Data Acquisition in Social Networks: Issues and Proposals

Claudia Canali, Michele Colajanni, Riccardo Lancellotti

Department of Information Engineering
University of Modena and Reggio Emilia
{claudia.canali, michele.colajanni, riccardo.lancellotti}@unimore.it

Abstract. The amount of information that is possible to gather from social networks may be useful to different contexts ranging from marketing to intelligence. In this paper, we describe the three main techniques for data acquisition in social networks, the conditions under which they can be applied, and the open problems. We then focus on the main issues that crawlers have to address for getting data from social networks, and we propose a novel solution that exploits the cloud computing paradigm for crawling. The proposed crawler is modular by design and relies on a large number of distributed nodes and on the MapReduce framework to speedup the data collection process from large social networks.

1 Introduction

An unprecedented explosion of user generated content is available on social networks that, thanks to their growing popularity, are gaining top importance as sources of valuable information. Two thirds of the world's Internet population visit a social network site weekly, and the time spent on these sites accounts for more than 10% of all Internet time, with this percentage growing three times faster than the rate of the overall Internet growth [NielsenWire (2009); Canali et al. (2009)].

The growth in terms of registered users motivates the increasing interest of academic and industrial researchers in social networks for different goals including workload characterization, marketing purposes, understanding and forecasting Internet use, identifying main challenges to support future Internet-based services. Unfortunately, collecting data from social networks is a real challenge with respect to other Internet-based sources, such as Web pages, blogs, peer-to-peer systems. Some problems are related to the size and the intrinsic complexity of a social network. Other problems are related to the heterogeneity of the target. Indeed, we have to consider that the term social network identifies a broad category of applications that may differ in many ways, ranging from user communication styles, types of exchanged contents, privacy settings, etc. Hence, the one-size-fits-all approach to data acquisition is unfeasible. The state of the art of academic research includes three main techniques for data acquisition from social networks: network traffic sniffing [Gill et al. (2007); Nazir et al. (2009)]; implementation of specific applications for each social network [Nazir et al. (2008)]; crawling of the user social graph. The last method is the most popular approach for social networks where user data are publicly available, such as MySpace. Traffic sniffing and specific applications are limited to specific contexts where crawling is unfeasible. For example, traffic analysis of dormitory networks was adopted to characterize the access patterns of college students in Facebook [Nazir et al. (2009)]. Throughout this paper we initially present the three main techniques adopted for data acquisition in social networks and we identify the main

open issues of each of them. We then focus on *social network crawling*, and identify the limits of current crawlers through analysis and experimental results. This study induced us to propose an innovative solution that exploits the potential of cloud computing to support data collection in the context of social networks. The novel framework is based on a crawling algorithm that exploits the MapReduce computing paradigm [Dean and Ghemawat (2008)]. This proposal has several advantages: it allows us to speed up the data collection process by distributing the operations on a large number of nodes performing parallel crawling; it reduces the risk of triggering the countermeasures adopted by social network operators against extensive and automatic crawling; it favors the implementation of a modular and easily customizable crawler that is able to acquire data from different social networks.

The remainder of this paper is structured as follows. Section 2 describes the main issues affecting data acquisition from social networks with a special focus on crawling. Section 3 presents the proposed architecture for crawling social networks based on the cloud computing paradigm. Section 4 discusses the related work. Section 5 concludes the paper with some final remarks.

2 Data acquisition in social networks

The social network term includes a broad category of applications. In this paper, we define *social network* any Web-based site that offers the user the possibility to register, to interact with other users, to share contents of any nature through any sort of social links. Similar sites include the most traditional Facebook, LinkedIn, MySpace, and Orkut, but also the sites that have added a lot of social network facilities, such as YouTube, Flickr, Digg, and Twitter.

Acquiring data from a social network requires an exploration of the user population with the goal of collecting different kinds of information, such as the network links among the users, uploaded and downloaded contents, rating, comments [Khrishanmurthy (2009)]. In this paper we are interested to the data collection process only, hence we do not delve into details of which data can be acquired from a social network and which analyses can be carried out on these data. In a similar way, we do not consider typical data anonymization techniques that typically are carried out only after the data collection process.

