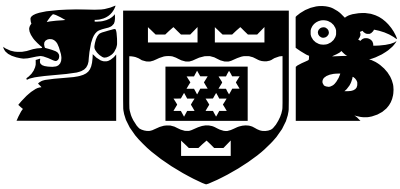
VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui



School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

Reputation Scraper - Social Media

Iain Walker

Supervisor: Kris Bubendorfer

Submitted in partial fulfilment of the requirements for
ENGR489 - Bachelor of Engineering.

Abstract

Online reputation has become increasingly important as more social and business interactions move online. This project is concerned with investigating how we can infer basic reputation metrics from non-traditional sources such as social media sites, including Facebook, Twitter, Google+, LinkedIn, and Slashdot. The method of gathering has been through web-scraping, since most social media sites do not (understandably) have an API to retrieve this information without logging in to the system. The project has looked at developing web-scrapers to retrieve publicly-available information on social media sites, and investigating how this can be used to infer basic reputation metrics.

Contents

Figures

Chapter 1

Methodology

1.1 Project Management Approach

The project was structured around a loose waterfall approach. At the start of the project, a long-term plan and major milestones were outlined. More detailed plans were added and target dates adjusted as the year progressed. Aspects of agile development practices were also used during the implementation and design stages. Throughout the project I had joint weekly meetings with Dr. Kris Bubendorfer, Ferry Hendrikx and Filip Dimitrievski. In these meetings, 30-45 minutes in length, we were able to outline progress achieved during the week, any issues encountered, and get answers to any questions about the project as a whole. Having meetings with Filip and Ferry present was beneficial, as there were aspects of the project which Filip and I were able to collaborate over. Ferry, co-designer of the GRAft system was also able to give useful technical feedback, with his experience in web-scraping.

This approach of combining aspects of waterfall and agile was reasonably effective as a project management approach. Agile methods tend to work best in a team environment, assisting with the coordination of team members. However the method of small, focused sprints contributing to the larger project were excellent in maintaining focus and direction. The waterfall element in turn outlined the more concrete and long-term goals of the project, which was useful with monitoring and controlling and ensuring my work did not fall behind.

These target goals were met in the majority of cases. I did experience delays with finishing my implementation and evaluation, as new goals such as community detection were added late in the project. Reasons for this late discovery and delay were in part due to the exploritory nature of the project, as discussed in the design approach.

1.2 Design Approach

Requirements analysis and design were completed through a combination of research and prototyping. As the project was focused on an exploration of understanding reputation data on social media, a large portion of time was given to background research. This will be covered in depth in chapter 3.

Later in the design phase, constructed a set of prototypes to evaluate the feasiblity and alternatives for a web-scraping solution. These covered preliminary scrapers for Twitter, Facebook, LinkedIn and Slashdot. The benefit of developing these was to both give me a better understanding of technologies involved with performing these functions, and to produce some meaningful effort early in the project, as these scrapers could be potentially useful later. There were some pitfalls in this approach however. It is noted that some effort

in prototyping can go to waste - and this was the case here. We eventually declared web-scraping Facebook largely infeasible, due to its user interface's constant state of flux. Also during prototyping, Twitter's API changed significantly, rendering much effort lost.

The weekly meetings with my supervisor allowed opportunities to obtain feedback on design choices, as well as suggestions where there was room for improvement.

1.3 Project Complexities

The complexity of the system stems largely from the aggregation of unstructured data from a variety of sources - a difficulty often encountered in web-crawling applications. The codebase itself is not overly complex. However, what contributed to the primary difficulty of prototype development was the development of code in the aggregation of data from disparate locations, and debugging often unclear and unexpected errors from various web requests.

Understanding the data gathered was a time-expensive challenge. The process of understanding and defining reputation data on social media was the task that occupied most of my time on the project.

The time constraint of 300 hours impacted the project at all stages. The implementation and evaluation components were particularly impacted - limitations had to be placed upon the scale of data collected from scrapers in order to compensate for time. In addition, the selected prototyping methodology was often expensive in terms of time, due to the necessity of revising code and collecting more relevant data, as I achived greater understanding of the problem domain.

