Data Mining in Dynamic Social Networks and Fuzzy Systems

Vishal Bhatnagar
Ambedkar Institute of Advanced Communication Technologies & Research, India

A volume in the Advances in Data Mining and Database Management (ADMDM) Book Series

Information Science REFERENCE
An Imprint of IGI Global

Managing Director: Lindsay Johnston
Editorial Director: Joel Gamon
Production Manager: Jennifer Yoder
Publishing Systems Analyst: Adrienne Freeland
Development Editor: Christine Smith
Acquisitions Editor: Kayla Wolfe
Typesetter: Erin O'Dea
Cover Design: Jason Mull

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

Library of Congress Cataloging-in-Publication Data

Data mining in dynamic social networks and fuzzy systems / Vishal Bhatnagar, editor.
pages cm
Includes bibliographical references and index.
Summary: "This book brings together research on the latest trends and patterns of data mining tools and techniques in dynamic social networks and fuzzy systems"--Provided by publisher.
ISBN 978-1-4666-4213-3 (hardcover) -- ISBN 978-1-4666-4214-0 (ebook) -- ISBN 978-1-4666-4215-7 (print & perpetual access) 1. Social networks--Research. 2. Social sciences--Network analysis. 3. Online social networks--Research. 4. Data mining. 5. Fuzzy systems. I. Bhatnagar, Vishal, 1977-
HM741.D384 2013
006.3'12--dc23

2013009733

This book is published in the IGI Global book series Advances in Data Mining and Database Management (ADMDM) (ISSN: 2327-1981; eISSN: 2327-199x)

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter 3
Emergent Data Mining Tools for Social Network Analysis

Dhiraj Murthy
Bowdoin College, USA

Alexander Gross
Bowdoin College, USA

Alex Takata
Bowdoin College, USA

ABSTRACT

This chapter identifies a number of the most common data mining toolkits and evaluates their utility in the extraction of data from heterogeneous online social networks. It introduces not only the complexities of scraping data from the diverse forms of data manifested in these sources, but also critically evaluates currently available tools. This analysis is followed by a presentation and discussion on the development of a hybrid system, which builds upon the work of the open-source Web-Harvest framework, for the collection of information from online social networks. This tool, VoyeurServer, attempts to address the weaknesses of tools identified in earlier sections, as well as prototype the implementation of key functionalities thought to be missing from commonly available data extraction toolkits. The authors conclude the chapter with a case study and subsequent evaluation of the VoyeurServer system itself. This evaluation presents future directions, remaining challenges, and additional extensions thought to be important to the effective development of data mining tools for the study of online social networks.

INTRODUCTION

With the increased pervasiveness of the internet, society has seen exponential growth in digital data that has been made available on global public networks. With this rise of 'Big Data,' researchers have seen the need to identify, organize, collect, and extract this information back out of the system and into useful forms (Hammer, Garcia-Molina, Cho, Aranha, & Crespo, 1997). The fields of data mining and web-content extraction are critical to this process and have remained active areas of re-

DOI: 10.4018/978-1-4666-4213-3.ch003

search, as the types and forms of data available on the Web have continued to grow and evolve. The continued growth of information on the Web - due in part to more recent trends of fully online, social, and context aware computing - have made more types of data available, which are of potential use in a highly interdisciplinary range of fields. Many disciplines are looking at 'Big Data' and ways to mine and analyze these data as the key to solving everything from technical problems to better understanding social interactions. For example, large sets of tweets mined from Twitter have been analyzed to detect natural disasters (Doan, Vo, & Collier, 2011; Hughes, Palen, Sutton, Liu, & Vieweg, 2008; Murthy & Longwell, in press), predict the stock market (Bollen & Mao, 2011), and track the time of our daily rituals (Golder & Macy, 2011). As our use of blogs, social networks, and social media continues to increase, so does our creation of more web-based hyperlinked data. The successful extraction of this web-based data is of considerable research and commercial value.

Data mining often goes beyond simple information retrieval and has moved towards a meta-discovery of structures and entities hidden in seas of data. As our social interactions become increasingly mediated by Internet-based technologies, the potential to use web-based data for understanding social structures and interactions will continue to increase.

Online social networks are defined as 'web-based services that allow individuals to (1) construct a public or semi-public profile within a bounded system, (2) articulate a list of other users with whom they share a connection, and (3) view and traverse their list of connections and those made by others within the system' (Boyd & Ellison, 2008). Individuals interact within online social networks through portals such as Facebook, which create social experiences for the user by creating a personalized environment and interaction space by combining knowledge of one users' online activity and relationships with information about other networked individuals. It is through data mining algorithms that Twitter, for example, determines recommendations for users to follow or topics that may be of potential interest. One way to study social networks is by examining relationships between users and the attributes of these relationships. However, data on a blog, Facebook, or Twitter is not directly translatable into network-based data that would be useful within research praxis, and this is where the ability to perform effective data mining becomes important. Social networks typically only provide individual portal access to one's egocentric network. Put in the language of social network analysis (SNA), the visible network is constructed in relation to ego (the individual being studied) and relations of ego, known as 'alters,' are seen (e.g. Facebook friends). However, in a restricted profile environment, the alters' relationships are not revealed. In order to understand network structure (which is key to a systems perspective), the researcher must use methods like data mining in order to gather information about all users and interactions by iterating over the data. A variety of different types of tools have been developed to collect this web-based information. These tools were created for a wide array of purposes. The majority of these tools have been commercially released. Some of these tools can be used to construct profiles of individuals based on data from multiple sources. Given issues of privacy, ethical uses of these tools should be strictly employed (Van Wel & Royakkers, 2004).

Despite the existence of a variety of tools, their ease-of-use and robustness can vary widely. There are many types of networks and online communities that could qualify as a subject of network-based research. Many of these virtual organizations and networks often share key ele-

ments and structures that are common across online social networks. These could include users, groups, communications, and relationship networks between these entities. Also, unlike the simple structured or semi-structured data that is subject of most data mining projects, SNA is not merely focused on generating lists of entities and information. Social networks are more organic in their growth and place emphasis on relational attributes. SNA seeks to understand how individuals and groups within networks (termed 'cliques') are connected together.

