<http://www.diva-portal.org>

This is the published version of a paper presented at *Network Intelligence Conference (ENIC), 2015 Second European*.

Citation for the original published paper:

Erlandsson, F., Nia, R., Boldt, M., Johnson, H., Wu, S F. (2015)

Crawling Online Social Networks.

In: *Network Intelligence Conference (ENIC), 2015 Second European* (pp. 9-16).

<http://dx.doi.org/10.1109/ENIC.2015.10>

N.B. When citing this work, cite the original published paper.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:bth-10993>

Crawling Online Social Networks

Fredrik Erlandsson*, Roozbeh Nia[†], Martin Boldt*, Henric Johnson*, S. Felix Wu[†]

*Blekinge Institute of Technology

{fredrik.erlandsson, martin.boldt, henric.johnson}@bth.se

[†]University of California, Davis

{rvnia, sfwu}@ucdavis.edu

Abstract—Researchers put in tremendous amount of time and effort in order to crawl the information from online social networks. With the variety and the vast amount of information shared on online social networks today, different crawlers have been designed to capture several types of information. We have developed a novel crawler called SINCE. This crawler differs significantly from other existing crawlers in terms of efficiency and crawling depth. We are getting all interactions related to every single post. In addition, are we able to understand interaction dynamics, enabling support for making informed decisions on what content to re-crawl in order to get the most recent snapshot of interactions. Finally we evaluate our crawler against other existing crawlers in terms of completeness and efficiency. Over the last years we have crawled public communities on Facebook, resulting in over 500 million unique Facebook users, 50 million posts, 500 million comments and over 6 billion likes.

Index Terms—online social networks, online social media, crawling, mining

I. INTRODUCTION

Recently, online social networks, OSNs, have gained significant popularity and are among the most popular ways to use the Internet. There have been efforts to make the social informatics on, for instance, Facebook available for applications (e.g., [1, 2]). Additionally, researchers and developers have become more interested in using the *social interaction networks*, SINs [3], to further enhance and personalize their services [4]. OSNs are also redefining roles within the publishing industry, allowing publishers and authors to reach and engage with readers directly [5]. However, SINs are not directly available today through the current APIs provided by most OSNs. Applications using SINs would therefore spend a lot of time trying to gather the data needed to create the SINs for their services. Therefore, our research problem is how we can design a crawler that makes social interactions in OSNs accessible. To the best of our knowledge it exists no crawler today with the capabilities of crawling *all* interactions in a timely manner. This also refers to the problem of how, when and what to crawl from OSNs in a structured way, which is the focus of this paper.

Researchers have studied and explored the information flow on online public communication boards. However, these communities are usually significantly smaller than the communities we find on Facebook. For instance, communication boards of Open Source Software are limited to a few thousand people out of which only a few hundred of them are active members [6, 7]. Such networks are considerably smaller than

Facebook communities. The number of members of Facebook groups that we have crawled range from a few thousand to tens of millions of people.

The nature of OSNs and the amount of information available makes the problem of what to crawl interesting. To narrow down the scope of the proposed research, we are focusing on the interactions in OSNs. By doing this, we noticed a gap and segregation between the content and the social graph. There have been efforts to make social informatics on Facebook available for applications; enabling social computing applications can simply worry about the computation part and not the social informatics. Performance and incompleteness issues of existing crawlers were the main reason for us to start developing *SINCE - SIN Crawler Engine*, that serves as a bridge between the content and the social graph in the online world, by not only providing which users have interacted with each other but around exactly which content these interactions have occurred. For readability in this paper, a *page* refers to a single community and a *post* refers to anything shared on a *page*. SINCE is crawling open pages on Facebook and can make informed predictions on how social interactions take place. It leverages this to prioritize what content to crawl next or decide which contents need to be crawled again, re-crawled.

SINCE makes the social information around *posts* easily and thoroughly accessible, which is necessary in order to create SINs and build applications such as Friend Suggestion [8] which are dependent on the complete set of interactions between users.

Although SINCE only crawls data from public pages and domains, as discussed in [9] we threat the crawled data with high respect to the users. For instance, we do not draw direct connections between the user id and the profile page of the user. This also means that we do not try to access information from users' *wall* so we never access or make available any data that was published with configured privacy settings. The content of public *pages* are by definition open information and we have discussed this with Facebook representatives and they do not have any concern with our data.

