Where Do Bots Come From? An Analysis of Bot Codes Shared on GitHub

BENCE KOLLANYI[1]
Corvinus University of Budapest, Hungary

An increasing amount of open source code is available on the Internet for quickly setting up and deploying bots on Twitter. This development of open-source Twitter bots signals the emergence of new political economies that redistribute agencies around technological actors, empowering both the writers of the bots and users who deploy a bot based on the shared code. However, we have no systematic knowledge about how these bots are produced or what role sharing and collaboration play in this process. This article examines the practice of writing and sharing bot codes on GitHub, currently the largest online code repository, by analyzing data about more than 4,000 public Twitter bot code repositories from 2008 to 2016. The data reveal an exponential growth in the number of Twitter bots and a strong dominance of U.S.-based bot developers. Based on data gathered from GitHub about the social aspect of programming on the platform, the article also discusses how developers collaborate and interact with one another on GitHub around bot codes.

Keywords: Twitter, GitHub, bot, open source, programming, social media, automation

Twitter bots are computer programs that control Twitter accounts and usually run on Internet-connected servers. Bots use an application programming interface (API) to communicate with these servers on Twitter. These APIs both provide some level of access to Twitter's public data and allow third-party applications, including bots, to control Twitter accounts. For example, bots on Twitter can post messages on their own, follow or unfollow users, set up polls, post a picture, or send direct messages to specific users. Twitter's open API policy has been an important factor in the platform's success; it has led to a lively scene of Twitter-based clients and other applications and a general abundance of Twitter bots. According to Makice (2009), Twitter's APIs were made open from the beginning of the service to encourage developers to come up with Twitter-based applications, from fancy data visualization projects to widgets to mobile applications.

Bence Kollanyi: kollanyi@gmail.com
Date submitted: 2016–07–31

[1] The author gratefully acknowledges feedback received from the Algorithms, Automation and Politics workshop organized by the European Research Council–funded Computational Propaganda project of the Oxford Internet Institute and held as a preconference to the International Communication Association meeting in Fukuoka, Japan, in June 2016. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the European Research Council.

Bots follow predefined rules and often act without human intervention. The rules do not have to be complicated. For example, some bots are coded to do nothing but repeat the same message or set of messages over and over again. Such mechanical repetition was, in fact, a typical activity pattern of early Twitter bots. Such bots have come to be seen as undesirable spam, and regulations have been put in place in Twitter's terms of service that give the company the right to suspend spam bots. Because mechanical repetition is relatively easy to recognize from message patterns, regular Twitter users also often report these bots as spam. These automated accounts are often detected by one of Twitter's internal algorithms.

Other bots automate human communication and action more convincingly: They do things on Twitter that could have been done by a human user. These bots are able to do the same tasks but on a larger scale, in less time, and without human intervention.

Overview of Research on Twitter Bots

The growing phenomenon of social media automation, and the use of bots on Twitter in particular, triggered a strong response from the information security research community in 2010. Many of the early publications documented the efforts of detecting automated Twitter accounts to prevent commercial spam and to filter out tweets with links pointing to malicious websites (Chu, Gianvecchio, Wang, & Jajodia, 2010; Lee, Eoff, & Caverlee, 2011). To detect Twitter bots, researchers have focused on various aspects of Twitter activity, such as analysis of the sender–receiver relationship (Song, Lee, & Kim, 2010) and behavioral patterns. The methods include machine learning (Ratkiewicz et al., 2011; Wang, 2010) and the use of honeypots to trap Twitter bots (Lee, Caverlee, & Webb, 2010). For a more detailed overview of the various methods to detect bots on Twitter, see Ferrara, Varol, Davis, Menczer, and Flammini (2015).

Twitter, like any other major social media platform, has deployed sophisticated algorithms to fight spam. Lin and Huang (2013) have studied spam accounts that were able to survive for an extensive period of time without being caught by these algorithms. The literature has also addressed a new class of more sophisticated social bots best described as software agents mimicking humans, which are harder to detect and combat (Ferrara et al., 2015).

