Can One Tamper with the Sample API?

- Toward Neutralizing Bias from Spam and Bot Content

Fred Morstatter, Harsh Dani, Justin Sampson, and Huan Liu
Arizona State University, Tempe, AZ, USA
{fred.morstatter, harsh.dani, justin.sampson, huan.liu}@asu.edu

ABSTRACT

While social media mining continues to be an active area of research, obtaining data for research is a perennial problem. Even more, obtaining unbiased data is a challenge for researchers who wish to study human behavior, and not technical artifacts induced by the sampling algorithm of a social media site. In this work, we evaluate one social media data outlet that gives data to its users in the form of a stream: Twitter's Sample API. We show that in its current form, this API can be poisoned by bots or spammers who wish to promote their content, jeopardizing the credibility of the data collected through this API. We design a proof-of-concept algorithm that shows how malicious users could increase the probability of their content appearing in the Sample API, thus biasing the content towards spam and bot content and harming the representativity of this data outlet.

Keywords

Data Mining; Data Sampling; Sample Bias

1. INTRODUCTION

Social media has become a very active research area in recent years. One limiting factor for many researchers is the amount of data they are able to collect. Many social media sites strictly limit or outright forbid the collection of data on their sites for research purposes. One major social media site, Twitter, stands out among these sites for its willingness to share data with researchers. Twitter does this mainly through its real-time feeds, called the "Streaming" APIs. There are three main APIs that researchers can access directly: 1) the "Filter API", which gives the user a directed sample based on the input of the user, 2) the "Sample API" which returns a random 1% sample of all public tweets generated on Twitter, and 3) the "Firehose" which yields 100% of all public tweets. While the Firehose seems like the obvious choice for data collection, the monetary cost of using this API as well as the server requirements for hosting the data prevent most researchers from being able to use this option. The Filter API and the Sample API can be used by researchers worldwide to download tweets in real-time, collecting at most 1% of all of Twitter's public statuses, or "tweets" completely free of charge [4].

The Filter API and the Sample API are appealing to researchers, and it is important that researchers understand the underlying mechanisms of these APIs so that they can have confidence in the data collected through them. Biases introduced in the data collection process can propagate to the research results. For instance, bias has been found in Twitter's Filter API. By comparing the results of a Firehose crawl with a Filter API crawl, the authors of one study [4] discover that bias in the way tweets are sampled yields completely different statistics, *e.g.* the top hashtags in the data and the topics in the text.

While evidence of bias has been found in the Filter API, studies on the Sample API have been largely positive, with indications that this data outlet is unbiased. By empirically studying the data, one study [3] found that the results were not statistically significantly different than the results of the Firehose. Another study [2] discovered the underlying mechanisms behind how Twitter samples the data in the Sample API. First, they show the different components that make up the identification number (ID) for each and every individual tweet. A portion of this ID contains the millisecond-level timestamp when the ID was generated. The authors find that the Sample API selects the tweets based on this timestamp, with any tweet whose ID was generated in the millisecond range of [657, 666] to be selected by the Sample API. Twitter, assuming that the millisecond-level information of the ID's timestamp is random, chooses this mechanism to distribute their tweets. This sampling mechanism seems reasonable. Humans don't have millisecond-level control over the tweet's timestamp. The muscle movements to click a mouse or tap a phone screen combined with the network delay make the timestamp nearly impossible for a human to predict at the millisecond-level granularity.

Unfortunately, many of the users of Twitter are not humans, but bots: accounts controlled by computers. Estimates of the number of bots on Twitter range from 5-9% [1]. Many of these bots have malicious intent: using their ability to generate massive amounts of noise to try to skew statistics of the data (e.g. sentiment towards particular topics, top keywords). Bots have the ability to control the *exact* time their tweets are posted, allowing them to control (to a certain extent) the time that Twitter receives their tweet. If bots are able to control if their tweets appear in the Sample API, it could affect the integrity of this stream: bots could

WWW'16 Companion, April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4144-8/16/04.
http://dx.doi.org/10.1145/2872518.2889372.

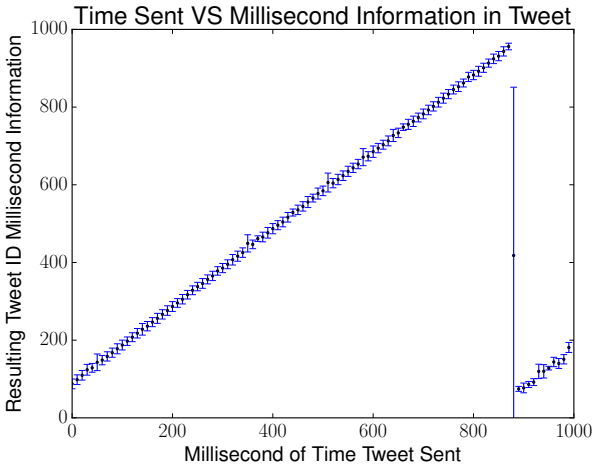

Figure 1: Millisecond of publish time from tweet ID as a function of the millisecond when the tweet was sent. The large error near the 850 mark is an artifact of the modulus property of the reading.

modify the statistics of this data stream, affecting both research and applications that depend on this stream.