Understanding reputation data, and writing policies to describe reputation effectively.

Debugging and collection of data, and asserting that this data is avalid. Ensuring that websites were not overloaded, resulting usually in scrapers being blocked. Recovery from detection, and how scrapers can respond. Construction of useful policies, and data analysis and aggregation.

Chapter 2

Requirements Analysis

To satisfy the goals of the project, the following issues need to be addressed. The requirements may be logically split into two divisions; web-scraping requirements and access policy requirements.

2.0.1 Functional Requirements

R1. Policies must consider data from reasonable time periods, to generate a shadow of the future.

R2. Aggregation of social media data into consistent and readable format

R3. Metric of reliability based on amounts of data collected.

2.0.2 Non-Functional Requirements

R4. Resistance to User-Interface Change

R5. Reasonable performance - expectation that policies and scrapers could be used as part of wider application.

R6. Accuracy of data collected - content should not be missing or incorrect

R7. Resistance to Blocking Detection - my scrapers should not be blocked.

2.0.3 Discarded Requirements

R8 Aggregation of Social Media data, for storage in GRAft.

2.1 Policy Construction

The first and most significant requirement is to generate a set of policies to assist with generating a snapshot of the reputation information of an individual.

2.2 A reliability Metric Based on the Amount of Data Considered

From the beginning of the project I acknowledged that it may not be feasible to scrape the profile or data of an entire user

2.3 Social Media Platform Portability

The ability to build scrapers for new sites on top of existing architecture I develop.

2.4 Maintainability and Resistance to User-Interface Changes

Along with portability of my scrapers, they need to be somewhat resistant to change at the user interface level. An oft-repeated downfall of web-scraping is that changes to interfaces may occur at any time, without warning. This is in contrast to changing APIs, which generally give some significant warning and phasing out period of functions. Often changes to the layout of pages will break crawlers, resulting in a need for frequent re-builgs on sites that go regular user interface change. As such my system must be designed in a manner that is as un-reliant on layout-specific information as possible. In cases where change will unavoidably break my scrapers, they should be designed in a fashion which allows for easy identification and solving of issues.

2.5 Scraper Performance

This requirement refers to the speed and accuracy of my web-scrapers. The web-scraping tools developed must perform with sufficient speed to generate snapshots of an individual's reputation, in a reasonable amount of time. In order to create policies that are useful for future works, these scrapers must be able to gather and make conclusions about an individual in a matter of seconds or minutes. Data gathered must be accurate and represent what is actually displayed on a site.

2.6 Ability to Resist Blocking Detection and Recover from Failure

A challenge presented in web-scraping is the ability to resist detection as a robot by webservers. Webservers do not look well upon robots, and will attempt to block aggresive crawlers. As such my scrapers need to implement strategies to avoid detection, and when blocked or detected, take appropriate action. A balance has to be achieved between performance of scrapers and detection by webservers - normally a scraper attempting to retrieve masses of results will be detected and blocked extremely rapidly.

2.7 Privacy Protection

This requirement refers to the fact that data I gather should be anonymised to a reasonable extent, such that actual personal details should not be traceable.

Maintain reasonable privacy of data I collect. Discuss how the information I am gathering is publicly avaialble on social media anyway, and users should have reasonable expectation that their data will be accessed.

2.8 Discarded Requirements

Aggregate data for storage into GRAft - because of community effort required. Business model would need to be constructed, out of scope for the project.

Visualisation component of the project- discarded as was able to source external tools to visualise and represent data.

Chapter 3

Scraper Design and Implementation

In this section I discuss the design and implementation of the web scraping components of the project. I highlight important decisions made during the course of the project, as well as steps taken to implement the system based on requirements detailed in chapter four.

3.1 Architecture

Two architectures were considered when designing the web-scraper components. A database-storage approach was considered, in which

3.1.1 Database Storage

One possible approach to meeting the goals of the project was to store fetched data in a relational database.

Advantages - relational model, persistance, backups. Simplicty of fetching exact data.

