Ontology-guided Social Media Analysis
System Architecture

Alexander Semenov[1,2] and Jari Veijalainen[1]
[1]*Dept. of Computer Science and Information Systems, University of Jyväskylä, Mattilanniemi 2, Jyväskylä, Finland*
[2] *NRU ITMO, St. Petersburg, Russia*

Keywords: Social Media Monitoring, Ontology Development.

Abstract: Social media sites have appeared to the cyber space during the last 5-7 years and have attracted hundreds of millions of users. The sites are often viewed as instances of Web 2.0 technologies and support easy uploading and downloading of user generated contents. This content contains valuable real time information about the state of affairs in various parts of the world that is often public or at least semipublic. Many governments, businesses, and individuals are interested in this information for various reasons. In this paper we describe how ontologies can be used in constructing monitoring software that would extract useful information from social media sites and store it over time for further analysis. Ontologies can be used at least in two roles in this context. First, the crawler accessing a site must know the "native ontology" of the site in order to be able to parse the pages returned by the site in question, extract the relevant information (such as friends of a user) and store it into the persistent generic (graph) model instance at the monitoring site. Second, ontologies can be used in data analysis to capture and filter the collected data to find information and phenomena of interest. This includes influence analysis, grouping of users etc. In this paper we mainly discuss the construction of the ontology-guided crawler.

1 INTRODUCTION

Social media sites have appeared to the cyber space during the last 5-7 years and have attracted hundreds of millions of users. For instance, Facebook has currently over 901 million users ("Facebook Newsroom," 2012) and YouTube also hundreds of millions. The sites are often viewed as instances of Web 2.0 technologies and support easy uploading and downloading of user generated contents, such as text images, and videos, as well as links to web contents. This content contains valuable real time information about the state of affairs in various parts of the world that is often public or at least semipublic. By public information we mean in this context information that anybody can access using a browser. If authentication is needed at such a site, passing it should grant access to all the contents of the site. YouTube and Twitter fall into these categories. By semipublic information we mean information that is shared inside a certain group, i.e. inside a virtual community, that can be (potentially) joined by anybody. Authentication of the users is necessary in this case. For instance, Facebook authenticates the users, before they can access the contents. After that one can access information that is declared public by any user, information produced in groups the user belongs to, and information released for the "friends" or "friends of friends" etc. of the user. Twitter also requires authentication. It also offers, however, an API through which one can get the tweets with or without authentication. The former has fewer restrictions.

Many governments, businesses, and individuals are interested in this kind information for various reasons. In this paper we describe how ontologies can be used in constructing monitoring software that would extract useful information from social media sites and store it over time for further analysis. Ontologies can be used at least in two roles in this context. First, the crawler accessing the sites must know the "native ontology" of the site in order to be able to parse the pages returned by the site in question, extract the relevant information (such as friends of a user) and store it into the persistent generic (graph) model instance at the monitoring site. Second, ontologies can be used in data analysis to capture and filter the collected data to find

Semenov A. and Veijalainen J..
Ontology-guided Social Media Analysis - System Architecture.
DOI: 10.5220/0004157303350341
In *Proceedings of the 14th International Conference on Enterprise Information Systems* (SCOE-2012), pages 335-341
ISBN: 978-989-8565-11-2
Copyright © 2012 SCITEPRESS (Science and Technology Publications, Lda.)

information and phenomena of interest. This includes influence analysis, grouping of users, etc.

2 SYSTEM ARCHITECTURE

2.1 Overall System Architecture

Requirements and the general architecture of our monitoring system were presented in (Semenov et al., 2011). It consists of four major modules, crawler, repository, analyser, and control. Crawler is the component accessing the social media site, repository stores the data gathered, and analyser analyses the data accessing the repository. The activity can be continuous and the data can be collected and analysed in parallel from different social media sites.

