Sampling Online Social Networks with Tailored Mining Strategies

Conference Paper · October 2019

DOI: 10.1109/SNAMS.2019.8931829

CITATIONS
0

READS
16

4 authors, including:

Mohamad Arafeh

3 PUBLICATIONS **1** CITATION

SEE PROFILE

Azzam Mourad

Lebanese American University

105 PUBLICATIONS **887** CITATIONS

SEE PROFILE

Ernesto Damiani

Khalifa University

768 PUBLICATIONS **10,064** CITATIONS

SEE PROFILE

Sampling Online Social Networks with Tailored Mining Strategies

Mohamad Arafeh[1], Paolo Ceravolo[2], Azzam Mourad[1], Ernesto Damiani[3]

[1]*Computer Science and Mathematics Department, Lebanese American University, Beirut, Lebanon*
[2]*SesarLab, Università degli Studi di Milano, Milan, Italy*
[3] *Center of Cyber-Physical System, Khalifa University, Abu Dhabi, UAE*

***Abstract*— With the vast amount of daily generated data that expands every day, Online Social Network (OSN) is considered a key source of information for many Big Data applications. Despite that, companies behind OSNs resort to putting more constraints on their APIs gateways, decreasing the number of information researchers can gather while increasing the time of data mining procedures. This paper proposes a new platform to run sampling strategies with maximum scalability, to decrease the budget and time required for mining a representative sampling set. By comparing the accuracy of several strategies, our experiments demonstrate the relevance of the proposed platform in supporting OSN mining.**

Keywords*— *social network, data miner, big data, data analysis, data sampling

I. INTRODUCTION

In recent years, significant attention has been spent on Big Data analytics. Online Social Network (OSN) has proven to represent a first-class source of input of such analytics. Twitter, for example, gives researchers a gateway for accessing billions of users' data such as links, written contents, and pictures. A variety of analytics benefit from such data. To mention a few, we can quote Natural Language Processing [7], Link Prediction [8], Community Detection [9] and Sentiment Analysis [10]. Additionally, data streams collected from OSNs have been widely used in real-time Sociometrics systems, e.g. for monitoring traffic events [1] or fire propagation [2].

With the increased importance of OSN, it emerges a need for getting the full benefit of it, limiting the time and the budget required for mining the continuous data flow they generate. Several approaches have been proposed for mining OSNs [11, 12, 13, 14, 15, 16, 17, 18 ,19]. However, to the best of our knowledge, none of them provide a general framework for tailoring a mining strategy on a specific data source. For instance, none of them is capable of using concurrent sampling and filtering procedures acting on vendor-specific parameters.

Our goal is to propose a mining platform to help researchers and data collectors to easily mine and analyze OSN, defining API-specific and budget-constrained strategies. We believe that to achieve good performance, any case requires its ad-hoc strategy. In our framework, a strategy is the combination of a data source, a network exploring algorithm, a set of filters, and one or more data analytics. Our proposed solution is implemented and tested against randomly generated data and mined Twitter accounts, collecting over 40,000 users with their connections.

More specifically, the paper is organized as follows: in Section 2 we present the related work. In Section 3 we descibe our proposed platform, including its architecture, components, and workflow. In Section 4, we present our system implementation. In Section 5, we present our experimental analysis, and, finally, in Section 6 we go to the conclusions.

II. RELATED WORK

Most OSNs have APIs allowing anyone to query and amass large amounts of information in a short amount of time. Usually, the process includes registering an application, the platform returns then a set of tokens granting access to the streaming API. As of 2015 Twitter, which is the main source of information for researchers, limits the number of queries that can be executed in a 15-minute window, which results in lowering the amount of information analysis can get while increasing the time to mine the required resources.

