Discovering, Assessing, and Mitigating Data Bias in Social Media

Fred Morstatter*, Huan Liu

Computer Science & Engineering, Arizona State University, Tempe, Arizona, USA

Abstract

Social media has generated a wealth of data. Billions of people tweet, sharing, post, and discuss everyday. Due to this increased activity, social media platforms provide new opportunities for research about human behavior, information diffusion, and influence propagation at a scale that is otherwise impossible. Social media data is a new treasure trove for data mining and predictive analytics. Since social media data differs from conventional data, it is imperative to study its unique characteristics. This work investigates data collection bias associated with social media. In particular, we propose computational methods to assess if there is bias due to the way a social media site makes its data available, to detect bias from data samples without access to the full data, and to mitigate bias by designing data collection strategies that maximize coverage to minimize bias. We also present a new kind of data bias stemming from API attacks with both algorithms, data, and validation results. This work demonstrates how some characteristics of social media data can be extensively studied and verified and how corresponding intervention mechanisms can be designed to overcome negative effects. The methods and findings of this work could be helpful in studying different characteristics of social media data.

1. Introduction

Social media is an important outlet to understanding human activity. Over the last few years many social media sites have given users a way to express their interests, friendships, and behavior in an online setting. Because of their ubiquity, these platforms have been critical in many global events. During the Arab Spring protests, these platforms helped protesters to organize. Across several natural disasters such as Hurricane Sandy, earthquakes, typhoons, and floods, social media has been used both by the affected to request assistance as well as by humanitarian aid agencies to spread information about critical aid

*Corresponding author

Email addresses: fred.morstatter@asu.edu (Fred Morstatter), huan.liu@asu.edu (Huan Liu)
resources. More generally, social media is used by everyday people to discuss current events, and their day-to-day activities. There are several sites with hundreds of millions of users, and a few sites with billions of users, all sharing, posting, and discussing what they see around them.

Noticing the richness, extent, scale, and dynamic nature of social media data, researchers welcome the new opportunities to use social data to answer questions regarding human behavior. Though not all social media sites provide data for research purposes, some sites do provide mechanisms through which researchers can obtain a sample of data to conduct their research. One example is Twitter, a microblogging site where users exchange short, 140-character messages called “tweets.” Ranking as the 8th most popular site in the world by the Alexa rank in August of 2016,¹ the site boasts 313 million monthly users publishing 500 million tweets per day. Twitter’s platform for rapid communication is a vital communication platform in recent events including Hurricane Sandy,² the Arab Spring [1], and several political campaigns [2, 3]. As a result, Twitter’s data has been coveted by both computer and social scientists to better understand human behavior and dynamics. Because of its open nature with sharing data as well as the richness and size of the data generated on Twitter, many research projects use Twitter data to understand human behavior online. This has led to Twitter being called the “fruit fly”, or model organism, for computational social sciences research [4].

![Figure 1: Overview of the process: human behavior becomes data upon which research is conducted. Humans generate data on social platforms (“1”) which is then collected by researchers in order to answer questions about their behavior (“2”). At both steps, there is a potential for bias. In this work we focus on API bias, denoted by “2” in the figure.](image)

While Twitter is an amazing resource for research on social media, there may exist bias during data collection of which researchers should be aware in their study, i.e., whether a representative dataset is obtained for planned computational social science research. If the goal of computational social science is to study society at scale, then the data we study must provide an accurate reflection of society. More specifically, we study whether or not the sampled data that researchers often use for their research is representative of the full, unsampled data on Twitter, Firehose data. If there is bias, we ask if we could detect bias under normal circumstances without the aid of Firehose. In this work, we focus on bias that arises from sampling strategies on social media and

---


potential sources of data bias, and present ways of detecting and mitigating bias in order to draw credible conclusions from sampled and limited data.

2. Related Work

This section consists of two parts that are relevant to this study. The first part is related to existing work on data collection bias and the second part is about the technical details of data gathering mechanisms available at Twitter.

2.1. Data Collection Bias

First, users on a site can introduce bias into a dataset. This corresponds with arrow “1” of Figure 1. This is often done unintentionally by the user base of the site. For example, Twitter’s user base consists mostly of young users [5]. Thus any body of tweets is likely to have a very different age distribution than the age distribution within a country. Partly due to this, we cannot generalize conclusions made from Twitter data without taking these differences into consideration. These demographic biases have been studied previously. For example, in [6] the authors discover key demographic dimensions in which Twitter demographics differ from the demographics of the real world. Addressing the concern about generalizability explicitly, [3] found that using Twitter alone to predict elections did no better than random chance. The authors attribute this poor performance to the demographic makeup of the site.

While the body of genuine users on the site can present bias to those studying the aggregate of their posts, malicious users can also introduce bias into the site. Bots, software-controlled accounts, can work in tandem to change the statistics of the site. They can cause a topic to trend [7] or they can misrepresent already trending topics. Bots can also be used to follow specific user accounts, making those accounts appear more prominent than they actually are [8].

Malicious users are not restricted to bots. In fact, there are many humans who coordinate to damage the reputability of social media sites. One way that this is done is through “crowdturfing,” [9] where humans are hired to perform specific tasks. These tasks often take the form of fake reviews [10], where coordinators organize negative reviews of competing products or positive reviews of their own. Additionally, a new phenomenon of “shills” has appeared on social media. This is where well-trained people disguise themselves on social media, usually to address negative information of a campaign.

