Scientific Research Publishing

Journal of Computer and Communications, 2023, 11, 172-193
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

GatherTweet: A Python Package for Collecting Social Media Data on Online Events

Claudia Kann[1], Sarah Hashash[1], Zachary Steinert-Threlkeld[2], R. Michael Alvarez[1]

[1]Department of Humanities and Social Sciences, California Institute of Technology, Pasadena, California
[2]Luskin School of Public Affairs, University of California Los Angeles, Los Angeles, California
Email: ckann@caltech.edu

How to cite this paper: Kann, C., Hashash, S., Steinert-Threlkeld, Z. and Alvarez, R.M. (2023) GatherTweet: A Python Package for Collecting Social Media Data on Online Events. *Journal of Computer and Communications*, **11**, 172-193.
https://doi.org/10.4236/jcc.2023.112012

Received: December 7, 2022
Accepted: February 25, 2023
Published: February 28, 2023

Abstract

Social media plays a crucial role in the organization of massive social movements. However, the sheer quantity of data generated by the events as well as the data collection restrictions that researchers encounter, leads to a series of challenges for researchers who want to analyze dynamic public discourse and opinion in response to and in the creation of world events. In this paper we present gatherTweet, a Python package that helps researchers efficiently collect social media data for events that are composed of many decentralized actions (across both space and time). The package is useful for studies that require analysis of the organizational or baseline messaging before an action, the action itself, and the effects of the action on subsequent public discourse. By capturing these aspects of world events gatherTweet enables the study of events and actions like protests, natural disasters, and elections.

Keywords

Data Science, Movements, Social Media Data, Twitter, Network Science, Data Mining, Python

1. Introduction

The spread of social media has transformed the world in recent years. Individuals are now able to easily connect to others who are thousands of miles away, creating online enclaves separated from their physical location. While this interconnectedness creates online communities based on shared discourse, many events such as protests, natural disasters, and elections maintain clear geographic foci. Moreover, these geographic foci are dynamic: a protest movement spreads, a hurricane travels, and presidential primaries occur across the country. This paper introduces the package gatherTweet to follow the temporal and geo-

graphic progression of offline events made up of discrete activities (a particular protest march, cities the hurricane hits, or election days). Many individuals still use their social media to report on, publicize and organize the offline events they participate in. Thus gatherTweet provides efficient data collection to determine individuals' broader event-related online community as well as the subset of those in their geographic vicinity. These data enable researchers to gain insight into the evolution of both the discourse network structure and content as an event progresses.

The package gatherTweet addresses a series of challenges facing researchers collecting social media data. Every collection effort must heed rate limits, manage storage space, and make choices about how to identify relevant users and conversations. The implementation of these choices is time consuming and requires advanced programming knowledge. This package circumvents many of these challenges, by enabling the user to:

1) Create data structures that represent events that are geographically and temporally dispersed. Specifically, gatherTweet creates codex "TwitterEvent" and codex "TwitterActivity" objects which can be populated manually, through importing a specialized .csv, or by reading existing directory structures.

2) Identify individuals[1] on Twitter who participated physically in event activities.

3) Identify individuals who echoed the sentiments of and influenced individuals around the event.

4) Analyze the Twitter presence of these groups of individuals over time.

The rest of the paper proceeds as follows. First, we present a brief literature, with a focus on how our approach differs from existing packages and approaches. The general use case of gatherTweet is then discussed in conjunction with the methodology it employs. This is followed with an illustrative example using gatherTweet to track the evolution of the Black Lives Matter discourse in three separate cities during the summer of 2020. The use of Twitter as a platform is then discussed, in terms of the limitations it imposes, how gatherTweet circumvents them, and the potential to use similar methods on different social media applications. This is followed by suggestions of ways in which gatherTweet could be deployed and an explanation of the contribution gatherTweet makes to the overall Twitter-related software suite in Python.

2. Literature Review

Social media data serve as the foundation for many studies of public discourse because they provide researchers with an inexpensive, immediate, and accessible way to gather large swaths of information. They have proved especially fertile for studying the interaction of offline and online activity [1] [2] [3] [4]. Yet,

[1]Here we refer to each Twitter account that gatherTweet collects as an *individual*. The package makes no attempt to differential accounts based on whether they are individual entities, organizations, or bots. Rather we leave it to users who may want to differential individuals, organizations or bots to further investigate the accounts found to sort them into account types, or to use bot classifiers to detect potential bot accounts.

projects' data collection methods are usually tailored to their precise event, forcing subsequent researchers to spend effort building the same tool instead of analyzing data and writing. Few of these papers present user-friendly packages or tools to enable the replication of their analysis methods for applications other than the particular one in the study. Packages that do exist, mostly Twitter API wrappers, tend to be extremely versatile. This versatility, however, leads to an increased required programming knowledge to take advantage of them. gatherTweet straddles the space between Twitter API wrappers and individual use case code.

The package develops a general methodology and implements that methodology in the package. Previous work in this area has heavily emphasized methods which can be used to inform data collection from Twitter. These include building keyword or hashtag searching pipelines [5] [6] or approaches using keyword expansion [7] [8]. Other social media data collection efforts prioritize the streamlining the data collection process and storage [9] [10]. In general, these methods have not been implemented in easy-to-use packages and researchers who are not comfortable with complex programming may be unable to take advantage of the knowledge shared. The methodology presented in gatherTweet uses geographic information as a tool to pinpoint relevant actors as well as reduce the quantity of data collected for movements that have prolific online discussions. Thus, gatherTweet implements a generalizable methodology that takes into account geographic information, filters the data to a tractable size, and is easy to use.

There are numerous Python packages such as tweepy [11], twitterAPI [12], and python-twitter [13], all of which provide helpful wrappers for the Twitter API. These packages are versatile and have been used widely in other studies; in fact, gatherTweet is an implementation using codex TwitterAPI. However, their implementation is difficult for many users because each requires detailed knowledge of Python and the Twitter APIs. We have designed gatherTweet to be easy-to-use for a wider range of researchers, and have implemented it as a data collection method that has broad applicability in social media analysis.