A mean to avoid the limitation is to upgrade from standard to premium or enterprise API that are subject to the payment of a fee. However, researchers are finding new ways to address this issue, a web scraper mentioned in [3] works by parsing hypertext tags and retrieving plain text information embedded into them. Since web scraper doesn't get information directly using API, they aren't restricted by the limitation posed by OSN providers and thus can mine large amounts of data with less time and budget. But using a web scraping tool has its limitations, a web scraper can't be used for a long term monitoring, websites are in constant changes over time, thus web scrappers must be updated constantly, this makes a web scraping tools a constant work and a burden on the developer behind it. Additionally, each OSN requires a custom web scraper, this issue isn't different from using OSN API since also each one has its own, but developing a reliable scrapper requires a fair amount of work and a development knowledge that a researcher may not have. Finally, scrapping may expose the researcher to trials, for example, in [4] LinkedIn sued peoples that anonymously scraped their website for different reasons like a violation of the computer fraud and abuse act (CFAA), trespass, and breach of contract.

Another approach discussed in the literature is sampling. A small fraction of the OSN users is mined to create a sampling representative of the whole OSN. The random walk exploring strategy provides the base method to ensure unbiased sampling [11, 12, 13, 14]. Random walk, however, requires a long mixing time, i.e. it requires a long startup period before guaranteeing good accuracy [15]. An effective way for overcoming this issue is to incorporate Uniform Node Sampling (UNI) into random walk sampling strategies to enable the procedure jumping to other parts of the graph. Different authors developed this *random walk with jump approach* [16, 17, 18]. An alternative solution to address the same issue is developing a multi-layered social network, where multiple sources can be followed in order to exit the

blind roads or the local boundaries that a random walk can enter [19].

Researchers have also pointed out that sampling based on social media APIs is biased by policies that are constructed to save vendor's resources [20, 21]. This means scholars using data obtained via API need to apply caution when drawing inferences from such data. In particular, it has been observed that sources of biases arise from the order connected nodes are returned [23], and from the fixed timeframes used for selecting the nodes to be included in the sample APIs [22]. For example, in Twitter, the list of followers is returned based on the age of the link created between two nodes and the Sample Tweets endpoint returns a random sample of Tweets based on the millisecond timestamp that the tweet arrived at Twitter's servers.

The approach followed in our work is to let the user compose strategies using a mixture of approaches and constraints. This supports the setting of API-specific solutions and the comparison of alternative strategies in order to assess them in real-world scenarios.

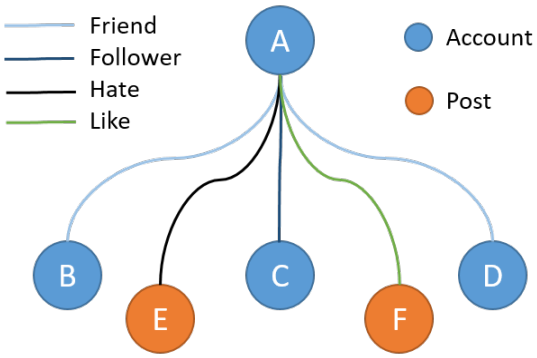

Figure 1 - Relationships between nodes in a graph database.

III. A FRAMEWORK FOR SAMPLING SOCIAL NETWORK

In this section, we present the background information about network space exploring algorithms and introduce our architecture. Accordingly, we discuss the system hierarchy, highlight the importance of the abstraction layer, detail the workflow organizing the mining procedures, and the different filtering strategies that can be used in the system.

A. Network Space Exploring Algorithm

An important part of our framework depends on the navigation algorithms. They represent the component navigating through the social network and to embed it as a graph in our data storing system.

Graph Databases are used to store OSN elements such as nodes or posted content, while the link between them is captured by edges. Edges can be labeled to define the type of relationship interconnecting two nodes, i. e. friends, follower, co-authors, etc. A post and its originator can be represented as two nodes connected by an edge labeled as *posted-it*. A post and a reader can be two nodes connected by a *like-it* or *dislike-it* edge.

Addressing the limits posed by OSN providers, navigation algorithms combined with filters are used as data sampler, working on a subset of a data selected to be presentative of the whole dataset. Sampling social network reduces the time and budget required to collect the minimum information needed. A similar approach has been discussed in [5] where the authors comparatively assess the accuracy of deterministic and probabilistic navigation algorithm.

