A Case Study of Collecting Dynamic Social Data: The Pro-Ana Twitter Community

Ian Wood

Australian National University, Canberra ACT 0200, Australia,
ian.wood@anu.edu.au

Abstract. The study of social processes in on-line social media content is a relatively new and rapidly growing endeavour. Many social media platforms provide a public API (Application Programming Interface) which can be used for the targeted collection of data from perceived communities, however existing software for this purpose focusses on a "snapshot" of the community and its communications, and ignores important aspects of its dynamics. We present a data collection system designed to capture tweets and the dynamics of Twitter user profile and friend/follower lists, and an approach to identify a set of tags or keywords that define an on-line community. This approach and system were used to collect a data set spanning 2 years and 7 months (including 3 Christmas periods) from the "pro-ana" (pro-anorexia) and eating disorder Twitter community.

Keywords: data set, Twitter, software, adaptive cluster sampling, network dynamics, social networks

1 Introduction

Social psychology is a dynamic process — people enter and leave social groups and groups change and adapt their sense of identity and social norms. It is these dynamics that make our societies what they are, that generate and define human social fabric. In order to study and eventually make predictions about group behaviour, it is essential that we capture the dynamics of the socially meaningful features under study.

With the exception of one very recent paper [2][1], data collection for the study of Twitter friend/follower network dynamics has been limited to a small number of network snapshots without finer grained timing [6, 1, 3] and focus on large scale and/or short term effects, without consideration of specific Twitter communities.

Working from the observation that Twitter hash tags can define a community [7, 4], we propose an approach analogous to adaptive cluster sampling [5] to iteratively expand a small selection of hash tags thought to be used by/define a

[1] the authors claim to have precise timing for friend/follower network changes, however they do not describe how they obtained the data

community by examining tags used in tweet samples from queries on the current set of tags. Frequent tags are added if they do not attract many tweets deemed to be from outside the community.

For some time, a phenomenon has existed on the internet that conveys positive messages about anorexia, the so called "pro-ana" movement. In addition to definitively "pro-" anorexia messages, there are other messages relating to eating disorders and connected social phenomena such as the "thin ideal". In order to study the "pro-ana" phenomenon and it's social context, we sought to collect "pro-ana" and eating disorder related Tweets. A selection of Twitter hash tags used almost exclusively by people with eating disorders was constructed following our approach.

We present a system that captures Twitter friend network dynamics by polling a users' friends and followers lists and user profile data each time we collect a tweet from them. User profile data is also polled with each tweet. Any changes are stored in a replicated MongoDB database and regular backups are performed. It incorporates several features to ensure reliable operation in the face of extreme tweet rates, outages and failures. The system was used to collect all tweets containing the identified hash tags and associated dynamic user data over a period of spanning three Christmas periods — over 1.2 million tweets, 300 thousand users and 200 thousand images.

In Section 2, I discuss communities defined by hash tags and the approach used to identify tags from the Twitter "pro-ana" community. In Section 3, I discuss the approach for sampling the dynamics of community network and user data. In Section 4, I discuss the software design and technical challenges. In Section 5, I present some general statistics from the collected data. In Section 6, I summarise our findings, and finally in Section 7, I talk about possible system enhancements and related research directions.

2 Identifying Communities

2.1 Hash Tags and Communities

Hash tags are used in Twitter and other micro-blogging sites as a way to organise, emphasise and otherwise colour posts. A hash tag is simply a word with the hash character "#" prepended, such as *#diet*. They allow users to specify aspects of their posts that they consider important and to direct their posts to what they feel is an appropriate audience [7, 4]. It is this second point that we attempt to harness in order to collect the output of a hypothesised community around "pro-ana" (pro-anorexia) and eating disorders.

2.2 Adaptive Sampling for Search Tags

Following the intuition behind adaptive sampling [5], a search query for collecting tweets was selected through an iterative process identifying hash tags used by people with eating disorders. At each step a sample of tweets was collected

containing the tags from the previous step, then a set of relevant new hash tags was selected from frequent tags in that sample.