Here, we outline the three main techniques proposed in literature to acquire data from social networks, and we then discuss in detail some issues we have experienced with crawling. The considered techniques are:

- Network traffic analysis [Gill et al. (2007); Nazir et al. (2009)]
- Ad-hoc applications [Nazir et al. (2008)]
- Crawling the user graph [Mislove et al. (2007); Cha et al. (2008); Lerman (2007); Cha et al. (2009)]

2.1 Network traffic analysis

This is a typical traffic sniffing and analysis technique that captures packet streams from a network link and then analyzes request-response pairs from network traces involving user interactions with a social network. From these request-response pairs it is possible to infer information about social browsing through the network content that is, which users are visiting other users pages. Furthermore, information about the users can be obtained by the analysis of the response payload, that contains the Web pages. While from a theoretical point of view, this approach is always feasible, accurate sniffing of high volume traces presents technical and legal issues [Crovella and Krishnamurthy (2006)].

– All countries impose restrictions on network traffic analysis to protect the privacy of citizens. Collection of network traffic can be carried out only in private contexts and for limited periods of time, such as university campuses or companies, where users are typically notified about the experiments being carried out. As a consequence, the collected data may be not representative of the entire social network user population.
– Traffic analysis in high speed links presents issues as it may overload the packet sniffing mechanism. The resulting loss of data may hinder the detection of request-response links and may reduce the amount of data that is viable for subsequent analyses. To address this issue it is necessary to deploy the packet capture system on a parallel architecture that may be difficult to deploy [Andreolini et al. (2007)].
– In order to extract the payload useful information from the payload of the responses it is necessary to parse the supplied Web pages. As the structure of the Web pages is specific for each social network, it is impossible to implement a general tool to collect data from different social networks, and the support of each social network requires a significant development effort.

2.2 Ad-hoc applications

Ad-hoc application are third-party applications that exploits a set of APIs, such as the Facebook Developer Platform or OpenSocial, to provide services and games to the social network users. In this architecture, a user does not interact directly with the application servers because the social network infrastructure provides an interface layer between user and application. Developing ad-hoc applications to acquire data from social networks allows to collect information about the users in a twofold way. First, the APIs typically allow the application to access information about the profile of users who are registered to the application. Furthermore, the analysis of the log on the application servers allows to extract information about the dynamic user behavior. The analysis of a social network through an ad-hoc application is not affected by the severe legal issues related to traffic sniffing because users must explicitly register to the application and accept the possibility of information disclosure, this technique is not free of issues.

First of all, it can be used only when the social network (e.g., Facebook [Nazir et al. (2008)]) provides third parties with specific APIs for adding new applications. Moreover, the size of the dataset that may be collected depends on the popularity of the applications: if the applications do not attract many subscribers, the available dataset is limited and useless for analysis purposes. We have also to consider that this approach requires the implementation of novel applications that are specifically designed for one targeted social network. This requires great efforts in software investments that do not guarantee returns if the application is not successful.

2.3 Crawling

Crawling is the most popular solution for data acquisition in social networks and consists on querying the social network for publicly available information about users. This approach is viable for most social networks including YouTube, Flickr, Digg, Orkut, MySpace, and Twitter [Mislove et al. (2007); Cha et al. (2008); Lerman (2007); Cha et al. (2009)]. Crawling may take further advantage of the availability of public APIs that some social network operators provide.

Crawling exploits the typical structure of a social network, that can be modeled as a directed graph $G = (U, E)$, where U is the set of nodes (users) and E is the set of edges (social links among users). Each node has *outgoing links*, and *incoming links*. For the goal of collecting information

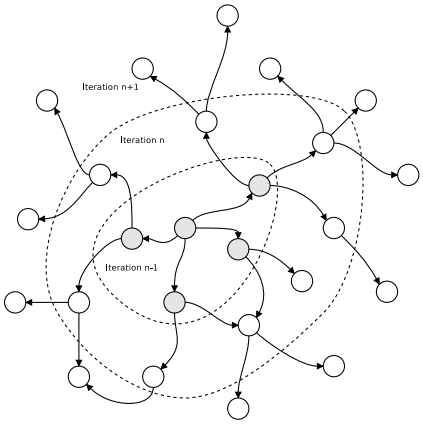


FIG. 1 – *Crawling iterative process*

about the users of a social network, we are not interested in distinguishing between the different types of social links among users (e.g., friends or followed/follower links) but only in devising a way to visit each user in the network.