In this work we evaluate two questions: 1) "can the millisecond-level information of the timestamps associated with tweet IDs be anticipated?", and 2) "can we leverage this to build software which generates tweets that have a greater chance of being included in Twitter's Sample API?".

2. ANTICIPATING THE SAMPLE API

Here we investigate whether the tweet ID's millisecond-level timestamp information can be predicted. In this first experiment we see how the millisecond-level information of the tweet ID corresponds to the millisecond at which the tweet was sent. To do this, we send tweets at each whole millisecond in the range [0, 990] in intervals of 10. For each tweet we send, we observe the millisecond-level information of the tweet's ID. We extract this using the process outlined in Kergl et. al [2], which identifies specific bits in the binary representation of the ID that contain this info. We repeat this experiment 6 times, and plot the results in Figure 1. We observe a strong linear relationship between the time the tweet is sent and the millisecond ID of the tweet ID. Furthermore, we find that the millisecond delta is consistent with an average delta of 171ms. Using this information we devise an approach that can generate tweets with a greater probability of landing in the Sample API in the next section.

3. MANIPULATING THE SAMPLE API

In the previous section we showed empirically that the millisecond-level timestamp information of a tweet ID is a function of the time the tweet is posted. Using this insight, we design an algorithm that adaptively adjusts the millisecond lag in order to maximize the amount of tweets that appear in the Sample API. The pseudocode for this algorithm is shown in Algorithm 1. The essence of this algorithm is that it adaptively measures the network lag between when the tweet is sent and when the tweet is received by Twitter, measured by the ID of the tweet. We use a moving average of the last w tweets to estimate the lag. Details on how to extract i from the tweet ID can be found in Kergl et. al [2].

To measure the effectiveness of our algorithm, and by extension the possibility to bias the Sample API, we conduct 5 trial runs. In each experiment we produce 100 tweets. We use $w = 3$ in our experiments. We set $m = 661$ as it is the midpoint of [657, 666] and should give us the greatest chance of hitting the window where the Sample API selects its tweets. Also, we set $\hat{\delta}_{init} = 491$ as it is the expected delta from 661 as we discovered in the previous section. We achieve $\mathbf{82 \pm 9}\%$ coverage, contrasted with $\mathbf{1 \pm 0}\%$ coverage when tweets are produced under normal conditions.

Data: w: window size; m: target millisecond; $\hat{\delta}_{init}$: initial delta
$h \leftarrow$ empty list;
$target \leftarrow m - \hat{\delta}_{init}$;
while *true* **do**
  wait until the $target$-th millisecond of the next second;
  post a tweet, t;
  $c \leftarrow$ current time in milliseconds;
  $i \leftarrow$ time in milliseconds from tweet t's ID;
  append $i - c$ to the end of h;
  $\hat{\delta} \leftarrow \frac{1}{w} \sum_{k=|h|-w}^{|h|} h[k] \bmod 1000$;
  $target \leftarrow 1000 \cdot \mathbb{1}(m - \hat{\delta} < 0) + m - \hat{\delta}$
end

Algorithm 1: Maximize the probability of a post being selected by the Sample API. $\mathbb{1}$ is the indicator function.

4. DISCUSSION & CONCLUSION

Twitter's Sample API is a popular tool among researchers in the area of social media mining. The results of this paper show that by timing the creation time of tweets, bots and spammers can have a large impact on the information that is provided through the Sample API, effectively injecting bias into this data outlet. This is a major problem for users who wish to ensure that their data is a representative sample of the real activity on Twitter. The approach presented in this paper is intended to give researchers an understanding that despite recent findings that say that the Sample API is unbiased, results should be taken with care when this data outlet is used in analysis. Furthermore, social media sites can use these results to improve the design of their APIs. API designers can surpass this issue by adding random bits to the timestamp portion of the ID, or by pre-generating IDs before distributing them.

Acknowledgements

This work is sponsored, in part, by Office of Naval Research (ONR) grant N000141410095 and by the Department of Defense under the MINERVA initiative through the ONR N00014131083.

5. REFERENCES

[1] J. Elder. Inside a Twitter Robot Factory. *The Wall Street Journal*, 11 2013. http://on.wsj.com/1Qo215n.

[2] D. Kergl, R. Roedler, and S. Seeber. On the Endogenesis of Twitter's Spritzer and Gardenhose Sample Streams. In *Advances in Social Networks Analysis and Mining*, pages 357–364. IEEE, 2014.

[3] F. Morstatter, J. Pfeffer, and H. Liu. When is it Biased? Assessing the Representativeness of Twitter's Streaming API. In *WWW*, pages 555–556, 2014.

[4] F. Morstatter, J. Pfeffer, H. Liu, and K. M. Carley. Is the Sample Good Enough? Comparing Data from Twitter's Streaming API with Twitter's Firehose. In *ICWSM*, pages 400–408, 2013.