Twitter bots are often perceived as content polluters or threats to security. However, not all Twitter bots are evil. Chu and his colleagues (2010) deem automated programs on Twitter double-edged swords. While some bots serve legitimate purposes, such as delivering news or automatically generated updates for a service, others only spread spam.

Beyond detection, communication scholars have examined the credibility of bots. Findings suggest that, in certain circumstances, social media users perceive Twitter bots as credible sources of information (Edwards, Edwards, Spence, & Shelton, 2014). Other scholars study so-called robot journalism, or the use of news bots in the various phases of content creation: identifying newsworthy events, curating, analyzing data, and even writing (Lokot & Diakopoulos, 2015; Steiner, 2014).

Researchers from the fields of information security, communication studies, and political science have studied the use of automated social media accounts for political purposes. This line of research investigates how political actors can gain political advantages by deploying bots. Ratkiewicz and colleagues (2011) have described Twitter as a battlefield for modern politics, where political actors can use computer codes to create the illusion of an artificial grassroots movement (astroturfing) or conduct smear campaigns. Mustafaraj and Metaxas (2010) gathered evidence about using automated accounts during a Massachusetts Senate race. The authors analyzed how messages from Twitter accounts could influence search engine results, a technique known as Twitter-bombing. Other research projects used Twitter data collected during U.S. elections to study and report on the use of automated accounts to retweet content linked to specific political groups (Lumezanu, Feamster, & Klein, 2012). More recently, Woolley (2016) provided an overview of incidents where social bots were deployed by powerful political elites during global political events.

GitHub: The Largest Online Code Repository

Previous research about Twitter bots looks primarily at the outcome of bot activity in order to better regulate the platform’s bot ecosystem. This research works to detect and categorize bots into those that are desirable and those that are unwanted, or to organize automation along similar simple distinctions. Although this approach has a strong focus on detection and description of bots already deployed on Twitter, such research has not studied the development of the code and the actors behind the bots. Where do bots come from? With this question in mind, this article explores bot creation using the rich data, including written and shared bot code, available on GitHub.

GitHub is currently the largest online repository for shared computer code. For open-source developers, it is a means of collaboration and sharing work. Because of this, GitHub is a good place to study open-source Twitter bot codes and the social arena around bot development.

GitHub has always advertised itself as an online hosting service with version control designed to support the collaborative development of software. In the early days of GitHub, one of its advantages, at least compared to other code sharing and version controlling solutions, was indeed the social functions on the website. For example, GitHub users are able to follow one another’s work on the platform. They can also acknowledge programs written by other developers, a practice known as starring a repository. Developers can create a copy (fork) of another user’s repository and work on this version independently. The changes in the forked repositories can be later integrated in the original repository.

Researchers have studied the social aspect of coding on GitHub with both qualitative and quantitative methods. These findings are relevant to the problem of understanding how users learn about Twitter bots through social exchanges. Margaret-Anne Storey and colleagues (2014) conducted a survey on GitHub by contacting 7,000 active users. They found a new generation of developers who can be characterized by a high use of social media for communication, coordination, collaboration, and, perhaps most importantly, learning. By providing social tools for sharing code, GitHub effectively lowers the barriers to joining and contributing to development communities.

Dabbish, Stuart, Tsay, and Herbsleb (2012) provide a good example of qualitative research on GitHub. The researchers use in-depth interviews to understand how social dynamics influence the ways in which individual developers engage with code. They look at how users follow other developers' work, and whether this kind of transparency—the public nature of working on GitHub—leads to learning and increased collaboration among developers. The researchers conclude that developers tend to produce cleaner code and commit changes less often if many people are watching their projects. However, they also posit that coders experiment less under these conditions due to this social pressure. Nevertheless, Dabbish and colleagues find that GitHub users ultimately learn from one another by watching how others solve similar problems.

