On data collection, graph construction, and sampling in Twitter

Jeremy D. Wendt, Randy Wells, and Richard V. Field, Jr.
Sandia National Laboratories
Albuquerque, New Mexico U.S.A.
Email: jdwendt@sandia.gov, rwells@sandia.gov, rvfield@sandia.gov

Sucheta Soundarajan
Syracuse University
Syracuse, NY U.S.A.
Email: susounda@syr.edu

***Abstract*—We present a detailed study on data collection, graph construction, and sampling in Twitter. We observe that sampling on semantic graphs (*i.e.*, graphs with multiple edge types) presents fundamentally distinct challenges from sampling on traditional graphs. The purpose of our work is to present new challenges and initial solutions for sampling semantic graphs. Novel elements of our work include the following: (1) We provide a thorough discussion of problems encountered with naïve breadth-first search on semantic graphs. We argue that common sampling methods such as breadth-first search face specific challenges on semantic graphs that are not encountered on graphs with homogeneous edge types. (2) We present two competing methods for creating semantic graphs from data collects, corresponding to the interactions between sampling of different edge types. (3) We discuss new metrics specific to graphs with multiple edge types, and discuss the effect of the sampling method on these metrics. (4) We discuss issues and potential solutions pertaining to sampling semantic graphs.**

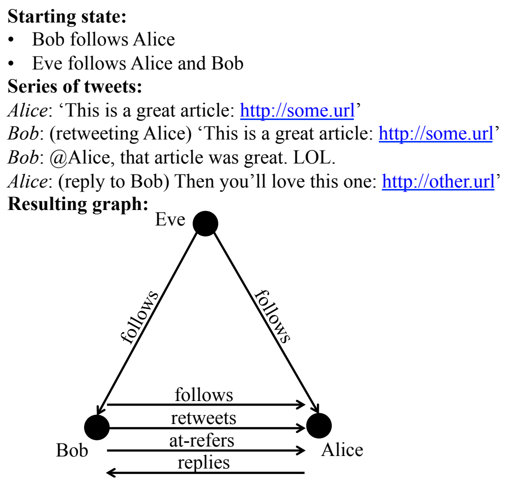
I. Introduction

Twitter provides an exciting venue for social network researchers, allowing for the study of topics such as information flow through large groups and network evolution. Twitter has 320 million monthly active users with almost four-fifths outside the United States.[1] These active users post, read, and respond to short messages (tweets) on a wide variety of topics – from the commonplace to the significant.

Twitter contains many helpful features for information flow analysis, including hashtags and timestamps on tweets. Importantly, Twitter contains a rich social network graph with various edge types, each with a different meaning. Users may subscribe to ("follow") any number of other users' tweets. Moreover, a user can call out ("at-refer") other users by username within a tweet's text. Finally, conversations can be tracked via explicit responses ("reply") or message forwarding ("retweet"). Each of these features can be captured in a rich multiple-edge-type semantic graph (see Fig. 1).

The different edge types indicate varying types of interaction, both in quality as well as quantity. This semantic richness is integral to the Twitter network, and provides unique opportunities for researchers to analyze information flow in a real-world, large-scale environment or study interesting social phenomena such as network maintenance via interactions (*e.g.*, at-refers and retweets) vs. static networks (followers).

[1]As of April, 2016 according to https://about.twitter.com/company.


Starting state:
- Bob follows Alice
- Eve follows Alice and Bob

Series of tweets:
Alice: 'This is a great article: http://some.url'
Bob: (retweeting Alice) 'This is a great article: http://some.url'
Bob: @Alice, that article was great. LOL.
Alice: (reply to Bob) Then you'll love this one: http://other.url'
Resulting graph:

Fig. 1. A tiny sample semantic graph built from Twitter state and between-user interactions.

Before Twitter data can be studied, it must be collected. Twitter provides data access at varying cost levels. Herein, we consider the data rates available via Twitter's free API. The free API has dramatic rate restrictions, resulting in limitations on how much of the graph we can retrieve in any reasonable amount of time. For instance, at the free query rates, it would require many hundreds of years to query for all users' data.

Any network analysis therefore requires sampling of this semantic graph. While considerable work has been done previously in analyzing various graph sampling techniques, these typically assume a single edge type and fewer, if any, limitations on the data collection. The challenges arising on semantic graphs are less well-understood.

Sampling a Twitter graph using the free API poses two major challenges. First, different edge types' samples may represent different regions of the graph. For example, a random-walk crawl on retweet edges might identify a completely different node set than a random-walk crawl on follower edges – even if both begin at the same node. Second, the Twitter free API has drastically different rate restrictions for different edge types. For example, the API allows over an order-of-magnitude more timeline requests than follower requests over the same time interval. Thus, even ensuring that the different edge types cover the same region of the underlying graph, the sample may contain many more edges of one type as a result of API bias.

In this paper, we examine Twitter graph sampling specifically, with an eye toward more general semantic graph sampling. Novel elements introduced herein include the following:

- We describe problems encountered in naïve breadth-first search of Twitter and suggest possible improvements.
- We present two methods for forming semantic graphs from node visits and edge collects.
- We present novel metrics specific to semantic graphs, and analyze our graph-forming methods with these metrics.
- We describe problems with sampling semantic graphs, and propose models that could explain why they occur.

Beyond reporting on our own analyses and results, this work is intended to alert the social networks community to non-obvious difficulties in collecting graphs from Twitter specifically, and in semantic graph sampling more generally. We hope to spark interest in, and further the study of, these problems.