Crawling the social network graph is an iterative process that starts from a set of initial users and proceeds by discovering new users at every step. The initial setup of the crawler is typically composed by a list of randomly selected users, because starting from multiple random locations is one way to improve the data collection process in terms of duration and data representativeness [Gjoka et al. (2010)].

Different methods can be used to proceed through the crawling graph. They differ in the order through which they visit new users at every step: popular approaches include Breadth-Search-First (BSF), Depth-First-Search (DFS), Forest Fire (FF) and Snowball Sampling (SBS). We adopt the BSF algorithm because it is used extensively in literature for collecting data from social networks [Mislove et al. (2007); Ahn et al. (2007); Wilson et al. (2009)]. Figure 1 illustrates the iterative crawling process following a BSF approach. For the generic step n of the data collection process the crawlers start from the set of previously discovered users (that have been identified at the crawling step $n-1$) and explore the outgoing links to not yet visited users. The set of newly identified users represents the basis for the next crawling step $n+1$. The main reason for the popularity of this method is that an (even incomplete) BSF sample collects a full view (all nodes and edges) of some region in the graph. However, BSF may lead to a bias because it tends to overestimate the node degree by privileging users with high number of social links. However, this bias may be removed by techniques that are able to collect the majority of users in the graph [Wilson et al. (2009)].

To understand the pros and cons of crawling social networks, we implement two crawlers based on the BSF algorithm and we apply them to collect data from YouTube and Digg Canali et al. (2010). We choose YouTube and Digg as examples of popular Web sites that allow users to subscribe, to create social links with other users and to share contents. According to data published by the market research company comScore [comScore comScore (2010)], YouTube is the dominant provider of online video, with a population of over 50 millions of users. In a similar way, Digg is a highly popular bookmark sharing site with more than 15 millions monthly US unique visits in 2008 [Schonfeld (2008)]. Furthermore, YouTube and Digg offer a set of public APIs that allow a

crawler to simplify its access to data of registered users and their social links.

We implemented a crawler for each site because YouTube and Digg offer different API interfaces and adopt different countermeasures to limit extensive crawling. For both crawlers our exploitation of the YouTube and Digg APIs is fully compliant with the terms of use for non-commercial purposes. The efforts for software development and the time required by the execution of the crawlers were significant. Each crawler exceeds 3 thousands lines of code in addition to a DBMS that is used as data storage and for synchronization among different runs of the same crawler. Each crawler was executed on a Pentium IV system with a processor clock of 2.3 GHz and equipped with 2 GBytes of RAM. The YouTube and Digg sites were crawled in the second half of 2009 for a period of 10 days. Table 1 reports the number of users visited and the number of social links as they are the most relevant information for the navigation on the graph $G = (U, E)$ that describes the social network structure. (As anticipated in the Introduction, in this paper we are not interested to the nature and size of collected information.)

TAB. 1 – *Information from social network graph navigation*

Parameter	**YouTube**	**Digg**
Period of crawling	20-30 Aug. 2009	15-25 Nov. 2009
Number of users	1,708,414	349,035
Number of social links	12,935,561	3,212,454

We collected data on nearly 2 million of Youtube users and more than 10 million social links, that correspond nearly to 1% of the network. Moreover, we gathered data for almost 9% of the Digg social network users. Although the data acquisition process lasted for 10 days, the crawler reached a small percentage of the social network structure. Our experience evidenced that crawling the entire user graph in this way may require months of work, thus making impossible to obtain a continuously updated knowledge about the overall network. This problem is primarily caused by the huge size of the user graph to explore. Moreover, the data collection process is further delayed by the countermeasures deployed in the social network to hinder extensive crawling, such as:

– IP banning;
– restrictions on amounts of data results.