In the first iteration, a brief study of tweets and Tumblr[2] posts containing *#proana* revealed related tags *#thinspiration #anorexia #bulimia* and *#pro-mia*[3].

For the second iteration, tweets were collected from August 18–22 2012 on these tags, a total of 1182 tweets. Hash tags were counted in the collected tweets, and those with more than 3 occurrences were considered. The majority of the tags appeared to have much wider usage than the community of interest (e.g.: *#diet*). A quick manual check by conducting a twitter search on each tag confirmed these suspicions, and those tags were discarded. Manual checks on the remaining tags revealed many more with wider usage. Such tags were discarded if most of the relevant tweets (those deemed to represent communications within the hypothesised community) in which they were used also contained other selected tags — hence the candidate tags were unable to identify new tweets not already identified by other selected tags.

For each tag, a judgement had to be made about the number of irrelevant tweets compared to the number of new tweets not collected by other tags in the query. In general the distinction was clear, and no compromise was necessary. However a few, such as *#depression* and *#selfharm* were discarded despite identifying a small number of relevant tweets not identified by other tags. It was deemed preferable to maintain a higher degree of data relevance and a smaller query. Such a query stands a smaller risk of saturating the Twitter API rate limits which could result in lost data. Table 1 lists some typical tags that had wider usage with descriptions of the wider contexts in which they were used.

This process applied to all the remaining tags resulted in a core of 14 tags identified as used almost exclusively by people with eating disorders, as well as capturing nearly all tweets in the sample that were deemed produced by those people. Tweets with these tags were collected for a further 4 days from September 15 to 19 2012 and the analysis repeated, however no extra tags were identified. Table 2 lists the final set of selected tags.

3 Collecting Dynamic Twitter Data

Purchased twitter data as well as tweets collected with the Twitter API's, contain a snapshot of the tweeting users' profile information with each tweet, giving some information about user profile changes a user may have made between the collected tweets from that user[4]. However the profile snapshot does not contain the lists of friends and followers of the user, and to the best of my knowledge

[2] www.tumblr.com

[3] *bulimia* is short for bulemia nervosa, an eating disorder related to anorexia nervosa; *pro-mia* is short for "pro-bulemia".

[4] though user data embedded in tweets can be stale.

Description	Tags
diet and weight loss	*#diet #weightloss*
body shape and parts	*#skinny #thin #supermodelbody #legs #bones*
adjectives and verbs	*#perfect #motivation #recovery #perfection*
used to discuss eating disorders	*#anorexia #eatingdisorder #eatingdisorders*
fitness and exercise	*#fitspiration #fitspo #gym*
depression and it's symptoms not specific to eating disorders	*#selfharm #depression #cutting*
with other meanings irrelevant to eating disorders	*#ana #ed #mia*

Table 1. Non Personal Eating Disorder Tags

#proana #promia #anasisters #bulemia #bulimic #ednos #edproblems #hipbones #thingsanataughtme #thinspiration #thinspo #abcdiet #thighgap

Table 2. Tags Selected For Search Query

historical data on changes in the friend/follower network are not commercially available[5].

Several previous studies of Twitter friend/follower network dynamics have used a small number of network snapshots without finer grained temporal information [6, 1] and collected data for a fixed set of users, without focus on particular communities. One very recent study of note [2] was able to obtain a large data set containing precise timing for friend/follower network changes, however they do not describe how. Their data is sufficiently large that it would contain comprehensive information from many social groups, however it spans just one month.

The Twitter APIs do not provide a feed containing network changes as they happen, however they do provide a REST (Representational State Transfer) interface for collecting a snapshot of a users' friends and followers lists. The data collection strategy presented here uses that endpoint to poll a users' friend/follower lists each time they tweet. In this way the dynamics of the friend/follower network of users active in the data is recorded with similar frequency to their tweets and the user profile data contained therein.