II. Previous work

Over the past decade, the quantity of available network data has increased dramatically. Collecting and analyzing large datasets can be prohibitively expensive, and so much of the research literature has focused on the graph sampling problem: How can one select vertices and edges from a large graph so that the structure of the sampled subgraph is a good representative of the complete data?

There are two distinct strains of network sampling. The first corresponds to crawler-like sampling techniques: Given a limited view of the network, how to decide which data to add to the sample?

For instance, a typical web crawler begins at a web page, and transitions to another page, *etc.*, and at each step must make its next decision based on the neighborhood of the current page and what it has seen in the past [1]–[3]. The second class of network sampling techniques assumes full access to the complete data, and the goal is to downsample the data to some desired size to make analysis feasible [4], [5]. This model of sampling is most relevant when analysis, rather than collection, of the data is costly.

In both types of sampling, the goal is generally to produce a sample that is substantially smaller than the complete network, but retains important structural properties. These properties vary depending on the goal. For example, Maiya and Berger-Wolf attempt to preserve community structure [5], while Leskovec and Faloutsos evaluate the quality of a sample by measuring how well it preserves various graph statistics, such as the degree distribution [4].

Sampling Algorithms. In this paper, we deal with the case when collecting data is time-consuming, and so focus on the sampling via graph exploration model. Many techniques exist for such sampling – random walks, breadth-first search (BFS), and related algorithms are especially popular. Bias in such methods can be reduced through Metropolis-Hastings or reweighting techniques (*e.g.*, [6], [7]). Forest-fire sampling is a modification of a BFS traversal where each neighbor is added to the sample and visited with probability p (instead of all neighbors). In [4], Leskovec and Faloutsos show that forest-fire sampling preserves several important graph properties, such as diameter and size of the largest connected component. Snowball sampling (similar to BFS, but only a fixed number of each node's neighbors (rather than all) are visited) is also commonly used [8]. For further background on sampling, [9] provides an excellent survey of the sampling literature.

Sampling Bias. The bias introduced by sampling techniques has received considerable interest. Achlioptas, *et al.* show that the traceroute sampling method can introduce a power law degree distribution when the underlying network has a Poisson degree distribution [10]. Stumpf, *et al.* show similar results for the case when the sample is generated by randomly sampling nodes, and that sampling from scale-free networks may not produce samples with scale-free degree distribution [11]. Kurant, *et al.* show that BFS introduces a bias toward high degree nodes (*e.g.*, they show that a BFS sample overestimates the average node degree of a Facebook network by a factor of 3.5) [12]. Random walks are known to be biased toward high degree nodes [13]. Modifications can help alleviate this bias; for example, Henzinger, *et al.* use a Metropolis-Hastings random walk to sample URLs uniformly at random [6].

Maiya and Berger-Wolf show that certain types of sampling bias can be beneficial to graph applications [14]. In particular, they argue that bias towards samples with high expansion allows for discovery of previously-unseen regions of the graph.

Sampling Twitter. Researchers have also studied sampling on the Twitter network specifically. Morstatter, *et al.* compare the sampled data provided by the Twitter Free API to the full Twitter Firehose, and showed that the Twitter Free API samples can be poor representatives of the complete network [15]. A subset of the same authors address the problem of finding bias in the free 1% sample of Twitter data. Avrachenkov, *et al.* address the problem of finding high degree nodes with as few API requests as possible [17]. Ghosh, *et al.* compare a 1% sample generated by random sampling vs. a sample generated by collecting content from topical experts, and find that the sample of experts is more diverse and richer [18]. Salehi, *et al.* use respondent-driven sampling, and show that it outperforms Metropolis-Hastings random walk sampling with respect to capturing certain Twitter-specific features [19]. De Choudhury, *et al.* study how the sampling method can affect discovery of information diffusion paths in Twitter, and argue for including both topology as well as content in the sampling process [20].

III. Twitter data collection

In this section, we provide details on Twitter's free API. The rates increase with cost. We describe our data collection implementation and the results from three different collects.

Twitter Free API. Twitter's API provides access through specific requests that are rate limited within a 15 minute window. Although the API allows for many different requests, we focus on the four requests we used to create a semantic user graph (Twitter API names in parenthesis; see Fig. 2):

- Get friends (GET friends/ids),
- Get followers (GET followers/ids),
- Get timeline (GET statuses/user_timeline), and

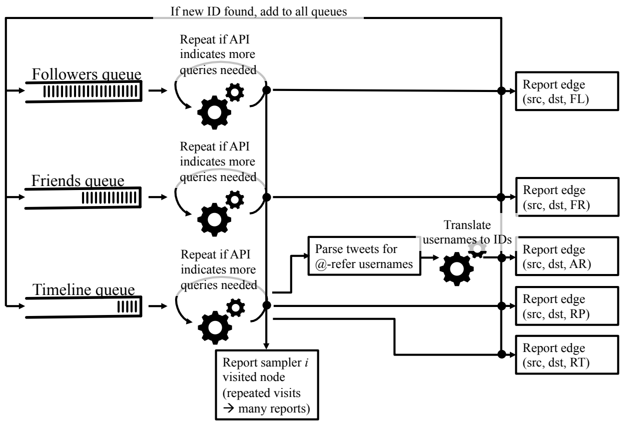

If new ID found, add to all queues
Followers queue
Friends queue
Timeline queue
Repeat if API indicates more queries needed
Repeat if API indicates more queries needed
Repeat if API indicates more queries needed
Parse tweets for @-refer usernames
Translate usernames to IDs
Report edge (src, dst, FL)
Report edge (src, dst, FR)
Report edge (src, dst, AR)
Report edge (src, dst, RP)
Report edge (src, dst, RT)
Report sampler *i* visited node (repeated visits → many reports)

Fig. 2. An overview of our data flow for retrieving data from Twitter via its API. Each gear represents a different API call. As the different requests have different rates and may require different number of requests to get full data for any user, we use separate queues for each request.

