twAwler: A lightweight twitter crawler

Polyvios Pratikakis
FORTH-ICS
polyvios@ics.forth.gr

December 6, 2018

Abstract

This paper presents twAwler, a lightweight twitter crawler that targets language-specific communities of users. twAwler takes advantage of multiple endpoints of the twitter API to explore user relations and quickly recognize users belonging to the targetted set. It performs a complete crawl for all users, discovering many standard user relations, including the retweet graph, mention graph, reply graph, quote graph, follow graph, etc. twAwler respects all twitter policies and rate limits, while able to monitor large communities of active users.

twAwler was used between August 2016 and March 2018 to generate an extensive dataset of close to all Greek-speaking twitter accounts and their tweets and relations. In total, the crawler has gathered 750 million tweets of which 424 million are in Greek; 750 million follow relations; information about 300 thousand lists, their members (119 million member relations) and subscribers (27 thousand subscription relations); 705 thousand trending topics; information on 52 million users in total of which 292 thousand have been since suspended, 141 thousand have deleted their account, and 3.5 million are protected and cannot be crawled. twAwler mines the collected tweets for the retweet, quote, reply, mention, list membership and list similarity graphs, which, in addition to the follow relation crawled, offer vast opportunities for analysis and further research.

1 Introduction

Social media content offers many opportunities for research in numerous fields and disciplines, including machine learning, natural language processing, epidemiology, sociology, economics, etc. Data mining social media is, however, increasingly difficult, due to technical and policy constraints.

Specifically, twitter has been used in multiple studies and analyses due to its more open and public content. Twitter restricts the reuse and publication of data crawled using its public API to only sharing anonymized information (user and tweet IDs), and moreover restricts the number of queries made to its API to a limited rate. This limitation may be overcome by combining multiple

users' rate limits and aggregating multiple crawls, but (i) this may be considered sharing non-anonymized information by twitter, or (ii) it may require access to expensive resources such as several cloud VMs or a cluster, for a prolonged period of time.

This paper presents twAwler, an open-source[1], cost-effective, lightweight twitter crawler that can explore, discover and target users related to a specific topic or using a given language. The crawler takes advantage of multiple twitter API endpoints, maximizing the total crawled information while respecting all limits and policies. Moreover, it requires a single users' credentials and can run on a single machine over large periods of time, tolerating reboots and downtime without issue. twAwler aims to be complete, *i.e.*, does not crawl a sample of the user content but instead all of the traffic of the users belonging to the crawled community. The crawler can analyze tweets and generate a multitude of relations and information, including the follow, retweet, mention, reply, quote, and favorite graphs, temporal patterns, topic and word frequencies, etc. The author has used the crawler for a period of 20 months to discover and track a set of all Greek-speaking twitter accounts, using a low-cost machine, the author's desktop PC. twAwler can perform similarly well for even larger twitter communities, especially when targetting language-specific parts of the twitter graph.

2 Twitter Crawler

twAwler is a custom crawler for twitter data that discovers and monitors Greek-speaking twitter users, monitors all their publicly accessible content. The crawler stores this information and is able to extract multiple relations, including the follow graph, the mention, reply, retweet and quote graphs, the favorited graph, etc. These relations are timestamped, enabling further analysis using dynamic graph techniques. twAwler maintains a set of tracked users, a set of users that have been marked as greek-speaking, a set of stop-users that are definitely not greek-speaking and the sets of dead, suspended, and protected users. twAwler is structured as a set of small tools and the scripts using them to perform on-demand or continuous crawls.

2.1 Tweet Crawler

twAwler uses the twitter /statuses/user_timeline API endpoint to crawl the tweets of tracked users. To crawl the selected users' tweets, twAwler downloads all the user tweets that it can using the request twitter API, up to the 3200 tweets that twitter allows, or until it reaches the last tweet seen when the user was crawled previously. To save on the number of rate limited requests, twAwler prioritizes crawling of users that have a high probability of having tweeted a lot since last crawled. To do that, it computes the average tweets per day for every user and

[1]The software will be released under Apache License before this publication is in print. twAwler is about 11KLoC of Python.

uses two processes of tweet crawling: On the one hand, the crawler sorts all users based on their expected tweets since they were last crawled and crawls users with a high number of expected tweets. On the other hand, the crawler sorts all users based on the time since they were last crawled and crawls users not visited in a long time. This way, twAwler minimizes the probability of lost tweets, and also make the best use of the available requests per minute that twitter allows. Note that twitter throttles the number of API requests per 15 minutes, that only the last 3200 tweets can be requested, and that a maximum of 200 tweets can be returned per request. A simple back-of-the-envelope calculation shows that by delaying crawling to keep the average number of tweets per user crawled close to 1000 since the last visit, we consume half the number of requests per tweet compared to an average of 100 tweets since the last visit.

In addition to the stardard twitter user timeline crawling, twAwler also looks for tweets from tracked users that are retweets, reply to, or quote other tweets, and use the /statuses/lookup endpoint to ask specifically for information regarding these tweets and their authors. This way, the crawler will eventually discover all interactions of the tracked users and crawl them for analysis of the whole "thread".

To discover new users in the targetted community, we seed the tracked set of users by performing a search for common greek stopwords on the /statuses/filter twitter streaming API. There is no need to run the filter continuously, as it is only used to seed the tracked users, since the crawler is user-centric and aims for a full crawl of the involved users. In addition to seeding based on stopwords, the crawler adds new users by tracking any retweets in Greek from currently tracked users. Specifically, if the crawler discovers 10 tweets in Greek by an unknown user that have been retweeted by tracked users, then it also starts tracking the new user as there is a high probability they are also Greek-speaking. Moreover, a daily pass discovers tracked users with more than 500 tweets, of which less than 2 per cent are in Greek, and stops tracking them, adding them to the set of users where the crawl stops, as they were found to not be greek-speakers.