IP banning is a typical technique used by social network infrastructures as a protection against Denial of Service (DoS) attacks. A limitation is imposed on the number of requests allowed within a specific time interval (e.g., a few thousands of requests per day) coming from the same IP address. If the amount of requests generated from the same IP address exceeds a security threshold, the servers filter out subsequent requests from that IP for a period of time (e.g., 20 minutes for YouTube). To cope with this problem, we use two network interfaces on the same machine to distribute the crawling traffic among multiple IPs. However, this solution is not sufficient to avoid the risk of IP banning, hence we have to delay subsequent requests to avoid exceeding the limits imposed by the social network operators. In summary, this solution may avoid the risk of IP banning, but it does not resolve the problem of an excessively slow crawling process.

As a further countermeasure, the social network providers limit the maximum number of results that may be returned by the APIs [Youtube Developer's Guide (2010)]. Whenever the APIs are used to download a list of data, such as the list of user outgoing links, the number of returned results is limited to a maximum value. For example, the list of returned outgoing links for YouTube is truncated to 100 entries for each user. This limitation is critical as the node degree distributions within a social network typically determine the presence of *hub* users with several thousands of links. Our crawlers address this issue by integrating pure crawling through APIs with the possibil-

ity of exploiting some information available on the user home page, that contains the complete list of his/her outgoing links. Our crawlers are able to identify the users for whom the API returns an incomplete list of outgoing links, and to integrate this list by parsing the HTML code of the user home pages. This solution is inevitable but it contributes to lengthen the crawling process because it requires separate requests to the user pages to avoid IP banning countermeasures.

To summarize, our experience evidenced that extensive crawling of graphs with millions of users is a challenging task. Traditional crawlers, like the one used in our experiments, are not able to collect a sufficiently large amount of data in reasonable periods of time due to three main issues.

- As the APIs and the structure of users Web pages are specific for every social network, each crawler must be customized and it is not possible to develop a generic crawler common to multiple social network.
- Due to huge network size (in the order of tens of millions of users) it is difficult to reach a large part of the user population in reasonable periods of time.
- The amount of data that can be collected from social networks may be limited. For example, the maximum number of user social links returned by APIs is limited in the order of one hundred, the number of requests allowed within a specific time interval coming from the same IP address is limited to a few thousand requests a day (otherwise *IP banning* occurs). To overcome these limitations, the crawler my parse information available directly on the user Web page. However, this approach is more complex from a programming point of view (to to the complexity of parsing HTML text) and is more time consuming (multiple pages must be accessed to collect the same types of information available through a single API invocation.

These issues motivate our effort to explore innovative approaches for crawling of large social networks.

3 Cloud-based crawling

In this section we describe an innovation solution for crawling, that exploits the cloud computing platform to improve data acquisition of social network data. There are three main goals behind cloud-based crawling:

- to speed up the data collection process by parallelizing the user graph exploration;
- to avoid countermeasures, such as IP banning, introduced by some social network operators to limit extensive crawling, in the respect of terms of use of provided APIs;
- to have a modular software that can guarantee data acquisition from different social networks through the same easily customizable crawler.

The proposed crawling software exploits two main features available in cloud computing platforms: the possibility of distributing the crawling processes over several virtual machines where the exact number can be decided at runtime; a set of software functions for parallel programming and management in the style of Platform as a Service (PaaS).

It is important to have the possibility of specifying at runtime the number and type of necessary virtual machines because the necessary amount of RAM and computational power depends on the size of the social network and on the number of users that are considered at each crawler iteration. The PaaS libraries guarantee a set of software functions for creation/termination of virtual machines, synchronization of parallel jobs, data exchange among jobs. We rely on the *MapReduce* programming paradigm [Dean and Ghemawat (2008)] which is supported by some important cloud computing providers, such as Amazon Elastic Compute Cloud (EC2) [Amazon Elastic Compute Cloud

(2010)]. Our design is compliant with the Hadoop framework [Hadoop MapReduce (2010)] that is the platform for the deployment of our crawler.