- Translate username to ID (GET users/lookup).

A friend is the opposite of a follower: The friends request returns those who the input id follows; the follows request returns those who follow the input id. The timeline request returns the input user's most recent tweets, which we use to identify retweets, replies, and at-refers. As the at-refer is the only of these data which uses usernames, we translate these to user IDs to integrate with the other results.

The free Twitter API limits friend and follower requests to 15 requests every 15 minutes. A request must specify a user ID, and returns 5,000 of its friends or followers (or all, if fewer than 5,000). As some users have more than 5,000 friends or followers, it may require multiple requests to get all of a user's friends and followers. As some users have no followers, some requests result in no new users or edges.

The free API limits timeline requests to 300 requests every 15 minutes. Each request can return up to 200 tweets, beginning with the most recent, and indicates if more tweets are available. Repeated requests can yield up to 3,200 tweets. Each returned tweet has a marker to indicate if it is a reply or retweet and the other user the reply/retweet is attached to. By parsing the tweets, we identify at-refer usernames.

The free API limits username-to-ID requests to 60 requests per 15 minutes with up to 100 users per request.

Our Collection Method. For several reasons, it is impractical to request data for each user via all APIs simultaneously. First, the query rates for the three main requests (friends, followers, timeline) vary by more than an order of magnitude. Second, the number of requests required to exhaust a user's obtainable data varies considerably. Therefore, we use a separate queue for each of these collectors.

With three separate queues, we must decide how to let the queues interact with one another. In this work, we choose to let each queue add newly discovered users to all queues. As friend and follower requests point in opposite directions from the seed user, if collectors did not feed each others' queues, it would quickly result in completely disparate users being visited by the collectors. This decision seeks to maximize the overlap of those users visited by more than one collector.

Therefore, each collector (friend, follower, timeline) asynchronously makes requests from its own queue. After getting the top user id from its queue and ensuring it has not already visited this id,[2] it requests data from the API. Requests are processed and repeated for each user until all available data for that user has been obtained. Each collector separately ensures it does not exceed its API-mandated rate.

We wrote our Twitter collection in Java using Twitter4j,[3] an unofficial library for the Twitter API. We implemented the collectors' queues as priority queues based on the depth at which an ID was discovered. Thus, BFS uses ascending depth order while depth-first uses descending depth order.

For the collections described below, we used BFS starting from a quasi-random seed user id. Specifically, we watched 15 minutes of the Twitter Streaming API (a 1% sample of tweets as they are published), and randomly selected an id from those ids. Note that this strategy biases toward starting with a more active user.

Challenges. We face several challenges in collecting Twitter data. First, due to the drastically different query rates, the resulting network may be dominated by a particular type of edge. For example, in one dataset, we discovered over 8,000 unique nodes in the first 15 minutes of requests. As friend and follower requests are limited to 1,440 per day, (if all users require only one request) it will take over 5 1/2 days to collect all of the friends and followers for these users discovered in 15 minutes. Due to some users requiring more than 1 request each for friends and followers, this duration can easily extend past 8 days. Second, given that it is infeasible to obtain all edges of all types for each sampled user, determining which edges to request poses a major challenge.

Twitter Collection Results. We describe the results of three separate runs of our Twitter collection method. In Section IV we describe how we developed graphs from these results.

Table I lists the duration and number of requests for our three datasets. The variation between number of friend and follower requests is affected by slight variation between the two collectors' timing, and how we counted requests. The Twitter API sometimes returned error types when we requested values. These errors indicated protected accounts, or other unspecified Twitter problems. We did not repeat these error-returning requests. The timeline collector can make far more requests during the same period of time.

Although we made many API requests, a large number of the visits reported in Table I were repeats to the same user. The number of users we visited per collector and dataset is shown in Table II. To collect all data for a user, we required on average 1.8 friend, 36 follower, and 13 timeline requests per user. The followers result was skewed heavily by our finding users with millions of followers.[4]

[2]By checking upon removal from the queue, we avoid the more costly check before adding – which would require checking if an ID was visited and if it was already in the queue.

[3]http://twitter4j.org/

[4]In collect #1, one user had 14.5 million followers (2,864 requests); in collect #3, a user had 10.9 million users (2,159 requests).

TABLE I
REQUESTS PER COLLECTOR

Dataset	Duration (days)	Friend	Follower	Timeline
1	7	7,773	7,259	139,540
2	9	8,690	9,002	168,822
3	7	6,511	6,670	118,682

TABLE II
USERS VISITED PER COLLECTOR

Dataset	Duration (days)	Friend	Follower	Timeline
1	7	4,435	118	13,573
2	9	4,797	878	11,319
3	7	3,780	166	10,050

TABLE III
REQUESTS WITH ZERO RESULTS PER COLLECTOR

Dataset	Duration (days)	Friend	Follower	Timeline
1	7	1,629	13	769
2	9	1,159	289	517
3	7	1,656	36	2,011

When accounting for the actual number of users visited (limited by both request rate and number of requests needed), the disparities seen in Table I shifts. While the number of users visited by friends is now around 1/3 of those visited by timeline, the number of users visited by followers is far fewer.