The rest of the paper is organized as follows. We start with a comprehensive discussion of related work in Section II and from that we can validate the need and originality of our work. In Section III we discuss our requirements and various challenges our crawler have been facing, including resource allocation and capabilities to predict when posts needs re-crawling. In Section IV we describe the design decisions taken

while developing our crawler, including the first published solution on how to crawl shares using the Facebook API. We also present our created API that makes the data gathered by SINCE available for other researchers and developers. The methods of prioritizing the crawling queue is described in Section V. We finish the paper (Section VI) with an evaluation and a comparison between SINCE and other crawlers. We also show that the location of the crawler is important together with statistics of measured crawling time for 2.5 million posts. Finally, the paper is concluded in Section VII and future work in Section VIII.

II. RELATED WORK

Despite the huge number of social network publications, few have been dedicated to the data collection process. Chau et al. [10] briefly describe using a parallel crawler running breadth-first search, BFS, to crawl eBay profiles quickly. The study conducted by Mislove et al. [11] is, to the best of our knowledge, the first extensive OSN crawling study published. From four popular OSNs, Flickr, Youtube, LiveJournal, and Orkut, 11.3M users and 328M links are collected. Mislove et al. confirms known properties of OSNs, such as a power-law degree distribution, a densely connected core, strongly correlated in-degree and out-degree, and small average path length.

Gjoka et al. [12] are proposing two new unbiased strategies: Metropolis-Hasting random walk (MHRW) and a re-weighted random walk (RWRW). Where Catanese et al. [13] describes the detailed implementation of a social network crawler. It used the BFS and uniform sampling as the crawling strategies to run the crawler on Facebook, and then compared the two strategies.

Most studies are based on subgraphs, thus it is important to know how similar the sampled subgraphs and the original graphs are. Leskovec and Faloutsos [14] evaluate many sampling algorithms such as random node, random edge, and random jump. The datasets used by Leskovec and Faloutsos [14] are citation networks, autonomous systems, the arXiv affiliation network, and the network of trust on opinions.com, the largest of which consists of 75 k nodes and 500 k edges.

Ahn et al. [15] obtain the complete network of a large South Korean OSN site named CyWorld directly from its operators. They evaluate the snowball sampling method (which is in fact breadth-first search) on this 12 M node and 190 M edge graph. Their results indicate that a small portion ($< 1\%$) of the original network sampled in snowball fashion approximates some network properties well, such as degree distribution and degree correlation, while accurate estimation of clustering coefficient is hard even with 2 % sampling.

Gjoka et al. [16] propose a sampling method to select nodes uniformly without knowledge of the entire network, and use this method on a large sample (1 M nodes) of the Facebook graph. The link privacy problem raised by Korolova et al. [17] concerns how an attacker discovers the social graph. The goal of the attacker is to maximize the number of nodes/links it can discover given the number of users it bribes (crawls).

Several attacks evaluated actually correspond to node selection algorithms for crawling, such as BFS and greedy attacks. The same problem is considered by Bonneau et al. [18] who took a survey of several approaches for obtaining large amounts of personal data from Facebook, including public listings, false profiles, profile compromise, phishing attacks, malicious applications and the Facebook Query Language. The research dataset in [19, 11] was mined through public listing in [18].

III. REQUIREMENTS AND CHALLENGES

Our crawler highly depends on Facebook's API, and therefore, bugs in Facebook's API will cause problems that we have no control over. Also, resource limitations have forced us to be picky about which communities to crawl. Given enough resources, our crawler can be modified to automatically crawl all public communities on Facebook and other OSNs given an initial set of seeds.

A. Requirements

SINCE, from a high level, takes the identifier of a Facebook community as input and outputs a stream of documents. In addition to capturing the response of API requests, our crawler has to satisfy the following requirements:

Coverage: It is important and desirable to be able to crawl each and every post thoroughly and completely. However, if resources do not allow this, it is more desirable to get all the data from a limited set of posts, *depth*, rather than less data from a larger set of posts, *breadth*. Example of an application that leverages breath crawling could be a *Dynamic News Feed*, where users are not only bound to see the posts shared by their immediate friends and pages they have liked but can quickly see emerging topics of interest for the user, which will create a more dynamic news feed.