2.2 3-Level Modelling Framework

Semenov and Veijalainen (2012) analyzed the social media monitoring environment from the modelling point of view. One can argue that monitoring social media sites involves three modelling levels. The original Universe of Discourse (UoD) is the immediate social reality of the people, i.e. their entire life. The social media sites capture some aspects of this social reality and store and communicate them as digitally encoded information. What is captured is determined by the *site ontology* of a particular site. The site ontologes are different at different social media sites and capture some aspects of immediate social life. They have their own ontologies that are populated by instances from peoples' lives and the contents they produce. For instance, Facebook has "profiles" and it makes visible "friends" of a person by recording this relationship between profiles. It also offers "groups" that are explicit representations of virtual communities. "Family" is also offered as a special subcategory of "friends". Recently, Facebook began to offer "time line" that contains the history of "status updates" along inserted "pictures", and various "life events" attached to the "profile" in chronological order.

The second level model is needed by the monitoring site. It should be able to capture a number of first level models at various social media sites and it is the core model of the monitoring site. In (Semenov and Veijalainen, 2012) we argued that such a model is a special temporal graph. All the models implemented by the various social media sites should be subsumed by this "pivot" model. Based on it, the data retrieved from different sites can be stored into homogeneous structures. Relying on it, one can also use the same algorithms to analyze the data from different sites.

The third modeling level is then the data base level, because we want to store the retrieved data persistently. DBMSs have their own data models and we must represent the above graph using the database modeling facilities. We have designed a relational database scheme for the second level model and implemented it into a PostgreSQL installation. We could also install the schema in a special graph database and store the information there. In order to hide the differences between different implementations of the graph above at DBMS level there must be an abstraction layer on top of the DBMS interface. For this reason we call the persistent storage repository.

2.3 Ontologies

As alluded to above, the sites model social reality relying on certain ontologies. In order to capture correctly the information, the monitoring site must "understand" the ontology of the site – or at least that portion that is relevant in the second level model. Social media sites store its data internally and usually do not display implementation level descriptions explicitly. However, unlike usual web-sites where users are able to store any data (mostly, semistructured HTML data), social media sites introduce certain concepts varying from site to site. These are implemented by fields with titles (in Facebook e.g. favorites, applications, friends, groups, status, etc) and buttons. Using buttons and fields users can enter new data that correspond to the site ontology. Data can be comments, plain text, images or other multimedia data, or semistructured HTML entities (e.g. blog posts). Sites allow interfaces for the modification of the data only within the conceptual structure. Thus, users may for instance send text messages using predefined form or update the fields of the profile, add comments or links, etc.

These data are stored internally at social media site, and later made accessible through a web-interface, in the form of HTML pages.. Usually the web pages are enriched using such technologies as CSS, JavaScript/AJAX, Adobe Flash, etc. Sometimes social media sites have several different versions of web-interfaces, offering e.g. a special simplified interface for mobile devices.

Another way to access a social media site is a special API. An API is meant to be used by software

agents extracting information from the site, but they can also serve e.g. mobile clients.

Explicit site ontology may be constructed based on the analysis of the concepts it uses in its user interface, such as “friend”, ”group”, “status update”, “following”, ”followers”,”(re)tweet”, etc, or in its API description. Theoretically, formal conceptualizations of the entities existing within a site might be constructed in a number of different ways, and in any case the site ontology must be constructed by a human being. It is also important to notice, that the representation or encoding of the ontology on the social media site might change without modification of the ontology itself: page structure might change visually, or HTML mark-up may change even without the modification of the visual results. Also, as noted in (Semenov and Veijalainen, 2012) social media ontology is being modified over time: concepts and relations might be added or deprecated. This necessarily means that the web page will change in one way or the other. A recent example is introduction of “timeline” to Facebook.

Reflecting the above to the monitoring software, the data extraction part of it should contain the target site ontologies themselves, and procedures which would populate the ontologies at the montoring site. After the above modelling task the crawling of the social media site can begin.