The Social Network Innovation Lab (SNIL) is an interdisciplinary research lab dedicated to understanding online social networks, social media, and cyberinfrastructure for virtual organizations. Research at the SNIL often involves the need for tools that are able to extract social network-based data for analysis from varied online social communities. The SNIL currently has projects that require data mining of popular microblogging services, shared interest forums and traditional social networks. We found that many currently available data mining tools were insufficient or poorly suited toward applications in social network research. This led us to begin to investigate the development of our own custom data mining tools. As part of this project, we researched existing tools, developed a conceptual framework for general data mining of online social networks, and tested prototype implementations of these ideas while acquiring data for use in current ongoing projects.

In this chapter, we will consider a variety of common methodologies and technologies for generic data mining and web content extraction. We will highlight a number of features and functionalities we see as key to effective data mining for social network analysis. We will then review several current data mining software tools and their fitness for data mining online social networks. The remainder of the chapter discusses our development of a data-mining framework for online social networks. Specifically, we introduce our work in extending the Web-Harvest 2.0 framework to data mine online social networks. This is followed by a case study of some of our initial results to acquire data from an online virtual community organized around social network technologies. The final sections summarize what we have learned through this process and maps out a course of action for future development in this area.

WEB-CONTENT EXTRACTION TECHNOLOGIES

With online social networking sites, the information and data that constitutes the network and its entities are, by necessity, distributed over a vast array of unique and dynamically generated page instances. Even when only a basic set of common SNS features (user profiles, friend lists, discussion boards) are considered, it is easy to see how the number of pages required to encounter and capture all the activity of the social network could quickly grow exponentially. In order to study the structure and operation of virtual communities within social networks, researchers need to parse and capture this sea of distributed data into formats more appropriate for research and analysis. In the absence of direct access to the database systems that drives a social network or a site-provided API, one must utilize other means to capture SNS data for research.

The majority of information on the Web is circulated in the form of HTML content, which wraps information in a nested set of tags that specify how data needs to be visually rendered in the browser. This is suitable for making data easily read and understood from the screen or through printing, but not as useful when clean organized machine-readable datasets are desired. Most online data extraction tools take advantage of the fact that the HTML is itself a structured data interchange format and leverage the HTML format to create parsers, which can extract the simple content and

structure of data on a page in an organized way while discarding the irrelevant material. Generally, most online data extraction technologies can be classified into several categories.

Formats, Conventions, Utilities, and Languages

Technologies in this class are low-level constructs that often derive from some sort of published standard grammar. This grammar may then be implemented in whole or in part by other higher-level programming languages and technologies. They often simply define a way in which data can be ordered, searched, manipulated, or transformed. For instance, the XPath standard defines a format for finding and isolating pieces of information from a structured XML document (Clark & DeRose, 1999). Similarly, regular expressions are a structured format, which are useful for performing advanced searches and manipulation on unstructured strings of characters. XSLT is a language defined to assist in the transformation of one type of structured XML document into another (e.g. transforming an HTML document into a simpler RSS feed or vice-versa) (Clark, 1999). In the absence of well-defined standards for interacting with various types of data, extraction becomes more difficult. However, because of the low level nature of these structures, they present challenges when used in isolation within advanced data extraction projects (without constructing a broader framework for their application to a set of data).

Libraries

Data extraction libraries often perform the job of wrapping one or more lower level data manipulation/extraction constructs into an organized framework within the context of a specific programming language. These libraries then manage the implementation of a given construct within a framework useful for further development within a programming language. Development libraries leave the end goals completely open to the developer. Depending on the time, investment, and goals of the developer, development libraries can be used to create anything from simple one-off scripts to high-level applications with many advanced features.

Application Programming Interface (API)

Web-based APIs provide a standardized abstraction layer for secure portal access to certain Web-based environments. APIs provide serialized access to information from online data ecosystems. APIs allow administrators to enforce restrictions and terms of use on data access. They also obscure and protect proprietary implementation details. Furthermore, they will generally apply and be structured around the content available from one data source (e.g. a particular website, web-enabled technology, or application). A prominent example is Google's OpenSocial API framework (Häsel, 2011). The OpenSocial API is an open standard for a set of API features specific to social networking. Code developed against such frameworks would be potentially adaptable to any social network using the open standard. Other examples include APIs provided by popular social networking sites including Twitter and Facebook.[1]

Applications

The majority of data extraction solutions take the form of applications. Applications often make use of a large set of lower level extraction technologies and development libraries and bundle them into an interface designed around a set of desired functionality. These differing function sets and the varying levels of expertise expected of the user results in the availability of a wide range of different types of data extraction applications. These

applications range from self-adapting, learning, fully GUI-based extractors for non-technical users to applications for advanced data extraction that require some in-depth knowledge of programming or data extraction utilities. Many applications fall into this latter category. Some of the most common are Helium Scraper, Djuggler, Newprosoft, Deixto, and Web Harvest.[2]

Enterprise Suites

This class of data extraction solutions is characterized by providing very high level, multi-featured, and advanced software solutions. They are often delivered as a suite of highly specialized applications. The implementations of these software packages are usually not open-source (as the code is often developed from proprietary development libraries). Like many enterprise solutions, these software products are often intricately complex and necessitate special training and/or ongoing technical support from the company itself to effectively use these tools. This support and training usually is an additional cost beyond the original software license. Pentaho[3] and QL2[4] are two examples of enterprise level data extraction and mining solutions that offer custom solutions, training, and support.