Since we are dependent on depth, we are aiming on applications built for leveraging social interaction networks, such as *Friend Suggestion* [8]. Hence SINCE is implemented as a deep interaction crawler.

Real-time: The information and interactions on Facebook public communities is time-sensitive. When crawling a post as deeply as SINCE does, the crawling time is an important factor. Although many of the observed posts are quite small with a short crawling time, just a few seconds, we also have big posts with crawling time up to a few hours. Since the interactions on posts and the social interaction networks around these posts are constantly changing, we need means to decide if a re-crawl is needed or not, which is described in Section V.

Important questions that arise due to the nature of how the interactions around posts evolve are “Which posts do we have to re-crawl to get the most updated information?” and “When would be the best time to re-crawl these posts?”.

Scale: As of today there are over a billion users and millions of public communities on Facebook [20]. There are over 2.7 billion likes and comments posted on Facebook on a daily basis as of February 1st 2012 [21]. A crawler must have capabilities to scale to become more and more efficient as content grows.

Data Quality: The crawler should output reliable and uncorrupted data. Therefore, it needs to be able to detect failures in Facebook’s current API and be able to restart from exactly where it stops when a failure occurs.

B. Re-crawling

Unlike traditional blogs or websites where only the administrators are able to post updates to their websites, OSNs are constantly receiving new posts from hundreds of millions of users around the clock. Therefore, traditional web crawlers and their algorithms that identify re-crawling time would not satisfy our requirements. OSN users are more active with ongoing interactions, collaborating and posting new content. Therefore, not only do we need to crawl everything efficiently in a given amount of time but we have to detect whether we would need to re-crawl what we have already crawled in order to get additional content. This issue most often arises for popular posts that are constantly receiving new interactions such as likes, comments, and shares from users; hence, crawling such posts are extremely expensive and it is crucial to make an informed decision whether and when we should issue a re-crawl.

The data we have gathered has been analyzed by Wang [22] suggests that most interactions on posts happen within three hours after the post was made. Table I, shows the number of comments on popular posts divided between four different time intervals after the post was initially made. In (a) we see that more than 70 % of first comments take place within the first 30 minutes after the post was initially created. In addition, if the post has more than 20 comments, then 98 % of the posts have the first comment within the first 30 minutes, and merely 2 % of the posts get the first comment after 30 minutes. (b) shows that if the post has more than 20 comments, only 2 % of the posts will get the fifth comment over two hours from when the post was initially created. Another point we can take from this table is that 95 % of posts that get their tenth comment later than three hours from when it was first initially shared, will get fewer than 15 comments total. (c) shows that 92 % of posts that get their tenth comment later than three hours from when it was first initially shared, will get less than 20 comments total. (d) shows that 95 % of posts that get their 20th comment later than three hours from when it was first initially shared, will get less than 40 comments total. This information is used and discussed further in Section V, and we are using it in order to decide whether we will need to re-crawl a post that have already crawled and further, when would be the best time to re-crawl the post in order to get the complete view of each post.

Another approach which helps to decide whether we should re-crawl a page or post is by looking at how the SIN forms around a particular page or post and how the members of SIN have interacted with the posted items before. Psychology studies show that people tend to interact with posts that they personally agree with, and that most people would not initiate an opposing point of view [23]. Although, once an opposing

point of view has been posted, the rest of the people are very likely to follow.

Based on the ideas described above, SINCE not only is able to efficiently detect which posts need to be re-crawled, but also decides when would be the best time to re-crawl such posts to capture the maximum amount of interactions and reduce the probability of needing to re-crawl the post in the future.

IV. A PLATFORM TO MAKE INTERACTIONS ACCESSIBLE

SINCE is able to crawl all public pages and groups on Facebook, given just basic privileges like a normal Facebook user. Even private communities can be crawled given that the user id of the crawler has access. Furthermore, it is easy to modify our tool in order to extract information from other OSNs.

A. Design

SINCE is designed to perform crawling in two stages. *Stage one* uses the Facebook’s unique identifier of a public community (page or a group) to find the id of all posts, messages, photos, and links posted on the given community by admins and members. This is a straightforward process that has to consider Facebook’s pagination [24] of API requests as discussed below. A stage one crawl will simply access the feed connection on the community-id of the community we are interested in and continue to read the next page until it reaches the last page. This will give us a complete list of posts in a particular community.