2.2 Follow Graph Crawler

The public twitter API offers four ways of crawling user follow relations, namely using the friends/ids and friends/list API endpoints for crawling the IDs or fully populated user structs of the user's friends, and followers/ids and followers/list to crawl followers respectively. As rate limits are set separately for each endpoint, twAwler crawls all users' friends and followers using four crawler processes, marking each crawled user to not be crawled again for the following 30 days. This limit is currently arbitrary, as the follow relation does not change often or oscillate, and could be set to a much lower time window while still remaining within the rate limits for a community of this size. Note that two of the four endpoints twAwler uses to crawl the follow graph produce detailed user information, whereas the other two produce solely user IDs. These are easily separated and stored as the follow graph and a table of users, where some IDs may not yet be populated with all user information. twAwler periodically runs a separate

processes to populate missing user information, using the /users/show/:id endpoint, and also utilize the tweet crawler to save the user information for any user for whom it finds it missing or out-of-date. Note that this will generate multiple entries of the user information per user, allowing an analysis to monitor the evolution of the user's profile, bio, language, etc, over time. Currently, twAwler is configured to use a time window of two weeks for crawling user information, and do not store the new information if it only differs in the number of tweets from the user profile last seen.

2.3 List Crawler

Twitter lists have been used to mine user-curated information in previous research. As they aggregate the opinion, or classification, of users into groups by other users, they amount to valuable crowdsourced information that can be mined to infer common interests, relations, etc. The twitter API offers 4 ways to crawl lists; namely, lists/subscriptions returns information regarding the lists to which a user subscribes, lists/memberships returns information regarding the lists of which a user is a member, lists/members returns the users that are members of a list, and lists/ownerships returns lists curated by the given user. As these have different rate limits, twAwler uses four separate crawler processes that crawl lists and users in a round-robin fashion and populate or update the list membership relations.

2.4 Favorites Crawler

Twitter allows users to signal interest in a tweet by marking it as "favorite" (or "like"). The twitter API offers one way to get a user's favorites, favorites/list, returning "likes" ordered by date of "liked" tweet. This is one of the most limiting constraints, as any "likes" on tweets older than the newest "like" seen will be lost. To avoid that as much as possible, twAwler does not stop after observing previously seen "likes" as with tweets, but continues crawling until having seen more than 190 previously known "likes". The probability of missing user "likes" remains, but as users often "like" tweets that they observe in the top of their timeline, missing old "likes" happens less often. The twitter web client allows viewing a user's "likes" in the order they were done, not the order they were tweeted; manually observing the "likes" of five very active users over a period of two days showed a small percentage of missed "likes". twAwler could scrape the web page for these, but does not, so as to remain very clearly within the limitations of the twitter API agreement. Note that the rate limiting for favorite crawling is an order of magnitude less than the tweet API limits, resulting in only a sample of the favoriting graph compared to the near-full coverage for the other information.

2.5 Crawler Storage

twAwler currently uses a MongoDB installation to store crawled data. The MongoDB contains the following collections, used to store the crawled data as well as metadata used by the crawler.

The users collection contains user objects as returned by the Twitter API. There may be multiple entries per single user, as discussed above. Each entry is annotated with the insertion date. To avoid sorting and aggregation when scanning for the latest entry for a given user, the data includes an additional field called screen_name_lower that contains the screen name in lowercase. This field is unique when it exists, and in aggregate helps depict the currently existing users as last seen by the crawler.

The tweets collection contains tweet objects as returned by the Twitter API. Each tweet is unique. As there may be missing or truncated tweets in the data store, twAwler uses additional curation processes that filter out truncated tweets and use the /statuses/lookup API to properly populate them with the non-truncated version. The author has used this tool to also properly ingest textual dumps of tweets generated by early versions of the crawler, and handle the switch of the tweet length limit from 140 to 280 characters seamlessly.

The trends collection contains trend objects as returned by the trends/place Twitter API for Greece, timestamped and crawled every 15 minutes. The rate limit for trends is never reached, as Greek-speaking accounts tend to have a very large geographical correlation with Greece, and there is currently no need to crawl trending topics for multiple locations.

The shorturl collection contains key-value pairs of shortened URLs and their corresponding target URL, as resolved by twAwler. To populate this collection, twAwler uses both the expanded URL provided for each shortened URL by twitter within each tweet, and also uses a set of crawler processes that interact with URL shortener services or directly with the web, to resolve shortened URLs.

The follow collection stores directed, timestamped follow edges as generated by the follow graph crawler. This amounts to a dynamic graph that approximates the follow relations between tracked users and their friends and followers, *i.e.*, it is a superset of the follow relation between tracked users, and includes their friends and followers regardless of whether their tweets are being tracked.

The greeks collection contains IDs (key) and screen names (not necessarily up-to-date) of users that have been classified as Greek-speaking. twAwler is parametric as to the criterion; the reported deployment is configured to classify users as Greek speakers and track when they satisfy any of the following conditions:

- Users with more than 100 tweets, of which at least 20% are in Greek.
- Users with more than 500 tweets, of which at least 10% are in Greek and their username and bio are in a set of common Greek names or written in the Greek alphabet.

Clearly, the Greek language having a unique alphabet aids significantly in recognizing Greek speakers. However, that is not central to twAwler, as Twitter's language recognition is very precise in several languages that use the latin alphabet. Conversely, twAwler marks users as non-Greek speakers and stops the crawler from following them and discovering new users through them, when it has crawled more than 500 of their tweets, of which less than 1% are in Greek. The author has found that these conditions succeed in classifying most users into either Greek-speaking or not, leaving only a small number of inconclusive users. twAwler applies an additional constraint to these, where if more than 30% of a user's followers and friends have already been classified as Greek-speaking, the inconclusive user is marked as Greek-speaking.

The suspended collection contains IDs and screen names of users that have been observed to be suspended by twitter. Screen names may not necessarily be up-to-date, as the account may have changed its screen name between the time it was last crawled and when it was suspended.

The ignored collection contains the IDs of users that have been dropped by the crawler using the criteria described above, or by manual curation.

The cemetery collection contains IDs (key) and screen names of users that have been seen to have deleted their account. Note that these users may reactivate their accounts at some point, at which they may be re-discovered and removed from the deleted user's collection. This is done only using user IDs and not screen names, as the latter may change or be taken by a different user in the mean time.

The crawlerdata collection contains enough information for the crawler to properly track the user without wasting any API requests. Namely, it contains the ID of the first and last tweets seen by the user, the earliest and latest times they have been crawled, whether their API limit of 3200 tweets in the past has been reached and thus only new tweets can be crawled, when the user profile and avatar picture was last downloaded, and when the user's favorites were last scanned.