The proposed crawler is designed according to a modular architecture with two main parts. The first part contains the engine responsible for the coordination of parallel execution of crawlers. This module is responsible for managing the crawling process, including the choice of the parameters for the number and the characteristics of the virtual machines used for crawling, and the definition of the *map* and *reduce* functions that are used to implement over the cloud platform the crawling algorithm described in Section 3.1. The second part of the crawler software consists of specific code for data acquisition related to each social network. This module contains a standard interface with methods to retrieve information about a user and about his/her social links. However, each social network requires a different implementation of this module because the APIs, when provided, are different, and even the code to parse user home pages and to navigate them cannot be general.

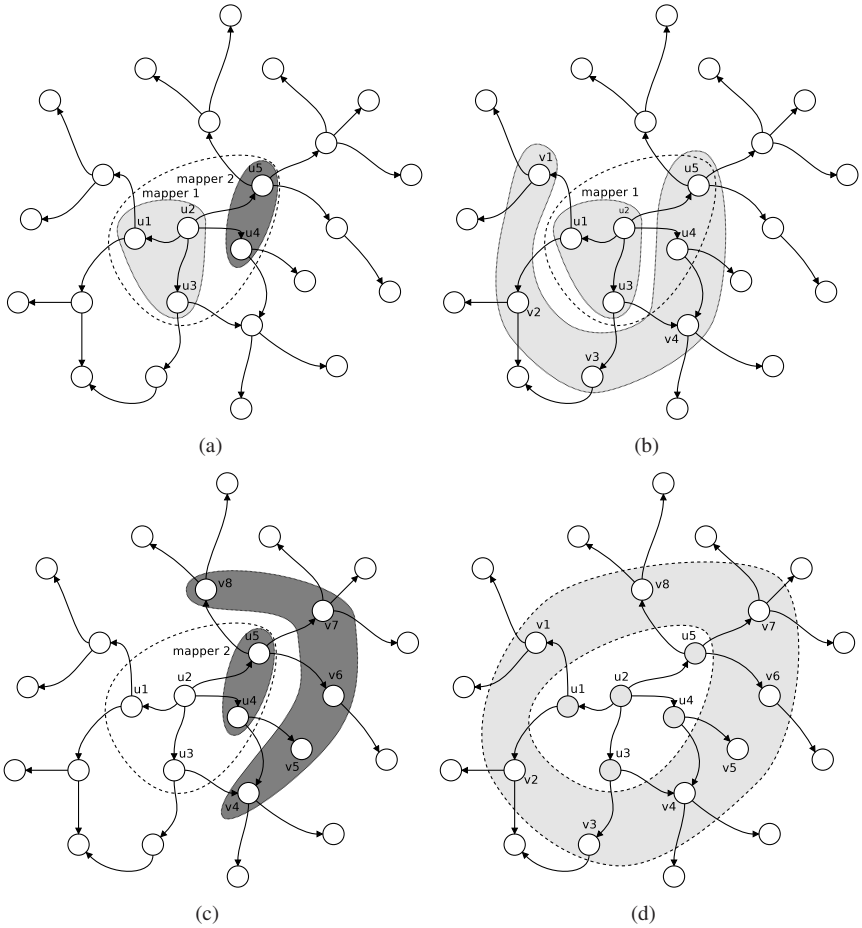
3.1 Crawling engine module

The module responsible for the coordination of parallel crawling tasks exploits the MapReduce programming paradigm, that requires the definition of the *map* and *reduce* phases to model crawling operations. We iterate the map and reduce phases in order to support the Breadth-Search-First approach for data collection within the social network graph, as described in Section 2.

We recall that crawling is an iterative approach where at each step a set of *border users* U is explored for a twofold reason. First, data about the users are retrieved and added to the set of collected information about the social network. Second, all the outgoing social links of the users are scanned to identify the new set of users that will be explored in the next crawling step. Both tasks are implemented using the second module of the crawler that is specific for each social network.