Basic polling-based crawling needs some seed nodes to start. These are typically public user profiles or similar concepts in the site ontology that the crawler accesses. Fresh nodes, subject for the further traversal should be discovered from the retrieved data using inference rules. Thus, crawling would be carried out on a “semantic” level, which would offer certain advantages for the crawling of the social media sites. Since HTML pages of the social media sites are usually generated dynamically, thus emanating from the “deep web”, traditional traversal of the web-sites using hyperlinks for discovery of new nodes or relationships might not work, since many links point to the pages, which do not store relevant information (e.g. logout page). Also, the same instances of the same concepts could be represented on syntactically different HTML pages having different URLs or, in the case of using AJAX at the browser, without modification of the URL at all. At many sites, however, the user has permanent identifier that can be used to relate him or her and the further data.

Another functionality is the wrapper that must relate the concrete data on the retrieved page to the site ontology level concepts. E.g. if the “friend” relationship is used to span the second level graph, where on the fetched page are the friends of person X. Fresh nodes for crawling need to be discovered based on processing tuned to the site ontology and page structure. Presence of the explicit site ontology would allow its selective population; for some crawling tasks all the data from the social media site may not be necessary. In addition, ontology would allow specification of stopping conditions in terms of the used ontology. E.g. depth of the crawling.

In case of the crawling of the subset of the ontology of the site (when we do not need to monitor all the concepts), *crawling ontology* should accordingly correspond to the necessary subset.

Another usage of the ontology in the crawler is marking up the already stored data. Crawler may crawl the social media sites according to certain tasks which could differ from one another by time of the crawl, the ontology used, keywords or some other parameters used. In general, ontology should mark some groups of interest and possibly mark elements within those groups, e.g. most influential entity, or different marking of members of different communities. Thus, we need two kinds of ontologies: crawling ontology, which is used during the crawl for extraction of entities from a social media site, and analysis ontology, which describes higher level concepts and is used when the contents of the site is analysed.

2.3.1 The Crawling Ontology

As stated above, each social media site is characterized by its site ontology, which represents concepts and relations, occurring at the site. Since social media sites usually contain a number of different entities, it might not be necessary to collect values for all entities, particularly when not needed for the current analysis. Some entities might have a large size (in bytes) and collecting them would consume unnecessary network capacity space on the monitoring site, and decrease the speed of the crawling (e.g. downloading videos might not be necessary in some cases). Taking into account that all the entities of a certain site are described within its ontology and share common representation template, task of selective crawling – extraction of the entities of certain types only, thus skipping extraction and download process of other types of entities is highly important. In general, for this purpose is necessary to construct an ontology that is subset of the ontology of the social media site, and only contain concepts whose instances should be extracted during the current crawling task. Additionally, as stated above, it is necessary to

especially single out some relations, and use objects of these relations (we assume paradigm "subject predicate object") as elements which should be crawled further.

Elements of the ontology for the current crawling could be selected by a number of means, e.g. manually, which would greatly simplify descriptions of the tasks for the crawling and could be easily used by the user without special computer science background. For example, selection of ontology concepts could be done using mouse selection of some subset of entities of entire site ontology. Attributes of the concepts whose values should be retrieved should also be selected. Later this ontology (with attributes) should be transferred to internal crawler storage and used during the data extraction process.

The following problems might appear during the implementation of the idea described above. The first one is the selection of a subset of entire site ontology and its transfer to the crawler internal storage. The second one is picking up relations used for a further traversal. The following approach is used in the current crawler to select a subontology of the main ontology of a site. After the selection of the concepts and their attributes, instances of the following classes are created, and temporal values are assigned to their data attributes. These objects represent "crawling schema", i.e. a subset of the site ontology that describes the elements which should be crawled. Inter-node references are transferred to the crawler separately. In the crawler, ontology describing crawling schema is reconstructed (using SPARQL queries), and is used during the data extraction process. Nodes for further traversal are selected from discovered nodes using SPARQL queries.