A growing body of literature based on GitHub data explores how developers use the platform's collaborative code repository. The availability of data and of big data technologies for collecting and distributing vast amounts of data and the emergence of methods to overcome the data collection limits imposed by GitHub all account for the rise, and general popularity, of such research.

Lima, Rossi, and Musolesi (2014) look at both the network-like functions of GitHub and the collaborative aspects of the platform. Their project is based on a very large data set and finds that social ties have a relatively low level of reciprocity when compared to other platforms. Furthermore, active users do not necessarily have large social networks on the site. Although there are some highly visible cases of collaboration on GitHub, such as the construction of the Linux source code helmed by Linus Torvalds, Lima and colleagues suggest that the strength of the site lies in fostering a social learning process wherein users can look at and appropriate published code for their own purposes.

GitHub provides easy access to its repositories, the shared code, and to metadata about repositories through an API. The metadata available through the API include the number of contributors, some measures of popularity of the repository (stars and forks), and information about which programming language the code was written in. Additional information is available on users: their location, the date they joined the platform, the number of repositories or programming codes they have shared, and their contribution to other repositories. The site has a more liberal API policy than many other websites both in terms of the number of API calls that a user can initiate and the level of detail of the data. GitHub users generate a large amount of data. According to Gousios (2013), GitHub records 200,000 events on an average day, or more than 8,300 events per hour.

Data-driven research projects also look at the choice of programming language by developers (Sanatinia & Noubir, 2016), the geography of the open-source community (Takhteyev & Hilts, 2010), and the use of the platform in educational settings (Zagalsky, Feliciano, Storey, Zhao, & Wang, 2015). For a more detailed overview of GitHub-related software research, see the review by Kalliamvokou et al. (2014).

Alongside research projects investigating specific questions, the collection and curation of data from GitHub is an important research endeavor on its own right. There are at least three major attempts to download every GitHub repository for the purpose archiving and research (Gousios, 2013; Grigorik, 2012; Sanatinia & Noubir, 2016). Researchers behind the GitHub Archive and GHTorrent data set have decided to make their data sets public and share them with the research community in order to facilitate further research. They claim that the data available about GitHub users is still largely underexplored in terms of academic publications (Gousios, Vasilescu, Serebrenik, & Zaidman, 2014).

This article follows the quantitative approaches described above in terms of data collection and analysis. It is unique, however, in that it focuses on a specific domain of programming: the development of codes for running bots on Twitter. The aim of the article is to look at the available open-source bot codes, and the metadata about the development of this code, to better understand both the code-related mechanics behind the bots and the social aspects of bot development.

Method

Working with online social data, especially when accessing collectively generated data through multiple API calls, is in many ways an exploratory and iterative process. It is also a grounded process: Research questions are formulated as the researcher becomes familiar with the data. In other words, initial collection and analysis is often geared toward exploring the questions a researcher can ask from a cursory exploration of the data. A cyclical and systematic process of data interrogation creates context for further analysis.

One of the GitHub APIs allows users to obtain a list of repositories that contain specific search terms in their name, description, or in their readme file. First, an API call found the GitHub repositories containing both the words *Twitter* and *bot*. This initial search provided 3,783 results, and an additional search for the term *Twitterbot* as one word led to another 764 results. After combining the two data sets and removing the duplicates (the overlap between the two terms), the final data set contained 4,106 repositories.[2]

Although the most important information about the owners of the repositories is readily available from the search API, further information can be obtained only with separate API calls, one for each repository. These extra API calls are relatively simple to generate because they follow the same syntax for every repository. This additional step in the data collection process revealed important insights about the users, such as the time of registration, location, the number of public repositories, and how many followers they have. More requests could provide additional details, such as the readme file for each repository or the user names of those who forked the repository on GitHub.

Work with GitHub data began in early 2016, though all data analyzed for this article were collected in April 2016. To access the above-mentioned data through the GitHub API and to parse the information from the API response, the Python programming language was used. Most of the data processing and the analysis was also done in Python, while the data visualization was done with Tableau.

