systems blow up in my face, it just takes a toll on you. These days i just dont give a shit, theres no competition, nothing, im just drifting, the difference in the PNL numbers is embarassing,

I really want to get back to where i was when i started trading where i was dedicated to trading every single day, and grinding for every dollar, dedicated to “winning” like Charlie Sheen, so im thinking if i start a journal i might feel more committed.

My trading is pretty scattered in terms of what i look for, but id have to say if im looking for anything its gap ups/parabolics when it comes to stocks, and my entries are mostly based on support/resistance, then Volume/VWAP, but bottom line is Price action “Trumps” all.

Anyways, if you want to join the “super terrific happy hour” and stare at me while i spin an umbrella please feel free, hopefully we can lose a fortune together.

So lets start scanning right away in the morning, something that catches my eye is PLCE, that thing gapped up yesterday on earnings and got crushed while the market was rocking up, my thoughts yesterday was any move through 110 its going to 105, there will probably be upgrades this morning with banks looking to defend it, so look to short in too that cause there is just no way anyone is buying that piece of shit the highs after that bar yesterday, especially with how much it diverged from the sandp, i think that was the only stock in the market that closed on its lows yesterday.

ASNA, i dont usually like trading gap downs but the volume bar is massive, there was big time rotation yesterday, i will pay attention too it if it looks like it is holding over 2, i could easily see a push to 2.50 and it will be easy to take size.

GLYC it seems insane, but i know tons of people were shorting that thing from 9 if it gaps up i might think about buying it because its probably going to turn into a squeeze towards 15, normally i would never take this kind of trade but it seems like such an obvious trap. Don’t short a Gap up on this under any scenario, you might win, but you are just asking for trouble.

BEDU biggest whiff in history cant believe i didnt buy the breakout yesterday probably goes up another couple bucks.

I know i probably shouldnt have to say this but DO NOT BUY/SHORT these symbols im just openly going through my process, i will short list this too 5-6 stocks then eventually just pick 1-2

NVDA/AMD both bouncing, I really like the look of AMD, buying AMD gap up might be one of my top plays today but i got to wait till closer till the open.

If you are bored of the stock market learn crypto currencies. Bitcoin has already been one of the largest asset bubbles of all time and it is not over yet.

Just took a couple thousand AMD long from 11.48 i guess with commissions thats my top play i will look to build into that over 11.50 or sell it if it cracks down normally i would keep scanning but im pretty focused on AMD now, bought a couple more k at .45
The Super Terrific Happy Hour
Discussion in 'Journals' started by Max E., May 19, 2017.

Max E. 11,456 Posts 1,522 Likes

I've been trading for a long time, been through the highest highs, and the lowest lows, but have always been able to make a really good living when i need too. Biggest problem im having right now is that its just too easy to walk into work a couple days a week make a few k a month and just keep getting by, i've just lost ambition.

When i started trading our firm posted the top 5 traders out of 2000+, every day, and i used to fight incredibly hard to be in that top 5 every single day id make it atleast 1 or 2 out of 5 every single week.

Then got burned by a broker once who stole all my money, had multiple systems blow up in my face, it just takes a toll on you. These days i just don't give a shit, theres no competition, nothing, i'm just drifting, the difference in the PNL numbers is embarassing.

I really want to get back to where i was when i started trading where i was dedicated to trading every single day, and grinding for every dollar, dedicated to "winning" like Charlie Sheen, so im thinking if i start a journal i might feel more committed.

My trading is pretty scattered in terms of what i look for, but id have to say if im looking for anything its gap ups/parabolics when it comes to stocks, and my entries are mostly based on support/resistance, then Volume/VWAP. but bottom line is Price action "Trumps" all.

Anyways, if you want to join the "super terrific happy hour" and stare at me while i spin an umbrella please feel free, hopefully we can lose a fortune together.

First tip: Buy Enron.
Seinfeld - "The Checks" Super Terrific Happy Hour

Max E.

So lets start scanning right away in the morning, something that catches my eye is PLCE, that thing gapped up yesterday on earnings and got crushed while the market was rocking up, my thoughts yesterday was any move through 110 its going to 105, there will probably be upgrades this morning with banks looking to defend it, so look to short in too that cause there is just no way anyone is buying that piece of shit the highs after that bar yesterday, especially with how much it diverged from the sandp. i think that was the only stock in the market that closed on its lows yesterday.

ASNA, i dont usually like trading gap downs but the volume bar is massive, there was big time rotation yesterday, i will pay attention too it if it looks like it is holding over 2, i could easily see a push to 2.50 and it will be easy to take size.
GLYC it seems insane, but I know tons of people were shorting that thing from 9 if it gaps up I might think about buying it because it's probably going to turn into a squeeze towards 15, normally I would never take this kind of trade but it seems like such an obvious trap. Don't short a gap up on this under any scenario, you might win, but you are just asking for trouble.

Max E.  11,456 Posts  1,522 Likes

BEDU biggest whiff in history can't believe I didn't buy the breakout yesterday probably goes up another couple bucks.

Max E.  11,456 Posts  1,522 Likes

I know I probably shouldn't have to say this but DO NOT BUY/SHORT these symbols im just openly going through my process, I will short list this too 5-6 stocks then eventually just pick 1-2

Max E.  11,456 Posts  1,522 Likes

NVDA/AMD both bouncing, I really like the look of AMD, buying AMD gap up might be one of my top plays today but I got to wait till closer till the open.

just21  4,736 Posts  250 Likes

If you are bored of the stock market learn crypto currencies. Bitcoin has already been one of the largest asset bubbles of all time and it is not over yet.

Max E.  11,456 Posts  1,522 Likes

Just took a couple thousand AMD long from 11.48 I guess with commissions that's my top play I will look to build into that over 11.50 or sell it if it cracks down

https://www.elitetrader.com/et/threads/the-super-terrible-happy-hour.309621/
normally i would keep scanning but im pretty focused on AMD now, bought a couple more k at .45
Appendix B

Link to full source code of the web crawler:

https://github.com/KarasMI7/Text_extracting_crawler

Some questions asked to the stakeholder - No answers recorded

- What are the requirements of the web scraper?
  - Should it skip links it already visited?
- How will the text extraction process proceed?
  - How should the extracted data be stored?
  - How should the extracted links be handled?
- What data is considered relevant in a forum?
- How will the scraper interact with their existing program?