Figure 1 depicts an example of the ontology of a social media site, consisting of three concepts: Profile, Post, and Community, where Profile can be friend of another Profile (relation "isFriendOf"), Post is written by Profile (relation "isWrittenBy"), and Profile can be a member of Community ("relation memberOf"). The following data attributes are present; name, bio, and interest for Profile (text entities, connected by relations "hasName", "hasBio", and "hasInterest" respectively), text for the Post ("hasText" relation), and community name and details for the Community ("hasCommName", and "hasDetails", respectively).

If the task is to only collect user profiles, their names, and isFriendOf relations among them, sub-ontology (see Fig. 2) should be selected.

A subontology describes profile node, its relation "isFriendOf", and "hasName" data relation. As relation used for further crawling in this case "isFriendOf" should be selected by the user.

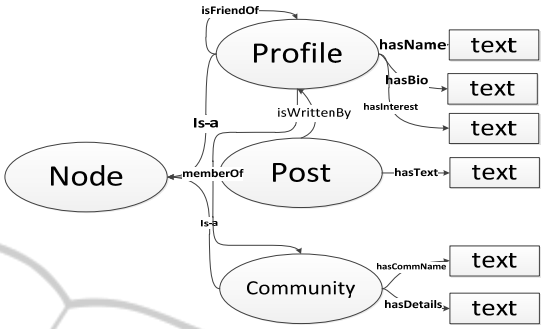

Figure 1: Example ontology.

Figure 2: Subontology.

2.3.2 Analysis Ontology

The analysis ontology contains characteristics of the phenomena of interests. Some interest in social media are e.g. memberships of the profiles in groups, relations between them and information which can be extracted from this data using various social networking analysis algorithms. A typical question to be answered would be "How many friends does a person have at this site on average?" Or "who is the most influential person in this particular group?" These are questions that can be answered based on the information at the site. In this respect the information at the site must be contain the answers, but the ontology needed to grasp this kind of questions is different from the site ontology. On the other hand, it is necessarily related with the site ontology and the crawling ontology chosen must contain enough information in order to make such analysis tasks possible. We do not handle this ontology further in this article.

3 DETAILED DESIGN OF THE ONTOLOGY-BASED CRAWLER

In this section we present a detailed architecture of ontology based web crawler. Crawler is implemented using Python 2.6 programming

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

language using Twisted library ("Twisted," 2012). Crawler consists of the following modules (see Fig. 3):

Database: PostgreSQL database which stores the crawled data. Since Database is interacting with DB Wrap module only, it can be replaced to another DBMS system, for instance Graph database Neo4j.

DB Wrap: interface for the database. The purpose of the module is to allow using of different DBMS, such as PostgreSQL, or Neo4j. Interface implements the access to the database. In the case of PostgreSQL it would use Psycopg2 library. The module exports functions which allow selection of the stored entities from the DB and insertion of the crawled entities. Together, DBWrap and DB can be considered as a Repository that was presented in (Semenov et al., 2011).

Insert Cache: stores the list of the elements that are subject to be inserted to database.

Read Cache: stores the queue of the elements to be traversed

Wrapper: Contains procedures of data extraction, specified for the current site ontology. Uses XPath and regular expressions.

Query translator: module that translates semantic entity description to request which should be handled by the HTTP connection module (either request to web-page or to API), dependent on description from the Ontology

HTTP: connection module towards WWW sites in Internet.

Ontology: ontology, having description of current crawling task, along the specification of relations, which should be considered as sources for new entities for traversal

Scheduler: scheduler of the crawler. Contains control modules for the Twisted library.