Outsourcing, Contracting, and Crowdsourcing

Alternative paradigms that merit mention are outsourcing, crowdsourcing, and freelance contracting. In outsourcing and freelance work, the researcher partners with a company or individual and explains the data extraction project and agrees upon a price and timeline for the delivery of the data. Crowdsourcing uses a slightly different model where a large data mining task might be divided into a large number of small tasks and a small fee may be offered for the delivery of each incremental piece of data delivered. Amazon's Mechanical Turk is a popular platform for streamlining this process and has been successfully deployed (Sheng, Provost, & Ipeirotis, 2008). Individual micro-tasks are constructed such as asking someone to record the tags on an online article or to classify a given webpage. Mechanical Turkers are often offered between 0.01¢ and 0.20¢ for the completion of each microtask. For some researchers, outsourcing works well because the tasks are cost-effectively completed in a short timeframe (Sheng, Provost, & Ipeirotis, 2008). For those who do not regularly need to acquire new data, this one-time fee structure may work well. Crowdsourcing may also be cost and resource effective. This method can bring additional concerns of uncertainty of when the task will be completed and, more importantly, quality control of data as contributors usually very highly in terms of quality of work (Snow, O'Connor, Jurafsky, & Ng, 2008). These approaches do not represent new technologies for data mining per se, but illustrate new solutions in the absence of tools and expertise for acquiring their own data sets.

CONSIDERATIONS FOR DATA MINING OF ONLINE SOCIAL NETWORKS

The utility of data mining applications for social network research is dependent on what functionalities are most appropriate to the domain. This section explores what functionalities are most valuable in social network research. This discussion is guided and informed by experiences and needs identified through our own research in the study of life science virtual communities of practice as well as work exploring health related communication networks on Twitter.

Input and Output

Data extraction is ultimately about acquiring formatted information from a data source and then translating, manipulating or filtering this information into other formats as appropriate to one's research objectives. In basic online content

extraction, the initial input is often a simple web address of the location of the information one seeks to capture. The distributed organization of most social networks means the information you need could be dispersed over a large number of pages. The URLs for these pages may need to be dynamically generated from one or more lists of attributes. Furthermore, one will most likely need to extract information from one location and use the results to identify and locate other pieces of information. This paradigm is best served by tools which allow for the most possible types of automated data extraction and manipulations. Data mining tools should be able to both read from and output to as many potential data formats as possible. Common formats most useful within a data mining tool kit include various kinds of structured text files like HTML/XML, delimited text, spreadsheets, or JSON. Data mining tools can be even more powerful when they include the ability to read and write to database systems, APIs, or even to have the ability execute local system commands to generate input variables. The power and usability of a tool increases as it is able to take input and give output in diverse formats.

Dynamic Query Specification

An important feature to consider when evaluating the utility of a data extraction tool to one's needs is to understand the ways in which the tool allows you to request and gather information. Many basic data mining applications use a GUI to allow one to specify the desired extractions. This will always have certain limitations. Other tools use a command-based query language like SQL to scrape data. In our work with social networks, we often need to traverse a list of forum locations, record all the user names encountered, collect information from each user's unique profile page, and then conditionally acquire extra information about certain users (if they are found in the previous step to have some specific attribute). Queries of this complexity often cannot be defined as a single request without resorting to multiple individual queries managed by a researcher. A query of this sort requires grammar for conditional branching, looping structures, and benefits from the ability to define functions (as well as local and global variables). Tools that implement full programming logic allow complex, dynamic, and context-aware queries to be defined. All the entities from a social network could conceivably be acquired via one request. This frees the researcher from having to micromanage many aspects of complex data mining projects, once the extraction rules and logic of the extraction job have been defined.

Social Network Interfacing

Online social networks often recognize the importance of allowing access to their data. Site developers often provide a programming interface for third parties developers to create value-added applications leveraging this social data (the use of which might further enhance participation in the network by its users). These application programming interfaces (APIs) often provide alternate methods for requesting information from a site beyond simply observing the information in situ. For example, Twitter and Facebook both have well-developed APIs to access vast stores of data. As opposed to simply requesting a page and extracting data from it, APIs allow developers to make a special kind of request to the API and return just the raw data one is looking for. APIs can greatly ease the process of gaining access the information on a social network. Some of the smaller, less well-known, networks we are studying include API-like features. Data mining tools for the analysis of social networks benefit greatly if they allow content extraction from common APIs.

Job Scheduling

In contrast to many common data mining tasks, social network researchers are more likely to be interested in near real-time data. Using the methods already described, one could create a data mining specification which would capture a

snapshot of the activity on a social network at the time the extraction was run, but, what about the next day or a month later? The constant use and modification of networks by their user base can cause networks to change greatly in short periods of time. One could simply capture a snapshot once a month, but these snapshots could contain large amounts of redundant information. A more robust solution might involve using programming logic coupled with automated time aware functionality to develop a data extraction request with the power to detect changes within the source network and incrementally update itself over time. In addition, a system of this type would likely require a perpetual extraction process which is able to detect and log changes periodically, and then call appropriate update scripts autonomously as needed. There are several key types of job scheduling attributes that could be associated with data extraction scripts:

- **Now:** This option would immediately execute a job. This is the most basic type of scheduling operation.
- **Later:** This option allows a user to schedule jobs at specific times. This could be useful to extract data from a site during low traffic hours, or for a situation where it is known that new information will be posted or made available at a specified times.
- **Chain:** The ability to chain tasks would allow one job to be scheduled to start once another has completed. This is very useful when one data extraction task is dependent upon the completion of one or more other tasks. With this option, the whole extraction flow for a complex and self updating extraction job could be specified in advance and sent to the mining application as a single project.
- **Recurring:** Recurring jobs are valuable in data mining of online social networks. Social network data presents challenges due to the fact that social networks are often in continuous flux. Most research of online social networks begins by capturing a snapshot of data as it exists at a specified point in time. A robust function for recurring data extraction allows researchers greater control over when to update their data and at what intervals to poll the site for new information.