Disadvantages - extra complexity deemed unnecessary. Worse integration with GRAft.

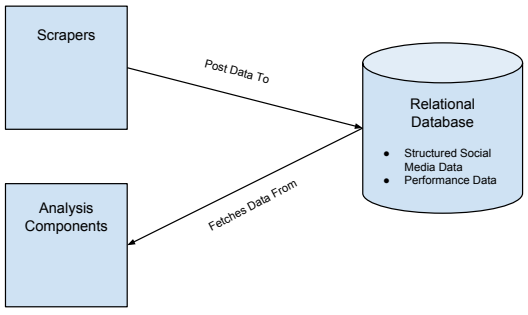


Figure 3.1: Proposed Architecture for Database Storage Solution

3.1.2 XML Storage

Advantages - simplicty of scripting. Simplicty of integration with GRAft. Simplicity of changing structures.

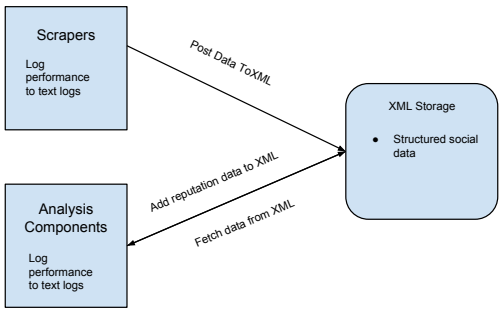


Figure 3.2: Proposed Architecture for XML Storage Solution

3.2 Technology

My scrapers were written entirely in Ruby. Developing screen scrapers is largely independant upon language choice; however Ruby was selected due to its large number of libraries suitable for scraping, and straightforward scripting nature. Ruby also has a significant OpenSource culture in comparison to many other languages.This was considered important early in the project when looking at developing a model for future maintenance of the scrapers, especially when considering the discarded requirement of aggregating data for GRAft.

Alternatives to Ruby were considered; Java, which does not have the same open source following as Ruby, as well as a larger degree of lower-level network coding for web-scraping software; and PHP, which was rejected due to personal time constraints restricting personal capabilities of learning the language to a competent level during the project. PHP would have given the advantage of being more consistent with past works, however [**?**], as well as being the same language as my policies are described in.

The frameworks upon which my scrapers are built included:

- Nokogiri: a widely used HTML and XML parsing library, allowing for straightforward interpretation of raw HTML documents.
- Mechanize: a library simulating browser interaction on web-pages, without requiring a real browser.
- Rest-Client: Ruby's most popular HTTP client.

3.2.1 Systems Design and Structure

Developing the project solution consisted generally of a cyclic, iterative prototyping approach as detailed in figure 5.1. This generally consisted of four phases; Implementing and adjusting scrapers, fetching data, analysing the data, and prototyping policy concepts as a result of this analysis.

Various frameworks were considered when implementing my standalone scraping application. Option one was to extend scrAPI, a Rubyforge project allowing for rapid development of web scrapers. The primary benefit claimed by scrAPI is that it would allow me