[2] By default, the GitHub search API shows only original repositories and excludes forks. By adding the forked versions—the copies of the original code that may have been modified—the number of bot repositories would be higher than 6,000. This research proceeds with the initial data set, which does not include forks.

API calls need to follow a specific syntax, which has an extensive documentation on GitHub. To learn the syntax and understand the extent and context of the available data, three distinct sources were used: the documentation of the API, the structure of the information on the website, and the data available by querying the API.

1. GitHub, similar to many bigger websites, provides an extensive documentation for its API. The language of the documentation is mainly geared toward developers, and it describes the right syntax to use to access the data. It is also a useful tool for gaining insight into the kind of questions a social scientist may ask of the data.

2. Using the interface of GitHub was also helpful for understanding the mechanics of the platform. To this end, GitHub hosted the code for this research. I regularly browed the various repositories created by others and looked at the interfaces of the site to help contextualize the data. Due to the liberal API policy of GitHub, almost every piece of information available on the website is accessible by different API queries, so studying the website also helps to form ideas about the kinds of questions that can be asked from the API.

3. The research strategy was to start with a smaller data set of bot repositories, and analyzing the data allowed me to learn about the limitations of the API, the challenges of data collection, and the possibilities of extending the range of data collected.

After developing a strategy for downloading the data, it was important to decide how to store information. Figure 1 is an overview of the iterative process of data collection and data storage that resulted in the final data set. This process began with the initial data set of 4,106 repositories containing the words *Twitter* and *bot* or *Twitterbot* in their name, description, or readme file. The data set was then extended using additional targeted API calls to GitHub. Using the unique identifier GitHub provided for each repository made it simple to add new entries and integrate extra information into the data set. It also makes it possible to update the data set with newer bot repositories in the future.

Results

The Exponential Growth of Twitter Bot Repositories on GitHub

GitHub was launched on April 10, 2008, and Twitter bots have been present from the very beginning of the site's history. The oldest bot repository found was created only four days after GitHub's official launch date. The number of Twitter bots has been growing quickly. In GitHub's first two years, 2008 and 2009, almost 100 different bot codes were published. The number has been growing ever since, and by 2013 it reached 1,000. In recent years, the number of bots has quadrupled, and at the time of this study (April 2016) there are more than 4,000 repositories. Note that deleted bot repositories no longer show up in the search.

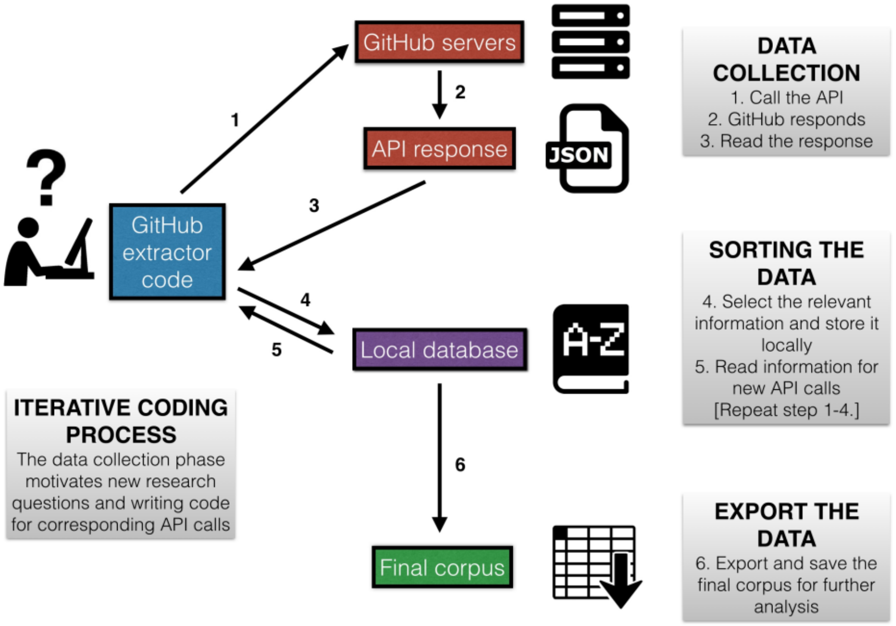
GitHub servers
API response
JSON
GitHub extractor code
Local database
A-Z
Final corpus
1
2
3
4
5
6
DATA COLLECTION
1. Call the API
2. GitHub responds
3. Read the response
SORTING THE DATA
4. Select the relevant information and store it locally
5. Read information for new API calls
[Repeat step 1-4.]
EXPORT THE DATA
6. Export and save the final corpus for further analysis
ITERATIVE CODING PROCESS
The data collection phase motivates new research questions and writing code for corresponding API calls