[5] a recent Quora post claims that the TwitterCounter service provides this for up to one year.

It should be noted that the dynamic data thus collected is not complete. A user who watches tweets in the data set, but themselves tweets rarely, will only be polled on the occasions they tweet — profile changes and follow/unfollow actions made between recorded tweets by users not active in the data are only captured in aggregate (though follow/unfollow actions will be detected if the recipient tweets). The temporal data, then, has an element of sampling error, a systematic lag, which is particularly pronounced with infrequent tweeters. None the less, one might hope that changes to both the user profile and friend/follower lists which are relevant to social processes within the hypothesised pro-ana twitter community will have a high probability of occurring near the time of each tweet to that community (i.e.: each tweet within my data set). We should, however, be careful when analysing behaviour of infrequent tweeters, as there may be a bias in the recorded dynamics of their data.

4 Algorithms and Technical Challenges

This section outlines the design of the dynamic twitter data collection system and describes some of the challenges that had to be faced in order to create a stable system, robust to the the vagaries of tweet data flow, that efficiently utilises the restricted data bandwidth provided by the Twitter APIs. Over the course of two years of continuous data collection, a stable system that is robust to twitter outages and sudden increases (by orders of magnitude) in data volume, and efficiently utilises the narrow data bandwidth for collecting friend/follower data has been built.

4.1 Overall Architecture

The system uses a multi-threaded architecture, enabling asynchronous HTTP requests to the various Twitter API end points and media URLs (see Figure 1). Communication between threads is achieved with thread-safe queues. A tweet collection thread regularly polls the *search/tweets* REST API with the query tags shown in Table 2. Each tweet and it's meta-data are stored in the database and the authors user id is added to the friends, followers and user profile queues. Any media URLs and corresponding media entity ids are added to the entities queue.

The systems main thread (the initial thread when the system launches) initially constructs the shared, thread-safe queues and launches the other threads. It then monitors the other threads, restarting them if they crash and gracefully coordinates system shut-down when requested. To help debug frozen threads, as can happen in early development of multi-threaded systems if thread locks are not properly released, the main thread also responds to operating system QUIT signals, dumping a stack trace of each thread. As a further precaution, a unix *cron* script (which is run regularly by the operating system) monitors the system as a whole and relaunches it if an hour passes without activity. The system also has a simple thread which initiates regular database backups.