Since each case requires a different strategy, we have built our platform to support the implementation of different navigation algorithm, allowing us to compare their performance and accuracy under different settings. The algorithms that are available in the platform can be divided into two groups: deterministic like Breadth First, and probabilistic like Forest Fire, Random Walker, and Metropolis Hasting. Additional focus has been given to probabilistic approaches, such approaches can be supplied with hyper-parameters that are capable of changing the shape of the mined data, fitting the data selection task, while further widening the traversed space of the network. Two hyper-parameters frequently used in the platform are forward weight, and number of iterations. Forward weight controls the onward and backward jump rate of the random walker, this value range between 0 and 1, the higher the number to deeper the level explored by the algorithm. This parameter significantly affects the accuracy of the results. Number of iterations controls how long the random procedure has to be repeated.

B. System Architecture, Components and their Roles

A Social Network is an ever-expanding data source, mining data in such a context requires constant updates to the embedded data structures. To address this issue, we used different technologies that help to achieve maximum scalability in our architecture. Figure 2 presents the hierarchy of our architecture listing the abstract classes that compose it.

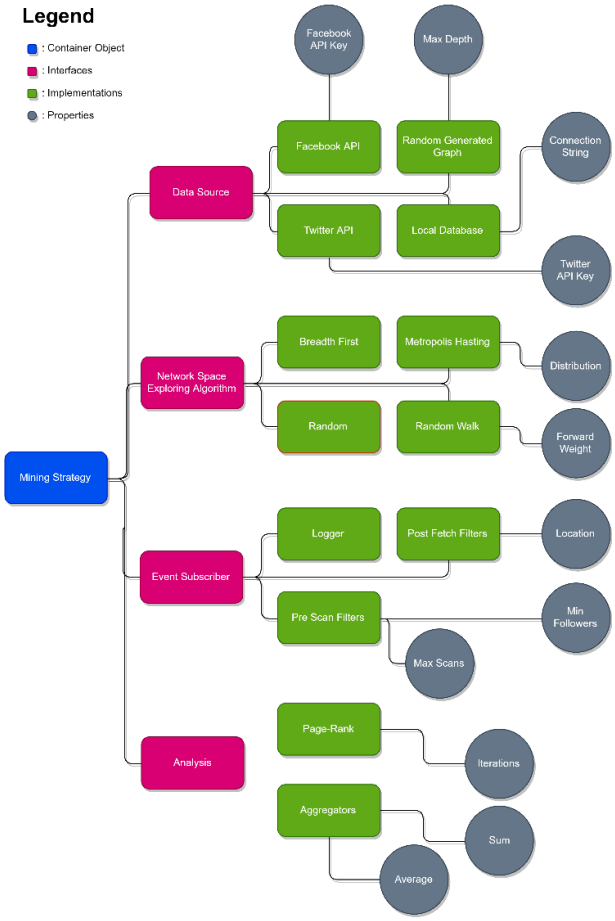

Figure 2 – The Architectural Hierarchy.

At the root level, we have a class for defining mining strategies, each strategy is a combination of multiple settings managed by separated components.

Data Source. A strategy has to contain a connection to an input source which the miner uses for querying data. Sources could be online like Twitter or Facebook, or locally available like a local SQL Database, or data files (CSV, Excel). Abstracting such component allowed us to have a limitless source in which we were able to sustain the daily increasing number of the available data.

Network Space Exploring Algorithm. As discussed in Section 3, a network space exploring algorithm is used to navigate through a data source, embedding the mined data as a graph. Such algorithms define the base procedure to sample the data source, with a great impact on the time and budget spent on mining and analysis.

Event Subscribers. This component allows integrating further components to the core architecture, the idea came from the need of supporting the addition of new features in one hand and data transformation in another hand. Upon each step, this component broadcast the current status to subscribers, thus allowing them to modify and transform the data. This component adopts a limitless amount of custom-defined filters and data embedders. Pre-Scan and Post-Fetch are event subscribers that run before scanning and after fetching, such subscribers can be used as data filters.

Pre-Scan: used to reduce the number of API request by limiting the maximum depth that can be reached by the miner, or by limiting the number of nodes that can be scanned under each branch node.

Post-Fetch: information about the node attributes are only available after the invocation of the fetch method, thus any filter that requires such information is categorized as a post filter, and thus need to subscribe under the post-fetch event.