Concurrency

Most web-content extraction tools acquire data by creating a virtual agent to make automated requests from a web host. This is the same way a web browser works. A browser requests a web site and the host sends a file containing information (i.e. in HTML) needed to display the web page. In data mining, this data is simply grabbed and parsed in a variety of methods to obtain data. Most tools and extraction workflow simply utilize one agent to fetch each required page in sequence. For non-dependent extraction tasks, this process could be streamlined by creating multiple agents to make multiple concurrent requests to the same web host (for different information). For large jobs, this feature places a key role in speeding up the acquisition of data, (as these requests are rarely taxing on local hardware or network speed). Concurrency should be evaluated as a key component of any data mining tool designed for online social networks.

Progress Management

Many of the features we have discussed focus on ways to allow for some level of automation in a data extraction task to be specified in advance (so the researcher is not required to micromanage the numerous aspects of large and complex extraction jobs). Usually, the analysis of social networks requires collecting large amounts of data from a network. In data mining tasks, data extraction is often limited by the speed that the hosting server allows clients to access data. Aside from concurrency, there is often little that can be done if the

social network you are mining is slow at returning requested information or is very large. If the network is both, it could take hours or perhaps days to complete certain large data extraction tasks. Our research often involved extracting information from numerous user profiles and forum pages. We noticed that although we could not predict when a job would complete, we could often determine the number of entities that needed to be captured before the process took place. This helped us realize that if we could also keep track of the number of completed entity extractions, we would be able to implement a progress monitor and estimate completion time for all our extraction tasks. We feel this type of progress tracking and management are important features for the data mining of online social networks. Wherever possible, extraction tools should attempt to keep track of the progress of data extraction tasks as well as expected time to completion. This feature will be of great value to researchers responsible for managing one or more large data extraction projects by giving them the information they need to maximize their own productivity and to be prepared in advance as to when data will be ready for post-hoc processing.

Extraction Meta-Behavior

A natural instinct in the design of data extraction tools for large social networks is to find ways to acquire the desired data as quickly as possible. Further consideration reveals an important counter argument to this instinct. The operator of an extraction process tool might be tempted to create large numbers of page request agents, which in turn might generate large amounts of traffic on the hosting site. This not only is considered bad 'netiquette' (Mozry, 2011), but has ethical and legal considerations. If one's data mining project is part of academic research, the relevant Institutional Review Board (IRB) should be consulted to confirm ethical compliance with human subjects. A large volume of page requests with a web host could degrade the quality of the experience of other users of the site. Also it could result in the web host banning all requests from your IP address if the host believes your requests are malicious. Furthermore, many social networking sites have specific policies or Terms of Service (TOS) in place that would dictate how much data can be requested per agent. Whenever possible, it is recommended that permission and guidelines be obtained from the administrators of any site one wishes to extract information from. It is in the best interest of the data extractor to be responsible and follow any appropriate rate-limiting conventions whether explicit, or implicit when extracting data from a host. Having one's IP banned from a site could be potentially catastrophic to a project. Any data mining toolkit for online social networks should implement some standard to protect the user, but also provide the ability to create custom guidelines depending on the known TOS of a web host or for when the user knows it is acceptable to request large volumes of data. These limits should be able to be defined in multiple ways by the number of requests per time unit or staying below some defined throughput or bandwidth measure. The idea is to be able to maximize program efficiency to acquire data as fast as possible, but not so fast that you may be garnering ill-will from a network and its users. In other words, data mining social networks should be - if possible - an open and collaborative process. Often, the social network being studied will also be interested in research results gained from the data that was mined.

Client-Server Paradigm

Extracting data from the web can require significant processing power as well as bandwidth. Many types of data extraction projects may be ongoing and most users do not want their computer constantly running potentially resource expensive scripts. We feel an ideal solution involves tools that use a client server paradigm (where each user

simply submits their jobs to a server for handling). That way, the designated server can handle all the heavy processing and high data load while the clients' machine remains free for use. With robust scheduling, concurrency, and progress management in place, the server application just needs to notify the client when the final collected data is available. The use of a client side application gives a lot of flexibility to the user requesting certain extraction jobs. They can use the client to log onto the main server and manage all their running jobs regardless of where they are physically located. The server should provide the client with options such as checking job progress, creating new jobs, aborting running jobs, changing scheduling, and changing the extraction specification. Also, this provides the ability for multiple users with different data extraction needs to utilize one centralized server.

REVIEW OF EXISTING DATA-MINING TOOLS TO MINE ONLINE SOCIAL NETWORKS

Our ultimate goal in this research was to begin development of a custom data mining solution optimized for extracting information from online social networks. Though we began the process with the aim of developing a custom "built-from-scratch" data extraction toolkit, we decided to first evaluate existing tools. As discussed in the previous section, we were interested in whether we could incorporate these existing tools into a hybrid data extraction toolkit which implemented our functionalities. In this section, we present this evaluation of common online data extraction tools. These reviews speak both to a tool's usefulness in social network research and in regard to their ability to be used as the basis for a prototype extraction tool. After the evaluation of several commonly available tools and technologies for online data extraction, we determined that Web-Harvest 2.0 was an ideal choice for the needs of our project goals. Among the tools considered were Helium Scraper, Newprosoft, Happy Harvester, Djuggler, Rapid Miner, Deixto, and Web-Harvest.

Common Data-Mining Tools

Based on our evaluation it was determined that Helium Scraper, Newprosoft, Happy Harvester, and Djuggler were all powerful GUI-based scraping applications. However, these tools also shared the same limitations. All four tools were single operating system applications that only allow scraper configurations to be defined within the context of the application. They also have no ability to be controlled or configured from the command-line. Their source code is not open-source and scripts could not be written against their various executables. When taken in consideration with our project goals (which would require software modifications for large social network scrapes to be conducted with minimal impact on the host), it became clear that these tools could not be leveraged to achieve our desired functionality.