3. Method

While various steps of this methodology can be done independently, both using gatherTweet as well as replicated for other social media platforms, in this paper we assume that the user will be implementing the package using all of its capabilities. As a broad overview, gatherTweet first identifies the most central and active participants in each activity within the event (users who are part of the *core*). This is accomplished by identifying individuals who are both discussing the event online and whose geo-location suggests their involvement in the physical activity. A directed one-step crawl is then implemented to find users who retweet the core. These users are labelled *echos* as they propagate the online content of the core. In addition, the users whom the core retweets are found in a

similar manner. These users, *influences*, create content that the core disperses.

These three groups of individuals make up the *discourse network*. In general, the discourse network does not invite the use of typical network analysis tools, as the connections between echos and influences are unknown. However, there are studies that can be done more generally on the discourse network's structure. For example, structural knowledge can answer research questions like whether individuals participate in multiple activities, if their roles (core, echo, or influence) are consistent over activities, and whether individuals associated with different activities have similar tweeting behaviors. The content of conversation could also change across an event's activities [14] [15]. Following the initial discourse network collection, gatherTweet can access the full timeline of individuals who are part of the activity. This creates a panel study for each individual, it is similar to each of them filling out multiple surveys over the period, so the researcher can search for individual level patterns. Individuals' tweet history can help answer questions about how they act over time (previous to, during, and following the activity they are associated with) and whether individuals participate in multiple offline activities, for example if individuals participate in multiple protests, are impacted by weather events at multiple stages or vote in multiple election [16] [17].

The full methodology can be seen in **Figure 1**. During preliminary event specification, the researcher determines the activities of an event. The next step, individual level identification, has three components, one for each type of individual. First, individuals participating in the event activities both online and offline

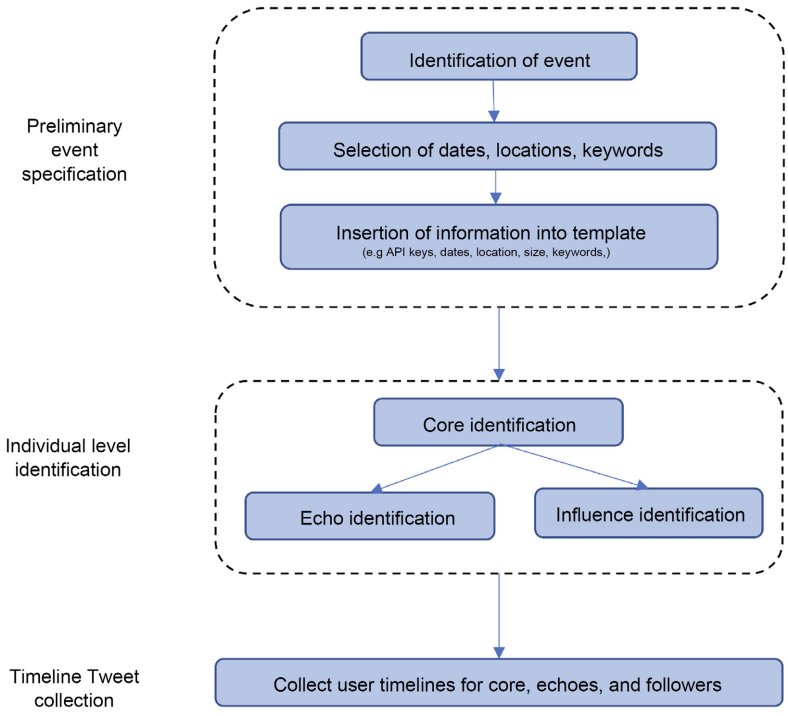

Figure 1. Proposed data collection workflow.

are identified. These individuals are the core for each event activity. An individual could belong to the core of multiple activities within an event, however these core groups are kept separate by activity. Such users are recorded multiple times, with each instance considered a unique user. In the second component of individual level identification, individuals who members of the core retweet are found and labeled as influences, because they likely influence the views and tweets of the core. In the third component, a list of users who retweet the core is created, and these users are labeled echos since they echo the tweets of the core. These three groups make up the discourse network of Twitter users associated with each event. Finally, in timeline tweet collection, the full timeline is collected for all three types of individuals. The researcher must specify a start time and end time for the event as a whole, the period for which the timelines will be collected.

The start time should precede the start of the actual event and the end time come after the event is over in order to include pre- and post-event information as a baseline. Researchers can also add distinct timeline spans for each activity. This may be useful in the case when the activities cover a large time span. If an individual shows up in multiple activities or plays multiple roles within an activity, each instance is viewed as an independent observation and the timeline is collected for each situation. This duplication ensures that individuals are counted in every role and activity in which they appear. These panel data provide researchers a complete understanding of each individual's Twitter discourse surrounding an activity or the event.

In the remainder of this section the details of applying the methodology using gatherTweet will be covered. For each of the functions used, the full specification of them can be found in the supplemental materials as well as in the github repository.

3.1. Preliminary Event Specification

The current version of gatherTweet uses Twitter API v2. To access the API users must have a developer account[2]. The specific limitations referenced in this paper assume that researchers have Academic Researcher level access[3]. The application and acceptance process can take a few days.

To use gatherTweet, an empty event object must first be created; this object is then filled with the relevant information and activity objects. This event object is instantiated as:

```
import gatherTweet as tw
event = tw.TwitterEvent (name, base_directory = " ", separator = "CityTown")
```

This code creates a codex TwitterEvent object which can then be populated. The object has a name, a path, and a grouping factor which defaults to the "CityTown" in which the activity takes place. The details regarding the inputs of

[2]Application instructions can be found at https://developer.twitter.com/en/support/twitter-api/developer-account.

[3]More information on Academic Researcher credentials and applications can be found at https://developer.twitter.com/en/products/twitter-api/academic-research.

these functions are listed in Table 1. Once created, the codex TwitterEvent object is ready to be populated by a series of codex TwitterActivity objects as well as event specific data like the keywords and the event period. Each of the three processes used to populate the event with activities are outlined in the next section: from an Excel sheet, manually in Python, or from an existing data structure. Each of the processes requires the same researcher-defined information. The difference between the processes is how the researcher puts that information in the object—the final result is the same.

The researcher must compose an event consisting of discrete activities. For each activity, the researcher must specify a time-frame and geographic boundaries for where it occurred. The geographic boundaries are a bounding box consisting of North and South latitudes and East and West longitudes. They must then determine a list of keywords pertinent to the discourse surrounding the event. To construct this list, it is best to look at Twitter itself as well as other studies that analyze the frequency of usage of certain key terms and hashtags. Keywords are not only restricted to hashtags but also include words and phrases used in the tweets. This information is then added to the codex TwitterEvent object in one of the three processes that follows.