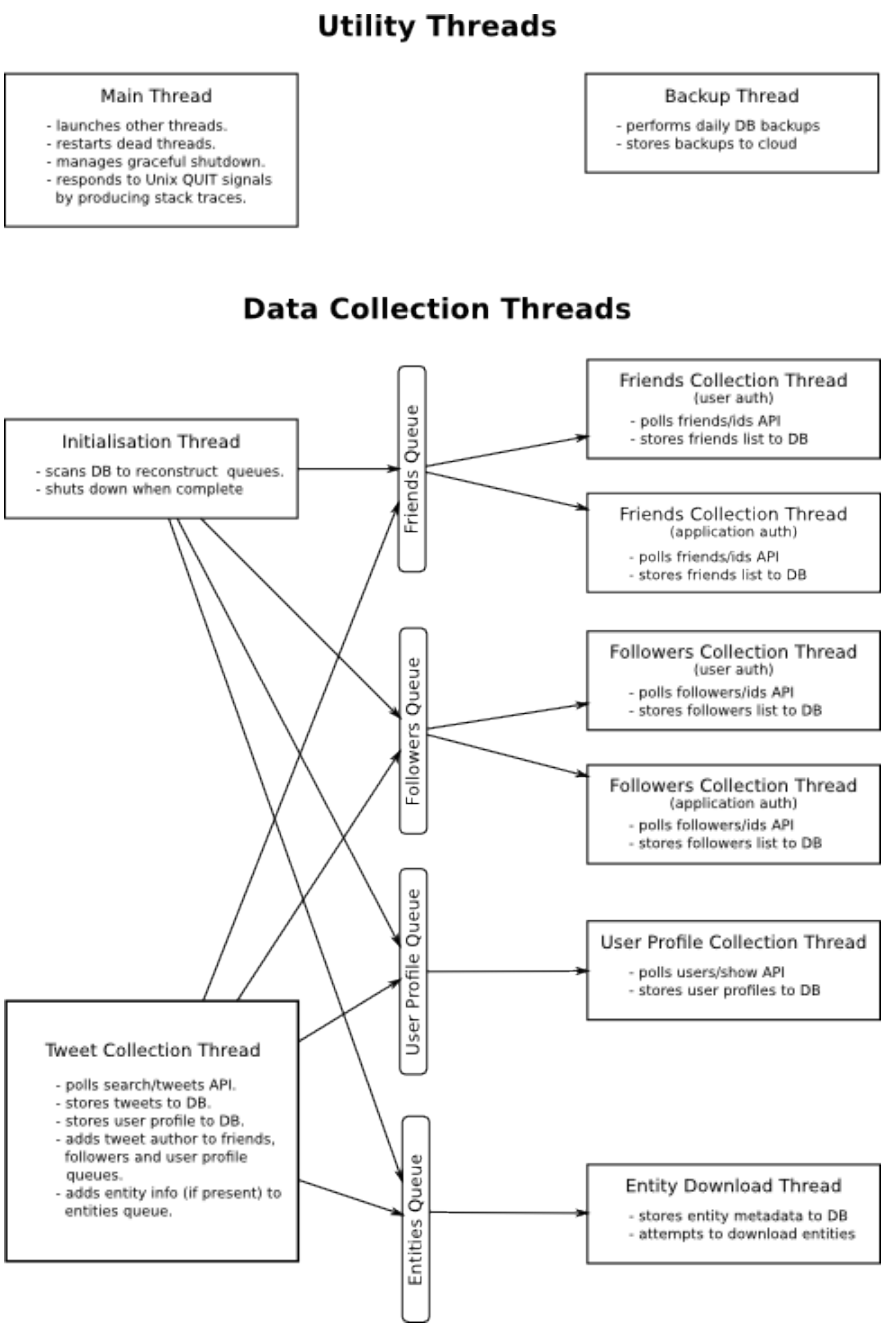

Fig. 1. Overall Architecture. Arrows indicate data passed between threads.

The initialisation thread reconstructs the data collection queues by scanning the tweet and user databases for partial and out of date data (e.g.: friend or follower lists that have not been polled since the last tweet from the user). Once the scan is completed, the initialisation thread shuts itself down. Other data collection threads are launched and run synchronously with initialisation.

The tweet collection thread regularly polls the *search/tweets* API endpoint, collecting all tweets that contain tags from Table 2. Collected tweets and embedded user profile data are stored to the database and then the tweets authors' id is placed in the friends, followers and user profile queues and any media entity meta-data in the entity queue. On system restart, all tweets since the last collected tweet are requested, however twitter does not guarantee that some will not be missed. Our experience indicates that for this query, tweets are usually accessible for several days and down times of that order do not result in lost data. This is probably not the case during heightened tweet rates, however.

Twitter has two modes for accessing its REST APIs: with user authentication and with application only authentication. With application only authentication, you cannot perform tasks on behalf of a user (which is not needed here), but you are given separate API rate limits. For a data collection application such as this, using both forms of authentication essentially doubles the rate limit — in the case of polling friend and follower lists, this is significant. Thus four threads were used for collecting friends/follower information (each of friends or followers with each of user and application only authentication). When storing friend or follower list information, any changes to previously recorded lists are stored in the database and the lists updated. Similarly, when new user profile information is obtained from tweets or collected by the user profile thread, changes are stored alongside the new data.