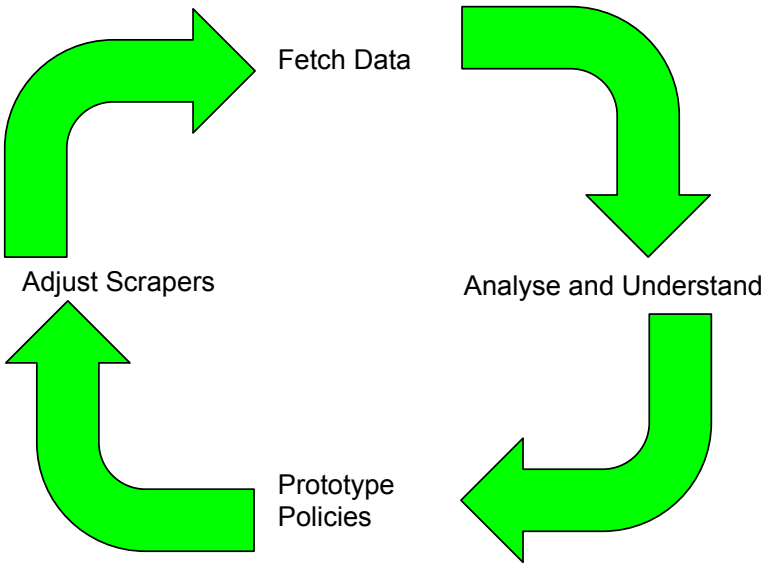


Figure 3.3: Scraper Development Feedback Loop

to fetch data from HTML pages using CSS selectors. It also hid processes such as the actual fetching of pages, and sending of HTTP requests.

Browser automation options were considered as an alternative architecture. Libraries such as Watir allow users to simulate user interaction with web pages, by driving an actual browser instance. Watir - browser automated option. More for automated tests. Performance would be significantly worse than nokogiri, as pages are loaded through the browser. Also personal experience from building such applications shows that implementation of such a solution would be very time consuming. Advantages of this would be low detection rates.

3.2.2 Code instrumentation

The code was instrumented using a Ruby logger system written for the application. Given complexities and difficulties debugging errors on web-scraping applications, logging had to be performed to a very fine level of granularity. Any action changing system state such as fetching a page triggers the logging mechanism. The logger would then take note of the timestamp and write to the appropriate file the nature of the action. For example, if the scraper sent an HTTP request to retrieve a given URL, it would record the timestamp and URL requested.

The logger would write to the appropriate log file based on the nature of the supplied action. Because these scrapers were running over long periods of time, using traditional IDE debugging tools was not effective at detecting errors. As a result, I used multiple debugging files with different purposes in an attempt to catch these errors. The debug.log captured all interaction information at a basic level. Error.log captures error information that is non-fatal to scraping an individual's profile, e.g. on Twitter. Commonly these errors were due to application logic flaws, such as performing operations on null entities. As a result the error.log assisted greatly in identifying these edge cases. Finally the failure.log was used to record fatal exceptions that would prevent me from scraping a profile. Occurences such as 404, 500 or 503 responses from servers are examples that would result in a record being written to the failure log. The failure log would write system state at the time of failure to a

high level of detail, sometimes even writing the entire HTML document before the failure to disk. This again assisted with debugging, when reviewing how a particular run had gone.

To limit the performance overhead of writing to these logging files, a buffered approach was taken in order to achieve the least impact.

3.3 Dataset

3.3.1 Extend scrAPI

scrAPI is a Rubyforge project, allowing for fast implementation of web scrapers. The benefit of scrAPI is that it would allow me to fetch data from HTML pages using CSS selectors. It also hid processes such as the actual fetching of pages, and sending of HTTP requests. Ultimately scrAPI was a more high-level approach to scraping content.

Extending scrAPI was ultimately discarded however, due to its heavy reliance on CSS files remaining constant. Any change to stylesheet files would likely have broken my scrapers. Arguably these stylesheets are less likely to change than layout manipulations (e.g. consider xpath on HTML as an alternative); however on sites such as Twitter and Facebook large design teams frequently make changes to these files. Given that a key requirement of the project was to make scrapers resistant to user-interface change, this resulted in scrAPI being deemed unfit for purpose.

3.3.2 Extend A Browser-Automation Model

Browser automation options were considered as an alternative architecture. Libraries such as Watir allow users to

Watir - browser automated option. More for automated tests. Performance would be significantly worse than nokogiri, as pages are loaded through the browser. Also personal experience from building such applications shows that implementation of such a solution would be very time consuming. Advantages of this would be low detection rates.

3.4 Social Media Selection

In order to scrape useful information, I needed to look at the various social media sites in the context of gathering reputation data, and seeing what could be selected.

3.4.1 Standalone Crawling Application

A standalone crawling application architecture was considered, and ultimately chosen as the most suitable strategy.

3.5 Product Implementation

3.5.1 Technology Choice

3.5.2 Twitter Scraper

Initial prototype

Design improvements, including multi-threading and parallel fetching of tweets. Decisions around discarding requests that fail in scraping individual tweets.