3.1.1. From Excel

The first process discussed is using an Excel document to populate the event. This approach is the easiest, particularly for those with less experience in Python. A template Excel sheet is provided in the **gatherTweet** repository on GitHub. The researcher must add the information provided in Table 2 to the template. The **bolded** arguments are optional. Alternatively, the researcher can create their own Excel sheet where the *italisized* values are the sheet names and the Input column represents the columns of the sheet. Once the sheet is completed, it should be saved in the codex base_directory specified in the event. It can then be populated through the function:

event.upload_from_excel(path = "event_template.xlsx")

The details of this function can be found in Table 3.

Once populating the Excel sheet is complete, the user can begin using the codex TwitterEvent object codex event to collect data.

3.1.2. From Python

The researcher can also input activity information directly in Python. In order to do this, each individual activity must be created. Each codex TwitterActivity object is created as:

activity = tw.TwitterActivity(ID, starting_date, ending_date, CityTown,
StateTerritory, Date, BestGuess)

If, for a given activity, the timeline time span is different than that for the event in general, this independent timing can be added using:

activity.add_timing(period_start, period_end). Details for the function can be seen in Table 4.

This activity is then added to the event through the function (Table 5).

```
event.add_activity(activity)
```

Table 1. TwitterEvent inputs.

Name	Name for the event.
base_directory	Path to where all of the data will be stored.
separator	Describes how the activities are split in the data saving structure, it can be any of the activity objects. For geographical differences "CityTown" makes the most sense.

Table 2. Pages and fields to be filled out in event_template.xlsx to import Event from Excel. **Bold arguments** are optional.

Input	Description
Activities	
ID	Unique identification string.
starting_date	Beginning date of the activity within the event in the format dd mm yyyy hh:mm:ss.
ending_date	Ending date of the activity within the event in the format. dd mm yyyy hh:mm:ss.
CityTown	City or town where you want to search for individuals in the core.
StateTerritory	State or territory where you want to search for individuals in the core.
Date	Date of the activity in dd mm yyyy format.
BestGuess	Best guess of the size of the activity, if unknown, write 0.
period_start	Start of the event (when you want the timeline gathering to start) if different from the rest of the activities in the event. In the format dd mm yyyy hh:mm:ss.
period_end	End of the event (when you want the timeline gathering to end) if different from the rest of the activities in the event. In the format dd mm yyyy hh:mm:ss.
Keywords	
Keywords	List of keywords you want to use to identify the core. Can be written individually or in Twitter accepted format. For example, it is equivalent to have: George Floyd George Floyd vs. (GeorgeFloyd OR (George Floyd)).
Time span	
Start time	Beginning of the timeline gathering period in dd mm yyyy hh:mm:ss format.
End time	End of the timeline gathering period in dd mm yyyy hh:mm:ss format.
Keys	
Key	Twitter keys, make sure not to share these with other people.
Secret	Twitter secret keys, make sure not to share these with other people.
Location	
CityTown	The city or town with which the coordinates are associated.
StateTerritory	The state or territory abbreviation with which the coordinates are associated.
Coordinates	West longitude, south latitude, east longitude, north latitude to find bounding box for each location, a row needs to be added for each unique CityTown-StateTerritory pair found in the activities table.

Table 3. Upload_from_excel inputs.

Path	The path to the Excel file. If the path given is not an .xlsx file it will replace it with base_directory + "event_template.xlsx". If the path is "event_template.xlsx", it also assumes its in the base_directory.

Table 4. TwitterActivity inputs.

ID	Unique identification string.
starting_date	Beginning date of the activity within the event in the format dd mm yyyy hh:mm:ss.
ending_date	Ending date of the activity within the event in the format dd mm yyyy hh:mm:ss.
CityTown	City or town where activity occurs. This value does not correspond to the Twitter API, but is used for the researcher to link with instantiation of the bounding box, and thus simply must be consistent throughout the use.
StateTerritory	State or territory where activity occurs. This value does not correspond to the Twitter API but serves as a method of grouping activities for the researcher.
Date	Date of the activity in any format—this is a string for the user.
BestGuess	Best guess of the size of the activity. If unknown, write 0.

Table 5. TwitterActivity.add_timing inputs.

period_start	Start of the event (when the timeline starts) if different from the rest of the activities in the event. In the format dd mm yyyy hh:mm:ss.
period_end	End of the event (when the timeline ends) if different from the rest of the activities in the event. In the format dd mm yyyy hh:mm:ss.

To finish creating the event, the researcher's Twitter keys must be added. In addition the keywords and general location bounding boxes must be included. First, each set of keys must be converted into a codex TwitterKeyPair object.

key_j = tw.TwitterKeyPair(key, secret). Details for this function can be found in Table 6.

These keys, the keywords, the timing, and the location must then be added to the event object by the researcher through the series of functions listed in Table 7. The order in which each of these four items is added does not matter, but all must be provided to create a full codex TwitterEvent object.

After completing the codex TwitterEvent object, the user must apply a quick check in which gatherTweet both ensures internal consistency and produces an easily readable object that the user can examine. The command is codex protest_ids = event.print_protests() and the output is a dataframe of the activities with the relevant information. The event object is now ready for use.

3.1.3. From Data Structure

The TwitterEvent objects populated using the previous two methods are needed to pull the Twitter data associated with each activity. In order to run the checking and analysis functions explained later in the paper a codex TwitterEvent object is still needed. However, a sparser version of the object is sufficient. Once the data are found, the file structure encodes the information necessary to create this sparser version. In this case, building codex TwitterEvent objects from the

existing data structure may be simpler. The function codex upload_from_file_structure is used to do this. It loops through the file system, building up the codex TwitterEvent object. Specifically, the application is:

event.upload_from_file_structure()

The function has no inputs and assumes that the codex base_directory used in the creation of the event is the same as was originally used and thus is the directory in which the data is stored. Following this command, the codex TwitterEvent object codex event is ready to be used to collect timelines and analyze data.

3.2. Individual Level Identification

Individuals in the event core are identified after the creation of the event object. From there, the echo and influence accounts are found. While on the back end these are similar processes, small differences in the commands sent to the Twitter API result in substantial differences in data collected.

Table 6. TwitterKeyPair inputs.