In this work we focus on the general area of bias in social media data collection. For this reason we largely focus on arrow “2” of Figure 1 in the rest of the paper. However, we would be remiss to omit some important extensions to this area. For example, some work discusses how one can use expert sampling to surpass the randomness of data collection through APIs [11]. In

---


3
another comparison, research has been carried out to compare Twitter’s Search and Streaming APIs [12]. This allows a deeper understanding of which dataset to use when the Firehose is not available.

2.2. Overview of Twitter’s API Mechanisms

In this work, we use Twitter’s APIs as an example of biased data. We provide a brief introduction to the two APIs studied in this work. Twitter provides several APIs which allow researchers and practitioners to collect data to answer their particular research question. The “Twitter Streaming API”\(^ 5\) is a capability provided by Twitter that allows anyone to retrieve at most a 1% sample of all the data by providing parameters. The sample will return at most 1% of all the tweets produced on Twitter at a given time. Once the number of tweets matching the given parameters eclipses 1% of all the tweets on Twitter, Twitter will begin to sample the data returned to the user. The methods that Twitter employs to sample this data is currently unknown.

2.2.1. The Twitter Firehose

One way to overcome the 1% limitation is to use the Twitter Firehose—a feed provided by Twitter that allows access to 100% of all public tweets. However, the Firehose data is very costly. Another drawback is the sheer amount of resources required to retain the Firehose data (servers, network availability, and disk space). Consequently, researchers as well as decision makers in companies and government institutions are forced to decide between two versions of the API: the freely-available but limited Streaming, and the very expensive but comprehensive Firehose version. To the best of our knowledge, no research has been done to assist those researchers and decision makers by answering the following: How does the use of the Streaming API affect common measures and metrics performed on the data? In this article we answer this question from different perspectives.

2.2.2. Streaming API

Using the Streaming API, we can search for keywords, hashtags, user IDs, and geographic bounding boxes simultaneously. The filter API endpoint\(^ 6\) facilitates this search and provides a continuous stream of Tweets matching the search criteria. The limitation of the Streaming API is that it will return, at most, 1% of all of the tweets on Twitter. When a query stays below the 1%, then the Streaming API can return all of the tweets pertaining to that query. Once the volume of tweets surpasses 1% of all of the tweets on Twitter then the results will be sampled. How this sampling process is carried out is not published by Twitter.

The benefit of the Streaming API is that all of the data returned by it pertain to your query. While other APIs may return more data, much of that

\(^5\)https://dev.twitter.com/docs/streaming-apis

\(^6\)https://dev.twitter.com/streaming/reference/post/statuses/filter
data is irrelevant. With the Streaming API a researcher can get arguably the most relevant data for their research task.

2.2.3. Sample API

The other API we have in this work is the Sample API. Like the Streaming API, data is returned in a streaming fashion as it unfolds on Twitter. Unlike the Streaming API, the Sample API takes no parameters. One simply connects to it and receives a 1% sample of tweets.

The major benefit of the Sample API is that it provides a random sample. While we confirm this statistically in this work, Kergl et al. were able to reverse engineer the Sample API to see how it works [13]. They found that Twitter observes when a tweet is sent to their servers. If that tweet falls within a predetermined 10 millisecond window, then the tweet will be selected for the Sample API. If a tweet arrives at Twitter’s servers while the server clock’s millisecond reading is between [657, 666], then the tweet is provided through the API. Since humans do not have the ability to control the millisecond lag between when they click the “tweet” button to when it hits Twitter’s servers, this reading is considered as random.

3. Detecting Bias in Social Media Data Distribution Mechanisms

To detect bias in Twitter’s distribution mechanisms, we compare a sample of Twitter’s Streaming API data with the unsampled Firehose. For 28 days, we were given access to the Firehose API. During this time we collected data pertaining to Arab Spring activity in Syria. The keywords used were: #syria, #assad, #alepvolcano, #alawite, #homs, #hama, #tartous, #idlib, #damascus, #daraa. The authors also crawled the user SyrianRevo, and drew a geographic bounding box around Syria. Using this data, we assessed how well
the data from the Streaming API matched the Firehose. We report the number of tweets from each source in Figure 2. The number of tweets per day in the Streaming API exceeds the 1% threshold on each day, indicating that some sampling should be in effect for each day.

3.1. Top Hashtag Analysis

Hashtags are an important communication device on Twitter. Users employ them to annotate the content they produce, allowing for other users to find their tweets and to facilitate interaction on the platform. Also, adding a hashtag to a tweet is equivalent to joining a community of users discussing the same topic [14]. In addition, hashtags are also used by Twitter to calculate trending topics, which encourages the user to post in these communities. By tweeting a hashtag, a user is explicitly annotating their tweet for a community.

Recently, hashtags have become an important part of Twitter analysis [15, 16, 17]. For both the purpose of community formation and trend analysis it is important that our Streaming dataset convey the same importance for hashtags as the Firehose data. Here we compare the top hashtags in the two datasets using Kendall’s $\tau$ rank correlation coefficient [18].