The map phase algorithm starts from the set U that includes the users who are to be explored during the current iteration of the MapReduce algorithm. Figure 2(a) shows an example where the map phase is carried out by two mapper processes: mapper 1 gathers information about the set of users $\{u_1, u_2, u_3\}$, while mapper 2 crawls the users $\{u_4, u_5\}$: the distribution of the users in the original set U is typically carried out by evenly splitting the original input file into chunks. During the map phase, each user u is explored and data are collected about the user and his/her links. A new set V contains the users that are connected to the user u through outgoing social links. In the example, the mapper 1 returns as output the set $\{u_4, u_5, v_1, \ldots v_4\}$ (Figure 2(b)), while the mapper 2 returns the set $\{v_4, \ldots, v_8\}$ (Figure 2(c)). The output of the mapper process is in the format of couples key-value, where the key is the ID of a new user.

The output of the mapper contains duplicate values (in our example v_4) and values that do not belong to the final output. In our example, u_4 and u_5 belong to the initial set of users U and should be filtered out. During the *reduce* phase, duplicated and previously visited users are removed from the list of users that will be explored in the next crawling iteration, as shown in Figure 2(d). The duplicated users are automatically removed by the MapReduce framework as all the key-value pairs with the same key in the output of the map phase are aggregated and sent to the same reducer instance. The task that is carried out by the reduce phase is removed from the output the nodes that were visited in the previous crawling iterations.



3.2 Data collection module

The crawling coordination engine interacts with the data collection module for the acquisition of data from the social network. This latter module contains all the network specific functions of the crawler and provides a standard interface to the mapper. This modular approach guarantees a general purpose crawler that can be easily customized to cope with different social networks.

Figure 2 outlines the data collection module that is activated when the generic i^th mapper invokes the module functions to collect data about a user u. The data collection logic is provided by the *data manager* that can gather information from the social network using two subsystems. First, if the social network provides APIs to third parties, the data manager can use the *API interface* for data collection because this action is much more efficient than Web page parsing. If APIs are not supported or if data obtained through APIs are incomplete, the *Web parser* is activated. It requests the personal Web page(s) of the user u and parses their content. All gathered information is sent to a data storage that in our implementation is a DMBS. Then, the list of social links for the user u is sent back to the coordination engine for the successive phases of crawling.

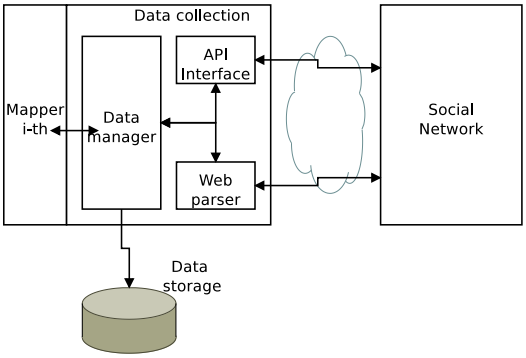


FIG. 2 – *Crawling iterative process*

3.3 Advantages of cloud-based crawling

The proposed cloud-based crawler can address the main issues for large scale data collection in social networks for several reasons.

Using a large number of virtual machines increases data collection speed because the crawling process is distributed over multiple nodes. As data acquisition is a network-bound operation (as it involves only a limited amount of CPU and I/O resources), virtualization provides a further benefit supporting a large number of separated virtual machines, each with its own public IP, on a smaller set of computers. The final effect is a reduction of the crawling time with respect to traditional crawlers, or the crawling of a significantly larger part of the network in the same amount of time. Moreover, since each crawling process runs from a different IP, the risk of IP banning is limited. Furthermore, virtualization allows the fast restart of virtual machine from previously-saved images, and the restart of a virtual machine may be exploited as a way to refresh the pool of IP addresses used for crawling.

We recall that the BSF approach for crawling (described in Section 2.3) may be subject to bias in the collected data when only a small portion of the social network is visited by the crawler. Cloud-based crawling can address this issue because it allows us to collect a large amount of information in a short time period.

It is also important to observe that the crawler modular architecture simplifies the adaptation of the crawler to support data acquisition from multiple social networks because it separates the functions for the coordination of parallel processes (common to any social network) from the interface with the social network specific APIs, including parsing of user home pages, that are the only part of the crawler that needs to be customized for every social network.

4 Related work

Three main techniques have been exploited in literature for data acquisition from social networks: network traffic sniffing and analysis, ad-hoc applications, and crawling the social network graph.