Key	Twitter developer key as a string
Secret	Twitter developer secret key as a string

Table 7. Completing event functions and inputs.

event.add_key(key_j)	
Key	Twitter developer key as a string
Secret	Twitter developer secret key as a string
event.add_keyWords(words)	
Words	List of keywords to use to identify the core. Can be written individually or in Twitter accepted format. For example, it is equivalent to have: George Floyd GeorgeFloyd vs. (GeorgeFloyd OR (George Floyd))
event.add_timing(start, end)	
Start	Beginning of the timeline gathering period in dd mm yyyy hh:mm:ss format.
End	End of the timeline gathering period in dd mm yyyy hh:mm:ss format.
event.add_location(CityTown, StateTerritory, west, south, east, north)	
CityTown	List of CityTown entries that occur in the activities.
StateTerritory	List of StateTerritories associate with CityTown list.
West	List of West longitude for bounding box of CityTowns.
South	List of South latitude for bounding box of CityTowns.
East	List of East longitude for bounding box of CityTowns.
North	List of North latitude for bounding box of CityTowns.

Figure 2 presents a schematic for the core identification for each activity. Beginning with an activity from the codex TwitterEvent object, the keywords, activity timeframe, and location are queried. For each keyword in the keywords object, tweets are found that are published within the bounding box during the timespan with that keyword. The API returns a maximum of 500 tweets per query. In cases when there are more than 500 tweets in the timeline, the code paginates through the API results until all tweets with that keyword are downloaded[4]. At the end of each keyword search, the collected tweets are saved as a JSON file in a folder called "Core" within an activity-specific folder in the specified codex base_directory.

Figure 3 shows that the collection of echo and influence users follows a similar process. A list of the individuals belonging to the core is created. Their tweets are then examined from two days proceeding the start of the activity until two days after. Echos are designated as individuals who retweet any of the tweets during that period, while influences are the original creators of core user's retweets. The user IDs of these users are then stored in the Echo (Influence) folders within the Activity folder in the base directory as text files names echo (influence)-ids.txt.

In order to run any tweet or user identification processes using gatherTweet, the researcher uses the codex get_tweets command; Table 8 shows its arguments. This function has the ability to retrieve the core, echos, and influences user ID numbers as well as each of their timelines. This command, which in its default form is

```
event.pull.get_tweets(event, types = [], max_results = 500,
tweets_per_file = 1000, expansions = ["author_id",
"in_reply_to_user_id"], tweetfields = ["author_id",
"created_at", "geo", "entities", "public_metrics",
"text", "referenced_tweets"]),
```

Table 8. Get_tweets inputs.

Event	TwitterEvent object
Types	a list of the types of Tweets and users you want to collect, options are: [Core, CoreTimeline, Echos, EchosTimeline, Influences, InfluencesTimeline]. If left as an empty list, all will be evaluated. In order to run the Timeline versions, all activities are checked to make sure they have the base version. For Echos and Influences it checks that a core exists. Everything but the Core can restart after being interrupted with minimal redundancy.
max_results	Number of tweets to attempt to pull in each query, must be an integer between 1 and 500.
tweets_per _file	Number of tweets to save per file before beginning a new file.
Expansions	A list of tweetfields the researcher would like more information on, taken from https://developer.twitter.com/en/docs/twitter-api/expansions, options are: author_id, referenced_tweets.id, in_reply_to_user_id, attachments.media_keys, attachments.poll_ids, geo.place_id, entities.mentions.username, referenced_tweets. id.author_id
twEetfields	A list of values within each tweet to be returned from each call are taken from https://developer.twitter.com/en/docs/twitter-api/data-dictionary/object-model/tweet: id (default), text (default), attachments, author_id, context_annotations, conversation_id, created_at, entities, geo, in_reply_to_user_id, lang, non_public_metrics, organic_metrics, possibly_sensitive, promoted_metrics, public_metrics, referenced_tweets, reply_settings, source, withheld)

[4]500 is accurate for Academic Researcher developer keys on the Twitter API v2 as of October 2022.

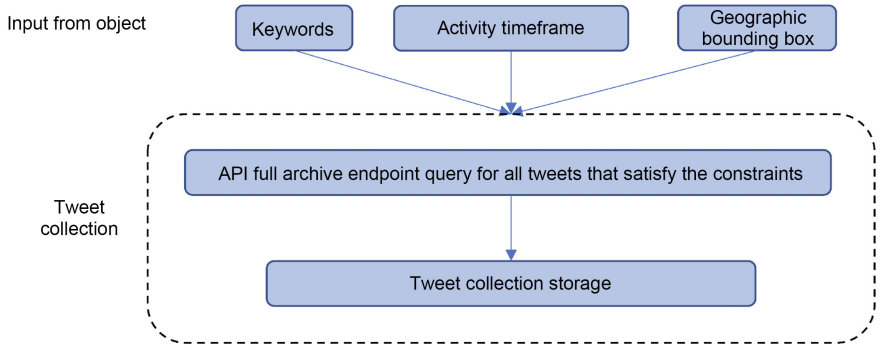

Figure 2. Core Identification.

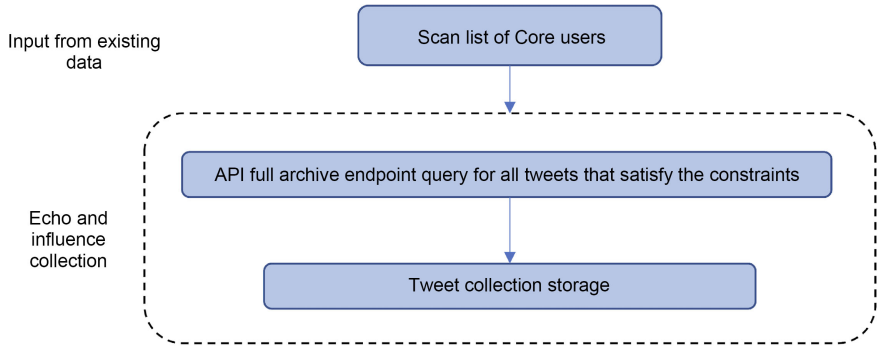

Figure 3. Echo and Influence Identification.

is flexible and customizable. If "codex Core" is not in the codex types input and has not already been pulled for the event, the process stops and issues an error. gatherTweet checks for other dependencies and issues errors as appropriate. Additional examples of such dependencies include finding the echo or influence timeline before having found the echo or influence accounts.