Rapid Miner

Rapid Miner is one of the leading open-source applications for data mining and analytics and has been successfully used in data extraction projects (Graczyk, Lasota, & Trawiński, 2009; Han, Rodriguez, & Beheshti, 2008). Rapid Miner was evaluated as a potential fit for our project's needs. It is open-source, cross-platform, uses XML-based configuration files, which can be developed through the interface or written directly, and the code base can be scripted against both in application wrapper interfaces as well as from the command line. Though Rapid Miner is a powerful tool, it has a steep learning curve and includes a large number of features which would not be

needed for our project's needs. Using Rapid Miner would prevent us from being able to develop a fast lightweight utility in a reasonable amount of time.

DEiXTo

DEiXTo (also known as ΔEiXTo) (Kokkoras, Lampridou, Ntonas, & Vlahavas, 2008) is another web extraction technology that was evaluated for our project's needs. DEiXTo is a single platform GUI-based web extraction application built on top of an open source Perl-based scraper utility. DEiXTo also uses an XML based configuration language that potentially allows configurations to be defined outside of the GUI. The Perl module that forms the backbone of DEiXTo's extraction technology could also be scripted against any operating system or code framework that supports the Perl scripting language. The DEiXTo file format (.wpf) is obtuse and the documentation is not easily accessible. Ultimately, most .wpf files must be developed within the GUI application, which is single platform and is not open-source. DEiXTo is also limited in the ways output can be written only to specific file formats and in specific ways. While the features and options available in DEiXTo would allow us to accomplish our project goals, it was determined that tools which were more configurable and more open (in terms of input and output capabilities) would be better suited to our project.

Web-Harvest

Web-Harvest 2.0 is a Java-based open source data extraction tool, which has been successfully used in the data mining literature (Yang, Liu, Kizza, & Ege, 2009). It is a hybrid tool that consists of a GUI based application wrapped around an open-source Java development library. This library, in turn, implements several of the most common and powerful extraction utility formats such as XPath and regular expressions. The Web-Harvest 2.0 platform also defines syntax for defining custom data extraction workflows. This was ideal for several reasons. First, quick start-up of development was possible using the features of the graphical user interface to easily debug, learn, and understand how to develop complex workflows in the Web-Harvest scraper configuration format. In many ways, defining workflows via this format is better than coding library solutions that require workflows to be defined in the context of that codebase. This is because the configuration rules are written in a simple XML-based language, and can be written with any text editor (Nikic & Wajda, 2006). This frees the developer from the additional nuances of any high-level programming language. Furthermore, once tested, these workflows could be easily shared with others and passed to the development package, which could execute the scraper configurations through code. The fact that at the core of Web-Harvest is an open-source data extraction engine allowed for our project to wrap this engine in our own lightweight Java code. Web-Harvest 2.0 was a good fit because of its hybrid nature. Most pure application-based scrapers are not extendable and few define an open configuration format (forcing the developer to work within the confines of what the application allows). Pure development package-based based extraction tools can have a steep learning curve and are often difficult to debug. Relying purely on data extraction utilities and standards like XPath and regular expressions requires that an entire framework be built around them in order to execute complex extraction workflows. This can be time-consuming and resource intensive. Given the remit of our project, Web-Harvest served as an ideal solution in terms of features and the ability to separate between code and UI (which allowed us to quickly develop our own tools using the Web-Harvest engine). Other tools were not found to efficiently allow us to work in this way. See

Table 1. A comparison between data mining tools

	Availabilty	Expertise	Specification Language	Automation	Concurrency
Newprosoft	$, PC	Moderate	No	NA	NA
Helium Scraper	$$, PC	Easy	No	NA	NA
Djuggler	$,PC	Moderate	No	NA	NA
Rapid Miner	Free, Cross	Very High	No	Possible	Possible
Deixto	Free, Cross	High	Yes	Indirect	Possible
Web-Harvest	Free, Cross	High	Yes	No	Possible

Table 1 for a comparison between Web-Harvest and other data mining tools.

Extension of Web-Harvest for Data Mining of Online Social Networks

After a review of existing data mining tools and a consideration of the desired features of data mining for online social networks, we decided to develop extensions and an application wrapper for the open-source Web-Harvest 2.0 data extraction engine to serve as a prototype of a full-fledged social network-centric data extraction engine. As discussed previously, Web-Harvest features a robust query specification language with capabilities to import and export data to a number of important formats including MySQL database integration. We sought to add features important to data extraction for social networks including time-based and repeating jobs, running multiple jobs simultaneously, client-based process and progress management, and server-based execution. Our tool incorporates all of these features by building on top of the open-source Web-Harvest source code.

As discussed in the previous section, Web-Harvest has been used successfully in various studies as a basic scraper. One example is Nagel and Duval's (2010) work, which used Web-Harvest in order to collect large amounts of publication data. For their study, they required a simple web scraper and used Web-Harvest in its original form to mine publication data from Springer, an academic publisher. They used the software to collect data including titles, authors, affiliations, and postal addresses.

Katzdobler and Filho (2009) use Web-Harvest extensively. They combined Web-Harvest with JENA, a tool used to build semantic web applications, as well as an ontology which described what type of information they wanted to extract. The JENA API then accesses the ontology and Web-Harvest extracts the information from the site. However, manual creation of the configuration file and manual startup of Web-Harvest is needed.

TagCrawler is another program written using Web-Harvest and is one of the few cases of Web-Harvest being directly extended (Yin, 2007). The creators of TagCrawler required a web crawling tool which would be able to retrieve information from tagging communities. While the end goal of the project was not related to our project, their use of Web-Harvest as a base and building from it illustrates that this can be a successful model.

Our project, 'VoyeurServer,' uses Web-Harvest's existing functionality, but adds several layers of additional features. The features we identified in Section 3 served as a guide to our development efforts. By extending the Web-Harvest core framework, we were afforded the opportunity to discard the overhead of a GUI in favor of a lightweight command-line interface implemented with a client-server pattern. We were also freed from the limitations placed on the extraction engine by the GUI. We were also able to take advantage of the Java programming

language to develop our own features not present in the GUI or the engine itself.