As shown in Table III, it is not uncommon for a user to have no friends, followers, or relationships expressed with tweets. In fact, requests made that return no results can account for a significant number of requests (as many as 1/4).

IV. SEMANTIC GRAPH CONSTRUCTION AND ANALYSIS

In this section, we describe our process for developing semantic graphs from the available Twitter data. We first introduce notation needed for formal study of graphs with multiple edge types, then define metrics for multi-edge-type semantic graphs. We define two graph construction techniques, and study these techniques' implications using our metrics.

Notation. Let $G = (\mathcal{V}, \mathcal{E})$ be a directed, semantic graph with node set $\mathcal{V}$ and edge set $\mathcal{E}$ that represents the structure of the Twitter network; we will often write $\mathcal{V}(G)$ and $\mathcal{E}(G)$ to denote the nodes and edges, respectively, for graph G. Each edge $e = (u, v, i) \in \mathcal{E}(G)$ is a triple with source node $u \in \mathcal{V}(G)$, destination node $v \in \mathcal{V}(G)$, and integer $i \in \{1, \ldots, q\}$ representing the edge type. It follows that the edge list $\mathcal{E}$ can be split into disjoint sublists $\mathcal{E}_1, \ldots, \mathcal{E}_q$, that satisfy $\cup_{i=1}^{q} \mathcal{E}_i = \mathcal{E}$. For our application, there are $q = 4$ edge types corresponding to follow, at-refer, reply, and retweet.

Metrics for analysis. We use and adapt many common metrics from graph analysis to the case of multiple-edge types. This generalization provides new opportunities for study. There are numerous possibilities including the distribution of node degree on follower edges jointly with node degree on at-refer edges, and the amount of correlation between node degree of follower edges and clustering coefficient of retweets. When studying these semantic graphs, it is important for us to be specific about where we are computing these metrics for each separate edge type i, vs. across multiple edge types. We therefore formally define some graph metrics here.

Traditionally, n and m refer to the number of nodes or number of edges in a graph, respectively. Similarly, we define n_i to be the number of nodes incident to edge type i and m_i to be the number of edges of type i in G. Let

$$\mathcal{N}_i(u) = \{v \in \mathcal{V}(G), v \neq u \colon (u, v, i) \in \mathcal{E}(G)\} \quad (1)$$

denote the set of nodes incident with node $u \in \mathcal{V}(G)$ by an edge of type $i \in \{1, \ldots, q\}$. Therefore, $\mathcal{N}_i(u)$ is the neighborhood of node u across edge type i. Further, let

$$\mathcal{T}_i(u) = \{(v, w, i) \in \mathcal{E}(G) \colon v, w \in \mathcal{N}_i(u); u \neq v \neq w\} \quad (2)$$

denote the collection of triangles that include node u and are connected by edge type i. The number of i-edge triangles that include node u is then denoted by $t_{u,i} = |\mathcal{T}_i(u)|$.

The i-edge degree of node $u \in \mathcal{V}(G)$ is equal to the number of edges of type i that contain u, that is,

$$d_{u,i} = |\{(u, v, i) \in \mathcal{E}(G)\}| + |\{(v, u, i) \in \mathcal{E}(G)\}|. \quad (3)$$

The i-edge clustering coefficient of $u \in \mathcal{V}(G)$ is defined as

$$c_{u,i} = \frac{2\, t_{u,i}}{d_{u,i}\,(d_{u,i} - 1)} \quad (4)$$

if $d_{u,i} > 1$ and 0 else, and the i-edge graph-average clustering coefficient is simply $(1/n_i) \sum_{u=1}^{n_i} c_{u,i}$.

For any of the above node-specific, edge-type-specific metrics, we can compute a population distribution across our collected graph. For instance, we define the degree distribution of edge type i as

$$\delta_i(a) = \sum_{u=1}^{n} \mathbf{1}(d_{u,i} = a), \;\; a = 0, \ldots, n \quad (5)$$

where $\mathbf{1}(x)$ returns 1 whenever the boolean is true, else 0. We define distributions for all of the above metrics similarly.

When considered separately, each of the above defines a marginal statistic. Any of these statistical metrics can be used to make comparisons between two or more edge types within a semantic graph. For instance, we can define a joint degree distribution across two edge types i, j as

$$\delta_{i,j}(a, b) = \sum_{u,v=1}^{n} \mathbf{1}(d_{u,i} = a, d_{v,j} = b) \quad (6)$$

for $a, b = 0, \ldots, m$. Joint distributions for clustering coefficients, number of triangles, and any combination of the three, are also possible, as are higher-order joint distributions by considering groups of three or more variables. For simplicity, we limit our discussion of joint statistics to the Pearson correlation coefficient, which describes pairwise edge-type metrics. Assortativity, used to quantify the correlation of degree between pairs of linked nodes [21], is a similar concept but specific to node degree.

Graph construction. We now have a list of users visited by our three collectors (friends, followers, timeline). These are the nodes of our graph. Each collector also outputs a list of edges observed by one of four types (followers, at-refers, retweets, replies). Note that we flip the direction of friend edges to make them follower edges and equalize their type. However, we did

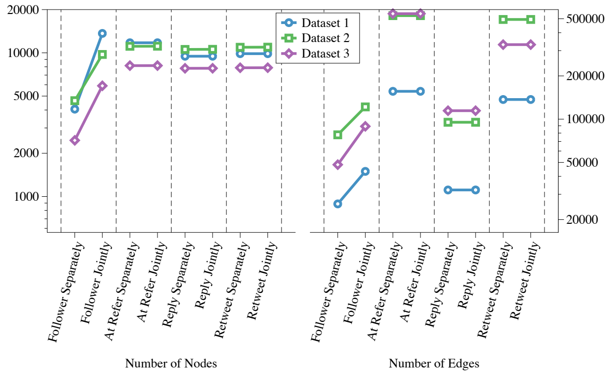


Fig. 3. The number of nodes (left) and edges (right) for all three datasets collected separately and jointly. Note that only the follower graph's nodes and edges change considerably between collecting separately or jointly.

not visit the endpoints of all observed edges, and therefore we cannot simply form a graph from the edges returned (*e.g.*, edge (u, v) was observed upon querying node u, but node v was never visited, then the edge should not be included).