3.3. Timeline Tweet Collection

The timeline tweet collection process is the same for all three groups of individuals. Given a list of Twitter user IDs, all tweets from the users during the event period are accessed. While timeline tweet collection is programmatically simple, it can be the most time intensive part of gatherTweet as it has to collect the largest number of tweets. To do this collection, the function codex event.pull.get_tweets() must be run with codex types = [CoreTimeline, EchosTimeline, InfluencesTimeline]. This function is the same one that identifies individuals who belong to the core, echos, or influences; the difference is in the codex type declaration. Full details can be seen in Table 8. If the individuals for each type have not been pulled gatherTweet will notify the researcher with the warning message

At least one activity does not have a list of "type" and the function will exit.

4. Parsing

It is useful to provide the researcher a sense of the size of an activity or activities' data. This can be used as a check as the method is applied to get a sense of the

size of the final dataset. The function codex check.number_of_tweets estimates the numbers of tweets in each activity and type, and Table 9 shows its inputs.

The package gatherTweet generates a series of JSON files in an informative file structure. The file structure is such that the location of the file clearly informs the researcher of which information is in the JSON file. This structure, however, keeps information separate and therefore requires the user to individually load numerous files into a processor to access the raw data. To aid the researcher in circumventing this process, gatherTweet also includes functions for merging the individual JSON files into a usable and accessible format.

Two functions, codex read.read_tweets and codex read.read_tweets_basic, combine all of the tweets from an activity into one file within the activity's codex analysis folder. The functions transform the JSON files into one comma-separated values (.csv) file. The difference between the two functions is in the form which each outputs. The former includes all of the information in the original file, while the later includes only the text, author ID, tweet ID, created at date, and a string indicating which activity the tweet is associated with.

The information provided in codex read.read_tweets_basic lowers the storage requirements and in many cases provides sufficient data for analysis. For most purposes, this function is the appropriate one for placing the data within activity-specific files. If the researcher wants to further simplify their storage structure and have all of the event data in one location, the functions codex read.wrap_csv and codex read.wrap_csv_basic can be used. These functions further combine the results into a singular file within the codex analysis folder of each group of activities (a directory for each value of the separator variable). The four functions have the same input options (as seen in Table 10) and the original and basic versions can be used interchangeably.

Table 9. Check.number_of_tweets inputs.

Activities	A list of the activities in which to estimate tweets.
Users	The types of users to count tweets from, options are [Core, Echo, Influence].
base_directory	Where to begin searching for the data.
Separator	Describes how the activities are split in the data saving structure, it can be any of the activity objects. For geographical differences "CityTown" makes the most sense.
Timeline	Boolean of whether to count tweets in the timeline or not, if not, only the original tweets of the core will be counted.

Table 10. Read.read_tweets(_basic) and read.wrap_csv(_basic) inputs.

Event	The name of the TwitterEvent object.
Users	A list of the types of Tweets and users you want to aggregate the data for options are any subset of: [Core, CoreTimeline, Echos, EchosTimeline, Influences, InfluencesTimeline].

5. Research Results

5.1. 2020 Black Lives Matter Protests

This section demonstrates usage of gatherTweet to aid in the analysis of Twitter discourse about the Summer 2020 Black Lives Matter (BLM) protests. Between May 26th and August 22nd, 2020, approximately 7750 BLM demonstrations occurred in over 2440 locations in all 50 US states [18]. These protests were some of the most well attended and longest lasting in American history [19]. With the sheer number of individuals and tweets involved in the protests, collecting every tweet that used relevant keywords would overwhelm most systems. By restricting analysis to specific cities, gatherTweet significantly reduces the strain of this collection.

5.1.1. Preliminary Event Specification

We first choose which cities and protests to include as well as the event time span. We focus on Los Angeles, Chicago, and Houston—the second through fourth largest cities in the United States—and choose activities (protests) starting immediately after the murder of George Floyd. The timeline collection occurs from the 20th of May 2020 until the 1st of October. This span is 6 days before the first recorded activity until 5 days after the final one. The location, date, estimated size, and overall statistics from the process can be seen in Tables 11-13. The resulting data encompasses 13 activities in Los Angeles, 24 in Chicago and 7 in Houston.

Table 11. Overview of Data: Los Angeles.

City	Protest Date	CCC Estimated Size	# Core	# Influence	# Echo	Avg Core Timeline Size
Los Angeles	05-27-2020	250	351	47,506	62,978	1593
Los Angeles	05-28-2020	750	604	0	184,151	1582
Los Angeles	05-29-2020	2000	516	48,301	136,280	1286
Los Angeles	06-06-2020	3000	325	46,197	86,181	1473
Los Angeles	06-13-2020	200	326	38,583	59,360	1785
Los Angeles	06-27-2020	100	143	35,151	50,522	2396
Los Angeles	07-14-2020	50	79	7450	8977	4773
Los Angeles	07-25-2020	150	88	14,634	8735	2673
Los Angeles	07-26-2020	500	78	13,818	23,805	4499
Los Angeles	08-24-2020	112	112	12,274	13,560	4346
Los Angeles	08-25-2020	200	111	29,268	26,568	3537
Los Angeles	08-26-2020	300	156	10,734	5297	4200
Los Angeles	09-23-2020	500	196	21,585	47551	4055

Table 12. Overview of Data: Chicago.

City	Protest Date	CCC Estimated Size	# Core	# Influence	# Echo	Avg Core Timeline Size
Chicago	05-29-2020	300	13	99	17	2350
Chicago	05-30-2020	1750	35	2639	3386	1210
Chicago	05-31-2020	200	44	2624	938	395
Chicago	06-05-2020	2000	48	11,534	12,616	466
Chicago	06-06-2020	2500	69	18,611	3299	476
Chicago	06-08-2020	200	22	2832	133	535
Chicago	06-12-2020	1400	13	2348	32	802
Chicago	06-13-2020	200	18	74	6	238
Chicago	06-14-2020	2500	16	2277	69	520
Chicago	06-17-2020	200	12	13	11	219
Chicago	06-19-2020	2000	28	4306	999	161
Chicago	06-24-2020	500	7	353	47	1324
Chicago	06-28-2020	2000	5	11	2	212
Chicago	07-02-2020	200	9	58	46	409
Chicago	07-17-2020	1000	9	728	11	545
Chicago	07-20-2020	100	3	162	77	308
Chicago	07-24-2020	200	3	0	0	113
Chicago	07-25-2020	300	5	652	319	1709
Chicago	08-08-2020	100	3	1	3	267
Chicago	08-18-2020	135	3	0	2	174
Chicago	08-29-2020	200	5	45	65	280
Chicago	09-23-2020	24	10	2428	0	607
Chicago	09-24-2020	500	11	3650	6271	4174
Chicago	09-26-2020	200	7	518	47	5755

Table 13. Overview of Data: Houston.