Data is stored in a replicated MongoDB instance. MongoDB was chosen due to its easy deployment, easily modified schema, easy replication and because the native format of stored data is JSON, the same as that returned by the Twitter APIs. Three main collections are maintained: tweets, user profiles and media meta-data (see Table 3). The Nectar research cloud[6] was used to house database replicas and for reliable storages of database backups.

Collection	**Twitter Data**	**Added Meta-Data**
tweets	tweet data	– how and when it was collected
entities	media entity data	– tweets that contain the entity – a history of any changes to it's data – download attempts/success
user profiles	user profile data	– history of profile changes – friends and followers lists – a history of friend/follower list changes – when and how the data was last polled

Table 3. Database Collections

[6] http://nectar.org.au/research-cloud

4.2 Polling friend/follower lists — the main bottleneck

Tweet rates approximately follow a power law distribution, and the pro-ana/eating disorder query is no exception. Tweet rates remain at a low level most of the time, however occasionally, the rate increases by an order of magnitude, and very occasionally by many orders of magnitude. During these "tweet storms", the download requirements of the system often exceed the Twitter API rate limits, causing the data collection queues to grow. The four public Twitter REST API endpoints used to collect data; tweets by query string, user profiles, user friend lists and user follower lists; all have data rate limits[7] which were on occasion met. Prioritisation schemes were developed to ensure timely collection of more important data.

Collection of tweets for the pro-ana and eating disorder community query fell behind during the highest tweet rates, however due to the ability of the tweet search endpoint to retrieve past tweets, the pro-ana/eating disorder query did not apparently lose data as a result.

The user profile endpoint can poll 100 users per query with 180 queries per 15 minutes. This high rate quickly caught up with even the most extreme "re-tweet storms", and it was sufficient to prioritise users whose previously stored data was oldest (unseen users first, ties resolved by user id).

The API endpoints for friends and followers lists poll only one user per query and 15 queries per 15 minutes. Also, each query returns a maximum of 5000 friend/follower ids — occasional users with millions of followers require hundreds of queries. Frequent 'moderate' re-tweet storms often took days to clear the queue, and extreme events could take weeks. This substantial delay was considered unacceptable.

Investigation of the re-tweet storms indicated that the majority of tweeting users had no other tweets in our data, especially for the more extreme events. Thus a strategy was implemented where users with at least one other tweet in our data were given priority. Of those, the user whose friend/follower data was oldest (i.e.: least recently polled) was given priority. With this strategy, more frequent tweeters were quickly re-polled, while the queue of less interesting one-tweet users can take many days to eventually clear. A newly seen user who tweets again before having been polled is moved to the front of the higher priority queue ('never' is considered least recent). In an attempt to get a snapshot of at least some one-tweet users friend/follower lists at the time they tweet, the most recently seen first-tweet users are polled first. In both priority schemes, rare ties are resolved by lexical order of user names.

User seen before?	Priority	Which user first?
Repeat User	First Priority	Least recently polled
New User	Second Priority	LIFO queue

Table 4. friend/follower lists polling priority scheme

[7] https://dev.twitter.com/rest/public/rate-limits

4.3 Image Collection

Many tweets, and especially tweets in this data, contain images. Twitter includes media URLs in tweet meta-data, and assigns a unique id to each image. When a tweet containing images is collected, the images meta-data is stored in the database including a link to the tweet. If the image with that twitter image id has not been downloaded yet, it is downloaded and stored as a file on disk. A simple cron script is used to backup the stored images to the servers running the database replicas. It is common for duplicate images to be assigned different twitter image ids. The system does not attempt to identify such duplicates.

4.4 Other Technical Challenges

Tweets can be directed to a recipient twitter user. Collecting the friend/follower lists of recipients was also attempted, however it soon became evident that recipients were frequently celebrities with millions of followers, causing extra burden on the already stretched follower API endpoint. Users of interest that are part of the pro-ana/eating disorder community would be tweeting regularly, and we would be polling their friends and followers lists regularly anyway, so it was decided that polling tweet recipients should be abandoned.