Kendall’s $\tau$ of Top Hashtags: Kendall’s $\tau$ is a statistic which measures the correlation of two ordered lists by analyzing the number of concordant pairs between them. Consider two hashtags, #A and #B. If both lists rank #A higher than #B, then this is considered a concordant pair, otherwise it is counted as a discordant pair. Ties are handled using the $\tau_\beta$ statistic as follows:

$$\tau_\beta = \frac{|P_C| - |P_D|}{\sqrt{(|P_C| + |P_D| + |T_F|)(|P_C| + |P_D| + |T_S|)}}$$

where $P_C$ is the set of concordant pairs, $P_D$ is the set of discordant pairs, $T_F$ is the set of ties in the Firehose data, but not in the Streaming data, $T_S$ is the number of ties found in the Streaming data, but not in the Firehose, and $n$ is the number of pairs in total. $\tau_\beta$ ranges from -1 (perfect negative correlation), to 1 (perfect positive correlation).

To understand the relationship between $n$, the length of the list, and the resulting correlation, $\tau_\beta$, we construct a chart showing the value of $\tau_\beta$ for $n$ between 10 and 1000 in steps of 10. This means that we are investigating how correlation plays out as a function of the length of the list. To get an accurate representation of the differences in correlation at each level of Streaming coverage, we select five days with different levels of coverage. The results of this experiment are shown in Figure 3(a). Here we see mixed results at small values of $n$, indicating that the Streaming data may not be good for finding the top hashtags. At larger values of $n$, the Streaming API does a better job of estimating the top hashtags in the Firehose data.

\[\text{We select five days with different levels of coverage: The minimum (December 27th), lower quartile (December 24th), median (December 29th), upper quartile (December 18th), and the maximum (December 19th).}\]
Comparison with Random Samples: After seeing the results from the previous section, we are left to wonder if the results are an artifact of using the Streaming API or if we could have obtained the same results by any random sampling. Would we obtain the same results with a random sample of equal size from the Firehose data, or does the Streaming API’s filtering mechanism give us an advantage? To answer this question we repeat the experiments for each day in the previous section. This time, instead of using Streaming API data, we select tweets uniformly at random (without replacement) until we have amassed the same number of tweets as we collected from the Streaming API for that day. We repeat this process 100 times and obtain results as shown in Figure 3(b). Here we see that the levels of coverage in the random and Streaming data have comparable $\tau_\beta$ values for large $n$, however at smaller $n$ we see a much different picture. The random data gets very high $\tau_\beta$ scores for $n = 10$, showing a good capacity for finding the top hashtags in the dataset. The Streaming API data does not consistently find the top hashtags, in some cases revealing reverse correlation with the Firehose data at smaller $n$. This could be indicative of a filtering process in Twitter’s Streaming API which causes a misrepresentation of top hashtags in the data.

There are some peculiarities with these results. For example, in Figure 3(a), we notice that the performance is not always perfectly correlated with the amount of coverage. The day with minimum coverage outperforms the day with median coverage. This can be attributed to the bias in the datasets. While there is not a perfect mapping from coverage to $\tau_\beta$, there is a strong correlation between the two ($\rho = 0.62, p < 10^{-50}$). This level of correlation rises to $\rho = 0.68 (p < 10^{-60})$ on the synthetic data. While imperfect, there is a strong level of correlation between coverage and performance. This is something that researchers and practitioners should keep in mind when assessing their results.
Table 1: Geotagged Tweet Location by Continent. Excluding boundary box from parameters.

<table>
<thead>
<tr>
<th>Continent</th>
<th>Firehose</th>
<th>Streaming</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Africa</td>
<td>156 (5.74%)</td>
<td>33 (3.10%)</td>
<td>-2.64%</td>
</tr>
<tr>
<td>Antarctica</td>
<td>0 (0.00%)</td>
<td>0 (0.00%)</td>
<td>±0.00%</td>
</tr>
<tr>
<td>Asia</td>
<td>932 (34.26%)</td>
<td>321 (30.11%)</td>
<td>-4.15%</td>
</tr>
<tr>
<td>Europe</td>
<td>300 (11.03%)</td>
<td>139 (13.04%)</td>
<td>+2.01%</td>
</tr>
<tr>
<td>Mid-Ocean</td>
<td>765 (28.12%)</td>
<td>295 (27.67%)</td>
<td>-0.45%</td>
</tr>
<tr>
<td>N. America</td>
<td>607 (22.32%)</td>
<td>293 (27.49%)</td>
<td>+5.17%</td>
</tr>
<tr>
<td>Oceania</td>
<td>54 (1.98%)</td>
<td>15 (1.41%)</td>
<td>-0.57%</td>
</tr>
<tr>
<td>S. America</td>
<td>3 (0.11%)</td>
<td>2 (0.19%)</td>
<td>+0.08%</td>
</tr>
<tr>
<td>Total</td>
<td>2,720 (100.00%)</td>
<td>1,066 (100.00%)</td>
<td>±0.00%</td>
</tr>
</tbody>
</table>

from this API.

3.2. Analysis of Geotagged Tweets

Another aspect of the data we present here is the distribution of the geotagged tweets in the data. We inspect this data as it is a crucial part of analysis of social media data [19, 20]. These geographic signals are immensely important as they tie the message from a user to the actual location where it originated, and it helps us to analyze a geographic region [21].

The number of geotagged tweets is low, with only 16,739 geotagged tweets in the Streaming data (3.17%) and 18,579 in the Firehose data (1.45%). We notice that despite the difference in tweets collected on the whole we get 90.10% coverage of geotagged tweets. We start by grouping the locations of tweets by continent and can find a strong Asian bias due to the boundary box we used to collect the data from both sources. To better understand the distribution of geotagged tweets we repeat the same process, this time excluding tweets originating in the boundary box set in the parameters. After removing these tweets, more than 90% of geotagged Tweets from both sources are excluded from the data and the Streaming coverage level is reduced to 39.19%. The distribution of tweets by continent is shown in Table 1. Here we see a more even representation of the tweets’ locations in Asia and North America.