City	Protest Date	CCC Estimated Size	# Core	# Influence	# Echo	Avg Core Timeline Size
Houston	05-26-2020	200	2	281	54	116
Houston	05-29-2020	200	36	16,193	14,953	772
Houston	05-30-2020	200	24	7997	7846	441
Houston	06-02-2020	200	180	25,426	0	637
Houston	06-08-2020	40	26	5448	12,923	700
Houston	06-13-2020	50	4	23	1	1026
Houston	07-04-2020	2000	8	752	259	647

The keywords used are listed in Table 14. They fall into three different categories: 1) calls to mobilize others to actively join protests, 2) names of the individuals who were victims of injustice, and 3) phrases that are commonly chanted during protests. They are designed to capture the organizational period immediately before the protests, the protest itself, and the topics most likely discussed during the protests.

Having created keywords, we then populate the package's Excel template; these values are in the template in the GitHub repository. To generate the event, the following code is used:

```
import gatherTweet as tw
dirr = "directory"
BLM = tw.TwitterEvent("BLM", base_directory = dirr)
BLM.upload_from_excel(dirr+"event_template.xlsx").
```

5.1.2. Individual Identification and Timeline Tweet Collection

Once the codex TwitterEvent object is set up, the collection can begin. One command accomplishes all of the user and tweet collection:

```
BLM.pull.get_tweets(BLM, base_directory = dirr)
```

The default of this function sets:

types = [codex Core, codex CoreTimeline, codex Echos, codex EchosTimeline, codex Influences, InfluencesTimeline].

Thus, it executes both of the major data collection steps of the method. *This function signifies the main contribution of gatherTweet*; in one call, given appropriate setup, researchers are able to collect all of the Twitter data for their purpose.

Table 14. Keywords for protesters.

Keywords
Black Lives Matter
BLM
George Floyd
Justice for Floyd
Walk with Us
Kneel with Us
March with Us
I Can't Breathe
March for Peace
Take a Knee
Breonna Taylor
No Justice No Peace
Say Their Names
Ahmaud Aubrey

5.1.3. Analysis

Parsing and analysis follow data collection. First, we populated **Tables 11-13** using the codex check.number_of_tweets command for each activity and group. These tables present summary statistics for each event. Los Angeles has the most tweets and accounts for all three types.

The rest of this application focuses on the actions of the core. In order to place all of the tweets related to the core users in one place, we run the following four commands.

BLM.read.read_tweets(BLM, users = "Core")

BLM.read.wrap_csv(BLM, users = "Core")

BLM.read.read_tweets_basic(BLM, users = "CoreTimeline")

BLM.read.wrap_csv_basic(BLM, users = "CoreTimeline").

Despite descriptive differences, the events are similar when it comes to their discussion frequency. **Figure 4** plots the percent of tweets from individuals in the core of activities taking place in each city each day. It shows that despite differences in each city's design, size of protests, and local events, the discourse pattern is remarkably similar. These individuals had a large increase in Twitter chatter in the days following the murder of George Floyd with similar patterns in the months following.

In addition to proportional tweeting frequency, we see that very few individuals appear in the core more than once. We believe that individuals who are in the core are at the physical protests since their location places them in the city, and prior work has found that this assumption accurately measures protest size variation [20]. The lack of repeating individuals suggests that the group protesters may not have been consistent over time [21] [22]. The statistics regarding repeat members of the cores can be seen in **Table 15**. Most of the members of the core of each city only go to one protest. The number of people who attend multiple protests is significantly fewer, and the numbers continue to decrease as the number of protests increases. Results like these can give researchers insights into protest behavior and help categorize the protests.

5.1.4. Data Collection Time Estimate

Given the large amounts of data that events can generate, it is important to estimate runtime associated with gatherTweet. The exact time required for the collection of core users cannot be determined *a priori*. Depending on the event and activities within it, there is no precise way of estimating the size of the core. However, once the core has been collected it is possible to estimate the time required to collect the echo and influence users. A directed search for a five day period takes on average 6 seconds per user[5]. To collect the echo and influence users, this search must be run for each member of the core. Therefore, a time estimate for the collection of all echo and influence users can be approximated by:

$$\#\text{coreusers} \times 2 \times 6 \text{ seconds} \quad (1)$$

[5]This value was taken by running the directed search for 180 individuals, found in the Black Lives Matter core, locally on a standard machine. The total time (1108.4092 seconds) to find the echos was then divided by 180 to obtain the average.

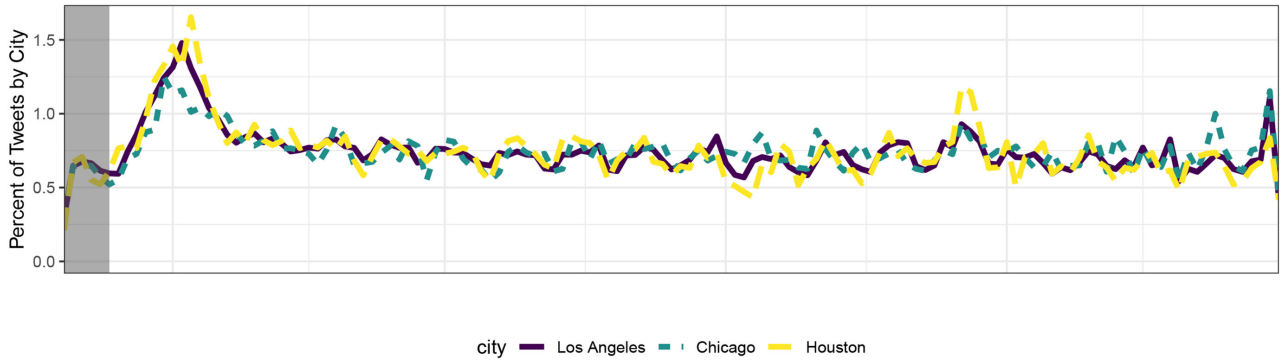

Figure 4. Discussion of BLM over time by protesters (members of the core).