Capturing network data traffic has been exploited by the academic community to collect information about social network user behavior through the analysis of user request-response pat-

terns [Gill et al. (2007); Nazir et al. (2009)]. This technique is typically limited to restricted scenarios such as university campuses and company networks.

Implementing an ad-hoc application for social network analyses has been proposed by Nazir *et al.* [Nazir et al. (2008)] that integrated three novel applications in Facebook to gather the dataset of the subscribed users. This is an interesting and original idea that requires several implementation efforts and that does not guarantee return of investments if the proposed application does not become popular. In any case, the gathered data refer only to the registered users with the high risk of limited analyses and biased results.

In this paper, we focus on crawling that is the most flexible technique to gather data from social networks as testified by many studies [Mislove et al. (2007); Cha et al. (2008); Lerman (2007); Cha et al. (2009)]. We have seen that traditional crawling, although popular, arises many issues, such as the limitations on the number of requests allowed in a specific time period imposed by the social network operators. Many popular social networks offer public APIs that simplify the extraction of data. However, when the data retrieved from APIs are not sufficient to explore the social network structure (for example, because the returned data set is truncated), specific software that gets and parses the user Web pages is required. Due to these and other difficulties that hinder the implementation of a general purpose crawler suitable to multiple social networks, existing studies have analyzed individual or small collections of sites, such as Flickr [Cha et al. (2008); Lerman (2007); Cha et al. (2009)]. The results are limited to detailed views of one popular social network, or provide some comparison between sites that are very similar. A significant exception is the paper [Mislove et al. (2007)], that considers several social networks. However, the study is limited only to the static properties of the network graph, for which data collection is very simple.

We propose to rely on the cloud computing paradigm to build a crawler that allows extensive and complex data gathering and that can be easily customized thanks to its modular structure consisting of a common engine for any social network and a software part specific for each site.

Other tools, such as Nutch [Bialecki (2009)], aiming to exploit parallel data collection and the MapReduce paradigm were proposed initially for Web crawling. We should consider that these tools are designed to navigate through Web pages and hyperlinks and are not designed to explore the graph of social links. Furthermore, these tools cannot take advantage of the social networks APIs to access information about the users, but they are specifically implemented to parse and process Web pages.

A related idea to our proposal has exploited multiple geographically distributed nodes of the Planet-lab network to collect data from a popular social network [Nazir et al. (2009)]. However, the proposed solution as been applied only to one social network and is not designed to cope with different networks with incompatible APIs. Furthermore, the Planet-lab architecture does not provide a flexible support to support the coordination of parallel crawlers, that must be implemented without the facilities provided by the MapReduce approach proposed in our paper.

5 Conclusions

Social networks represent a novel and valuable source of information. In this paper we consider data acquisition techniques and we leave to future work all issues (e.g., legal, privacy) and solutions (e.g., anonymization, sampling) related to the analysis of the sources.

Literature presents three main techniques for data acquisition in social network: network traffic analysis, ad-hoc applications and crawling. We analyze the main features and the limits of each technique with a special focus on crawling that represents the main interest of this paper. In partic-

ular, we evidence that traditional crawling is limited by the large size and complexity of the social network structure that requires excessive times to collect a significant portion of the available data. Moreover, the social network operators tend to use countermeasures to limit excessive crawling from the same IP sources. Finally, each social network provides its own APIs and is characterized by peculiar Web pages and layouts, hence accessing data requires the implementation of a specific crawler for each site.

We propose a novel approach that exploits the MapReduce programming paradigm and the cloud computing platform for a new generation of social network crawlers. Our proposal allows us to design a modular and easily customizable crawler that relies on a large number of distributed nodes to speedup the data collection process and to address the other issues of traditional crawling over social networks.

Acknowledgements

The authors acknowledge the support of FP7–SEC–2009-1 project VIRTUOSO "Versatile InfoRmation Toolkit for end-Users oriented Open-Sources explOitation", Grant Agreement nr. 242352.

References

Ahn, Y.-Y., S. Han, H. Kwak, S. Moon, and H. Jeong (2007). Analysis of topological characteristics of huge online social networking services. In *Proc. of the 16th International Conference on World Wide Web (WWW'07)*, Banff, Alberta, Canada.