Stage two is a bit more advanced. Since we are interested in all social interactions for each post, we have to make the following requests for each post gathered at stage one. We first gather the *post* itself, this post contains basic information like author, type, the message, and in applicable cases, links to the posted photo, link, video. In the first request we also get a preview of the posted comments and who have liked the post but this is not a complete view. In order to get all *likes* we iterate through the *like handle*. To get all comments on a post we have to iterate through the post’s *comment handle* and since each comment can have *likes*, we have to iterate through the *like handle* for each comment as well. As discussed in Section IV-B, Facebook does not provide a direct API to the *shared by* information. However, we have found a work-around for this problem and that will add an additional call to the graph. The methods described for *stage two* crawling means that for posts with a lot of interactions we have to make multiple requests to the graph. For instance, we have crawled posts with hundreds or thousands of comments each with a few likes, where we have to make a request for each comment to get its likes, resulting in crawling times of several hours for one single post.

Pagination: As mentioned before, Facebook has a limit on how many entities to be returned from calls to their graph-API [24]. This is by default set to 25 for Facebook’s internal calls. We have modified this, so all calls we make to Facebook requests 200 entities. The trade off is that, the higher this limit is configured, the more likely a failure might occur on

TABLE I
NUMBER OF COMMENTS ON POSTS AFTER THE POST WAS INITIALLY MADE.

		Number of Comments intervals										Total
		1-5	6-10	11-15	16-20	21-25	26-30	31-35	36-40	41-45	46-50	
First Comment (a)	< 30 mins	2762	462	348	259	185	152	104	71	41	27	4411
	> 30 mins	1599	74	20	12	1	0	2	2	5	3	1718
Fifth Comment (b)	< 2 hours	68	337	329	255	184	150	104	72	45	29	1573
	> 2 hours	117	199	39	16	2	2	2	1	1	1	380
Tenth Comment (c)	< 3 hours		3	189	217	179	141	104	70	43	30	978
	> 3 hours		76	179	54	7	11	2	3	3	0	335
20th Comment (d)	< 3 hours				1	40	95	80	62	40	26	344
	> 3 hours				43	146	57	26	11	6	4	293

Fig. 1. Our distributed crawling mechanism

Facebook’s servers; since, every request has a short time limit to be completed. For each failure we will need to re-crawl the number of posts equal to the limit that we have set. We have found 200 to be the ideal limit for our use cases with concern of the issues described above.

Facebook restrictions: Our crawler is built as a distributed system as discussed by Chau et al. [10]. This satisfies our demand of high crawling rate and works as a work around to the fact that Facebook only allows 600 requests per 600 seconds. We have one controller that is keeping track on current status and what data (in our case which *page* or *post*) to crawl next. The controller supports multiple agents. Figure 1 shows a basic sketch of how the controller and the crawling agents are connected. Each agent runs independently and can have its own Facebook applicationid and userid. In our current version we have seen that it is possible to reuse each applicationid for ten to fifteen agents (based on the physical location of the agent as discussed in Section VI-B). Running more agents with the same applicationid will hit Facebook’s 600/600 limits and force the agents’ with the same applicationid to wait up to 600 seconds before they can continue crawling.

Efficiency measures: As described before, our crawler is designed as a distributed system. The controller is keeping

track of interesting *pages* and the corresponding *posts* with support for n-agents to do the actual crawling. At most we have had just over one hundred active agents. The controller holds and prioritizes a queue of which *pages* and *posts* to crawl, when we see that one *page* has many interesting interactions we can point the agents to crawl that page. As of today, we have up to a few hundred agents that are able to grab community and post ids from the controller and crawl the context based SINS. Given enough active crawling agents, our tool can easily adapt to crawl every public community on every OSN in a timely manner.

B. Crawling Shares

One of the shortcomings in the current Facebook API is the lack of ability to crawl shares. Share is a term used by Facebook to show posts that have been shared by a user other than the initial poster. A user can share a post to their own, their friends, or any community that they have access to. The problem to crawl shares has been reported as a bug [25], but to the best of our knowledge there are no solutions to this issue. In the documentation Facebook has provided for developers, the shares are not covered. Results from API calls like http://graph.facebook.com/<community-id>_<post-id> only returns the number of shares. Opposed to similar calls for likes and comments where we get the full list of who has taken which action and when. It is interesting in terms of weighting different posts among each other to see how many shares, likes and comments each post have, where we consider the shares to have the highest impact of importance for a post. In our crawled dataset we have seen that a user is much more prone on doing a simple *like* or perhaps leaving a *comment* on a post than to re-sharing the post among its social graph. Not only is the number of shares important, but to use the crawled data to build ego-networks it is also interesting to see who have shared a post.