3 Post-processing and Vectorization

In addition to user discovery and tracking, twAwler is able to mine the crawled data and produce a large set of features for every twitter user, to facilitate subsequent analysis. Tables 1 and 2 shows a list of user features produced. Many of these features have been previously documented and used in related work on various classification or generalization use cases. In implementing these analyses out-of-the-box, twAwler provides a framework for rapid prototyping of new research and rapid replication of existing work. The features include all information provided per-user by the twitter API, as well as features extracted from the tweets of each user, from the metadata of the tweets, and from a user's relations to other users. For use cases where it is important to follow the evolution of the data, twAwler includes tools to compute time-interval versions of all features for specific users or groups of users.

id	The twitter User ID.
screen_name	The user's last seen screen name.
screen_name_len	Length of the user's screen name [16].
screen_name_upper	Number of uppercase letters in the user's screen name [16].
screen_name_lower	Number of lowercase letters in the user's screen name [16].
screen_name_digit	Number of digits in the users' screen name [16].
screen_name_alpha	Number of letters in the users' screen name [16].
name	The name of the user.
name_len	Length of the user's last seen name [16].
name_upper	Number of uppercase letters in the user's name [16].
name_lower	Number of lowercase letters in the user's name [16].
name_digit	Number of digits in the user's name [16].
name_alpha	Number of letters in the user's name [16].
name_greek	Number of greek letters in the user's name.
created_at	The date the user first joined twitter [16].
tweet_count	Total tweet count, as reported by twitter [16].
favourites_count	Number of tweets this user has favorited, as reported by twitter [22].
followers_count	Number of users that follow this user, as reported by twitter [16].
friends_count	Number of users this user follows, as reported by twitter [16].
fr_fo_ratio	Ratio of friends to followers [16].
location	User's location as reported by the user [16].
has_location	Presence of the location field [16].
time_zone	User's time zone as reported by the user.
lang	The language this user selected for the twitter UI.
protected	Was the user protected the last crawl time.
verified	True if this is a verified user [22].
dead	True if this account was seen to be deleted.
suspended	True if this account was suspended the last time this user was crawled.
user_url	URL field of the user's profile, if any.
bio_words	Number of words in the user's profile.
bio_upper_words	Number of all-uppercase words in the user's bio.
bio_lower_words	Number of all-lowercase words in the user's bio.
bio_punctuation_chars	Number of punctuation characters in the user's bio.
bio_digit_chars	Number of digits in the user's profile.
bio_alpha_chars	Number of letters in the user's profile.
bio_upper_chars	Number of uppercase letters in the user's bio.
bio_lower_chars	Number of lowercase letters in the user's bio.
bio_greek_chars	Number of greek letters in the user's bio.
bio_total_chars	Length of the user's profile.
seen_total	Total number of this user's tweets used in computing this vector.
total_inferred	Total number of this user's tweets including encountered tweet IDs that could not be crawled and are probably deleted.
seen_greek_total	Total number of this user's tweets marked as being in Greek by the twitter API.
all_intervals	A histogram of time-between-tweets for all tweets seen.
seen_top_tweets	Seen tweets that were not retweets or replies.
top_tweets_pcnt	Percentage of seen tweets that were not retweets or replies.
top_intervals	A histogram of time-between-tweets for seen tweets that were not retweets or replies.

Figure 1: List of features per user (1/4)

mention_indegree	Number of users that mention this user.
mention_outdegree	Number of users mentioned by this user.
mention_inweight	Number of tweets that mention this user.
mention_outweight	Number of mentions by this user [22].
mention_avg_inweight	Average mentions of this user per mentioner.
mention_avg_outweight	Average mentions per mentioned user.
mention_out_in_ratio	Out-degree/in-degree ratio (mention reciprocity).
mention_pcnt	Percentage of seen tweets that are mentions.
most_mentioned_users	A list of the users most mentioned by this user, and the mention counts.
most_mentioned_by	Users that are seen to most mention this user, and the mention counts.
retweet_indegree	Number of users that have retweeted this user.
retweet_outdegree	Number of users this user retweeted.
retweet_inweight	Number of retweets of this user's tweets.
retweet_outweight	Number of retweets by this user [22].
retweet_avg_inweight	Average retweets per retweeter.
retweet_avg_outweight	Average retweets per retweeted user.
retweet_out_in_ratio	Out-degree/in-degree ratio (retweet reciprocity).
retweet_pcnt	Percentage of retweets in tweets seen by this user.
most_retweeted_users	A list of the users most retweeted by this user, and the retweet counts.
most_retweeted_by	A list of the users that most retweeted this user, and the retweet counts.
rt_intervals	Histogram of time-between-retweets by this user.
reply_indegree	Number of users that have replied to this user at least once [8].
reply_outdegree	Number of users to which this user replied at least once [8].
reply_inweight	Number of replies to this user [8].
reply_outweight	Number of replies tweeted by this user [22, 8].
reply_avg_inweight	In-Replies per in-degree [8].
reply_avg_outweight	Out-Replies per out-degree [8].
reply_out_in_ratio	Out-degree/in-degree ratio (replies sent for each received) [8].
replies_pcnt	Percentage of replies in tweets seen by this user.
most_replied_to	Users to whom this user most replied, and tweet counts.
most_replied_by	Users that most replied to this user, and tweet counts.
reply_intervals	A histogram of time-between-replies by this user.
seen_replied_to	Number of tweets that received replies from other users [22].
most_engaging_tweet	The tweet by this user that gathered most replies by other users.
plain_tweets	Number of tweets without hashtags, mentions, or URLs [22].
most_used_sources	List of twitter application clients used by this user and tweet counts per client, as reported by twitter.
time_between_any	Minimum, maximum, average, median, and standard deviation of times between any seen tweets [16].
time_between_top	Minimum, maximum, average, median, and standard deviation of times between seen tweets that are not retweets or mentions.
time_between_rt	Minimum, maximum, average, median, and standard deviation of times between seen tweets that are retweets.
time_between_replies	A histogram of time-between-replies by this user.
max_daily_interval	Minimum, maximum, average, median, and standard deviation of the largest interval between actions in any given day [10].
last_tweeted_at	Time and date of the last seen tweet by this user.