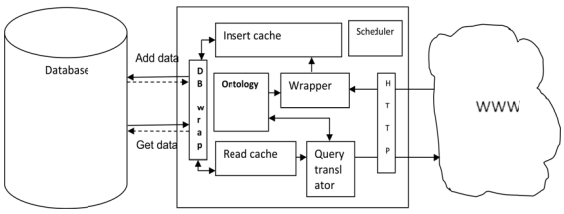

Figure 3: Technical diagram of the crawler.

The crawler was developed with the assumption, that social media site subject to be crawled can be modelled with multirelational directed graph, where nodes of the graph correspond to various entities from the social media site, and edges are relations between them. Currently, one minimal instance of multirelational directed graph is represented by "crawling ontology" described above. Before starting the crawl seed nodes should be added to the database (at start they would be extracted and put to frontier in the read cache).

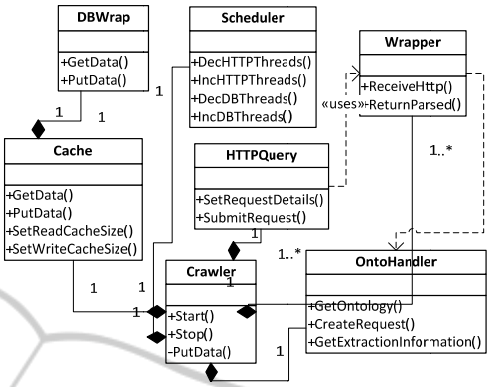

Figure 4: UML Class Diagram of the Crawler.

When the crawler is started, it takes a predefined number of seed nodes from the queue, constructs the HTTP request to extract the data for this entity using HTTPQuery class and information from the OntoHandler class, which would be sent to HTTPQuery class object (crawler may contain a number of HTTPQuery class objects, since it is working asynchronously). When a HTTP request completes, it generates the event (via firing the deferred in Twisted) and call Wrapper class object method ReceiveHttp(), which would populate the fields of the current crawling ontology and send it to the database (through Cache and DBWrap).

DBWrap is currently implemented for PostgreSQL DBMS and also uses asynchronous calls, since DB connections which send or retrieve large number of the elements to/from the database can block the main process. Thus, the main loop ("reactor" in Twisted) is waiting for the events from the network and sends received data to be processed, and simultaneously waits and processes the DBWrap events. Loop, processing the events can be modelled as the state machine: crawler is in the wating state, and it invokes procedures ReceiveHttp when event HTTP_EVENT occurs, and ReceiveDB when DB_EVENT occurs.

The scheduler class provides methods for adjusting the degree of parallelism, i.e. decrementing and incrementing the number of HTTP requests or DB requests being performed simultaneously.

The Wrapper class parses the input data and instantiates the ontology. Before that, a functioning wrapper class should be linked with parsing procedures that would extract the data for the current ontology entities from the retrieved page. In case when information needed for ontology population is

spread through several web-pages. Then the wrapper creates additional HTTP requests and waits for their completion.

4 RELATED WORK

A considerable number of articles related to crawlers' development were compiled both within academia and industry. There are a number of proprietary crawlers, used in search engines, such as Google, or Yahoo. Article (Liu, Liu, and Menczer, 2011) defines crawlers (also named spiders or robots) as programs which automatically download web-page. Crawlers are classified on universal crawlers which download every page, and topical (focused) crawlers which download pages related to certain topic only.

One of the open-source crawling frameworks, Scrapy ("Scrapy," 2012) is also built using Twisted framework ("Twisted," 2012). Scrapy allows for generation of multiple spiders within the framework, which could crawl different web-sites according to specified templates and allows creation of both focused and universal crawlers. Paper (Noordhuis et al., 2010) describes architecture of Twitter crawler, deployed on Amazon cloud platform. Book (Russell, 2011) describes a number of methods for extracting data from popular social media sites such as Twitter, and LinkedIn.

A considerable amount of research is devoted to the description of the architectures of focused crawlers, built using ontologies (Fang et al., 2007; Dong et al., 2009; Liu, Liu, and Dang, 2011).