As mentioned, our collector system collects some edge types separately. Unless all visits are constrained to occur at the rate of the slowest collector, there will be nodes visited by some edge type but not another. Therefore, the resulting graph will contain some nodes visited by some collectors, but not by others. We interpret each edge type as a different "layer" in the graph, denoted by G_i. In building our semantic graph, we noted two approaches in how to construct each G_i.

Recall from Section III that data is collected by the follower, friend, and timeline collectors. One approach for constructing G_i is to adopt the traditional single-edge-type approach where each collector marks all nodes visited separately. Therefore, we refer to this approach as *collecting separately*. In this approach, we construct node list $\mathcal{V}_{i,j} \subset \mathcal{V}$ to be the collection of nodes that were visited by collector j ($j = 1, 2, 3$; timeline, friend, or follower) and exist on an edge of type i. This approach results in the following definition for G_i:

$$G_i = (\mathcal{V}_{i,j}, (\mathcal{V}_{i,j} \times \mathcal{V}_{i,j}) \cap \mathcal{E}_i), \; i = 1, \ldots, q. \qquad (7)$$

Thus, when collecting separately, for an edge to be included in G_i, both the source and destination nodes had to be visited by collector j that identified edges of type i.

Another approach for constructing G_i takes advantage of multiple collectors visiting largely but not perfectly overlapping sets of nodes. Furthermore, some collectors visited far more nodes than others. To take advantage of these features, in this approach, we construct node list $\mathcal{V}_i \subset \mathcal{V}$ to be the collection of nodes that were visited by any collector j ($j = 1, 2, 3$ as defined above). The resulting graph can be obtained by Eq. (7) with $\mathcal{V}_{i,j}$ everywhere replaced by $\mathcal{V}_i$. We refer to this approach as *collecting jointly*. Thus, when collecting jointly, for an edge to be included in G_i, only the source had to be visited by collector j (else we could not have found the edge), but the destination had to be visited by any collector. Note that the two approaches result in overlapping node sets that satisfy $\mathcal{V}_{i,j} \subseteq \mathcal{V}_i$.

Effects of Collecting Separately vs. Jointly. Figure 3 shows the increase in graph size for the three graphs created by the three datasets. As at-refers, replies, and retweets were all derived from the timeline collector which visited far more nodes, unsurprisingly, collecting jointly barely changes the resulting graph's size (both number of nodes and number of edges). However, as the followers graph is built from data collected by the friend and follower collectors which each visited far fewer nodes, collecting jointly drastically increases the resulting graph's size.

Aside from the marked increase in size on the followers graph when collecting jointly, we examined marginal distribution metrics. As the timeline-based edge types (at-refer, retweet, reply) increased in size only slightly, unsurprisingly, there were no marked changes in any of their measures. For the follower edges, we note the following changes:

- *Degree distribution*: There are both more degree 1 nodes and the high degree nodes have higher degree.
- *Per-node triangle-count distribution*: The distribution is higher throughout: There are more nodes with just about every number of triangles.
- *Connectivity*: There are more small disconnected cliques on this edge type (mostly two-node), and many more nodes joined the large weakly connected component of the followers graph with very small or no increase in the diameter of the component.
- *Assortativity*: The assortativity changed slightly (an increase or decrease of 0.08 – depending on dataset).
- *Clustering Coefficient*: The clustering coefficient decreased around 0.04 (approximately 36%).

Figures 4–6 show how pairwise correlations across edge types vary between both our different datasets and collecting separately vs. collecting jointly. In all three plots, all three pairwise correlations involving only the timeline-collected edge types (at-refer, reply, retweet) remain consistent whether collected separately or collected jointly. Although generally slight, in all datasets and degree correlations (Fig. 4) including followers, the correlation decreases when collecting jointly. For the correlations across number of triangles (Fig. 5) including followers, the results are consistent within dataset: Datasets 1 and 3 both slightly decrease; dataset 2 slightly increases. For correlations of clustering coefficient (Fig. 6) including followers, the results parallel those seen with number of triangles: Datasets 1 and 3 both decreased; dataset 2 increases, but to even a lesser extent. This makes sense as clustering coefficient is a function of degree and number of triangles.

In short, collecting jointly functions well to increase the size of the resulting graph. However, it affects the internal consistency of the graph when compared against largely unaffected portions of the graph (as measured by the correlations between followers and timeline-based edges). Specifically, correlations between edge types appear to mostly decrease in our three collects. We are only able to observe these changes because of our novel multiple-edge-type measures.