Analysis. This component contains the algorithms used to extract the required information from the mined data. Currently, we support all the graph algorithms natively supported by Neo4j, including algorithms for computing node centrality metrics, community detection algorithms, pathfinding algorithms, similarity algorithms, and link prediction algorithms.

C. System Workflow

Figure 3 describes the workflow orchestrating our system. The process starts when the network space exploring algorithm chose a node from the social network graph, the results differ based on the algorithm selected while defining the strategy. The next step is scanning the selected node, to further explore the social network graph having an insight of the available nodes on the network and how to navigate to them without having any information about nodes attributes, such attributes are only available after invoking the fetch method on the scanned node.

To reduce the number of API requests, we introduced a caching system that answers the call, in case it was already available in the cache.

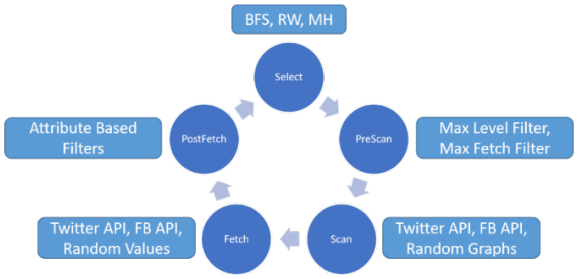

Figure 3 – The System Workflow.

IV. System Implementation

While the orchestrating workflow is always the same, data source, graph navigation, data handlers, and graph analysis vary between strategies. Accordingly, in order to reduce the complexity and to increase the flexibility, we abstracted the architecture main components, therefore, allowing them to have multiple implementations. A strategy defines which implementation we have to use while mining. Besides the core, multiple states of art technologies are used to achieve appropriate scalability and performance. Mined social network graphs are handled by the Neo4j graph database, using this technology we are capable of maintaining the dynamic nature of social networks. Additionally, Neo4j is known for performant querying with further access to various graph algorithms that assist data analysis. MongoDB is used to keep a history of the defined strategy, and to cache social network query results.

A. Mining Process

The mining process manages the orchestration between the components of our architecture, starting the process from a set of seed nodes. It provides the navigator with the required data source while broadcasting regularly to a set of event subscribers. Algorithm 1 describes the mining process in detail. It starts by creating a root for the graph to be mined, then the abstract components are initialized according to a strategy receipt that links to the specific implementations to be adopted in the mining process, e.g. Breadth First Navigator for exploring and Twitter as a data source.

Algorithm 1 Mining Process

```
Require: strategy
let root denote the starting node of the graph
initialise navigator from strategy
initialise dataSource from strategy
initialise eventSubscribers from strategy
observe(navigator, root)
analyse(root)

procedure OBSERVE(navigator, root)
    scan(root)                ▷ initialise seeds, provided when defining a strategy
    for each seed ∈ root do
        NAVIGATE(navigator, seed)
procedure NAVIGATE(navigator, node)
    do
        broadcast(node,'PreScan')
        if node ≠ dead then
            if node is scannable then
                scan(node)
            fetch(node)
            broadcast(node,'PostFetch')
            cache(node)
            node ← next(navigator, node)
    while node ≠ null
```

The initialized navigator scans the root to get the seeds nodes, for each seed it branches a navigator to explore further nodes. Navigation starts from the seed and ends when the navigator has no longer nodes to provide. The selected node that is going to be scanned will be broadcasted first to pre-scan event subscribers, allowing them to modify the node properties preventing it from being fetched or scanned by marking it as unscannable. After a node is fetched it will be

sent to post fetch event subscribers to decide if keeping the node or to prune it. As said the difference between pre-scan and post-fetch is that the later has access to an array of attributes that aren't available before. The fetched node is cached and used later when the same node is requested again.

B. *RandomWalk*

As an example of the navigator procedure invoked by the Mining Process, we illustrate the RandomWalk algorithm. This algorithm serves under the network space exploring algorithm components. The pseudocode in Algorithm 2 shows the process in details. It first initializes a random number and compare it with the weight defined in the strategy to decide going further deep in the graph or returning to a higher level. The bigger the weight the deeper the walker go.