Our solution to this is using the fact that most of the items on Facebook have a globally unique id. When looking at the standard method for crawling a post we always combine the page-id with the post-id and separate them with an underscore. For instance, http://graph.facebook.com/123_456 where 123 is the page-id and 456 is the post-id for the post. By making a request to post-id directly and then

adding the keyword `sharedposts` is it possible to see who have shared a particular story. In fact, to crawl the users who have shared a story the request have to look like this: <http://graph.facebook.com/456/sharedposts>. This request will return information of the users who have shared the post with time stamp and which audience they have shared it to.

C. Application Programming Interface

Together with *SINCE* we also have implemented a social aware kernel that is able to compute the social interactions, and make the produced data available through an API. Enabling developers and researchers to implement applications that can access these interactions and our crawled data. In addition, we compute and produce social interaction networks on the go around different content shared on OSNs. This functionality allows developers to only worry about how they like to leverage such social informatics to improve their services, instead of spending tremendous amount of time gathering the raw data and producing the networks themselves. Furthermore, we allow developers to directly access and even modify our database by adding new fields and creating new tables in order to be able to store their computed data so that they and other developers can benefit from their computations in the future. This functionality is not available through any API provided by OSNs as of today; therefore, developers and researchers are either not able to request social interactions or have to implement a crawler and compute this.

The functionalities provided by our API include retrieving more information with fewer requests compared to other APIs out there. In addition, we do an extensive amount of computing in order to create the SINs [3] based on the specified types of interactions in the request and providing the interaction networks in the returned response. Finally, the feature that separates our work from every other API is the extended functionality feature. We allow developers and applications to extend our object oriented designed system to add their own functions/modules to our code and later call these functions on our server.

V. PRIORITIZATION OF THE CRAWLING QUEUE

As stated before, we have one controller keeping track of the current progress and status of the agents. In addition, the controller does not do any actual crawling but is managing a queue of posts and pages that needs crawling. This queue was initially built as a simple FIFO queue, first in first processed, with the addition of keeping track of failures and timeouts from the agents. If a post is sent to an agent and the controller does not receive a response after 4 hours, the controller considers this post to have failed during crawling and will move it to the top of the queue. Although this is a simple and quite efficient method to get the system up and running, it is neither very intelligent nor efficient in terms of getting the most interesting posts. Therefore our controller has been updated to make use of the findings discussed in Section III-B

TABLE II
PENALTY MODEL USED BY THE CRAWLER TO PRIORITIZE THE QUEUE.

	Status	Probability (of new content)	Priority
Page	new (uncrawled)	high	5
	$\Delta t > interval$	medium	3
	$\Delta t < interval$	low	1
Post	new (uncrawled)	high	4
	$(\Delta t < 30min) \& (comments > 20)$	medium	3
	$(\Delta t < 3h) \& (comments > 40)$	medium	2
	$(\Delta t < 3h) \& (comments < 40)$	low	1
	$\Delta t > 2h$	low	0.5
	$\Delta t > 2months$	low	0

$$\frac{[current\ time] - [update\ time]}{[last\ post\ time] - [first\ post\ time]} \Rightarrow \Delta t \text{ (time since last crawl)}$$

$$\frac{[last\ post\ time] - [first\ post\ time]}{[number\ of\ posts]} \Rightarrow interval \text{ (post intervals)}$$

in order to prioritize the queue in a more intelligent way. The controller also uses the penalty model shown in Table II.

As seen in Table I, if a post is crawled less than 30 minutes from the creation and it has more than 20 comments we can expect this post to expand more. Therefore, the crawler keeps track of this post and re-crawls it again. Also, the crawler knows the last time a *stage one* crawl was performed of the communities (groups and pages). This information is used to know when to expect new content and issue a re-crawl of the community. Posts older than a few months are not prone to have many new interactions, based on the findings in Table I, so these posts typically does not need to be re-crawled.