Figure 2: List of features per user (2/4)

life_time	Time from account creation to last seen tweet [22].
tweets_per_hour_of_day	Histogram of total seen tweets by this user for each hour of day [16].
tweets_per_weekday	Histogram of seen tweets by this user for each day of the week.
tweets_per_active_day	Minimum, maximum, average, median, and standard deviation of tweets per active day, and days of minimum and maximum.
tweets_per_day	Minimum, maximum, average, median, and standard deviation of tweets for every day from creation to now, and days of minimum and maximum.
last_month	A histogram of tweet counts seen during the last month of this user's seen activity, per hour of day.
fr_scanned_at	Last time this user's list of friends was crawled.
seen_fr	Total number of twitter users seen to be followed by this user at least once.
gr_fr	Number of friends marked Greek-speaking.
gr_fr_pcnt	Number of friends marked Greek-speaking.
tr_fr	Number of friends tracked.
tr_fr_pcnt	Percentage of friends tracked.
fo_scanned_at	Last time this user's list of followers was crawled.
seen_fo	Total number of twitter users seen to follow this user at least once.
gr_fo	Number of followers marked Greek-speaking.
gr_fo_pcnt	Percentage of followers marked Greek-speaking.
tr_fo	Number of followers currently tracked.
tr_fo_pcnt	Percentage of followers currently tracked.
fr_fo_jaccard	Jaccard similarity between friend and follower sets.
fr_and_fo	Size of the intersection of the friend and follower sets, *i.e.*, all reciprocal follow edges.
fr_or_fo	Size of the union of the friend and follower sets, *i.e.*, all neighbors in the follow graph.
gr_fr_fo	Number of all friends and followers marked as Greek-speaking.
gr_fr_fo_pcnt	Percent of Greek-speaking neighbors (friends and followers).
greek	True if this user was inferred to be Greek or Greek-speaking.
total_words	Total number of words in all seen tweets.
min_wptw	Minimum number of words per seen tweet.
avg_wptw	Average number of words per seen tweet.
med_wptw	Median number of words per seen tweet.
std_wptw	Standard deviation of the number of words per tweet.
unique_words	Number of unique words that this user has used in seen tweets.
lex_freq	Lexical frequency (the ratio of unique words over total words) over all seen tweets.
total_bigrams	Number of bigrams constructed from seen tweets.
unique_bigrams	Number of unique bigrams in seen tweets.
bigram_lex_freq	Lexical frequency (the ratio of unique bigrams over total bigrams) over all seen tweets.
articles	Number of times this user was seen using an article (article list mined from Greek Wiktionary)
pronouns	Number of times this user was seen using a pronoun (pronoun list mined from Greek Wiktionary)
expletives	Number of times this user was seen using an expletive (expletive list mined from Greek Wiktionary).

Figure 3: List of features per user (3/4)

locations	Number of times seen using the name of a place (location list mined from Greek Wiktionary).
emoticons	Number of times this user was seen using emoticons.
emoji	Number of times this user was seen using an emoji.
alltokens	All tokens of text seen in this user's tweets (words, hashtags, mentions, emoticons, etc.)
all_caps_words	Words seen to be in all-capital letters, excluding words of length 1.
all_caps_words_pcnt	Percentage of words in all-capital letters excluding words of length 1.
all_caps_tweets	Number of tweets seen to be in all-capital letters.
all_caps_tweets_pcnt	Percentage of seen tweets in all-capital letters.
all_nocaps_words	Number of words seen to have no capital letters.
all_nocaps_words_pcnt	Percentage of words seen to have no capital letters.
punctuation_chars	Number of characters that are punctuation in this user's seen tweets.
punctuation_pcnt	Percent of characters that are punctuation in this user's seen tweets.
total_chars	Total number of characters in seen tweets.
digit_chars	Total number of digits in seen tweets.
digit_pcnt	Percentage of characters in seen tweets that were digits.
alpha_chars	Total number of letters in this user's seen tweets.
alpha_pcnt	Percentage of letters in seen tweets.
upper_chars	Total number of uppercase letters in seen tweets.
upper_pcnt	Percentage of uppercase letters in seen tweets.
lower_chars	Total number of lowercase letters in this user's seen tweets.
lower_pcnt	Percentage of characters in seen tweets that were lowercase letters.
greek_chars	Total number of greek letters in this user's seen tweets.
greek_pcnt	Percentage of characters in seen tweets that were greek letters.
total_hashtags	Total number of hashtags this user's tweets [22].
hashtags_per_tw	Minimum, maximum, average, median, and standard deviation of hashtags per tweet [16, 22].
uniq_hashtags	Number of unique hashtags seen in tweets written by this user.
total_rt_hashtags	Total number of hashtags in tweets retweeted by this user.
uniq_rt_hashtags	Number of unique hashtags seen in tweets retweeted by this user.
most_common_words	List of most common words (excluding stop-words) used by this user, and their counts.
most_common_bigrams	List of most common bigrams (excluding stop-words) used by this user, and their counts.
most_common_hashtags	List of most common hashtags used by this user, and their counts.
most_common_rt_hashtags	List of most common hashtags in tweets retweeted by this user, and their counts.
most_common_urls	List of most common domain names in URLs found in this user's tweets, and their counts.
most_common_rt_urls	List of most common domain names in URLs found tweets retweeted by this user, and their counts.
seen_urls	Total number of URLs in all tweets [22].

Figure 4: List of features per user (4/5)

urls_per_tw	Minimum, maximum, average, median, and standard deviation of URLs per tweet [16, 22].
avg_edit_distance	Average edit distance between every posted URL and profile name [22].
daily_sentiment	Average positive and negative sentiment per tweet mentioning each sentiment, per day, as timeseries [2].
entity_overlap	A graph of entities. Node weights are counts of tweets mentioning each entity, edge weights are counts of tweets mentioning both entities.
senti_entities	List of average positive and negative sentiment scores for all seen tweets mentioning an entity, per entity.
favoriters	Number of users seen to have liked a tweet by this user.
favorited	Number of users whose any tweet this user has liked.
most_favoriters	A list of users that have liked this user's tweets the most, and the number of likes per user.
most_favorited	A list of users whose tweets this user has liked the most, and the number of likes per user.
lexical_gender	A struct of two numbers: Percentages of self references that are done using the male and female gender of the word.
number_of_languages	Number of languages used by this user [19].
tweets_per_language	Number of tweets for the five languages most used.
vector_timestamp	The UTC timestamp of this vector (this is used by the crawler engine for caching user vectors).