Figure 1. The structure of the data collection.

Some online services collect automatic location information based on the IP address of their users or by geotagging data with location shared by a phone or tablet. GitHub uses a free-text question in the user profiles, so location data is based on self-declared information. This choice has numerous consequences for the quality, reliability, and usability of the location information on GitHub. First, users have the option to not declare any location information. In this data set, there are 1,443 bot repositories with no location information. Second, even if a user provides a location, there is no guarantee that it is truthful. Finally, location information may change over time, but the API only stores current information, so there is no way to know what the information was at the time a repository was created.

Besides missing, false, and outdated information, location data in the user profiles do not follow a strict syntax with street address, city, postal code, and country name. A developer living in New York may add this information in various ways, from a properly formatted street-level address to just typing NY or NYC. Google Places was used to interpret this information. Google Places is a robust system for coding geographical information, and with some limitations it is free to use for geocoding data. Google Places yielded proper geodata with longitude and latitude coordinates for almost every valid location. Some of the original location data on GitHub had street-level details, while others specified only the country. Another geocoding API was used to obtain the address in a structured format, allowing for identification of the country name for each repository. The results of the geocoding process are presented in Figure 2.

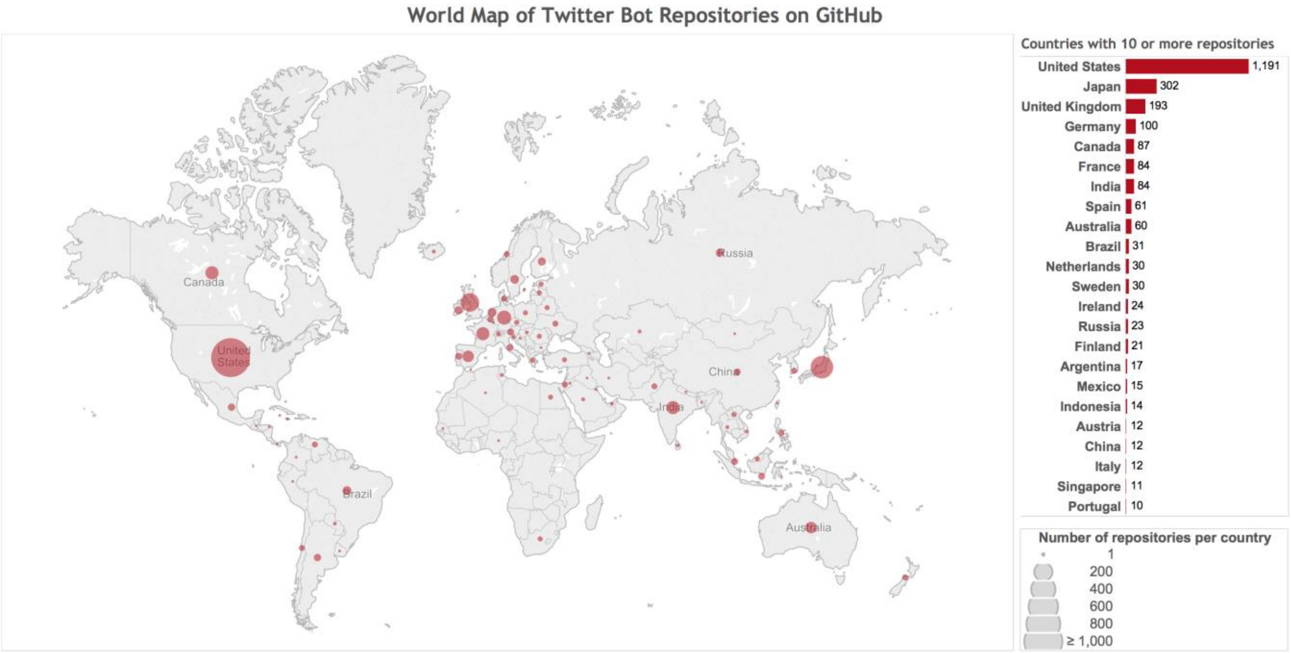

Figure 2. Global map of bot repositories (**N*** ***= 2,615).***

According to the location information in the data set, almost every second Twitter bot was produced by a developer from the United States. As shown in Figure 3, U.S. bots have dominated since 2008. In fact, in 2008 exactly 50% of the bots were from the United States, and the rest were from European and Chinese developers. Currently, Japan has the second largest number of bot repositories on GitHub. The first Japanese bot was published in 2009, and bot writers from the country have been remarkably active since then. In 2010, Japanese bot writers published more bots than developers from the United States.