We do not here draw conclusions about Twitter because this data was collected via a BFS and is thus biased. It is not clear

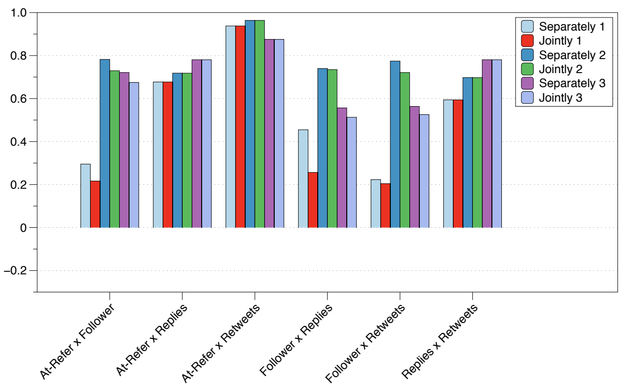


Fig. 4. The correlations across degrees between all pairs of edge types for each of our three datasets each collected both separately and jointly.

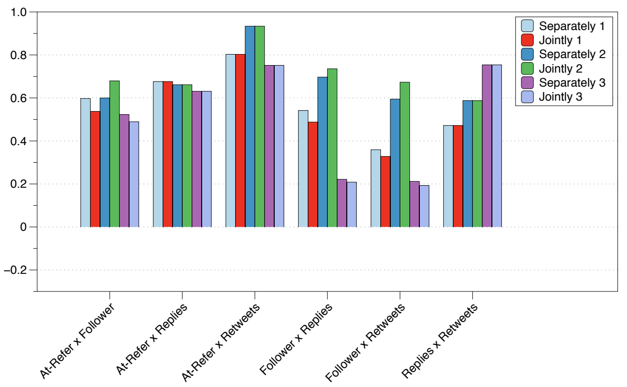


Fig. 5. The correlations across number of triangles between all pairs of edge types for each of our three datasets each collected both separately and jointly.

that the BFS bias here is the same as what is known from the literature on single-edge-type graphs.

V. Subgraph Sampling

Much of network analysis research uses publicly available datasets, which were collected via APIs or crawls.[5] When studying samples, one will typically sample from these datasets, which are themselves samples. Previous work has illustrated how different sampling techniques affect resulting graph properties (*e.g.*, [4], [14]), but it is not known to what extent these biasing effects hold for multi-edge type graphs.

[5]*e.g.*, the Stanford Large Dataset Collection

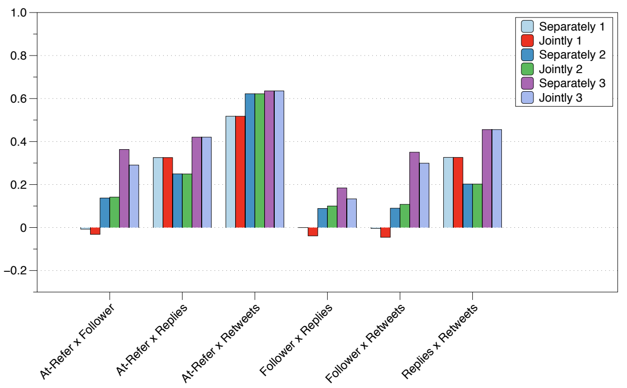


Fig. 6. The correlations across clustering coefficient between all pairs of edge types for each of our three datasets each collected both separately and jointly.

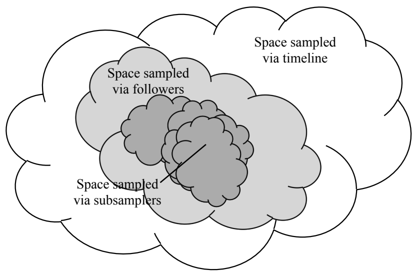

Space sampled via timeline
Space sampled via followers
Space sampled via subsamplers

Fig. 7. Our original conception of how our sampling analysis would perform. Although we realized timeline had collected well beyond how far followers would reach, we expected that many days of followers collection would exceed the reach of a single day of sampling.

Moreover, it is possible that sampling semantic graphs faces fundamentally different biases or challenges. In sampling from our collected Twitter data, we identified a surprising problem – one we have not seen reported previously in the literature.

We implemented a sampler with a similar method to our Twitter API collector (see Sec. III). It had three queues and requested data for nodes starting from a seed node (with each sampler adding new IDs to all three queues). This sampler also used (simulated) 15-minute windows with rate limiting, iterating between the three samplers. Unlike the Twitter API system we built, this sampling system requests data from the stored results from our previous collects. Therefore, we can easily differentiate between the users the sampler requests that our Twitter API collector visited but got no results back (*empty visited*) and the users the sampler requests that the collect saw but did not visit (*unvisited*).

In a graph with only one edge type, we expect that as long as we begin the sampling process toward the middle of an original BFS collect, the process would stay within the space that had been collected earlier (Fig. 7).

However, we observe a surprising result: When attempting to collect a BFS sample from our original BFS collect, we very quickly reach the outer boundaries of our original collect. Table IV shows the results of attempting to execute our BFS sampler starting from the original seed node of our Twitter-API collects for one hour's samples (two orders of magnitude smaller than the full datasets). For all datasets, we found that friends and followers requested far more unvisited users in the first hour than successful requests; timeline requested a similar number of unvisited users than successful requests. This level of failure rate would far overshadow any differences in bias we might see between different sampling techniques.

This surprising result is caused by one prime issue: Due to the asynchronous nature of our Twitter API collect software (and varying network return speeds), we cannot replicate the order that the various collectors were returning results from the API. That is, we can't know at any moment whether the timeline should return its results first, or if the followers request returns first. Thus, because the BFS sampling process may identify edges in a different order than the original collect process, it can find unvisited users very quickly.