VI. EVALUATION

Many of traditional metrics used for crawlers such as speed, bandwidth utilization and staleness are not usable when crawling interactions on Facebook. The reason is that the major issues are related to restrictions introduced by Facebook to limit application's ability to extract too much data as fast as described in Section IV-A. However as discussed in the same Section we have taken measures to get an efficient crawler by using multiple application ids and user ids to maneuver around these restrictions.

As seen in Table III, our crawler is the only one with full coverage, looking at all interactions around a specific *post*. It is also the one with the largest dataset as our dataset is much broader and covers all interactions on crawled content. Although our crawler limits in automatization of crawling, the current implementation where we are adding interesting *pages* manually have given us a few advantages over automatic crawling strategies; We can decide if a *page* seems to be of interest and if the interactions on the page can be of value for other research applications. To our knowledge, there exists no other crawler with the same capabilities and with full coverage of *pages* and *posts*.

A. Crawling time

Our tool is considerably faster and at the same time it does a more thorough job than other crawling tools crawling OSNs using HTML parsers. This is due to the following reasons.

TABLE III
COMPARISON OF DIFFERENT CRAWLERS AND CRAWLING STRATEGIES

Author	Open source	Coverage	Network(s)	Dataset	Performance	Parallelism	Crawling algorithm
Erlandsson et al.	yes	full	Facebook (extendable)	500 M comments, 500 M users, 50 M posts, 6 billion likes	good	yes	Manual
Mislove et al.		partial (just users)	Flickr, LiveJournal, Orkut, YouTube	11.2 M users	-	yes	BFS
Gjoka et al.	no	partial (just users)	Facebook	172 M users	good	yes	MHRW
Catanese et al.	no	partial (just users)	Facebook	8 M users	medium	no	BFS
Haralabopoulos & Anagnostopoulos		partial	Twitter	60 GB data, 93 M users	good	yes	BFS
S. Ye, J. Lang & F. Wu	no	partial	Flickr, LiveJournal, Orkut, YouTube	10.6 M nodes	-	no	Greedy, Lottery, BFS

Firstly, we use Facebook’s API and get all the content in JSON format meaning that we do not have to worry about parsing text or HTML, which by itself could be a complicated process. Secondly, other crawlers that rely on HTML pages will miss a lot of information since Facebook only makes a limited part of what has been shared available through HTML, i.e. what they provide within their API contains a lot more data that is simply not visible to the user. Finally, upon a failure, HTML crawlers will need to restart the process from the very beginning, while our tool will pick up from where it last successfully crawled the OSN.

We have had agents spread over the world, Figure 2a shows the distribution of crawling time over 2.5 million posts. The figure shows that the median crawling time is 1.86 seconds, but there are also posts that require a lot longer crawling time. We have recorded crawling times of over 10 000 seconds for posts that have hundreds of thousands different types of interactions. The box plot on the top illustrates the median, and the distribution of the crawling time.

B. Crawling location

Our agents are spread over the world (currently Sweden, Taiwan and the USA). What we have observed here is that the closer to Facebook’s data center in California the agent runs, the faster response it will get. While this could be considered to be an obvious fact; the closer you are to the server you are communicating with the faster the response will be. However we still think it is interesting to show that it have a significant impact on crawling rate based on where the crawler runs. The network latency must be considered when building large-scale data mining tools.

Figure 2b shows the distribution of crawling time based on the agents’ location using the same dataset as for Figure 2a. It confirms our assumption that the closer the crawling agents are to the Facebook’s data centers the shorter crawling time can be achieved. The long tail is created since some of the crawled post can be very large, compared to the distribution

of our crawled posts. The Amazon plot is utilizing Amazon’s EC2 cloud service and its shape differs because it only ran for a few days, where the rest of our agents ran for a few weeks, and during that time the posts served were considerably larger than the rest of distribution of our crawled posts.

VII. CONCLUSION

We have shown means for building an extensive tool to crawl public communities in OSNs. Our distributed crawler satisfies the requirements described in Section III-A and is capable of retrieving a complete set of non-corrupted data, including all the content shared and all the user interactions around them, from public pages on Facebook. We have given a description of how to design a mining tool for OSNs that can be used in social interactions.