Structure

The implementation of this project using the client-server paradigm required that we split the functionality for extraction into two applications: the VoyeurServer and the VoyeurClient. The latter was developed as a lightweight command line interface to provide users a way to submit and control their individual data extraction projects. It is through the VoyeurClient that users can submit their extraction jobs, manage the job's behavior, as well as monitor the progress of all their running jobs. The VoyeurServer application was designed to run continuously on a special server and to respond to requests from VoyeurClient instances. When the server receives a job from a client, the server creates an instance of the WebHarvest 2.0 extraction engine in its own thread, and manages the execution of the job as directed by users' interaction with the client. Figure 1 shows the relationships and flow of communication between these program entities.

Functionality

In addition to the ability to submit and manage multiple simultaneous extractions, we also sought to develop a set to features to allow for the smart management of extraction behavior and management of job progress. The ability to control and manage this behavior was incorporated into the VoyeurClient and VoyeurServer applications.

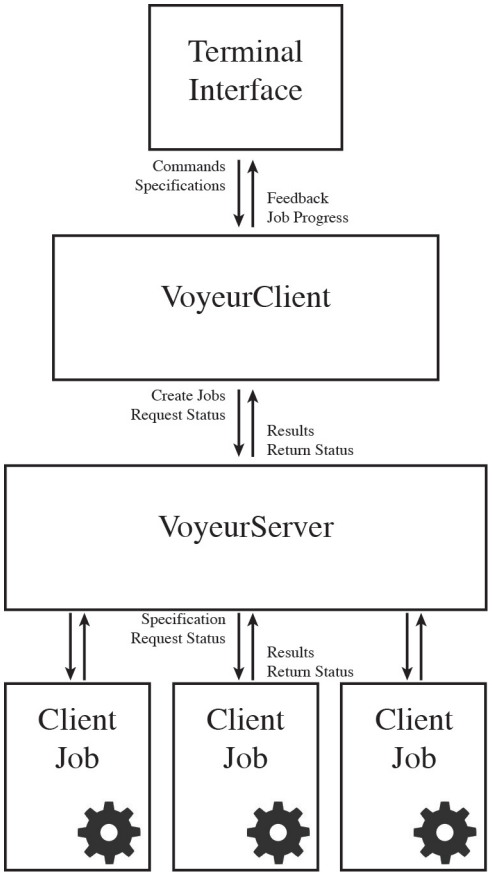
Temporal Control

The ability to control the temporal execution of job and recurrent behavior is one of the key features we identified for social network-aware data-extraction. This control allows users to create self-updating scripts as well as control how and when individual jobs will be run. Taking a cue from modern calendar systems, VoyeurServer was designed to allow jobs to be defined as a single event or as a repeating process to be executed in an ongoing manner.

Process Management

VoyeurServer users retain full control over their running and completed jobs. There is some danger that clients might feel they have lost control once their jobs were submitted to the server. In order to combat this, the VoyeurClient provides users with complete control over their running jobs at all times. This functionality includes the ability of users to start, stop, or pause as job at any point during its execution. Additionally users are given the ability to alter or remove a jobs repeating behavior at any time.

Figure 1. Schematic of data flow in Voyeur server



Progress Management

The ability to view and manage the progress of ongoing scrapes is an important feature for users managing large and complex data extraction jobs within online social networks. In order to add this functionality to the existing WebHarvest extraction framework, we developed an extraction specification template for progress aware content extraction jobs. This template adds two requirements to any extraction file. The first requirement is the initial definition of some measure of progress, which is stored in a special progress management database. The second involves having the extraction specification itself update this database regularly (based on its ongoing progress). The addition of these structures to any WebHarvest extraction specification file is sufficient to make the job progress-aware within the VoyeurServer framework. At the user's request, the VoyeurClient application will request and report this information to the user about their progress-aware jobs. Additionally, this feature will report on the start time, current running time, and any important log messages of any running job (regardless of whether it implements the progress-aware framework).

Summary

There a wide variety of valuable tools that can be immediately put to use or be modified to suit the needs of almost any data mining projects. In our case, we found that Web-Harvest 2.0 already incorporates many of the basic functionalities we had previously identified as important in the effective mining of online social networks. Because it is open source, we saw Web-Harvest as an ideal place to begin testing and developing a social network-centric data-mining tool. Our development plan centered on taking the core framework of Web-Harvest 2.0 and wrapping the code base to extend the application to serve as a multithreaded data extraction engine (implemented using a client-server interaction paradigm). Once the base extraction modules were wrapped in this way, we could focus on adding additional management features to the wrapper like task scheduling and process/progress management. Further work to develop and refine these ideas will be important to make this tool available to the broader data mining community.

EXPERIMENTAL RESULTS: A CASE STUDY

We engaged in this exploration of developing data extraction tools better suited towards collecting data for online social networks as part of a NSF funded research project on "Virtual Organization Breeding Environments." A key aspect of this project required us to gather large amounts of data from online life science communities of practice in order to explore the organization and cyberinfrastructure of virtual communities within these networks. As we collected this data, we realized there were ways in which the functionality of existing data extraction tools could be enhanced to streamline our extraction workflows.

The online network we studied had several key features we were interested in capturing. Specifically, we needed to collect over 200,000 user profiles, over 9,000 forums, a blogging ecosystem, and a friendship-based social network. This section details how we were able to use our tool to acquire the information we needed for our research.

Our needs included capturing information about the users within this community and their communications with one another. Our eventual goal is to use this data to study patterns of trust development and of online scientific collaboration. The community we studied did have an API for requesting information about each user's egocentric peer network. However, for all other data we had to rely on traditional web content extraction methodologies.