Three separate factors demonstrate the extent of this problem. First, the above analysis starts from the original seed,

TABLE IV
SAMPLING SUCCESSFUL AND FAILED REQUESTS

Dataset	Sampler Type	Successful requests (1st hour)	Failed requests (1st hour)
1	Friends	60	816
2	Friends	60	548
3	Friends	60	723
1	Followers	60	4,589
2	Followers	60	28,189
3	Followers	60	644
1	Timeline	1,200	1,052
2	Timeline	1,200	1,141
3	Timeline	1,200	625

but to do a sampling bias estimate, we must start from many random places within the space. If the original seed is this close to the edges, we expect that random seeds will be even closer. Second, BFS is the slowest expanding technique we would consider in our analysis. Forest-fire, snow-ball, or depth-first sampling would each run to the edges far faster. Finally, we were finding this significant number of failed requests in the first hour: Note that we need many hours of collecting for the graphs to grow large enough for reasonable results.

We propose an explanation for this surprising result. First, the semantic graph may suffer from the curse of dimensionality.[6] In brief, the different collectors and edge types add new dimensions to the data. As data expands in dimensionality, it's density decreases, so the number of points between any datapoint and an edge of the space decreases. Second, the problem is exacerbated by the disparity in the collection rates. That is, instead of a spherical space with a thick inner volume, the lower friends and followers rates creates a thin elliptical space where all points are even closer to the boundary.

This effect is dramatically exaggerated by the heterogeneous nature of semantic graphs. Because the different edge types adjacent to a node may lead to completely different regions of the graph (*e.g.*, an individual's followers may be disjoint from his at-refers), sampling in an order different from the original collect process may quickly lead to a lack of sufficient data. This is exacerbated by the large differences in collection rates for the different edge types. Because much of social network analysis takes place on network samples, it is critical to understand the special challenges arising in semantic graphs.

VI. CONCLUSIONS

New Problems from Semantic Graphs and Sampling. We briefly summarize here several problems specific to semantic graphs that we encountered herein but have not seen discussed in the literature.

First, different node and/or edge types may require separate collectors. This results in the collectors likely discovering different neighbors for the same node. Herein, we recommend addressing this with a shared-fed queue (see Figure 2). Second, different collectors may have different sample rates. If not handled by throttling all collectors to the slowest rate, this leads to many critical decisions in graph construction:

- *Multiple queues*: Each collector will need its own to-visit queue (see Figure 2).
- *Different visit counts*: Many nodes will be visited by only a subset of the collectors. How are such nodes to be included in the semantic graph? Herein, we recommend including all nodes visited by at least one collector (see Section IV).
- *Edges between nodes visited by different collectors*: When an edge is discovered by one collector visiting the source, but the destination was visited only by another collector, should that edge be included or discarded. Herein, we present both options as collecting separately and collecting jointly (see Section IV). Until we better understand how the two choices affect biasing, we cannot recommend one over the other.

Third, shared-fed queues with variable collect rates suggest possibilites for different strategies for different collectors. For instance, we identified a weak correlation between the number of visits required for friends and followers collectors on nodes of low-to-moderate degree (not shown above). This correlation broke down for the truly massive – those with millions of followers did not follow millions of others. However, if a graph of more "average" users were desired, we could have identified those with one of the collectors and pushed them as higher priority to the other. Note that any sampler where the user visited may be completely disconnected from others visited by this collector would have to be formed into a graph via collecting jointly, as it's likely that collecting separately would eliminate most (if not all) edges discovered by the visit.

Fourth, we propose that a model based on the curse of dimensionality explains why semantic graph sampling may lead to more unvisited nodes requested than in traditional graph sampling (see Section V).

Finally, a Twitter-specific issue arose from multiple collectors. Twitter allows queries for both followers and friends, but these sample the same set of edges but in different directions. Herein, we used BFS for both collectors (see Section III) and simply performed the union on the resulting collected edges (see Section IV). However, we believe this collection feature deserves more exploration both theoretically and practically.

Specific to Semantic Graphs? In this paper, we describe a new sampling problem in semantic graphs observed in our work on Twitter. This sampling problem differs from previous works in two principal extents. First, we have a multiple-edge type graph collected via multiple collectors with separate (but shared-fed) queues. Second, in our sampling analysis, we do not sample directly on the resulting graph, but back to the original data. When sampling against the original data, we could identify when we requested details on a node that was not visited (and therefore was not in the "full" semantic graph). This was appropriate as any improved collection technique for our needs would need to be implemented against the Twitter API – not against a resulting semantic graph built from the results of a Twitter API collect. An open question is thus how

[6]Term coined by Bellman [22]; an overview of applications in many areas at https://en.wikipedia.org/wiki/Curse_of_dimensionality

much less prevalent the edge problems we encountered are in graphs derived from single-edge data collects.

We proposed a model for explaining this problem. The first part of the model (curse of dimensionality) could be tested by reducing the number of original collects and samplers and seeing if the reduced-dimensionality result failed at lower rates. The second (varying dimensional thickness) could be tested by performing various collects where the different collectors run for different amounts of time. This allows the slower collects to expand out further to more nearly approach the faster collect's size. We would expect that more equally balanced collects would result in lower failure rates.

Enhancements to the Twitter API. We will now briefly suggest some API improvements which could result in improved graph collection. While these are informed by our experience as customers of the Twitter API, we do not intend them as any criticism of the current API. We appreciate the free access to useful, real-world information it provides.

Several sampling techniques require sampling a random edge or edges from the current node (*e.g.*, depth-first, and snow-ball sampling respectively). However, the Twitter API returns results in reverse chronological order. Moreover, the results are not returned as page i of n, but as an iterator. That is, if the results of a single query indicate more results are yet available, you may request more with the returned handle to the next set of results. This could be improved if the API supported requesting an arbitrary page i of n.