Figure 5: List of features per user (5/5)

Graph	$\|V\|$	$\|E\|$	Size	Directed	Weighted
Follow	26,339,971	204,969,957	4,2G	Yes	No
Retweet	3,851,055	46,084,224	1,0G	Yes	Yes
Mention	10,082,741	101,106,215	2,4G	Yes	Yes
Reply	4,552,175	24,364,103	552M	Yes	Yes
Quote	1,279,360	4,794,911	112M	Yes	Yes
List Sim	54,309,000	1,993,937,542	44G	No	Yes
Favorite	2,778,775	38,881,162	893M	Yes	Yes

Figure 6: Graphs mined from data

Graph	Density	Reciprocity	Assortativity	Transitivity	Radius	Girth	Diameter
Follow							
Retweet							
Mention							
Reply							
Quote							
List Sim							
Favorite							

Figure 7: Graphs mined from data

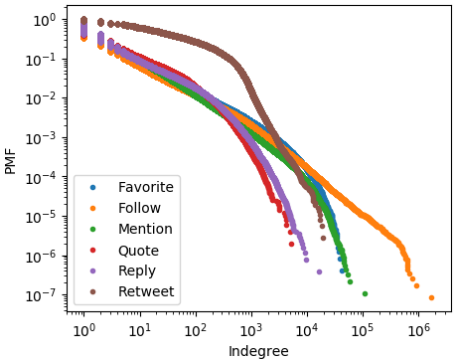


Figure 8: Distribution of in-degree

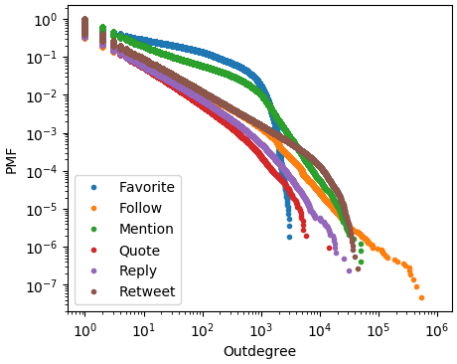


Figure 9: Distribution of out-degree

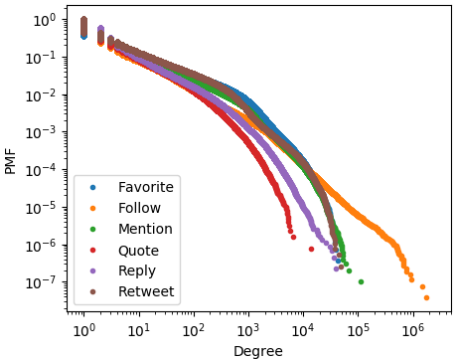


Figure 10: Distribution of degree

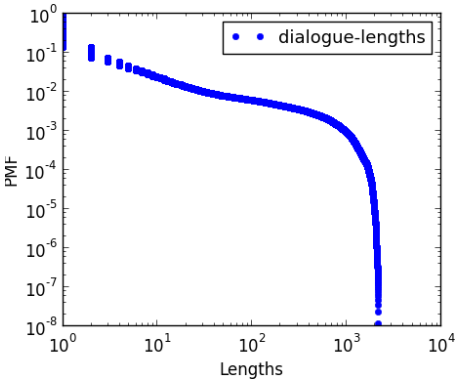
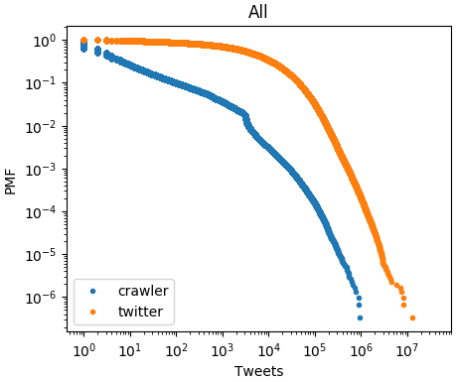


Figure 11: Distribution of thread lengths



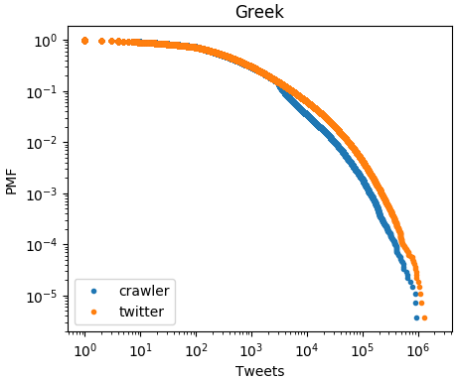


Figure 12: Distribution of tweets per user

4 Data Processing and Graph Extraction

To assist with social network analytics applications, twAwler performs additional post-processing of the harvested data to generate a set of graphs. In addition to the Follow Graph, stored as a dynamic graph of timestamped edges, twAwler also scans tweets to generate a set of graphs among users, namely the Retweet Graph, Mention Graph, Reply Graph, Quote Graph; all are directed and weighted, where each edge's weight is the number of observed interactions of the corresponding type, from the source user to the target user. Moreover, twAwler mines the list and list membership data crawled to generate the List Similarity Graph; it is an undirected graph where an edge between two users amounts to membership of the same list, and an edge's weight is the number of lists that include both users. Finally, using the crawled favorites per user twAwler extracts the user-to-user Favorite Graph.