Our findings show that to get a full view of social interactions on networks like Facebook it requires a considerable amount of resources. To increase performance of our crawler we have built a distributed system with one controller, responsible for keeping track of the current status, and multiple agents, making the actual crawling.

Our crawler is capable of handling failures and errors that might happen on the OSN’s API level. In addition we have taken steps to overcome the 600 API requests per 600 seconds that Facebook has by letting our crawling agents use different application keys.

VIII. FUTURE WORK

There is a lot of information being shared on different OSNs, but due to resource limitations we can not capture everything that is happening on Facebook. It is fair to say that Facebook is known to be the largest OSN with the most number of active users. It would be interesting to modify our crawler to capture the interactions on other OSNs, such as Google+, Twitter, and Youtube. The social graph introduced by these services differ from each other and it would be interesting to study and compare the effect of that on the

Fig. 2. (a) shows the crawling time distribution over 2.5 million posts. As we can see the average time is quite short but there is a spread over 10000 seconds, or roughly 3 hours, for posts with a lot of interactions. (b) shows the distribution of crawling time based on the agents' location.

communications and interactions that take place on different OSNs.

Our prioritization scheme discussed in Section V poses the risk of having “blind” interactions, interactions that occur very late in the post’s lifetime. Therefore we would like to further investigate what percentage of the interactions we potentially are missing by not re-crawling old posts.

Another limitation that we have due to the limited hardware access is that we cannot capture everything that is happening on OSNs, we have to pick the most interesting groups to us and prioritize which communities to crawl. Given enough hardware, our crawler can easily be modified to automatically detect new communities or *pages* based on their interactions with the communities that we start crawling from (i.e. our initial seed).

The prioritization discussed in Section V address the issue on when to crawl a post. It would be interesting to use meta-data gathered in *stage one* for checking if a post could be considered to not be interesting and thereby ignoring it. Here the traditional data mining term of interestingness should be evaluated on social media. A study to map interestingness to social media data in order to prioritize data would be a good start to evaluate data and prioritize if some posts can be ignored.

REFERENCES

- [1] R. Lee, R. Nia, J. Hsu, K. N. Levitt, J. Rowe, S. F. Wu, and S. Ye, “Design and implementation of faith, an experimental system to intercept and manipulate online social informatics,” in *Proceedings of the 2011 International Conference on Advances in Social Networks Analysis and Mining*, ser. ASONAM '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 195–202. [Online]. Available: <http://dx.doi.org/10.1109/ASONAM.2011.86>
- [2] H. Zhao, W. Kallander, H. Johnson, and S. F. Wu, “Smartwiki: A reliable and conflict-refrained wiki model based on reader differentiation and social context analysis,” *Knowledge-Based Systems*, vol. 47, no. 0, pp. 53 – 64, 2013. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0950705113001068>
- [3] R. Nia, F. Erlandsson, P. Bhattacharyya, R. Rahman, H. Johnson, and F. Wu, “Sin: A platform to make interactions in social networks accessible,” in *ASE International Conference on Social Informatics*, dec. 2012.
- [4] C. Wilson, B. Boe, A. Sala, K. P. Puttaswamy, and B. Y. Zhao, “User interactions in social networks and their implications,” in *Proceedings of the 4th ACM European conference on Computer systems*, ser. EuroSys '09. New York, NY, USA: ACM, 2009, pp. 205–218. [Online]. Available: <http://doi.acm.org/10.1145/1519065.1519089>
- [5] A. Mrva-Montoya, “Social Media: New Editing Tools or Weapons of Mass Distraction?” *The Journal of Electronic Publishing*, vol. 15, no. 1, Jun. 2012. [Online]. Available: <http://dx.doi.org/10.3998/3336451.0015.103>
- [6] C. Bird, D. Pattison, R. D’Souza, V. Filkov, and P. Devanbu, “Latent social structure in open source projects,” in *SIGSOFT '08/FSE-16: Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*. New York, NY, USA: ACM, 2008, pp. 24–35.
- [7] C. Bird, A. Gourley, P. T. Devanbu, M. Gertz, and