The widely disparate rate limitations between timeline and friend/follower requests leads to a wide disparity between the number of users visited in any time period. Our current working model for why semantic graph sampling may have failed indicates this as one of the chief problems. However, even if we had found no semantic graph sampling problems, this leads to a dataset with a heavily lopsided group of users that were visited by only the faster collector. We don't know why the disparate rates were specified, but if they could be brought to parity we believe it would improve sampling ability.

Concerns for Sampling and Information Flow. As described in Section I, our interest in collecting Twitter graphs was influenced by information flow in real-world social networks. When we began this exploration, we considered that (while any partial Twitter graph would be missing many possible paths between nodes) most of the graph would be well represented in a fairly full "middle". Our realization that even the seed node is very close to the edges (Section V) leads us to the critical question: If all nodes are close to the edges, is any information flow analysis on incomplete data likely to be missing possibly short paths between pairs of nodes?

Acknowledgment

The authors would like to thank Ali Pinar for many helpful comments during various phases of this work and for helping create this collaboration. This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

[1] B. Pinkerton, "Finding what people want: Experiences with the WebCrawler," in *2nd International WWW Conference*, vol. 94, 1994, pp. 17–20.

[2] J. Cho, H. Garcia-Molina, and L. Page, "Efficient crawling through URL ordering," in *7th International WWW Conference*, 1998, pp. 14–18.

[3] A. Maiya and T. Berger-Wolf, "Expansion and decentralized search in complex networks," *Journal of Knowledge and Information Systems*, vol. 38, no. 2, pp. 469–490, 2014.

[4] J. Leskovec and C. Faloutsos, "Sampling from large graphs," in *12th International Conference on Knowledge Discovery and Data Mining (KDD)*, Philadelphia, PA, August 20–23 2006.

[5] A. S. Maiya and T. Y. Berger-Wolf, "Sampling community structure," in *19th International WWW Conference*, Rayleigh, NC, April 26–30 2010.

[6] M. R. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork, "On near-uniform URL sampling," *Computer Networks*, vol. 33, no. 1, pp. 295–308, 2000.

[7] 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. [Online]. Available: http://arxiv.org/abs/0906.0060

[8] S. H. Lee, P.-J. Kim, and H. Jeong, "Statistical properties of sampled networks," *Physical Review E*, vol. 73, no. 1, p. 016102, 2006.

[9] M. A. Hasan, N. Ahmed, and J. Neville, "Network sampling: Methods and applications," https://www.cs.purdue.edu/homes/neville/courses/kdd13-tutorial.html, August 2013.

[10] D. Achlioptas, A. Clauset, D. Kempe, and C. Moore, "On the bias of traceroute sampling: Or; power-law degree distributions in regular graphs," *Journal of the ACM*, vol. 56, no. 4, p. 21, 2009.

[11] M. P. Stumpf, C. Wiuf, and R. M. May, "Subnets of scale-free networks are not scale-free: sampling properties of networks," *Proceedings of the National Academy of Sciences*, vol. 102, no. 12, pp. 4221–4224, 2005.

[12] M. Kurant, A. Markopoulou, and P. Thiran, "On the bias of BFS," in *22nd International Teletraffic Congress*, 2010, pp. 1–8.

[13] L. Lovász, "Random walks on graphs," *Combinatorics, Paul Erdos is Eighty*, vol. 2, pp. 1–46, 1993.

[14] A. S. Maiya and T. Y. Berger-Wolf, "Benefits of bias: Towards better characterization of network sampling," in *17th International Conference on Knowledge Discovery and Data Mining (KDD)*, San Diego, CA, August 21–24 2011.

[15] F. Morstatter, J. Pfeffer, H. Liu, and K. Carley, "Is the sample good enough? Comparing data from Twitter's streaming API with Twitter's Firehose," in *International Conference on Weblogs and Social Media*, 2013, pp. 400–408.

[16] F. Morstatter, J. Pfeffer, and H. Liu, "When is it biased?: Assessing the representativeness of Twitter's streaming API," in *23rd International WWW Conference*, 2014, pp. 555–556.

[17] K. Avrachenkov, N. Litvak, L. O. Prokhorenkova, and E. Suyargulova, "Quick detection of high-degree entities in large directed networks," in *IEEE International Conference on Data Mining*, 2014, pp. 20–29.

[18] S. Ghosh, M. B. Zafar, P. Bhattacharya, N. Sharma, N. Ganguly, and K. Gummadi, "On sampling the wisdom of crowds: Random vs. expert sampling of the Twitter stream," in *22nd ACM International Conference on Information and Knowledge Management*, pp. 1739–1744.

[19] M. Salehi, H. R. Rabiee, N. Nabavi, and S. Pooya, "Characterizing Twitter with respondent-driven sampling," in *IEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, 2011, pp. 1211–1217.

[20] M. D. Choudhury, Y.-R. Lin, H. Sundaram, K. S. Candan, L. Xie, and A. Kelliher, "How does the data sampling strategy impact the discovery of information diffusion in social media," in *Fifth International Conference on Web and Social Media*, vol. 10, 2010, pp. 34–41.

[21] M. E. J. Newman, "Assortative mixing in networks," *Phys. Rev. Lett.*, vol. 89, p. 208701, Oct 2002. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevLett.89.208701

[22] R. E. Bellman, *Adaptive Control Processes: a Guided Tour*. Princeton University Press, 1961.