Figure 16 shows the size of the mined graphs in vertices and edges, their size on disk, as well as whether they are directed or weighted. Figures 8, 9 and 10 presents the in-degree, out-degree, and undirected degree for the interaction graphs. The list similarity graph is not included, as it could not be easily analyzed by twAwler using a single machine. Note that the directionality of the graphs follows action, which in the case of retweets is not the direction information flows. That is, an edge in the Follow Graph points from the follower to the followee, an edge in the Retweet Graph points from the retweeter to the tweeter, an edge in the Mention Graph points from the mentioner to the mentioned, an edge in the Reply Graph points from the replier to the replied-to, an edge in the Quote Graph points from the quoter to the quoted, and an edge in the Favorite Graph points from the favoriter to the favorited.

To evaluate the coverage twAwler achieves for the crawled users, Figure 12 compares the distribution of tweets per user as reported by twitter in the user information returned for each user, with the total count of tweets crawled and saved by twAwler. Even though twAwler worked for a brief duration compared to the active period of most users, it was able to discover and crawl a very large percentage of the tweets of even the most prolific users. Part of this is possible for active users because twAwler follows retweets, replies, and quotes to the past and discovers very old tweets that are not otherwise reachable using the standard API.

As a simple use case for twAwler's usability, Figure 11 shows a simple computation of the maximum lengths for all threads of replies found starting with tweets by users marked as Greek-speaking. The distribution is extremely skewed, with 86% consisting of a tweet and a single reply and 6.3% consisting of two replies, while the single longest thread has length 2185. Interestingly, most of the participants in that thread seem to have been suspended, hinting towards the possibility of automated behavior.

As another simple use case, we applied the algorithm proposed by Wu *et al.* to detect and classify elite users. We extended the categories proposed to nine, namely Brands, Celebrities, Politicians, Political Parties, Newsmedia, Journalists, Organizations, and Government. Figure 13 shows the retweets be-

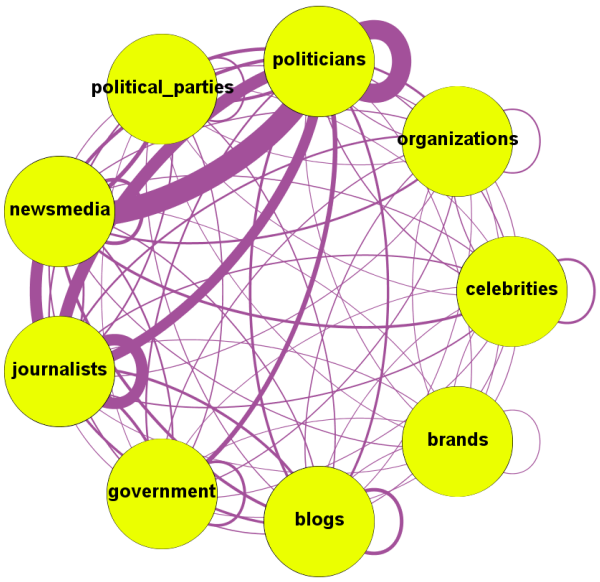


Figure 13: Retweets among User Groups

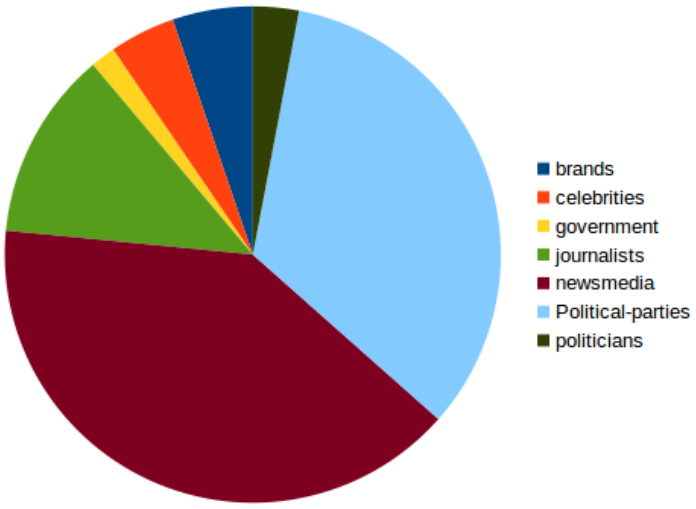


Figure 14: Percentage of elite users per category

Figure 15: Clusters found in the graph of elite users

Graph	$\|V\|$	$\|E\|$	Size	Directed	Weighted
Follow	26,339,971	204,969,957	4,2G	Yes	No
Retweet	3,851,055	46,084,224	1,0G	Yes	Yes
Mention	10,082,741	101,106,215	2,4G	Yes	Yes
Reply	4,552,175	24,364,103	552M	Yes	Yes
Quote	1,279,360	4,794,911	112M	Yes	Yes
List Sim	54,309,000	1,993,937,542	44G	No	Yes
Favorite	2,778,775	38,881,162	893M	Yes	Yes

Figure 16: Graphs mined from data

tween elite users in the corresponding groups, and Figure 14 shows the percentage of elite users classified in each category. To evaluate whether the proposed method was able to find true elite users, even though the categories we used were more than doubled, we computed the PageRank on the complete Follow graph[2], and found that in all categories, 95% of the elite users inferred by the algorithm were within the top 1% of the PageRank-sorted nodes of the follow graph. This confirms the observation by Wu *et al.* that elite users can be detected using list membership and are responsible for a disproportionally large part of the follow edges.

Finally, we extracted all elite user features, including the corresponding parts of the retweet, follow, and reply graphs. Figure 15 shows the clusters detected on the retweet graph of elite users, as retweet graphs have been previously found to have high homophily. Again, we found that clusters in the retweet graph amount to proximity in domain, by manual inspection. In categories of elite users where this was not the case, *i.e.*, politicians, political parties, and blogs, we noticed by sampling a small set of the elite users and manually inspecting them on the graph, that clustering seems to capture ideological proximity rather than domain proximity, although confirming this hypothesis is a matter of future work.

5 Related Work

There is a lot of related research focusing on the twitter social network; this section presents representative samples of such work and discusses how `twAwler` compares or advances the state of the art.