- A. Swaminathan, "Mining email social networks," in *MSR*, 2006, pp. 137–143.
- [8] R. Nia, F. Erlandsson, H. Johnson, and F. Wu, "Leveraging social interactions to suggest friends," in *International Workshop on Mobile Computing and Social Computing (MCSC 2013)*, july 2013.
- [9] F. Erlandsson, M. Boldt, and H. Johnson, "Privacy threats related to user profiling in online social networks," in *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Conference on Social Computing (SocialCom)*, sept. 2012, pp. 838–842.
- [10] D. H. Chau, S. Pandit, S. Wang, and C. Faloutsos, "Parallel crawling for online social networks," in *Proceedings of the 16th international conference on World Wide Web*, ser. WWW '07. New York, NY, USA: ACM, 2007, pp. 1283–1284. [Online]. Available: <http://doi.acm.org/10.1145/1242572.1242809>
- [11] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and analysis of online social networks," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, ser. IMC '07. New York, NY, USA: ACM, 2007, pp. 29–42. [Online]. Available: <http://doi.acm.org/10.1145/1298306.1298311>
- [12] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou, "Walking in facebook: a case study of unbiased sampling of osns," in *Proceedings of the 29th conference on Information communications*, ser. INFOCOM'10. Piscataway, NJ, USA: IEEE Press, 2010, pp. 2498–2506. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1833515.1833840>
- [13] S. A. Catanese, P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti, "Crawling facebook for social network analysis purposes," in *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*, ser. WIMS '11. New York, NY, USA: ACM, 2011, pp. 52:1–52:8. [Online]. Available: <http://doi.acm.org/10.1145/1988688.1988749>
- [14] J. Leskovec and C. Faloutsos, "Sampling from large graphs," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '06. New York, NY, USA: ACM, 2006, pp. 631–636. [Online]. Available: <http://doi.acm.org/10.1145/1150402.1150479>
- [15] Y.-Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong, "Analysis of topological characteristics of huge online social networking services," in *Proceedings of the 16th international conference on World Wide Web*, ser. WWW '07. New York, NY, USA: ACM, 2007, pp. 835–844. [Online]. Available: <http://doi.acm.org/10.1145/1242572.1242685>
- [16] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou, "A walk in facebook: Uniform sampling of users in online social networks," *CoRR*, vol. abs/0906.0060, 2009.
- [17] A. Korolova, R. Motwani, S. U. Nabar, and Y. Xu, "Link privacy in social networks," in *Proceedings of the 17th ACM conference on Information and knowledge management*, ser. CIKM '08. New York, NY, USA: ACM, 2008, pp. 289–298. [Online]. Available: <http://doi.acm.org/10.1145/1458082.1458123>
- [18] J. Bonneau, J. Anderson, and G. Danezis, "Prying data out of a social network," in *Proceedings of the 2009 International Conference on Advances in Social Network Analysis and Mining*, ser. ASONAM '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 249–254. [Online]. Available: <http://dx.doi.org/10.1109/ASONAM.2009.45>
- [19] J. Bonneau, J. Anderson, R. Anderson, and F. Stajano, "Eight friends are enough: social graph approximation via public listings," in *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, ser. SNS '09. New York, NY, USA: ACM, 2009, pp. 13–18. [Online]. Available: <http://doi.acm.org/10.1145/1578002.1578005>
- [20] TechCrunch. (2012, Sept) Facebook announces monthly active users were at 1.01 billion as of september 30th, an increase of 26% year-over-year. [Online]. Available: <http://techcrunch.com/2012/10/23/facebook-announces-monthly-active-users-were-at-1-01-billion-as-o>
- [21] T. Cherdar. (2012, Feb.) Facebook user data. [Online]. Available: <http://venturebeat.com/2012/02/01/facebook-ipo-usage-data/>
- [22] K. C. Wang, "Communication behavior in online social network," march 2013, ph.D. Qualifying Examination, UC Davis.
- [23] J. Stromer-Galley and P. Muhlberger, "Agreement and disagreement in group deliberation: Effects on deliberation satisfaction, future engagement, and decision legitimacy," *Political Communication*, vol. 26, no. 2, pp. 173–192, 2009. [Online]. Available: <http://www.tandfonline.com/doi/abs/10.1080/10584600902850775>
- [24] F. Developers. (2012, Sept) Facebook's graph api pagination. [Online]. Available: <http://developers.facebook.com/docs/reference/api/pagination/>
- [25] FacebookBlog. (2012, May) Shares count on facebook graph api not available. [Online]. Available: <http://developers.facebook.com/bugs/28967920771682/>