Algorithm 2 RandomWalk Navigator

```
1:  procedure NEXT(root)
2:      let R be a random generated number
3:      let W be the weight provided by the strategy
4:      initialise f := forward if node is root
5:      initialise f := backward if node reached Max Depth
6:      initialise f := backward if node is Leaf
7:      initialise f := forward if R < W and backward otherwise
8:      if f = forward then
9:          return a child selected randomly from node
10:     else
11:         if parent(parent(node)) ≠ null then
12:             initialise ancestor := parent(parent(node))
13:         else
14:             initialise ancestor := parent(node)
15:         return a child selected randomly from ancestor
```

V. Experimental Results

To hasten our experimental analysis, we point our miners to data sources, which helped us comparing different strategy under different conditions. We evaluate the accuracy of different strategies on two data sets. In the first one, we build and test strategies using data mined online from Twitter. In the second one, we mined data from a research dataset. The graphs resulted from the different mining strategies are compared using standard social network analytics.

A. *Random Generated Data Source*

We propose a case study where we need to mine the first level followers of a set of twitter accounts that respects predefined attribute conditions described in the strategy. Table 1 contains an overview of the control data generated using four seed nodes to serve the experiments, focusing on the percentage of the seed's followers of Italian origins. The results obtained by five strategies fetching followers of this particular country are compared with this control data. This gives us a measure for assessing the accuracy achieved by each strategy.

Seed	1	2	3	4
Italy	62%	27%	50%	25%

Table 1 – Set 1 Data Overview

The mining strategies tested are illustrated in Table 2. In all of them, we filtered the data and excluded any account that is not located in Italy. The first mining strategy uses Breadth First as exploring algorithm, fetching only 10% of the maximum account followers. The rest of the strategies alternate Random Walk and Metropolis Hasting as navigation algorithms. Different percentage of the maximum account number to be fetched has been used to test the algorithm accuracy, lower fetching percentage results in a high sampling ratio since a lower number of accounts will be fetched and included in the test results. For Random Walk and Metropolis Hasting, we have 500 Iterations and 0.2 forward weight. Additionally, for metropolis hasting, we used Normal Distribution to generate the next mining position.

	S1	S2	S3	S4	S5
Exploring Algorithm	BF	RW	MH	RW	MH
Account Fetched	10%	10%	10%	5%	5%
Location Filter	Italy	Italy	Italy	Italy	Italy
Iterations	-	500	500	500	500
Forward Weight	-	0.2	0.2	0.2	0.2
Distribution	-	-	Normal	-	Normal

Table 2 - Set 1 Mining Strategies

Experimental results in Figure 3 show the density of the Italian accounts for each seed node in each strategy. We observe that the first strategy displays the worst case, compared to the original data, using breadth-first as navigation algorithm yield 53% for seed 1 compared to 62%, 78% for seed 3 compared to 50% and 78% for seed 4 compared to 25%. This scenario happens when using breadth-first supplied with an attribute filter and a fetch filters for sampling while an important portion of the data of interest is located in the least of the data set. Other strategies show acceptable results since they reflect proportionally the main data even when using a higher sampling ratio.

B. *Twitter Data Sources*

The second experiment uses data provided from Kaggle [6], called Twitter Friends and hashtags. It is a mined list of Twitter users that includes users' information, friends, last seen, language, trending topics the user tweeted about, and others. We are interested in forming communities based on the frequent tags used by a user when posting on twitters. Our results show the density of the communities mined from the data. Since we don't have a global overview of the data, we include a strategy that iterated over a large portion of the dataset. Additionally, for the comparison, we have included other strategies that can be considered samples regarding the original one.

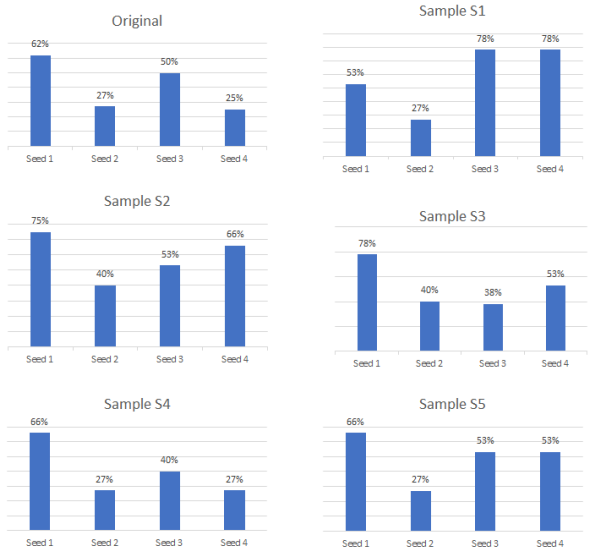

Figure 3 - Set 1 Experimental Results.