Unfortunately, up-to-date information about the geographical distribution of all GitHub users is not available. For 2012–2013 data, see Lima et al. (2014) or Graham and De Sabbata (2014). When comparing the geographical distribution of Twitter bot codes to Twitter use worldwide, we need to work with similar limitations. Twitter does not publish official statistics about the number of users per country, but statistics from research projects are available about worldwide Twitter use (Lipman, 2014). According to Sysomos, about 62% of the users are from the United States (Sysomos, 2014), while the United Kingdom and Canada represent 8% and 6%, respectively. The world map of Twitter bot repositories (Figure 2) is largely in line with these statistics, and the dominance of the United States is also clear on my map. Twitter has recently published that the number of monthly active users in Japan has reached 35 million (Purnell, 2016). This number can be compared to 320 million active users reported globally in the fourth quarter of 2015 by Twitter. The relatively high number of bot repositories for Japan is also in line with the country’s importance in the Twitter universe.

The Code Behind the Twitter Bots

As Dubbin (2013) points out, a Twitter bot can be written in almost any modern programming language. This GitHub corpus supports his observation, as it shows the diversity of Twitter bots in terms of technical realization.

The first bot code still available on GitHub was written in Ruby, and the developer worked on the project over a two-month period. This bot used the instant messaging system Jabber, popular at the time, to control a Twitter account. Ruby, the language used for this first bot was, in fact, the most popular Twitter bot language in 2008 and 2009, with more than half of bot codes written in Ruby. Python became popular around 2012, and by mid-2013, the number of bots written in Python reached the number of bots written in Ruby. Today, Python is the most popular bot writing language by a large margin: 1,488 out of 4,101 available bot codes on GitHub use the language. For an overview of the programming language changes over time, see Figure 3.

Learning more about the code behind the Twitter bots is important, because the chosen language has consequences for the group of people who are able to understand the code, make changes in the code, and deploy it on Twitter.