Table 15. Percent BLM repeat protesters by city.

Protests	Houston	Chicago	Los Angeles
1	100.00	87.38	92.37
2	0.00	9.23	7.38
3	0.00	1.85	0.24
4	0.00	0.92	0.00
7	0.00	0.31	0.00
17	0.00	0.31	0.00

A similar process was applied to find the time taken for timeline collection. Again, the specific individuals being queried can change these values, especially if the event being sampled is made up of prolific or reserved individuals. Our estimate for timeline collection is 0.1 seconds per day[6]. When the timeline gets shorter the time per day may increase. This is a result of the way Tweets are pulled. The program can receive 500 tweets per query, therefore, if the time is short enough that fewer than 500 tweets are found in each query the time to accomplish the task will remain constant. To find a rough estimate, the following equation may be followed:

$$\#\text{users} \times \#\text{days} \times 0.1\,\text{seconds} \tag{2}$$

We emphasize that the time required to collect tweets is heavily dependent on the subject matter the researcher is exploring as the frequency of tweeting is highly dependent on the type event.

5.2. Other Applications

This paper's methodology can be applied to other social media platforms and other activities. This subsection discusses examples where gatherTweet and the associated methodology would be useful.

[6]This estimate was taken from measuring the time it took to collect 180 timelines over our 134 day period from May 20, 2020 until October 1, 2020 and estimating the individual daily average time. The total time for this sample was 2400.3108 seconds.

One obvious application would be mass-mobilization events such as the Yellow Vest protests in France [23], the Black Lives Matter protests [24], or the Women's March [25]. Such protests are ideally suited because they involve a large following and spread out over time and space. Collecting all of the relevant social media data for events of these magnitudes would overwhelm many computer systems. Instead, for gatherTweet, the researcher would choose a set of geographic and time combinations to focus on. With such criteria, the monitor will identity the core and then expands out to find the rest of their network at the time of the activity.

A second application includes phenomena that move over time, such as heat waves or hurricanes, or epidemics like COVID-19. They require a slightly different approach than mass-mobilization events because the timing and geographic locations have to be well-specified. For weather episodes, the researcher has to discretize locations and timings to create the core. The researcher could choose a time increment, such as a number of hours or days, and pick a corresponding geographical area large enough to encompass the entire activity. That approach makes the most sense for events that move over time, such as a hurricane. For other natural activities, such as disease, where the geographic path is obscured, or there are jumps, or the presence of the activity is unclear, a threshold approach makes sense. Examples would be beginning the spread of COVID-19 when cases in a city surpass a *per capita* threshold.

Other examples of possible applications are elections and pop culture activities, such as performer's tours or sports teams' games. With these, there is a clear point when the activity occurs in a location. For example, elections, specifically U.S. presidential primaries, occur at different times in different locations. Sports teams, musicians, and comedians perform in different locations on certain dates and times. In each case, the approach is similar to the mass-mobilization since the location and timing of the core detection is often obvious. A main drawback is that these events may be smaller so the method may only pick up a small number of individuals.

6. Social Media Data Collection Restrictions

The collection process of Twitter data for research purposes are subject to the restrictions of its representational state transfer (REST) APIs[7]. While the documentation for the usage of the APIs is relatively straightforward, there are specific limitations pertinent to the implementation of gatherTweet. Perhaps the most binding limitation is rate limits that very by endpoint. **Table 16** provides common rate limits encountered.

[7]Here we discuss the technical uses of the Twitter APIs that are available for academic research. Twitter's developer terms of service also restricts how data collected can be used. For example, geolocated data cannot be used to track Twitter users spatially and can only be used in association with information in their tweets in aggregated form. Also, academic users are restricted in how they use data like these by their institutional review boards (IRBs); the data collection methodology in this paper was reviewed and approved by Caltech's IRB (Caltech IRB No. 21-1148).

Table 16. Summary of Twitter rate limits.

Endpoint	Per App (15 min window)	Per User (15 min window)
User tweet timeline	1500	900
User mention timeline	450	180
Full-archive search	300	180

To maximize the number of requests, gatherTweet sends 1 query per second to prevent errors and to avoid sending bad requests that count towards the rate limit. The 1 query per second is an explicit one for the full-archive search, however sometimes it may be helpful to slow down the query rate to prevent too many endpoint requests. For example the user mention timeline can be requested 450 times per 15 minute window, and by spacing the 450 requests to be within a 15 minute time window (sending a request every 2 seconds rather than 1) prevents querying after the rate limit is reached.

Compounded with the rate limit issue is the 10 million tweet per month per key restriction. While the only solution to the monthly cap is the utilization of multiple sets of keys, the data methodology presented in this paper seeks to maximize the 10 million tweets by gathering the most central information to the event.

The data collection process gets more complex as the time span studied increases. Therefore, to identify the core of the event we utilize keywords that are associated with the topics most discussed during the event. When querying utilizing keywords it is important to use every possible version of the phrase in regards to spelling and spacing. In addition to the keyword search, our method specifies a location and time component. Approximately 1% - 2% of all tweets are geotagged which means the creation of geographic restrictions on the core allows us to maximize the 10 million tweets as well as insure with a higher likelihood that the tweets collected belong to users involved in the event. Our second measure is to restrict the collection to users who tweet during the day of the activity, increasing the likelihood we collect users physically present during an activity.

The data collection process is designed to be versatile with regards to researchers' access to resources. The package gatherTweet can be run on university servers, in the cloud, or locally. The storage method of this data can also been done on a server or locally. By reducing the amount of post-processing required of the data it is extremely easy for the researchers to use the tweets in any type of natural language processing methodology.

7. Conclusions

Social media provides a wealth of information for researchers. gatherTweet, an easy-to-use Python package that requires very little prior programming knowledge, can aid researchers in the collection and analysis of large scale social media events. Since large events can produce more data than most researchers can

feasibly acquire, store, and analyze, gatherTweet's data methodology provides a solution by targeting the main participants in an event.

The methodology presented in this paper instructs researchers on a clear way to conceptualize massive physical events which have discrete geographic and temporal activities. In addition, it creates a non-random approach to limiting the data collected, while accounting for exogenous trends such as location and time. Finally, in collecting the individuals who echo and influence the participants, the network surrounding participants is included. Thus, the process can be used to easily find social media users who both participate in offline event activities and are in the same sphere as each other.