Table 3 shows the different mining strategies used by miners. For the first strategy, we don't intend to include any sampling parameter. So, our best option for navigation is breadth-first since we don't care about sampling. We didn't specify any restriction on the maximum accounts that can be fetched for each node. The limitation of a maximum 8th depth level can be considered very high since the data increases exponentially when depth level increase. For other strategies, we used the three remaining algorithms. Only 50 accounts can be fetched under each node. Breadth-first is limited for the first 3 levels, while Random Walk and Metropolis Hasting level are not set, including additional settings for "forward weight" 0.8 that allows them to expand their exploration space profoundly in the lower levels, therefore allowing them to have a more profound view on the data.

	S1	S2	S3	S4
Exploring Algorithm	BF	BF	RW	MH
Account Fetched	-	50	50	50
Max Depth	8	3	-	-
Iterations	-	-	500	500
Forward Weight	-	-	0.8	0.8
Distribution	-	-	-	Normal

Table 3 - Set 2 Mining Strategies.

Experimental results are available in figure 4, and the amount of communities detected by miners is available in Table 4. The first strategy contains the highest possible knowledge about the dataset, therefore can be used to measure the accuracy of other sampling strategies. In total 29 communities are identified in the first strategy, followed by 19 while employing Random Walk navigator. Metropolis Hasting (S4) ought the worst accuracy, compared to the rest of the strategies.

Strategy	Communities Mined
S1	29
S2	14
S3	19
S4	10

Table 4 Community Mined From Set 2

Figure 4 reveals the difference in communities' frequency in each strategy. Each graph presents the top 5 communities with the highest frequency in the data set. Apart from S1 which is the strategy that is used for comparison, S4 was the farthest with a frequent community equals to 72% compared to 83% in S1. Besides, 20% for the second most frequent community compared to 9.3% in S1. However, despite Metropolis Hasting performance was the worst in this case, it is still eligible for revisions as the algorithm is dependent on the hyper-parameters, as discussed in Section 3.

Figure 4 – Set 2 Experimental Results.

Finally, Table 4 measures the number of API requests made, and the total execution time, in seconds, of each strategy. The numbers are based on simulated twitter APIs, taking into consideration the limitation posed at the time of discussion. The results reveal that we can achieve, through a well-defined strategy, an accuracy comparable to the original, with a significant reduction in time, and API request.

	S1	S2	S3	S4
API Req.	13650	379	289	177
Exec. Time (s)	56456	1879	1400	970

Table 4 - Set 2 Execution Time.

VI. Conclusions

OSNs can be considered as a main source of information for any Big Data analysis study. Our aim in this paper is to develop a scalable platform that keeps pace with the continuous development of OSNs and to bypass their restrictions that limit the effectiveness of the mined data. In this regard, we have introduced a sampling strategy that serves as input for platform miners. Our experimental results illustrate a comparison between the sampling efficiency with respect to the original data. Moreover, they explore further the influence of a sampling strategy on the accuracy of the mined graph, thus confirming the importance of the strategy definition step proposed in the platform.

References

[1] A. Nsouli*, A. Mourad and D. Azar. Towards Proactive Social Learning Approach for Traffic Event Detection based on Arabic Tweets. In the Proceedings of the 9th IEEE International Wireless Communications and Mobile Computing Conference (IWCMC 2018), June 25-29, 2018, IEEE.

[2] Abel, Fabian & Hauff, Claudia & Houben, Geert-Jan & Stronkman, Richard & Tao, Ke. (2012). Twitcident: Fighting fire with information from Social Web streams. WWW'12 - Proceedings of the 21st Annual Conference on World Wide Web Companion. 10.1145/2187980.2188035.