TwitterEcho [6] is a Twitter crawler focused on discovering and crawling small communities. The authors design a targeted crawler and use it to recognize and track Portuguese accounts. TwitterEcho also prioritizes account crawling by ordering users according to their activity patterns, opting to crawl active users more frequently. TwitterEcho is a distributed system requiring multiple clients to crawl and aggregating the results into a single server. `twAwler` can also be used in this fashion for large communities, but we found that for communities having on the order of 100,000 to 200,000 active users like the Greek-speaking twitter users, one computer suffices. TwitterEcho collected more than 14 million tweets from 100,000 users within 11 months. `twAwler` manages to use many more available endpoints from the twitter REST API, crawling, in addition to tweets and user information, the follow graph, the favorites of all tracked users, as well as list ownership, subscription, and membership information. `twAwler` uses the fact that the Greek alphabet suffices to recognize the language, while TwitterEcho resorts to a more complex language classifier to separate Portuguese from Brazilian.

There is a very large body of literature on feature extraction from twitter content, most of which uses the features to perform classification. `twAwler`

[2]We used a Spark+GraphX cluster of 8 32-core nodes to compute PageRank on the full Follow graph.

is currently able to efficiently extract a very large subset of all the features mentioned in the following papers. Sakaki *et al.* [18] augment gender classification with image processing, classifying images as one of cartoon/illustration, famous person, food, goods, memo/leaflet, outdoor/nature, person, pet, screenshot/capture, and other. They use a "bot detection" criterion of whether a user has less than 150 tweets made using a mobile or web client. Liu and Ruths [13] use first name, as reported by the user, for gender inference. We expect that this feature is highly language-dependent, and will perform even better in the Greek language, where names, nouns, etc, are gendered. Zhang *et al.* [24] use content and interaction features to construct a model for classifying twitter users into age groups. Uddin *et al.* [22] use a wide set of features extracted from user information to classify users into six categories: personal users, professional users, business users, spam bots, news feed bots, and viral marketing bots. Hu *et al.* [11] correlate twitter and linkedin data to classify users into the categories: marketing, administrator, start-up, editor, software engineer, public relations, office clerk, or designer. The authors perform sentiment analysis and present sentiment results on the Pearson scale for each class. Pennacchiotti and Popescu present two versions of the same work [16, 15] in which they use features in four classes to classify users, namely user profile, user tweeting behavior, linguistic content of user messages, and user social network features; and employ a set of hand crafted regular expressions to mine ethinicty and gender.DARPA has organized a competition on bot detection [19]. All teams used an array of features, where the winning team managed to create visualizations that assisted in rapidly recognizing bots. The competition report describes multiple machine-learning techniques used to cluster users and find bot outliers, detect bot-to-bot similarities, etc.

The study of user activity over time has been studied in previous work, showing that extraction of timeseries data from content may offer valuable insights into user behavior. `twAwler` mines several timeseries from user activity, including daily sentiment per entity or in total, idle time intervals, etc., facilitating further experimentation in that direction. Paraskevopoulos *et al.* [14] use the time series of twitter activity to correlate users without location information, with already geotagged users. Ferraz *et al.* [10] study inter-action time intervals in twitter activities and create a theoretical model that closely explains and reproduces observations. They use their model to discover outliers and detect bots. Bild *et al.* [4] focus on the Retweet Graph and reason about the effects of sampling on a set of metrics such as the distributions of tweets per user, tweet rates, and inter-event time intervals. They find that the Retweet Graph is small-world and scale-free similarly to the follow graph, but with stronger clustering.

Backstrom *et al.* [3] study the evolution an growth of communities in two online social networks, DBLP and LiveJournal. They use a set of per-community features and relations between communities and individual nodes, and find that community growth depends not only on proximity but also on community density, and the density of each node's neighborhood. Cattuto *et al.* [7] use the Neo4J graph database to analyze a time-dependent social graph generated us-

ing wearable sensors that captured in-person meetings of all the participants. They were able to execute several complex queries with a temporal component over the graph and report scalability issues as the representation of the dynamic graph (as a static graph) tends to become very dense over time. Cheng *et al.* [8] study the twitter mention graph and predict reciprocity of communication, using a combined approach of decision trees and regression models, on a wide set of graph features extracted for the neighborhood of each vertex. They find social status difference to be a very important predictor of reciprocity.

Twitter lists have been used as crowdsourced similarity metrics in the past, interpreting the fact that twitter users independently classify other users into their own lists. `twAwler` extracts and processes list membership information and enables multiple similar use cases to be explored. Kim *et al.* [12] use twitter lists to recognize representative words for all of the list traffic and associate these words with list members. They also mine keywords from the list names as representative for the list members, even if these words are not used by the members. Wu *et al.* [23] use Twitter lists to recognize elite from ordinary users and study the usage patterns and content posted in each class. They use twitter lists to recognize elite users via their list-similarity with exemplar accounts. Culotta and Cutler [9] present a method for computing brand perception as a set of metrics of similarity to Entities. They mine entity definitions using twitter lists to identify characteristic accounts, and use Jaccard similarity on follower sets to compute a distance metric from chosen entities.

Often it is useful to analyze content in a non-user-centric way, to observe propagation patterns or orchestrated behavior, as is the case in bot detection work. Although `twAwler` performs user-centric crawling and aggregation of data, it includes tools to extract subsets of the reply, quote, mention, and retweet graphs conditional on time intervals or specific content, that allow detailed monitoring of information propagation as used in related work. Ratkiewicz *et al.* [17] search for astroturfing, or orchestrated campaigns appearing to be grassroots movements in order to influence opinion. They use hashtags, mentions, URLs and the entire text of every tweet as "memes" and look into propagation patterns in diffusion networks. They use a set of features per meme to classify memes, and assign six GPOMS sentiment dimensions [5]: calm, alert, sure, vital, kind, happy. Tsur and Rappoport [21] analyze use of hashtags in tweets and create a model that predicts the popularity of hashtags based on features like length, capitalization, abbreviations, and number of keystrokes required to type. Anderson *et al.* [1] study user similarity metrics to predict evaluations and election results. They capture both content similarity and similarity of user interactions, and find that relative social status affects how similarity influences user opinions.