3.5.3 LinkedIn Scraper

3.5.4 Facebook Scraper

Bibliography

[1] ADALI, S., ESCRIVA, R., GOLDBERG, M. K., HAYVANOVYCH, M., MAGDON-ISMAIL, M., SZYMANSKI, B. K., WALLACE, W. A., AND WILLIAMS, G. Measuring behavioral trust in social networks. In *Intelligence and Security Informatics (ISI), 2010 IEEE International Conference on* (2010), IEEE, pp. 150–152.

[2] ADALI, S., AND GOLBECK, J. Predicting personality with social behavior. In *Advances in Social Networks Analysis and Mining (ASONAM), 2012 IEEE/ACM International Conference on* (2012), IEEE, pp. 302–309.

[3] ADALI, S., SISENDA, F., AND MAGDON-ISMAIL, M. Actions speak as loud as words: Predicting relationships from social behavior data. In *Proceedings of the 21st international conference on World Wide Web* (2012), ACM, pp. 689–698.

[4] ARRINGTON, M. Facebook users revolt, facebook replies. `http://techcrunch.com/2006/09/06/facebook-users-revolt-facebook-replies/`, 2006.

[5] BACHRACH, Y., KOSINSKI, M., GRAEPEL, T., KOHLI, P., AND STILLWELL, D. Personality and patterns of facebook usage. In *Proceedings of the 3rd Annual ACM Web Science Conference* (2012), ACM, pp. 24–32.

[6] BACK, M. D., STOPFER, J. M., VAZIRE, S., GADDIS, S., SCHMUKLE, S. C., EGLOFF, B., AND GOSLING, S. D. Facebook profiles reflect actual personality, not self-idealization. *Psychological Science 21*, 3 (2010), 372–374.

[7] BRABHAM, D. C. Crowdsourcing as a model for problem solving an introduction and cases. *Convergence: the international journal of research into new media technologies 14*, 1 (2008), 75–90.

[8] BRODY, H. I don't need no stinking api: Web-scraping for fun and profit. `http://blog.hartleybrody.com/web-scraping/`, 2013.

[9] BRZOZOWSKI, M. J., HOGG, T., AND SZABO, G. Friends and foes: ideological social networking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2008), ACM, pp. 817–820.

[10] DE RAAD, B. *The Big Five Personality Factors: The psycholexical approach to personality.* Hogrefe & Huber Publishers, 2000.

[11] DEBATIN, B., LOVEJOY, J. P., HORN, A.-K., AND HUGHES, B. N. Facebook and online privacy: Attitudes, behaviors, and unintended consequences. *Journal of Computer-Mediated Communication 15*, 1 (2009), 83–108.

[12] DOAN, A., RAMAKRISHNAN, R., AND HALEVY, A. Y. Crowdsourcing systems on the world-wide web. *Communications of the ACM 54*, 4 (2011), 86–96.

[13] DUBOIS, T., GOLBECK, J., AND SRINIVASAN, A. Predicting trust and distrust in social networks. In *Privacy, security, risk and trust (passat), 2011 ieee third international conference on and 2011 ieee third international conference on social computing (socialcom)* (2011), IEEE, pp. 418–424.

[14] FACEBOOK. Facebook quaterly earnings slides, q4 2012. `http://www.scribd.com/doc/123034877/Facebook-Q4-2012-Investor-Slide-Deck`, 2012.

[15] GAL-OZ, N., GRINSHPOUN, T., GUDES, E., AND MEISELS, A. Cross-community reputation: Policies and alternatives. In *Proceedings of the IADIS International Conference on Web Based Communities, Amsterdam, The Netherlands* (2008), pp. 197–201.

[16] GOLBECK, J., AND HANSEN, D. Computing political preference among twitter followers. In *Proceedings of the 2011 annual conference on Human factors in computing systems* (2011), ACM, pp. 1105–1108.

[17] GOLBECK, J., KOEPFLER, J., AND EMMERLING, B. An experimental study of social tagging behavior and image content. *Journal of the American Society for Information Science and Technology 62*, 9 (2011), 1750–1760.