[3] Hernandez, Aldo & Sanchez-Perez, Gabriel & Toscano-Medina, K & Martinez-Hernandez, V & Sanchez, V & Perez-Meana, Hector. (2018). A Web Scraping Methodology for Bypassing Twitter API Restrictions.

[4] Case 5:16-cv-04463-LHK. (https://casetext.com/brief/linkedin-corporation-v-does-1-through-100-inclusive_motion-to-expedite-discovery-prior-to-rule-26-conference?sort=relevance)

[5] Paolo Ceravolo , Francesco Ciclosi, Emanuele Bellini, Ernesto Damiani. Assessing Strategies for Sampling Dynamic Social Networks.

[6] Hubert Wassner, Twitter Friends. 40k full Twitter user profile data (including who they follow!), Kaggle

[7] Verma S, Vieweg S, Corvey WJ, Palen L, Martin JH, Palmer M, Schram A, Anderson KM. Natural language processing to the rescue? extracting" situational awareness" tweets during mass emergency. InFifth International AAAI Conference on Weblogs and Social Media 2011 Jul 5.

[8] Pandey B, Bhanodia PK, Khamparia A, Pandey DK. A comprehensive survey of edge prediction in social networks: Techniques, parameters and challenges. Expert Systems with Applications. 2019 Jan 23.

[9] Dakiche N, Tayeb FB, Slimani Y, Benatchba K. Tracking community evolution in social networks: A survey. Information Processing & Management. 2019 May 1.

[10] Zhang L, Wang S, Liu B. Deep learning for sentiment analysis: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2018 Jul;8(4):e1253.

[11] Xu X, Lee CH, Eun DY (2014) A general framework of hybrid graph sampling for complex network analysis. In: Proceedings of the 33rd annual IEEE international conference on computer communications

[12] Gjoka M, Kurant M, Butts CT, Markopoulou A (2011b) Practical recommendations on crawling online social networks. IEEE J Sel Areas Commun 29(9):1872–1892

[13] Ribeiro B, Wang P, Murai F, Towsley D (2012) Sampling directed graphs with random walks. In: Proceedings of the 31st annual IEEE international conference on computer communications

[14] Lee CH, Xu X, Eun DY (2012) Beyond random walk and Metropolis–Hastings samplers: why you should not backtrack for unbiased graph sampling. In: Proceedings of the ACM special interest group (SIG) for the computer systems performance evaluation community.

[15] Mohaisen A, Yun A, Kim Y (2010) Measuring the mixing time of social graphs. In: Proceedings of the 10th ACM SIGCOMM conference on Internet measurement conference

[16] Avrachenkov K, Ribeiro B, Towsley D (2010) Improving random walk estimation accuracy with uniform restarts. In: Proceedings of the 7th workshop on algorithms and models for the web graph

[17] Ribeiro B, Wang P, Murai F, Towsley D (2012) Sampling directed graphs with random walks. In: Proceedings of the 31st annual IEEE international conference on computer communications

[18] Xu X, Lee CH, Eun DY (2014) A general framework of hybrid graph sampling for complex network analysis. In: Proceedings of the 33rd annual IEEE international conference on computer communications

[19] Zhao, J., Wang, P., Lui, J.C.S. et al. Data Min Knowl Disc (2019) 33: 24. https://doi.org/10.1007/s10618-018-0587-5

[20] Tromble, Rebekah & Storz, Andreas & Stockmann, Daniela. (2017). We Don't Know What We Don't Know: When and How the Use of Twitter's Public APIs Biases Scientific Inference. SSRN Electronic Journal. 10.2139/ssrn.3079927.

[21] Morstatter, Fred & Pfeffer, Juergen & Liu, Huan. (2014). When is it Biased? Assessing the Representativeness of Twitter's Streaming API. arXiv. 10.1145/2567948.2576952.

[22] Pfeffer, J., Mayer, K., & Morstatter, F. (2018). Tampering with Twitter's Sample API. *EPJ Data Science*, *7*(1), 50.

[23] Ghosh, S., Zafar, M. B., Bhattacharya, P., Sharma, N., Ganguly, N., & Gummadi, K. (2013). On sampling the wisdom of crowds: random vs. expert sampling of the twitter stream. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management* (pp. 1739-1744). ACM.