To test the relative reliability of the Twitter streaming and tweet search APIs, a separate process received tweets via the streaming API and stored them in an extra database collection. This collection was monitored by the main program, and any extra tweets were copied to the main tweet collection. We found that no tweets would have been lost without the streaming API data, so this part of the system is unnecessary.

During system development, a bug in the python http library and difficulties coordinating thread locks were identified from stack traces generated by the main thread in response to Unix QUIT signals. Early in development, data loss was avoided by automatically restarting the system via an hourly Unix cron script when further bugs triggered by infrequent combinations caused the system to crash. The system has now been running continuously for over a year without any of these problems.

During the first year of data collection, Twitter announced that it was making significant changes to it's APIs, and especially to rate limits and the ways they are reported and applied. The system attempts to utilise it's rate limits as fully as possible without exceeding them (which can prompt Twitter to block the application for a time), so the API changes required substantial adjustment to the rate limit monitoring logic. There were also a few changes to the meta-data for tweets and users. This did not directly require changes to program logic, however in order to keep database consistency, some logic was added to update old format records.

At the time of initial development, Tweepy[8] was chosen for access to the Twitter APIs. Unfortunately, at that time Tweepy did not have support for

[8] http://www.tweepy.org/

application only authentication. Twython[9] did however, and since both packages present the Twitter API in a similar way, it was not difficult to add threads that utilised this capability.

5 Summary of Pro-Ana Data

As of 30 January 2015, the data contained 1,283,875 tweets, 296,483 users and 307,723 image ids. There were 1,616,199 follow events, 1,616,188 unfollow events and 1,655,280 user profile changes. Hash tag usage followed a typical power law distribution, as did the number of followers and friends, though follower and friend counts were not correlated. The number of tweets per user, both in our data and overall also follow a power law.

Hash tags related to "thinspiration" (typically images of people, mostly women) dominate the data, with 73% of tweets. Retweets and images also account for a significant portion, with 57% of collected tweets retweets and 71% containing images. Thinspiration tweets account for 89% of the images, 80% of the retweets contain images and 76% of retweets contain thinspiration tags.

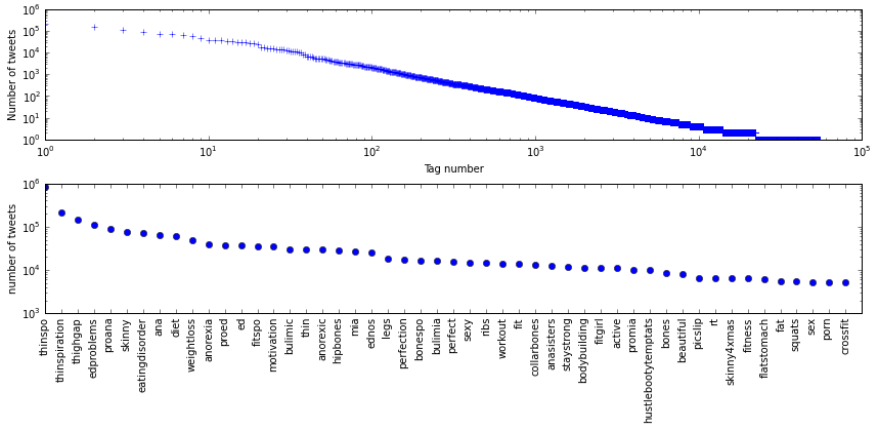

Fig. 2. Tag frequencies (converted to lower case, one mark per tag).