We were able to use the Web-Harvest query specification language to develop separate workflows to collect each type of information (user data

including profile information, forum posts, friend networks). Using the features developed in our VoyeurServer application, we were able to develop a broad automated workflow to simultaneously collect data from these three key areas (while at the same time respecting the bandwidth consumption limits agreed upon by communication with site administrators). Using the workflows and the VoyeurServer tool, we were able to successfully collect user and post information over the course of a week while limiting any daily micromanagement of the process. In this regard, our time aware processes, progress management features, and concurrent extraction jobs were able to prove themselves successful in application. The flexibility of Web-Harvest's I/O framework allowed us to capture data to structured XML as well as directly to a database simultaneously. In our research, we had previously developed an application to assist in the qualitative coding and classification of the community's data in this database. Our VoyeurServer content extraction workflows allowed the data to be integrated directly with our existing downstream research applications. This realized potential represents an extremely powerful and desirable workflow for network analysis. This experiment also helped identify issues that were not addressed by the current version of VoyeurServer. Some of these limitations and potential solutions are discussed in the next section.

FUTURE RESEARCH DIRECTIONS

Despite initial success in using Voyeur Server for mining data from online social networks, there remain further capabilities of Voyeur Server that require further development. We outlined key features for data mining of online social networks. Currently, the VoyeurServer extension of Web-Harvest implements these features at a basic level. Further testing and development would help determine whether this extension has a future as a general research tool or whether it suggests that extending Web-Harvest 2.0 is perhaps less preferable than starting from scratch to develop a data mining toolkit for online social networks. We consider further work in robust data-extraction tools for social networks to be of the utmost importance.

For those considering developing their own custom wrappers for the Web-Harvest 2.0 extraction engine, we suggest considering these suggestions of extended functionality:

- Utilize Web-Harvest's plugin architecture to develop integrated modules for common social network APIs to ease the process of developing extraction specifications for complex APIs (especially those requiring authenticated requests).
- Develop custom plugins for common database tasks. The VoyeurServer and WebHarvest database features rely on raw SQL statements and thus increases the level of expertise required to develop and implement database functionality.
- Develop a specification file for projects as opposed to per file scrapes. Incorporate time-aware functions, extraction meta-behavior, and progress monitoring options into this project specification format. This would allow for integrated individual workflows for large projects.
- Explore automated concurrency as opposed to having to design your individual jobs or projects for concurrency. This would help basic users take maximum advantage of parallel processing possibilities.

Our continued research seeks to address some of the issues and limitations discovered in developing this tool. We seek to further develop VoyeurServer in the following ways:

- Develop further functionality that allows for higher levels of concurrency.
- Investigate the feasibility of a broad-based public research server providing network-structured extraction as a service.

- Investigate high per-thread resources. Experience suggests that VoyeurServer is memory intensive. Our goal would be to reduce the memory overhead to a minimum for each running job. This will make running large numbers of jobs for various projects more efficient.
- Improve the interface for this Web-Harvest extension. Specifically, develop a GUI for the VoyeurServer client application.

We have shared these improvements so that developers can be aware of issues we currently face and some possible solutions. This will enable designers and developers to learn from the development challenges we have faced.

CONCLUSION

This chapter has reviewed various data mining tools for scraping data from online social networks. It has highlighted not only the complexities of scraping data from these sources (which include diverse data forms), but also introduces currently available tools and the ways in which we have sought to overcome these limitations through methodological extensions to existing software. After reviewing currently available data scraping tools, we developed a tool of our own, VoyeurServer, which builds upon the Web-Harvest framework. In this chapter, we outlined the challenges we faced and our specific solutions. We also included future directions of our data mining project. Ultimately, we introduce concrete methods to develop data mining solutions of online social networks using the Web-Harvest framework.

Though this research is preliminary and its remit has not been comprehensive, our experience in developing VoyeurServer tools has been positive and represents an important step towards the further development of this and/or other data mining tools specifically for online social networks. It is important to begin developing these domain specific solutions so that good open source options are available to researchers. Current tools tend to be focused around the domains of marketing and business knowledge. These types of solutions often fall short for use in academic contexts. As social media continues to become increasingly part of our online interactions, methods for data extraction from these networks will continue to remain critically important.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. 1025428.

REFERENCES

Bollen, J., & Mao, H. (2011). Twitter mood as a stock market predictor. *Computer*, *44*(10), 91–94. doi:10.1109/MC.2011.323.

Boyd, D. M., & Ellison, N. B. (2007). Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, *13*(1), 210–230. doi:10.1111/j.1083-6101.2007.00393.x.

Clark, J. (1999). XSL transformations (XSLT). *World Wide Web Consortium (W3C).* Retrieved from http://www. w3. org/TR/xslt

Clark, J., & DeRose, S. (1999). XML Path Language (XPath) version 1.0 w3c recommendation. *World Wide Web Consortium (W3C)*. Retrieved from http://www.w3.org/TR/1999/REC-xpath-19991116

Deixto. (n.d.). *Website*. Retrieved from http://deixto.com/

Djuggler. (n.d.). *Website*. Retrieved from http://www.djuggler.com/

Doan, S., Vo, B.-K., & Collier, N. (2011). An analysis of Twitter messages in the 2011 Tohoku earthquake. In P. Kostkova, M. Szomszor, & D. Fowler (Eds.), *Electronic healthcare: 4th International Conference, eHealth 2011* (pp. 58-66). Málaga, Spain, November 21-23, 2011. Berlin: Springer Berlin Heidelberg.

Golder, S. A., & Macy, M. W. (2011). Diurnal and seasonal mood vary with work, sleep, and daylength across diverse cultures. *Science*, *333*(6051), 1878–1881. doi:10.1126/science.1202775 PMID:21960633.

Graczyk, M., Lasota, T., & Trawiński, B. (2009). *Comparative analysis of premises valuation models using KEEL, RapidMiner, and WEKA* (pp. 800–812). Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems. doi:10.1007/978-3-642-04441-0_70.