After looking at the results of the two facets presented here, we see two stark results. While it is tempting to say that the geotagged tweets are not biased because there is such strong coverage of these types of tweets, this is not true. Both of these facets are in fact biased because they are not representative samples of the data. It just so happens that the geographic information is oversampled, which can be an advantage for those that wish to study this type of data. Thus, we cannot say that there is no bias in this data.

In this section we have shown evidence that the Streaming API is biased. This was done with the help of the Firehose, a resource unavailable to mostly all researchers. Thus, it is necessary for us to propose solutions to help those who study Twitter data to do so with the least amount of bias in their data. Going forward, we will propose two approaches that allow for this. First, we
4. Assessing Bias without the Firehose

In this section, we investigate whether another open data source, Twitter’s Sample API, can be used to find bias in the Streaming API [22]. We show that using the Sample API, one can accurately detect bias in the Streaming API without the need of the prohibitive Firehose. We focus on alternative methods to help the user understand when their data diverges from the true activity on Twitter. We continue to show that not only can one find the bias using this method, but that these results are consistent regardless of when and where the Streaming API query was issued.

4.1. Finding Bias in the Trend of a Hashtag

Herein, we propose a methodology that finds the bias in the Streaming API and reports to the user collecting the data when there is likely bias in the data.
With only one unbiased view from the Sample API, it is difficult to understand what we should expect from our Streaming API data. When the results from both sources match there is clearly no problem. When there is a difference how do we know if the relative error between the Sample API and the Streaming API at one time step is significant or if it is just a small deviation from a random sample? To better understand the Sample API’s response, we bootstrap [23] the Sample API to obtain a confidence interval for the relative activity for the hashtag at a given time step.

We begin by normalizing both the Sample API and Streaming API time series. This is done by calculating the mean, and standard deviation of each of the counts in the time series. Finally, we normalize each point by its standard score, which is calculated as:

\[
\text{Standard Score}(t_i) = \frac{t_i - \mu_T}{\sigma_T},
\]

where \( \mu_T \) and \( \sigma_T \) are the mean and standard deviation of all of the time periods in the time series, respectively, and \( t_i \) is an individual time series point. This is done to ensure that the distribution of points from both time series is \( N(0, 1) \).

We create 100 bootstrapped samples for each hashtag from only the Sample API. This is done to give confidence intervals about the variance of the hashtag. Essentially, we are leveraging the same approach we used to detect bias, but we are resampling instead of subsampling. We then extract the time series data from each sample and normalize them as we did before. This gives us a distribution of readings for each time period in the dataset. Next, we compare this distribution to the normalized time series from the Streaming API to detect the bias. We take the sample mean and sample standard deviation of this distribution at each point \( t_i \) as \( \mu^b_i \) and \( \sigma^b_i \). Borrowing the threshold used in control charts [24], we say that any Streaming API value at time \( t_i \) that is outside of \( \pm 3\sigma^b_i \) diverges significantly from what we see in the Sample API, and is likely to be biased.

We show a full example of our method in Figure 4. We enumerate the process for a single hashtag, “#believemovie” on August 5th, 2013. This was obtained by crawling one of the most trending hashtags on that day from the Streaming API. Simultaneously, we collect the data from the Sample API. The process begins with the time series data for this hashtag from both the Streaming and Sample APIs, shown in Figures 4(a) and 4(b), respectively. Looking at the two figures, we immediately see a difference in the trends of this hashtag from the two sources. To obtain a confidence interval on the difference of the two sources, we create 100 bootstrapped samples. The time series extracted from these samples are shown in Figure 4(c). Finally, taking the mean and 3 standard deviations of the bootstrapped samples at each time point, we obtain the confidence intervals seen in Figure 4(d). We make several observations from Figure 4(d). First, a spike that occurs between hours 10 and 11 is underrepresented in the Streaming

---

8The Streaming API was configured to only collect data from that parameter.
Figure 5: Differences in the data returned by each API endpoint. While the Streaming and Sample APIs are both capable of yielding the same amount of data (1% of all of the tweets on Twitter), only the Streaming API is capable of yielding data that is entirely relevant.

4.2. Why Use The Streaming API?

Previously we have discussed how the Sample API is an unbiased sample of the activity on Twitter. This begs the question of why to use the Streaming API at all, when we have an unbiased sample available. The answer to this question is the amount of relevant data that is available through each source. An illustration is shown in Figure 5. While both outlets provide the same volume of data, the amount of relevant data differs drastically. Since the Streaming API is based on a query provided by the user, all of the tweets will match that query. On the other hand, the Sample API returns a random 1% of all of the data on the site. This means that a small amount of the data returned will be related to your query. For example, suppose the analyst is collecting data pertaining to a local event which generates 500 tweets. If they are using the Streaming API, it is possible to get all 500 of these tweets as 500 is significantly less than 1% of all of the tweets on Twitter. If they were using the Sample API, however, they could only expect to get 5 tweets pertaining to the event in their entire collection. This crucial point is what leads most researchers to choose the Streaming API when doing their analysis.