Previous work has focused on greek-speaking twitter users, although at a much smaller scale, and focusing on tweets related to specific events, containing specific keywords or hashtags, etc. In comparison, `twAwler` targets users instead of tweets, allowing researchers to study specific events in the proper context of existing user relations and interactions. Antonakaki *et al.* [2] present an analysis of the Greek 2015 referendum and parliamentary elections. They use a stemmer

for word matching, and a lexicon-based sentiment analysis to assign sentiment to LDA topics. They produced an accurate prediction of the ballot results by counting number of tweets and tweet sentiment per topic, and assigning topics to outcomes. twAwler uses the same sentiment dictionary, kindly provided by the authors, to extract sentiment features per user and per entity. Theocharis *et al.* [20] analyze twitter content related to social movements and classify tweets into categories according to their intent. They find a small part of tweets are related to actions and organization, with most being reports about the events and conversations.

6 Conclusions

twAwler is a lightweight, user-centric crawler that targets twitter communities based on language. It satisfies twitter crawler constraints and does not require using multiple crawling accounts. Furthermore, it includes post-processing primitives that facilitate data analysis especially with respect to mining relationship graphs, as well as extraction of data in a quasi-anonymized (user id-only) form that can be shared or published without violating twitter's terms of use. twAwler can assist SNA researchers in gathering data from twitter without violation of its terms. It also facilitates the quick testing of hypotheses or quick replication of previous work by including a wide set of primitives and common computations, out-of-the-box. Although twAwler aims to be lightweight and can be easily deployed on a single machine, further analysis of the data may, however, require more resources. We demonstrate its effectiveness by easily replicating a set of tests found in related work.

References

[1] Ashton Anderson, Daniel Huttenlocher, Jon Kleinberg, and Jure Leskovec. Effects of user similarity in social media. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 703–712. ACM, 2012.

[2] D. Antonakaki, D. Spiliotopoulos, C. V. Samaras, S. Ioannidis, and P. Fragopoulou. Investigating the complete corpus of referendum and elections tweets. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 100–105, Aug 2016.

[3] Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 44–54. ACM, 2006.

[4] David R Bild, Yue Liu, Robert P Dick, Z Morley Mao, and Dan S Wallach. Aggregate characterization of user behavior in twitter and analysis of the

retweet graph. *ACM Transactions on Internet Technology (TOIT)*, 15(1):4, 2015.

[5] Johan Bollen, Huina Mao, and Xiaojun Zeng. Twitter mood predicts the stock market. *Journal of computational science*, 2(1):1–8, 2011.

[6] Matko Bošnjak, Eduardo Oliveira, José Martins, Eduarda Mendes Rodrigues, and Luís Sarmento. TwitterEcho: a distributed focused crawler to support open research with twitter data. In *Proceedings of the 21st International Conference on World Wide Web*, pages 1233–1240. ACM, 2012.

[7] Ciro Cattuto, Marco Quaggiotto, André Panisson, and Alex Averbuch. Time-varying social networks in a graph database: A neo4j use case. In *First international workshop on graph data management experiences and systems*, page 11. ACM, 2013.

[8] Justin Cheng, Daniel M Romero, Brendan Meeder, and Jon Kleinberg. Predicting reciprocity in social networks. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 49–56. IEEE, 2011.

[9] Aron Culotta and Jennifer Cutler. Mining brand perceptions from twitter social networks. *Marketing Science*, 35(3):343–362, 2016.

[10] Alceu Ferraz Costa, Yuto Yamaguchi, Agma Juci Machado Traina, Caetano Traina Jr, and Christos Faloutsos. RSC: Mining and modeling temporal activity in social media. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 269–278. ACM, 2015.

[11] Tianran Hu, Haoyuan Xiao, Thuy-vy Thi Nguyen, and Jiebo Luo. What the language you tweet says about your occupation. *ICWSM*, 2016.

[12] Dongwoo Kim, Yohan Jo, Il-Chul Moon, and Alice Oh. Analysis of twitter lists as a potential source for discovering latent characteristics of users. In *ACM CHI workshop on microblogging*, page 4, 2010.

[13] Wendy Liu and Derek Ruths. What's in a name? using first names as features for gender inference in twitter. In *AAAI spring symposium: Analyzing microtext*, volume 13, page 01, 2013.

[14] Pavlos Paraskevopoulos, Giovanni Pellegrini, and Themis Palpanas. When a tweet finds its place: fine-grained tweet geolocalisation. In *International workshop on data science for social good (SoGood), in conjunction with the European conference on machine learning and principles and practice of knowledge discovery (ECML PKDD)*, 2016.

[15] Marco Pennacchiotti and Ana-Maria Popescu. Democrats, republicans and starbucks afficionados: user classification in twitter. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 430–438. ACM, 2011.

[16] Marco Pennacchiotti and Ana-Maria Popescu. A machine learning approach to twitter user classification. *Icwsm*, 11(1):281–288, 2011.

[17] Jacob Ratkiewicz, Michael Conover, Mark R Meiss, Bruno Gonçalves, Alessandro Flammini, and Filippo Menczer. Detecting and tracking political abuse in social media. *ICWSM*, 11:297–304, 2011.

[18] Shigeyuki Sakaki, Yasuhide Miura, Xiaojun Ma, Keigo Hattori, and Tomoko Ohkuma. Twitter user gender inference using combined analysis of text and image processing. *V&L Net*, 2014:54, 2014.

[19] VS Subrahmanian, Amos Azaria, Skylar Durst, Vadim Kagan, Aram Galstyan, Kristina Lerman, Linhong Zhu, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. The darpa twitter bot challenge. *Computer*, 49(6):38–46, 2016.

[20] Yannis Theocharis, Will Lowe, Jan W Van Deth, and Gema García-Albacete. 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(2):202–220, 2015.

[21] Oren Tsur and Ari Rappoport. Don't let me be# misunderstood: Linguistically motivated algorithm for predicting the popularity of textual memes. In *ICWSM*, page 426, 2015.

[22] Muhammad Moeen Uddin, Muhammad Imran, and Hassan Sajjad. Understanding types of users on twitter. *arXiv preprint arXiv:1406.1335*, 2014.

[23] Shaomei Wu, Jake M Hofman, Winter A Mason, and Duncan J Watts. Who says what to whom on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 705–714. ACM, 2011.

[24] Jinxue Zhang, Xia Hu, Yanchao Zhang, and Huan Liu. Your age is no secret: Inferring microbloggers' ages via content and interaction analysis. In *ICWSM*, pages 476–485, 2016.