Figure 3 gives some indication of the user profile dynamics captured in the data. In considering if a profile had changed, automatically generated profile attributes such as the number of tweets, followers, favourites, etc. . . were not considered. Changes in *https* versions of profile image URLs were also not considered, as it was noted that Twitter provides these from different web domains depending on the way in which the user information is collected (embedded in a tweet or via the *user/search* API endpoint). It can be seen that for hundreds

[9] http://twython.readthedocs.org/en/latest/

of users, hundreds of profile changes have been captured, and for many thousands of users, tens of changes, thus our goal of capturing profile dynamics has succeeded. Changes to friend and follower lists follow a similar pattern.

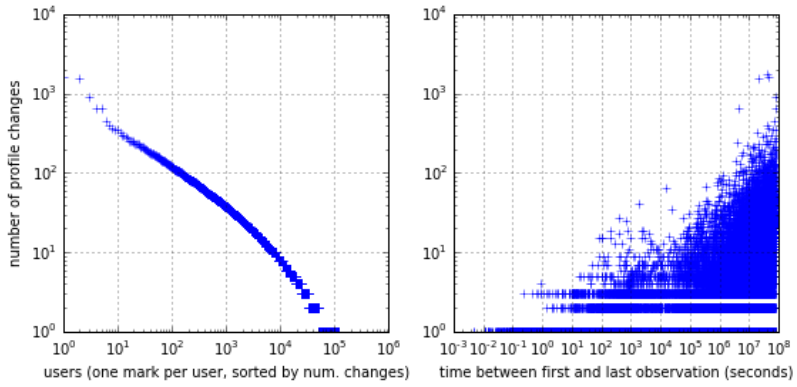

Fig. 3. Number of user profile changes and user observation windows (one mark per user).

6 Conclusion

Capturing data on dynamic aspects of social media communities is important for the study of online social behaviour. Systems designed to capture data from Twitter and other social media typically lack the ability to capture important dynamics, such as changes in the social network. We have constructed a system that captures these dynamics from Twitter communities that can be identified by their use of hash tags or other search terms. The system is very robust to the bursty nature of tweet streams, network problems and other difficulties associated with online data collection.

We present an approach similar to adaptive sampling to identify hash tags relevant to a community. With this approach, we identified a set of tags that is used almost exclusively by the Twitter "pro-ana" and eating disorder community and used our system to collect a nearly unbroken record of tweets, user and network data from that community covering three Christmas periods: over 1.2 million tweets, 300 thousand users and 200 thousand images.

7 Further Work

For a longitudinal study of an online community, it may be appropriate to revise the tweet search query on a regular basis (say, each month or quarter), as tag

usage may change over time, with new tags adopted by the community and old tags loosing favour.

Identification of users of interest (e.g.: apparent members of a social group), perhaps through analysis of the friend/follower, re-tweet and/or user mention networks, could be used to prioritise and enable collection of extra data from those users. For example, regular polling of friend/follower data and user profiles in periods of spare API bandwidth and/or at the expense of timely data collection from less interesting users.

The *search/tweets* REST API is not saturated by the current data collection approach, and substantially more tweets could at times be collected. Tracking tweets by all individuals in the data would quickly become intractable, however it could be valuable to collect more or all tweets by identified interesting users. A simpler strategy of collecting all tweets from users for a certain time since their last tweet could also be valuable.

The final search query from tags in Table 2 represents a balance between a wide net and saturating the twitter API limits. The choice was made to keep the query small in order to maintain a high degree of relevance in the data at the expense of not collecting a small number of relevant tweets. It is an interesting feature of the "pro-ana" phenomena on Twitter that the choice was quite clear, and that little compromise was needed. It would be of interest to investigate other potential Twitter communities to see if their boundaries can be so clearly delineated.

In Section 3 we mentioned the systematic temporal sampling error inherent in the data collection process. A valuable addition to research into social media as a metric for social processes would be the study of this and related sampling errors (e.g.: self selection bias).