4.3. Evaluation of Our Approach

In the previous section we showed how our approach could be used on real-world data to root out periods of bias in our data. In this section we use traditional metrics for machine learning algorithms to assess the efficacy of this approach for finding bias in sampled Twitter data.

First, we compare the results of the Sample API with the Firehose to understand how well the data from the Sample API can be used to recover the data from the Firehose. This is done to show how well we can recreate the Firehose based only on the Sample API. We scale the data from the Sample API by a factor of 100 in order to show the trends on the same scale. The results of this are shown in Figure 6. There are two important results from this figure.
Figure 6: An overview of our evaluation approach. As articulated in Section 4.1, the Sample API (dotted green line) can be bootstrapped to give a 99.7% confidence interval. Whenever the Streaming API (dashed black line) falls outside of these confidence intervals, we say that it is biased. Here we see how this predictor fares in comparison to the Firehose data from [25]. The Sample API is obtained completely independently of the Firehose.

Figure 7: Top hashtags from each data source for two representative days. Bold are tags in the Firehose that are not represented in the Streaming API.

First, that the Sample API tracks the Firehose very well, and second that three standard deviations from the mean seems to contain the Firehose line. Using these observations, we compute further accuracy measures going forward.

While we have shown that the bootstrapping method can track the Firehose information well, it is important to show that it is also revealing better time periods than those that are outside of the bounds. To give an indication of how this is affected, we show two examples of the top hashtag lists: one on December 19, when the coverage of the Streaming API is very close the Firehose, and another on December 27, when the coverage is significantly less. The results of this experiment are shown in Figure 7.

4.4. The Problem of Data Sparsity

One potential drawback of our method lies in the data sparsity of the Sample API. Accounting for only 1% of the Firehose, the “long tail” of hashtags will
largely be ignored by the Sample API. This is problematic for researchers who wish to verify their Streaming API query’s results when their query is focused upon hashtags that do not see a lot of activity as a fraction of the entire activity on Twitter. One counterargument is that these kinds of queries will likely not eclipse the 1% threshold, and in general will be unbiased.

One observation we make is that the times where our bootstrapping method will be futile is in times where there is data for the hashtag from the Streaming API and no data from the Sample API. In such cases, a bootstrapping approach will give us a degenerate distribution with mean 0, not allowing for meaningful comparison between the sources. We test the sparsity of the Sample API by finding “known zeros”, hashtags seen in the results of the Streaming API query, but not in the Sample API for a particular time unit.

Figure 8 shows the number of known zeros in the top 1,000 hashtags, with each hashtag ordered by frequency. Here, we see that the first hashtags are nearly perfect, with a total of only 4 known zeros in the top 10 hashtags. However, as we continue down the list, we begin to see more and more known-zeros. While this method helps researchers to find bias in their Streaming API queries, there are still many hours for many hashtags where no claim can be made about the validity of the data.

4.5. Geographic and Temporal Stability of Queries

In addition to tackling the bias problem, we also analyze the stability of the results when the data are collected in different geographic areas, and for queries started at different times. Do identical Streaming API queries started at different times get different responses during the overlap of their execution? Do identical Streaming API queries get different responses if they are executed from different geographical regions? To ensure that the results obtained in this paper hold for researchers outside of the US, we assess whether Twitter issues the same results to identical queries.

To answer these questions, we collected data from the Streaming API in both the United States (USA) and Austria (AT) with the following scheme: every 20 minutes, we start a query that lasts for 30 minutes. For example, query^{USA}_{1}
Table 2: Number of Comparisons, Median, Average, and Standard Deviation of Twitter ID Jaccard Scores across all comparisons. Because the temporal comparisons are between query, we have one less than in the geographic comparison.

<table>
<thead>
<tr>
<th>Comparison</th>
<th>N</th>
<th>Median</th>
<th>Mean</th>
<th>STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geographic</td>
<td>194</td>
<td>0.976</td>
<td>0.941</td>
<td>0.092</td>
</tr>
<tr>
<td>USA Time</td>
<td>193</td>
<td>0.996</td>
<td>0.995</td>
<td>0.003</td>
</tr>
<tr>
<td>Austria Time</td>
<td>193</td>
<td>0.996</td>
<td>0.942</td>
<td>0.186</td>
</tr>
</tbody>
</table>

and $\text{query}_{1}^{\text{AT}}$ collect tweets from 00:00 - 00:30 UTC, $\text{query}_{2}^{\text{USA}}$ and $\text{query}_{3}^{\text{AT}}$ from 00:20 - 00:50 UTC, and $\text{query}_{1}^{\text{USA}}$ and $\text{query}_{3}^{\text{AT}}$ from 00:40 - 01:10 UTC, and so on. Each query is configured with exactly the same parameters. In structuring our queries this way, we can control both for time, and for location. By looking at the 10-minute overlaps in the adjacent within-country queries (i.e. all $\text{query}_{i}^{C}$ and $\text{query}_{i+1}^{C}$), we can gain an understanding of whether identical queries started at different times get the same results. By looking at entire queries across countries (i.e. $\text{query}_{1}^{\text{USA}}$ and $\text{query}_{3}^{\text{AT}}$), we can understand the difference between identical queries started at the same time from different geographic locations.

The data spans from 2013-10-20 06:20 UTC - 2013-10-22 22:20 UTC. Each query starts exactly at each 20-minute interval and lasts for exactly 30 minutes. We collect 194 datasets in total from each country. In the between-country case, we compare the entire dataset as both queries are running in both locations. In the between-time case, we compare the overlaps between $\text{query}_{i}^{C}$ and $\text{query}_{i+1}^{C}$.