Amazon Elastic Compute Cloud (2010). http://aws.amazon.com/ec2.

Andreolini, M., S. Casolari, M. Colajanni, and M. Marchetti (2007). Dynamic load balancing for network intrusion detection systems based on distributed architectures. In *Proc. of 6th IEEE International Symposium on Network Computing and Applications (NCA'07)*, Boston, MA, USA.

Bialecki, A. (2009). Web-scale search engine toolkit. In *Proc. Of Apache Con 2009*.

Canali, C., S. Casolari, and R. Lancellotti (2010). A quantitative methodology to identify relevant users in social networks. In *Proc. of the IEEE International Workshop on Business Applications of Social Network Analysis (BASNA'10)*, Bangalore, India.

Canali, C., M. Colajanni, and R. Lancellotti (2009). Performance Evolution of Mobile Web-Based Services. *IEEE Internet Computing 13*(2), 60 – 68.

Cha, M., A. Mislove, B. Adams, and K. P. Gummadi (2008). Characterizing social cascades in Flickr. In *Proc. of the 1st Workshop on Online Social Networks (WOSP'08)*, Seattle, WA, USA.

Cha, M., A. Mislove, and K. P. Gummadi (2009). A measurement-driven analysis of information propagation in the Flickr social network. In *Proc. of the 18th international conference on World Wide Web (WWW'09)*, Madrid, Spain.

comScore comScore (2010). YouTube Streams All-Time High of 14.6 Billion Videos Viewed. comScore Releases May 2010 U.S. Online Video Rankings.

Crovella, M. and B. Krishnamurthy (2006). *Internet Measurement: Infrastructure, Traffic and Applications*. John Wiley and Sons, Inc.

Dean, J. and S. Ghemawat (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM 51*(1), 107–113.

Gill, P., M. Arlitt, Z. Li, and A. Mahanti (2007). YouTube traffic characterization: A view from the edge. In *Proc. of Internet Measurement Conference (IMC'07)*, San Diego, CA.

Gjoka, M., M. Kurant, C. T. Butts, and A. Markopoulou (2010). Walking in Facebook: A Case Study of Unbiased Sampling of OSNs. In *Proc. of the IEEE International Conference on Computer Communications (INFOCOM'10)*, San Diego, CA.

Hadoop MapReduce (2010). http://hadoop.apache.org/mapreduce/.

Khrishanmurthy, B. (2009). A measure of Online Social Networks. In *Proc. of the 1st International Conference on COMmunication Systems and NETworkS (COMSNETS'09)*, Bangalore, India.

Lerman, K. (2007). Social Information Processing in News Aggregation. *IEEE Internet Computing 11*(6), 16–28.

Mislove, A., M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee (2007). Measurement and analysis of online social networks. In *Proc. of the 7th ACM SIGCOMM conference on Internet measurement (IMC'07)*, San Diego, California, USA.

Nazir, A., S. Raza, and C.-N. Chuah (2008). Unveiling Facebook: a Measurement Study of Social Network Based Applications. In *Proc. of the 8th ACM SIGCOMM Conference on Internet Measurement (IMC'08)*, Vouliagmeni, Greece.

Nazir, A., S. Raza, D. Gupta, C.-N. Chuah, and B. Krishnamurthy (2009). Network level footprints of Facebook applications. In *Proc. of the 9th ACM SIGCOMM conference on Internet measurement conference (IMC'09)*, Chicago, Illinois, USA.

NielsenWire (2009). Social Networking's New Global Footpront. Nielsen Online Report.

Schonfeld, E. (2008). Digg Nearly Triples Registered Users In a Year, Says Sleuth Programmer. Research Report TechCrunch, 2008.

Wilson, C., B. Boe, A. Sala, K. P. Puttaswamy, and B. Y. Zhao (2009). User interactions in social networks and their implications. In *Proc. of the 4th ACM European conference on Computer systems (EuroSys'09)*, Nuremberg, Germany.

Youtube Developer's Guide (2010). Data API Protocol – API Query Parameters. http://code.google.com/apis/youtube/2.0/developers_guide_protocol_api_query_parameters.html.