This method is all the more relevant in an increasingly interconnected world. Events such as protests have a dispersion of geographic loci, and isolating them individually enables researchers to better pinpoint time spans and geographies. In creating a standardized method for data collection of large scale social media data of events, gatherTweet also enables comparison across future research projects. gatherTweet helps researchers directly collect the data most relevant to their cause.

Acknowledgements

We thank the Google Cloud Research Credits Program for providing credits for our use of the Google Cloud Platform for data collection and analysis. Related research was presented at the 2022 MPSA Annual Meetings, we thank participants for their comments. Thanks as well to Jian Cao and Danny Ebanks for their help with our research.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Tillery, A.B. (2019) What Kind of Movement Is Black Lives Matter? The View from Twitter. *Journal of Race, Ethnicity, and Politics*, **4**, 297-323. https://doi.org/10.1017/rep.2019.17

[2] Theocharis, Y., Lowe, W., Van Deth, J.W. and Garcıa-Albacete, G. (2015) Using Twitter to Mobilize Protest Action: Online Mobilization Patterns and Action Repertoires in the Occupy Wall Street, Indignados, and Aganaktismenoi Movements. *Information, Communication & Society*, **18**, 202-220. https://doi.org/10.1080/1369118X.2014.948035

[3] Conover, M.D., Ferrara, E., Menczer, F. and Flammini, A. (2013) The Digital Evolution of Occupy Wall Street. *PLOS ONE*, **8**, e64679. https://doi.org/10.1080/1369118X.2014.948035

[4] Kann, C., Hashash, S., Steinert-Threlkeld, Z. and Alvarez, R.M. (2021) Collective Identity in Collective Action: Evidence from the 2020 Summer Blm Protests. *The* 2021 *Annual Meetings of the Midwest Political Science Association*, Chicago, April 9, 2022.

[5] Bruns, A. and Liang, Y.E. (2012) Tools and Methods for Capturing Twitter Data during Natural Disasters. *First Monday*, **17**, 1-8. https://doi.org/10.5210/fm.v17i4.3937

[6] Cao, J., Adams-Cohen, N. and Alvarez, R.M. (2021) Reliable and Efficient Long-Term Social Media Monitoring. *Journal of Computer and Communications*, **9**, 97-109. https://doi.org/10.4236/jcc.2021.910006

[7] Bozarth, L. and Budak, C. (2022) Keyword Expansion Techniques for Mining Social Movement Data on Social Media. *EPJ Data Science*, **11**, 30. https://doi.org/10.1140/epjds/s13688-022-00343-9

[8] Zheng, X. and Sun, A.X. (2019) Collecting Event-Related Tweets from Twitter Stream. *Journal of the Association for Information Science and Technology*, **70**, 176-186. https://doi.org/10.1002/asi.24096

[9] McCormick, T.H., Lee, H., *et al.* (2017) Using Twitter for Demographic and Social Science Research: Tools for Data Collection and Processing. *Sociological Methods & Research*, **46**, 390-421. https://doi.org/10.1177/0049124115605339

[10] Barbera, P. and Steinert-Threlkeld, Z.C. (2020) How to Use Social Media Data for Political Science Research. SAGE Publications, Thousand Oaks. https://doi.org/10.4135/9781526486387.n26

[11] Roesslein, J. (2009) Tweepy. https://github.com/tweepy/tweepy

[12] Gedulig, J. (2021) Twitterapi. https://github.com/geduldig/TwitterAPI

[13] Taylor, M. (2007) Python-Twitter. https://github.com/bear/python-twitter

[14] Gonzalez-Bailon, S., Borge-Holthoefer, J. and Moreno, Y. (2013) Broadcasters and Hidden Influentials in Online Protest Diffusion. *American Behavioral Scientist*, **57**, 943-965. https://doi.org/10.1177/0002764213479371

[15] Eubank, N. and Kronick, D. (2021) Friends Don't Let Friends Free Ride. *Quarterly Journal of Political Science*, **16**, 533-557. https://doi.org/10.1561/100.00020143

[16] Rudig, W. and Karyotis, G. (2013) Beyond the Usual Suspects? New Participants in Anti-Austerity Protests in Greece. *Mobilization*, **18**, 313-330. https://doi.org/10.17813/maiq.18.3.r3377266074133w5

[17] Kryvasheyeu, Y., Chen, H.H., Obradovich, N., *et al.* (2016) Rapid Assessment of Disaster Damage Using Social Media Activity. *Science Advances*, **2**, e1500779. https://doi.org/10.1126/sciadv.1500779

[18] Raleigh, C., Linke, A., Hegre, H. and Karlsen, J. (2010) Introducing Acled-Armed Conflict Location and Event Data. *Journal of Peace Research*, **47**, 651-660. https://doi.org/10.1177/0022343310378914

[19] Putnam, L., Chenoweth, E. and Pressman, J. (2020) The Floyd Protests Are the Broadest in U.S. History—And Are Spreading to White, Small-Town America.

[20] Sobolev, A., Joo, J., Chen, K. and Steinert-Threlkeld, Z.C. (2020) News and Geolocated Social Media Accurately Measure Protest Size Variation. *American Political Science Review*, **114**, 1343-1351. https://doi.org/10.1017/S0003055420000295

[21] Saunders, C., Grasso, M., *et al.* (2012) Explaining Differential Protest Participation: Novices, Returners, Repeaters, and Stalwarts. *Mobilization*, **17**, 263-280. https://doi.org/10.17813/maiq.17.3.bqm553573058t478

[22] Saunders, C. and Shlomo, N. (2021) A New Approach to Assess the Normalization of Differential Rates of Protest Participation. *Quality and Quantity*, **55**, 79-102. https://doi.org/10.1007/s11135-020-00995-7

[23] Smith, S. (2018) Who Are France's "Yellow Jacket" Protesters and What Do They

Want? NBC News.

[24] Buchanan, L., Bui, Q. and Patel, J.K. (2020) Black Lives Matter May Be the Largest Movement in U.S. History. *The New York Times*.

[25] Stein, P., Hendrix, S. and Hausiohner, A. (2017) Women's Marches: More than One Million Protesters Vow to Resist President Trump. *The Washington Post*.