4.5.1. Between-Country Results

To compare the datasets, we calculate the Jaccard score of the Tweet IDs from $\text{query}_{i}^{\text{USA}}$ and $\text{query}_{i}^{\text{AT}}$. We then take the median, average, and standard deviation of these Jaccard scores. These results are reported in the first row of Table 2. Here, we see a very high average and a very low standard deviation between the Jaccard scores, indicating that the results from these two queries are nearly identical. These results indicate that no preference was given to the queries originating in the United States.

4.5.2. Between-Time Results

To compare the datasets, we fix the country $C$ and calculate the Jaccard score of the Tweet IDs from $\text{query}_{i}^{C}$ and $\text{query}_{i+1}^{C}$. The results for the USA and Austria queries are shown in rows 2 and 3 of Table 2, respectively. In the case of the USA, we see even stronger results, with an extremely high average and an extremely low standard deviation. Here, we can see that the overlapping times receive practically the same dataset in all cases. In the case of the Austrian datasets, we see that there is a wider distribution of Jaccard scores between query windows, however we continue to see an extremely high mean, which gives us confidence in the coverage in these results.

In this section we have shown how existing data can be corrected in order to remove the most bias possible from a pre-collected dataset. However, it may be possible for us to collect our dataset in such a way that the amount of
bias is minimized. In the next section we will investigate such data collection approaches.

5. Mitigating Bias in Social Media Data Retrieval

We have presented an approach to root out bias in a pre-collected dataset. In this section we investigate if we can prevent data bias in our data collection by increasing the coverage in our crawl. Bias can be minimized, making it possible to make better predictions based on principled statistical and machine learning techniques. In the absence of the ability to measure and correct for sources of bias, the only available recourse is to ensure that the coverage of the gathered sample is as close as possible to the total sample population during the gathering process. However, in order to measure the difference between a sample and the complete set, a useful population measure is required. Towards this, we discuss approaches that can prevent this type of bias from entering the dataset, such as those in [26].

The Twitter Streaming API uses a mechanism called “limit track” which informs the user of the number of tweets that were not delivered to them due to exceeding the 1% rate limit. The limit track data is provided periodically along with tweets delivered in a particular stream. Unfortunately, if the limit track value provided by Twitter is not a reliable measurement, then it becomes significantly more difficult to determine the overall sample population, and, as a result, the level of bias in the sample remains unknown. In addition, the usefulness of the limit track value is further reduced as it does not allow for any method to retroactively obtain the lost data. Unfortunately, since the method used for sampling tweets, as well as how the limit track value is obtained, is not yet published, it is imperative to know whether Twitter’s limit track is accurate and, if it is not, we must find another way to decide if information is being lost.

We motivate our approach with the example seen in Figure 9. On the left, we have #occupywallstreet, which takes up 1.4% of all of Twitter. Crawling this hashtag alone will cause us to be sampled, as it takes up more than 1% of all of the data on Twitter. Investigating the hashtags that co-occur with
#occupywallstreet, we see that #zuccottipark takes up 0.6% of Twitter (and
43% of #occupywallstreet), and #nypd takes up 0.8% of all of Twitter
(and 57% of the #occupywallstreet). Running two independent crawlers, one
for #zuccottipark and another for #nypd, will allow us to collect all of the data
for #occupywallstreet. This example provides an overview of our approach.
Details about how we handle special cases such as overlapping keywords and
tweets containing only the target hashtag are explained in subsequent sections.

In this section we propose an approach which splits the keywords so that
we can maximize the coverage of the approach. We hypothesize that this will
eliminate bias for two reasons. First, our findings indicate that, in general, more
coverage indicates less bias in the Streaming API. This approach will decrease
the amount of bias in the whole simply because we have more data. Secondly,
we believe the resulting dataset will be less biased because each individual query
will be designed to stay below the 1% threshold, meaning each individual query
should not be biased due to the API mechanisms.

5.1. Keyword Splitting Approaches

The Twitter Streaming API allows anyone to gather real-time data from
Twitter by specifying a set of parameters that include search terms, user names,
or location boxes. In the case that the search terms specified for a stream
surpass the 1% limit, the API informs the user of the total number of tweets
missed since the streaming connection began. Ideally, this would give the user
a quantitative measure of the overall sample size for their search. The total size
of the dataset would then be the sum of the number of unique tweets gathered
added to the limit track value provided by Twitter. Knowing the exact quantity
of missing data is of paramount importance when it is necessary to adjust the
data gathering method to account for gaps in the sample.

The Twitter limit track is designed to give a measurement of lost data for
a single stream. However, our proposed methods revolve around using multiple
streams to increase the effective sample size. In order to determine if the limit
track is a useful measure for the overall sample, when viewed from the context
of multiple streams, we ran a number of trials simultaneously. At the first level,
all keywords used in the search were tracked using a single stream. For each
level beyond the first, the keywords were separated as evenly as possible among
the streams. In the example shown in Figure 10, all keywords are split between
crawlers based on the number of crawlers required at each level. All split levels
are run simultaneously up to a maximum split level of five which required a
total of fifteen streaming clients. Since no keywords were added or duplicated
between the streams, the total number of possible tweets should be equivalent to
the unsplit streams number of caught tweets as well as the reported limit value.
However, in nearly every experiment, there was always a number of splits that
would result in a larger number of unique tweet IDs than should be possible
according to limit track. As shown in Figure 10, we accumulated 107.3% of
the tweets that were indicated by the limit track, meaning that we received
more tweets than were estimated by Twitter. Furthermore, using a four-split
Figure 10: Using a single crawler it is possible to gather tweets from at most 400 keywords. As can be seen, the rate of tweets caught remains stable for a single crawler. Splitting the same keywords across multiple crawlers results in immediate improvement in the number of unique tweets caught as well as allowing the sampled population to go beyond the population size indicated by Twitter.