An important component of the web-crawlers is the wrapper. It extracts structured data from semistructured web-pages. (Liu and Liu, 2011) classifies wrappers into three categories: manual data extraction, where software developer writes procedures for data extraction him/herself, wrapper induction, where rules for the data extraction are learned from set of manually marked up web-pages semi-automatically, and automatic data extraction where patterns are found automatically. There are a number of papers devoted to wrapper generation procedures, (Zhao et al., 2005). Also, there is a research which aims at development of methods for automatic classification of web-pages using machine learning (Zhang and Nasraoui, 2009).

However, it is frequently assumed that wrappers are generated for the pages which contain static HTML that can be decomposed into DOM tree. Nowadays a lot of pages employ dynamically generated content, using AJAX and/or Javascript, where important pieces of data may be encoded with JavaScript functions, or are displayed on the page after performing AJAX calls to the server-side procedures. Google publishes special recommendations for the owners of AJAX-enabled web pages ("Google," 2012). Approaches for creation of AJAX crawlers are described in (Mesbah et al., 2008; Duda et al., 2009). There are open-source interpreters of JavaScript and DOM handlers, such as Google v8 (Wikipedia contributors, 2012a), or Spidermonkey (Wikipedia contributors, 2012b). There is research devoted to construction of hidden web crawlers: (Madhavan et al., 2008). Also there is research devoted to crawling public domain RDF ontologies.

A substantial amount of research is devoted to structured knowledge, ontologies, and semantic web (Staab and Studer, 2011). Ontology is defined as a formal specification of a shared conceptualization (Grudin, 1994; Borst, 1997). There are papers devoted to methodologies of ontology construction, verification, and validation of ontologies. There are several data definition languages for describing the ontologies, e.g. RDF and OWL.

Currently there is large number of already existing ontologies such as FOAF: Friend of a Friend ("FOAF," 2012), and SIOC ("sioc-project.org," 2012): Semantically interlinked online communities. However, they do not specify detailed scheme of social media site ontologies but focus on higher levels. In addition, there exist ontologies for multimedia data (Staab and Studer, 2011).

There is also research devoted to storing the ontologies persistently (Staab and Studer, 2011) and querying them.

5 CONCLUSIONS

The present paper describes the architecture of an ontology driven crawler: software for extraction of the information from social media sites, whose structures are described by site ontologies at the monitoring site. We also discuss how the site ontologies are used to derive crawling ontologies and how these guide the crawler in its activities. The wrapper, a module inside the crawler, must be designed in such a way that it can extract the instances of the crawling ontology concepts from the web pages retrieved. In addition, possibilities of analysing the social media sites using ontologies are shortly discussed. The current crawler is a part of the social media monitoring system, originally discussed in (Semenov et al., 2011).

The future research concentrates on enhancing the repository and analysis part. One tricky issue is still how do deal with JavaScript and AJAX encoded pages inside the crawler. There must be an interpreter that runs the scripts and produces the page contents as if the crawler would be a browser. A further task is to analyse real sites. We have already crawled the entire contents of LiveJournal and made some initial analysis of it.

REFERENCES

Borst, W. N., 1997. Construction of Engineering Ontologies for Knowledge Sharing and Reuse.

Dong, H., Hussain, F., Chang, E., 2009. State of the Art in Semantic Focused Crawlers, in: Gervasi, O., Taniar, D., Murgante, B., Laganà, A., Mun, Y., Gavrilova, M. (Eds.), *Computational Science and Its Applications – ICCSA 2009, Lecture Notes in Computer Science. Springer Berlin / Heidelberg,* pp. 910–924.

Duda, C., Frey, G., Kossmann, D., Matter, R., Zhou, C., 2009. AJAX Crawl: Making AJAX Applications Searchable, in: *Proceedings of the 2009 IEEE International Conference on Data Engineering, ICDE '09*. IEEE Computer Society, Washington, DC, USA, pp. 78–89.