Since the creation of this software, Twitter has introduced several new API endpoints that could be integrated into the collection strategies to improve the resolution and fidelity of dynamic, particularly friend/follower network, data. Of particular interest is the *friendships/show* Twitter API endpoint, which returns information about the relationship between two twitter users and has a high rate limit of 180 calls per 15 minutes. Given a technique to regularly identify users of particular interest, their user relationships could be polled more frequently. Another Twitter feature that may be of interest is lists. Users can create and join lists, and use their list membership(s) to filter the tweets that appear in their Twitter feeds or manually view list tweets. List membership of users in the pro-ana data follows a typical power law distribution, with about 40% of users members of some list.

Acknowledgement.

Many thanks to Dr Henry Gardner and Dr Richard L. Jones for assistance preparing this paper.

References

1. Hutto, C., Yardi, S., Gilbert, E.: A longitudinal study of follow predictors on twitter. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 821–830. CHI '13, ACM, New York, NY, USA (2013), http://doi.acm.org/10.1145/2470654.2470771
2. Myers, S.A., Leskovec, J.: The bursty dynamics of the twitter information network. pp. 913–924. ACM Press (2014), http://dl.acm.org/citation.cfm?doid=2566486.2568043
3. Rainie, L.: The six types of twitter conversations (2014), http://www.pewresearch.org/fact-tank/2014/02/20/the-six-types-of-twitter-conversations/
4. Starbird, K., Palen, L.: Voluntweeters": Self-organizing by digital volunteers in times of crisis. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 1071–1080. CHI '11, ACM, New York, NY, USA (2011), http://doi.acm.org/10.1145/1978942.1979102
5. Thompson, S.K.: Adaptive cluster sampling. Journal of the American Statistical Association 85(412), 1050–1059 (1990), http://www.jstor.org/stable/2289601
6. Xu, B., Huang, Y., Kwak, H., Contractor, N.: Structures of broken ties: Exploring unfollow behavior on twitter. In: Proceedings of the 2013 Conference on Computer Supported Cooperative Work. pp. 871–876. CSCW '13, ACM, New York, NY, USA (2013), http://doi.acm.org/10.1145/2441776.2441875
7. Yang, L., Sun, T., Zhang, M., Mei, Q.: We know what @you #tag: Does the dual role affect hashtag adoption? In: Proceedings of the 21st International Conference on World Wide Web. pp. 261–270. WWW '12, ACM, New York, NY, USA (2012), http://doi.acm.org/10.1145/2187836.2187872

8 Appendix— Thread Locks and Events

The system has several resources that are shared between threads: the friends list threads, followers list threads and user data thread all write to the database collection containing user profile information. There is also an object for tracking rate limit status that is shared by all data collection threads. Locks are required to prevent threads from making changes simultaneously and potentially losing or corrupting data.

MongoDB and it's python API are threadsafe, and so long as the same record is not processed simultaneously, we are ok. In order to prevent excessive wait times as other threads process their data, a collection of locks was implemented, keyed by Twitter user id's. In this way, a thread must wait only when another thread is processing the same user id. The lock collection creates a new Lock object each time a new user id lock is requested, and so grows steadily. If we were dealing with millions of user id's, this could be a problem, however for this system it was considered acceptable and no strategy was implemented to remove old, long unused locks.

The system has two routines relating to rate limit status: one to poll the rate limit status and update the internal record, the other to simply query the internal record, but with an option to update it also. Due to the at times inconsistent

rate limit reports from Twitter, a number of heuristics are applied to detect and abandon incorrect reports. To do this, the update method needs to make calls to the query method. Since these methods call each other and potentially make changes to the internal record, a re-entrant thread lock is used (a re-entrant lock can be acquired by a thread multiple times). Note that query method calls from within the polling method never request an update, so infinite recursion is avoided.

Further communication between threads is implemented with events. An event is an object shared between threads which can be set, cleared, read or a thread can wait until the event is set (with an optional time limit). Events were used for two purposes: to request threads to wind up their activities and close down when the system is shutting down; and to inform data collection threads when there is new data in their respective queues.