Table 3: Impact of Additional Crawlers on Sample Coverage. Since multiple crawlers may have overlap in the tweets that they do not receive, it is not possible to determine the number of unique tweet IDs missed across each crawler - we use N/A when this is the case.

<table>
<thead>
<tr>
<th></th>
<th>Caught</th>
<th>Missed</th>
<th>Total</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsplit</td>
<td>3632</td>
<td>4488</td>
<td>8120</td>
<td>44.7%</td>
</tr>
<tr>
<td>2-split</td>
<td>5060</td>
<td>N/A</td>
<td>N/A</td>
<td>62.3%</td>
</tr>
<tr>
<td>3-split</td>
<td>8714</td>
<td>N/A</td>
<td>N/A</td>
<td>107.3%</td>
</tr>
<tr>
<td>4-split</td>
<td>11143</td>
<td>N/A</td>
<td>N/A</td>
<td>137.2%</td>
</tr>
</tbody>
</table>

approach, we collected 137.2%. Thus, limit track is not an accurate estimation of the total number of tweets.

In order to get the most tweets in a time period, we run multiple crawlers. Given a list of keywords, the Twitter streaming API will return every new tweet that contains any of the words specified. Therefore, by splitting the keyword list among multiple crawlers it becomes possible to gather tweets beyond the 1% limit. Under perfect conditions, each additional stream increases the effective limit rate by 1%. Unfortunately, when partitioning keywords, it is important to keep in mind the possibility of overlap between the streams. For example, a tweet that contains keywords that were split between a number of crawlers will be duplicated in each stream. Tweet duplication in this manner reduces the overall efficiency of each stream. The stream splitting methods must be able to account for and minimize the potential for overlap between keywords.

In order to gather samples closer to the population size, we propose and evaluate three different splitting algorithms - each with varying characteristics for
the initial rate of growth and growth stability as additional crawlers are added. In this work we introduce only two such methods. An in-depth description of these methods can be found in [26]. Here we present a Spectral Clustering based approach to this task, with a simple baseline to demonstrate its efficacy.

5.2. Coverage Experiments

Each of the following experiments is designed to test the efficiency of the given splitting method in obtaining a sample closer to the total sample population than is possible with the standard single stream method. The key factors that we will focus on include: speed of initial growth with a small number of crawlers, how stable the splitting method is for increasing growth as additional crawlers are added, and how many crawlers are required before we pass the population size estimation from Twitter.

In each of the experiments, we drew from a pool of twenty-one streams. This allows us to use a single stream with all possible keywords as a baseline measure for standard gathering rate and population estimation with the limit track. The remaining twenty streams are then used for performing keyword splitting up to twenty ways. Each of these streams was able to track up to 400 keywords, the maximum number of keywords allowed by Twitter in any given stream. The keywords used were chosen by taking the most frequently occurring keywords from the Twitter sample stream. These keywords contained a broad spectrum of topical words as well as multiple source languages. Keywords were chosen in this manner to ensure a high data rate from Twitter and represent a worst possible scenario for keyword splitting since a single keyword represents an atomic element that can not be further split.

5.2.1. Round Robin

The round robin method of stream keyword splitting is an effective baseline. Sampling the amount of tweets gathered and missed at each split level requires running one background stream that contains all selected keywords as well as k additional streams that contain the keywords split between each stream. While it is possible to sample all split levels simultaneously, the number of required accounts for a test of this type is

$$x_k = \frac{k(k+1)}{2}$$

where k is the number of split levels and x is the number of accounts. Sampling all splits up to a split level of 7 would require 28 separate account which is unfeasible for our purposes. Additionally, the processing power required to maintain the set of unique tweet IDs for each stream becomes problematic very quickly. Alternatively, using a single background stream that contains all keywords and comparing the results to each split level independently requires a much lower number of accounts. It is this latter method that we use for each stream splitting analysis. At the completion of the priming stage, the word pairs, from the most frequently occurring to the least frequently occurring, are assigned to streams in a round robin fashion. Each split level runs for 30 minutes before resetting all streams, including the background stream, and beginning the next split level. Resetting the background stream is key to analyzing each stream level in this method as
it allows a comparison of the difference between a single stream and multiple split streams over a given window of time and thereby making it unnecessary to maintain all data for every level of split at once.

The graph in Figure 11 shows that we were able to eclipse the limit track maximum by 12 splits at which point we were able to gather six times as many unique tweets containing proper keywords than was possible with a single stream. Reaching 20 split levels nearly doubled the number of unique tweets gathered over the maximum indicated by Twitter. The algorithm for constructing round robin splits can be found in Algorithm 1.