Facebook Newsroom [WWW Document], 2012. URL http://newsroom.fb.com/content/default.aspx?NewsAreaId=22

Fang, W., Cui, Z., Zhao, P., 2007. Ontology-Based Focused Crawling of Deep Web Sources, in: Zhang, Z., Siekmann, J. (Eds.), *Knowledge Science, Engineering and Management, Lecture Notes in Computer Science. Springer Berlin / Heidelberg*, pp. 514–519.

Google. Getting Started - Webmasters — Google Developers [WWW Document], 2012. URL https://developers.google.com/webmasters/ajax-crawling/docs/getting-started

Grudin, J., 1994. Computer-Supported Cooperative Work: History and Focus. Computer 27, 19–26.

Liu, B., Liu, B., 2011. Structured Data Extraction: Wrapper Generation, in: Web Data Mining, *Data-Centric Systems and Applications. Springer Berlin* Heidelberg, pp. 363–423.

Liu, B., Liu, B., Menczer, F., 2011. Web Crawling, in: Web Data Mining, *Data-Centric Systems and Applications. Springer Berlin Heidelberg,* pp. 311–362.

Liu, G., Liu, K., Dang, Y., 2011. Research on discovering Deep Web entries based on topic crawling and ontology, in: *Electrical and Control Engineering (ICECE), 2011 International Conference* On. pp. 2488 –2490.

Madhavan, J., Ko, D., Kot, Ł., Ganapathy, V., Rasmussen, A., Halevy, A., 2008. *Google's Deep Web crawl. Proc. VLDB Endow.* 1, 1241–1252.

Mesbah, A., Bozdag, E., Deursen, A. van, 2008. Crawling AJAX by Inferring User Interface State Changes, in: Proceedings of the 2008 *Eighth International Conference on Web Engineering, ICWE'08. IEEE Computer Society, Washington, DC,* USA, pp. 122–134.

Noordhuis, P., Heijkoop, M., Lazovik, A., 2010. Mining Twitter in the Cloud: A Case Study, in: Cloud Computing, *IEEE International Conference On. IEEE Computer Society, Los Alamitos, CA, USA*, pp. 107–114.

Russell, M., 2011. Mining the Social Web: Analyzing Data from Facebook, Twitter, *LinkedIn, and Other Social Media Sites. O'Reilly Media*, Inc.

Scrapy | An open source web scraping framework for Python [WWW Document], 2012. URL http://scrapy.org/

Semenov, A., Veijalainen, J., 2012. A modeling framework for social media monitoring. submitted to IJWET,

Semenov, A., Veijalainen, J., Boukhanovsky, A., 2011. A Generic Architecture for a Social Network Monitoring and Analysis System. IEEE, pp. 178–185.

SIOC, sioc-project.org | Semantically-Interlinked Online Communities [WWW Document], 2012. . URL http://sioc-project.org/

Staab, S., Studer, D.R. (Eds.), 2011. Handbook on Ontologies, International Handbooks on Information Systems.

FOAF, The Friend of a Friend (FOAF) project | FOAF project [WWW Document], 2012. URL http://www.foaf-project.org/

Twisted [WWW Document], 2012. URL http://twistedmatrix.com/trac/

Wikipedia contributors, 2012a. V8 (JavaScript engine). Wikipedia, the free encyclopedia.

Wikipedia contributors, 2012b. SpiderMonkey (JavaScript engine). Wikipedia, the free encyclopedia.

Zhang, Z., Nasraoui, O., 2009. Profile-based focused crawling for social media-sharing websites. J. Image Video Process. 2009, 2:1–2:13.

Zhao, H., Meng, W., Wu, Z., Raghavan, V., Yu, C., 2005. Fully automatic wrapper generation for search engines, in: *Proceedings of the 14th International Conference on World Wide Web, WWW'05*. ACM, New York, NY, USA, pp. 66–75.