Hammer, J., Garcia-Molina, H., Cho, J., Aranha, R., & Crespo, A. (1997). Extracting semistructured information from the Web. In *Proceedings of the Workshop on Management of Semistructured Data* (pp. 18-25). Tucson, AZ.

Han, J., Rodriguez, J. C., & Beheshti, M. (2008, December). Diabetes data analysis and prediction model discovery using RapidMiner. In *Future Generation Communication and Networking, 2008. FGCN'08. Second International Conference on* (Vol. 3, pp. 96-99). IEEE.

Häsel, M. (2011). Opensocial: An enabler for social applications on the web. *Communications of the ACM*, *54*(1), 139–144. doi:10.1145/1866739.1866765.

Heliumscraper. (n.d.). *Website*. Retrieved from http://www.heliumscraper.com/

Hughes, A. L., Palen, L., Sutton, J., Liu, S. B., & Vieweg, S. (2008, May). Site-seeing in disaster: An examination of on-line social convergence. In *Proceedings of the 5th International ISCRAM Conference*. Washington, DC.

Katzdobler, F.-J., & Filho, H. P. B. (2009). *Knowledge extraction from web*. Retrieved from http://subversion.assembla.com/svn/iskm/FinalDocumentation/FinalReport.pdf

Kokkoras, F., Lampridou, E., Ntonas, K., & Vlahavas, I. (2008). *Mopis: A multiple opinion summarizer. Artificial Intelligence: Theories* (pp. 110–122). Models and Applications.

Morzy, M. (2011). Internet forums: What knowledge can be mined. In Kumar, A. S. (Ed.), *Knowledge discovery practices and emerging applications of data mining: Trends and new domains* (pp. 315–335). IGI Global Publishing.

Murthy, D., & Longwell, S. A. (in press). Twitter and disasters: The uses of Twitter during the 2010 Pakistan floods. *Information Communication and Society*.

Nagel, T., & Duval, E. (2010, September). Muse: Visualizing the origins and connections of institutions based on co-authorship of publications. In *Proceedings of the 2nd International Workshop on Research 2.0. At the 5th European Conference on Technology Enhanced Learning: Sustaining TEL* (pp. 48-52).

Newprosoft. (n.d.). *Website*. Retrieved from http://www.newprosoft.com/

Nikic, V., & Wajda, A. (2006). *Web harvest: Overview*. Retrieved from http://web-harvest.sourceforge.net/overview.php

Pentaho. (n.d.). *Website*. Retrieved from http://www.pentaho.com

QL2. (n.d.). *Website*. Retrieved from http://www.ql2.com

Sheng, V. S., Provost, F., & Ipeirotis, P. G. (2008, August). Get another label? Improving data quality and data mining using multiple, noisy labelers. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 614-622). ACM.

Snow, R., O'Connor, B., Jurafsky, D., & Ng, A. Y. (2008, October). Cheap and fast---but is it good?: Evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 254-263). Association for Computational Linguistics.

Sourceforge. (n.d.). *Website*. Retrieved from http://web-harvest.sourceforge.net/

Twitter. (n.d.). *Website*. Retrieved from https://dev.twitter.com/, http://developers.facebook.com/

Van Wel, L., & Royakkers, L. (2004). Ethical issues in web data mining. *Ethics and Information Technology*, *6*(2), 129–140. doi:10.1023/B:ETIN.0000047476.05912.3d.

Yang, L., Liu, F., Kizza, J. M., & Ege, R. K. (2009, March). Discovering topics from dark websites. In Computational Intelligence in Cyber Security, 2009. CICS'09 (pp. 175-179). IEEE.

Yin, R. M. (2007). Tagcrawler: A Web crawler focused on data extraction from collaborative tagging communities. (Unpublished thesis). University of British Columbia, Canada.

KEY TERMS AND DEFINITIONS

Application Programming Interface (API): These interfaces are often defined within a given programming language or computer-based system to allow the system to communicate with other complex systems without revealing access to proprietary technology, private implementation details, or sensitive information.

eXtensible Markup (XML): A generic semi-structured format for the organization of data. The HTML format which defines how web pages should rendered in a browser is simply and extension of the more generic XML format, defined for a specific purpose. Extensions of the basic XML format are often developed and used to organize large amounts of repetitive structured metadata, like records of books in a library, or the organization of songs information (Album, Artist, Title, etc) within iTunes.

Graphical User Interface (GUI): A graphical user interface is an interface to a technological system that is presented to users through the use of visual widgets often displayed on a screen. These visual elements often make use of metaphors with real world objects to convey how they ought to be used to accomplish tasks in a virtual environment (i.e. virtual buttons, dials, etc.).

HyperText Markup Language (HTML): A standardized semi-structured format for the definition of content for a webpage on the World Wide Web. HTML files tell a browser all the information it needs to render content to the browser window. These files are often the subjects of data mining projects, which attempt to extract important information from the data directly and prior to rendering.

Social Network Analysis (SNA): Broadly, this entails the study of Social Networks and their component entities and transactions. SNA is often focused particularly on analytic and quantitative methods. Examples of SNA include the evaluating of social network maps using graph theory, utilizing machine learning techniques to attempt automatically classify different types of users, or identify sentiment of communications within a social network.

Structured Query Language (SQL): A generic language specification and format to request data from a data archiving system. There are many implementations of SQL, each of which is in use in different database systems. The most common of these is Microsoft SQL (MSSQL) and the widely used MySQL open-source database variant.

Terms of Service (TOS): A form of legal document that stipulates the terms under which a person may utilize a service. They often contain legally binding guidelines regarding what a user

of a service may or may not do, as well as what the guidelines for how the service provider will behave (including the transmission and storage of data).

ENDNOTES

1. See Twitter (n.d.).
2. See Heliumscraper (n.d.), Djuggler (n.d.), Newprosoft (n.d.), Deixto (n.d.), and Sourceforge (n.d.).
3. See Pentaho (n.d.).
4. See QL2 (n.d.).