**Algorithm 1**: Round Robin Splits Construction

```plaintext
Data: graph G, num_clusters k
Result: array of lists
for node v in G(V, E) do
    keywordList := keywordList + v
end
sortedList := Sort(keywordList by occurrence rate);
for word, index in sortedList do
    listNum := index % k;
    assign word to split list k
end
```

5.2.2. Spectral Clustering

Spectral clustering directly leverages the word occurrence graph. Unlike K-Means, spectral clustering requires an affinity matrix in order to create accurate clusters based on how the items relate to one another. This clustering method allows us to define a number of clusters, and the spectral clustering algorithm will incorporate the nuances of the similarity between the items in order to
Figure 12: The results of Spectral Clustering do show an increase of sample coverage overall. However, the clustering creates unbalanced splits where one stream, while still a good cluster, may contain significantly more words than others. The lack of balance manifests through instability in the rate of gain from each additional crawler.

improve cluster results. Like most clustering algorithms, spectral clustering does not make any guarantee on the size of each cluster.

Spectral clustering offers several properties. First, as the number of requested clusters increases, each cluster tends to become languages. This is a favorable trait for reducing overlap since keywords of differing language should have a low rate of word overlap. Secondly, cluster size is largely nonuniform. Though this behavior is preferable for most applications that employ clustering, a significant difference in the size of clusters causes some streams to significantly outperform the 1% limit and become severely rate limited while streams based on smaller clusters fail to reach a limiting rate at all. This effect can be seen in Figure 12. Clustering based on word occurrence quickly passes the Twitter limit with only 6 streams active but shortly thereafter struggles to gain much ground. Wild fluctuation can be observed between each split level and while there is overall growth it is possible to gather a smaller sample with a larger split level. Such inconsistencies were not observed in Figure 11 further indicating how detrimental sensitivity to cluster size is when considering methods for gathering tweet samples. Spectral clustering based splits can be found in Algorithm 2.

We have now proposed two solutions for mitigating bias in the Streaming API. So far, the bias we have investigated comes from the way Twitter distributes its data. This is not always the case. In fact, there are many other sources of bias common to social media data. Another source is the bias that is introduced by malicious users spreading content on social media. In the subsequent section, we will discuss how these two phenomenon can be combined in order to change social media content. We will discuss how malicious accounts
Algorithm 2: Spectral Clustering Split

can leverage the API mechanisms to poison the data from the Sample API.

6. Bias from API Attacks

Thus far we have discussed whether or not the APIs are biased. The discussion has been largely discussed from the perspective that the underlying mechanism of the API may be flawed in some way, which causes the API to yield data that is not representative of the true activity of Twitter. In this section we present another possibility: that malicious users themselves can contribute to this bias. This can come from the way the API selects its data.

One API that is susceptible to user manipulation is the Sample API. Its sampling mechanism is based on the millisecond timestamp of the tweet, as determined by [13]. This timestamp is ultimately determined by the users sending the tweets and could be leveraged by automated accounts to inject tweets into the API. In this section we discuss this possible vulnerability. We investigate whether the timestamp of the tweet can actually be predicted by a user account and then proceed to see if we can actually manipulate whether a tweet will be selected by the Sample API.

6.1. 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 [13], 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 13. 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.

6.2. 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 [27]. The pseudocode for this algorithm is shown in Algorithm 3. 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 extracting $i$ from the tweet ID can be found in [13].

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 $\delta_{\text{init}} = 491$ as it is the expected delta from 661 as we discovered in the previous section. We achieve $82 \pm 9\%$ coverage, contrasted with $1 \pm 0\%$ coverage when tweets are produced under normal conditions.

7. Conclusions

Social media data has become an effective tool and a novel lens for understanding society at scale. Because of its willingness to share a portion of its data
Algorithm 3: Algorithm to maximize the probability of a post being selected by the Sample API. \( \mathbb{1} \) is the indicator function, which returns 1 if the condition is met, and 0 otherwise.

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

with researchers, Twitter has become one of the foremost studied platforms for computational social science research. With so many researchers and practitioners relying on these data sources for their research results, it is important that be aware of the potential bias of various sources in our data collection effort. In this work, we discuss different sources of bias, and introduce strategies for correcting them.

We begin by discussing the bias due to the way Twitter shares its data through the Streaming API. This API provides, at most, a 1% sample of the activity of Twitter based on a query supplied by the crawler. We enumerated different facets of the data returned by the API, and discover that there are key differences in the way this API returns data. Differences include the most frequent hashtags, as well as topic models trained on the data. We propose a method which can identify periods of bias in the dataset. This method is generally applicable to any Twitter data to provide an assessment of data bias. We then look at the Sample API, another commonly-used source in search of a feasible approach to bias detection without the Firehose. Next, we propose a strategy of mitigating the bias by increasing the coverage of the Streaming API. Twitter’s Sample API is a popular tool among researchers in the area of social media mining. By inspecting the output of the API, we show that it is possible to time the creation time of tweets, which makes it possible for bots and spammers to attack the Sample API, in the sense that one can effectively inject 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. Our findings suggest that the Sample API is prone to attacks and researchers should be aware of this potential for attack, though in general the Sample API is unbiased. Furthermore, social media sites can use these results to improve the design of their APIs to prevent attacks. API designers can surpass this issue by adding random bits to the timestamp portion of the ID, or by pre-generating
IDs.

There are many ways in which social media can be biased. This work demonstrates that bias can be detected and mitigated by focusing on Twitter data and its APIs. The methodology designed and illustrated in this work can serve a starting point for similar tasks in the age of social media.

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.