[18] GOLBECK, J., ROBLES, C., EDMONDSON, M., AND TURNER, K. Predicting personality from twitter. In *Privacy, security, risk and trust (passat), 2011 ieee third international conference on and 2011 ieee third international conference on social computing (socialcom)* (2011), IEEE, pp. 149–156.

[19] GOLBECK, J., ROBLES, C., AND TURNER, K. Predicting personality with social media. In *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems* (2011), ACM, pp. 253–262.

[20] GUHA, R., KUMAR, R., RAGHAVAN, P., AND TOMKINS, A. Propagation of trust and distrust. In *Proceedings of the 13th international conference on World Wide Web* (2004), ACM, pp. 403–412.

[21] HENDRIX, F., AND BURBENDORFER, K. Malleable access rights to establish and enable scientific collaboration. Unpublished paper, submitted to eScience 2013 Conference.

[22] KEWALRAMANI, M. Sentiment analysis on twitter.

[23] KLOUT. Klout.com. `http://klout.com`.

[24] KOSINSKI, M., KOHLI, P., STILLWELL, D., BACHRACH, Y., AND GRAEPEL, T. Personality and website choice. In *ACM Web Science Conference* (2012), pp. 251–254.

[25] KOSINSKI, M., STILLWELL, D., AND GRAEPEL, T. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences* (2013).

[26] KUNEGIS, J., LOMMATZSCH, A., AND BAUCKHAGE, C. The slashdot zoo: mining a social network with negative edges. In *Proceedings of the 18th international conference on World wide web* (2009), ACM, pp. 741–750.

[27] LESKOVEC, J., HUTTENLOCHER, D., AND KLEINBERG, J. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web* (2010), ACM, pp. 641–650.

[28] LURIE, I. How to:scrape search engines without pissing them off. http://searchnewscentral.com/20110928186/General-SEO/how-to-scrape-search-engines-without-pissing-them-off.html, 2011.

[29] OPENID. http://openid.net/, 2013.

[30] QUERCIA, D., KOSINSKI, M., STILLWELL, D., AND CROWCROFT, J. Our twitter profiles, our selves: Predicting personality with twitter. In *Privacy, security, risk and trust (passat), 2011 ieee third international conference on and 2011 ieee third international conference on social computing (socialcom)* (2011), IEEE, pp. 180–185.

[31] QUERCIA, D., LAMBIOTTE, R., STILLWELL, D., KOSINSKI, M., AND CROWCROFT, J. The personality of popular facebook users. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work* (2012), ACM, pp. 955–964.

[32] RESNICK, P., KUWABARA, K., ZECKHAUSER, R., AND FRIEDMAN, E. Reputation systems. *Communications of the ACM 43*, 12 (2000), 45–48.

[33] RESNICK, P., AND ZECKHAUSER, R. Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system. *Advances in applied microeconomics 11* (2002), 127–157.

[34] SABATER, J., AND SIERRA, C. Review on computational trust and reputation models. *Artificial Intelligence Review 24*, 1 (2005), 33–60.

[35] STELZNER, M. 2013 social media marketing industry report. http://www.socialmediaexaminer.com/social-media-marketing-industry-report-2013/, 2013.

[36] WADWHA, T. Lessons from crowdsourcing the boston bombing investigation. http://www.forbes.com/sites/tarunwadhwa/2013/04/22/lessons-from-crowdsourcing-the-boston-marathon-bombings-investigation/, 2013.

[37] WARDEN, P. How i got sued by facebook. http://petewarden.typepad.com/searchbrowser/2010/04/how-i-got-sued-by-facebook.html, 2010.

[38] XU, Y., CAO, X., SELLEN, A., HERBRICH, R., AND GRAEPEL, T. Sociable killers: understanding social relationships in an online first-person shooter game. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work* (2011), ACM, pp. 197–206.

[39] ZIEGLER, C.-N., AND LAUSEN, G. Propagation models for trust and distrust in social networks. *Information Systems Frontiers 7*, 4-5 (2005), 337–358.