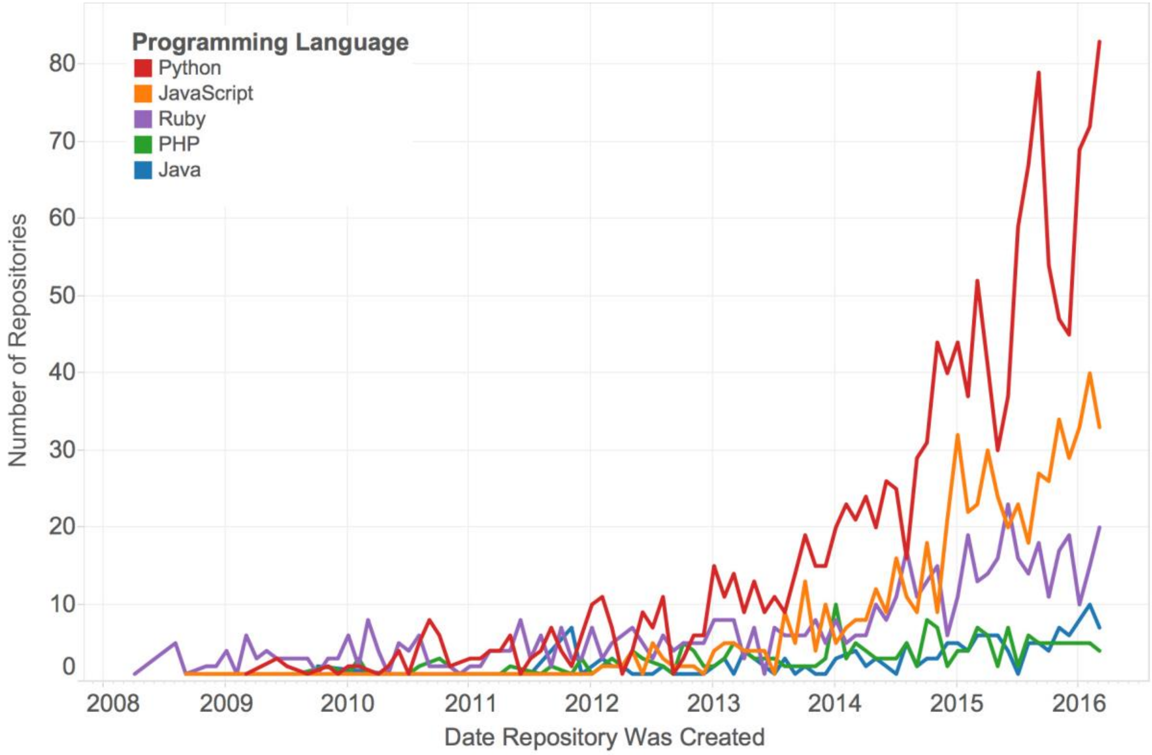

Figure 3. Bot programming languages used in GitHub, 2008–2016.

Python is known as a relatively easy language, and it is often the first programming language a person learns. The language is also known for being optimized for code readability, and developers try to write simple code which, makes it easier to understand someone else's program in Python (Hamilton, 2008; Peters, 2004). Hence, it also makes it easier to take a Python code from GitHub and apply it to a new problem. Finally, Python is particularly well suited for creating Twitter bots, because it has a range of packages that support several aspects of bot codes, from network access to text analysis and manipulation.

The size of the repositories is a good indicator of the complexity of the code. Interestingly, the median size of the repository is very similar for all major bot writing languages (between 125 KB and 150 KB). The only exception is Java, which tends to produce larger repositories (about double in size). Unfortunately, the size of the repository contains all the images and data files—not just the code—so this represents some limitations for analysis. Code length in lines of code would be a much better indicator of the complexity of the bots.

The Life Cycle of a Twitter Bot Repository

GitHub is designed to manage multiple versions of a program, and the platform is ideal for maintaining code when a user is constantly updating a repository. However, GitHub users often use the site for code dumping—storing and sharing programs that they are not planning to work on—rather than for actively working on a code and committing changes.

The lifetime of the repositories on GitHub can be measured as the difference between the date of creation and the last version of the repository (see Figure 4). The lifetime of a Twitter bot repository could include periods of active work on the code and longer gaps between maintenances and code updates. For most cases, however, a longer lifetime indicates that the project was actively used and supported by one or more developers for a longer time.

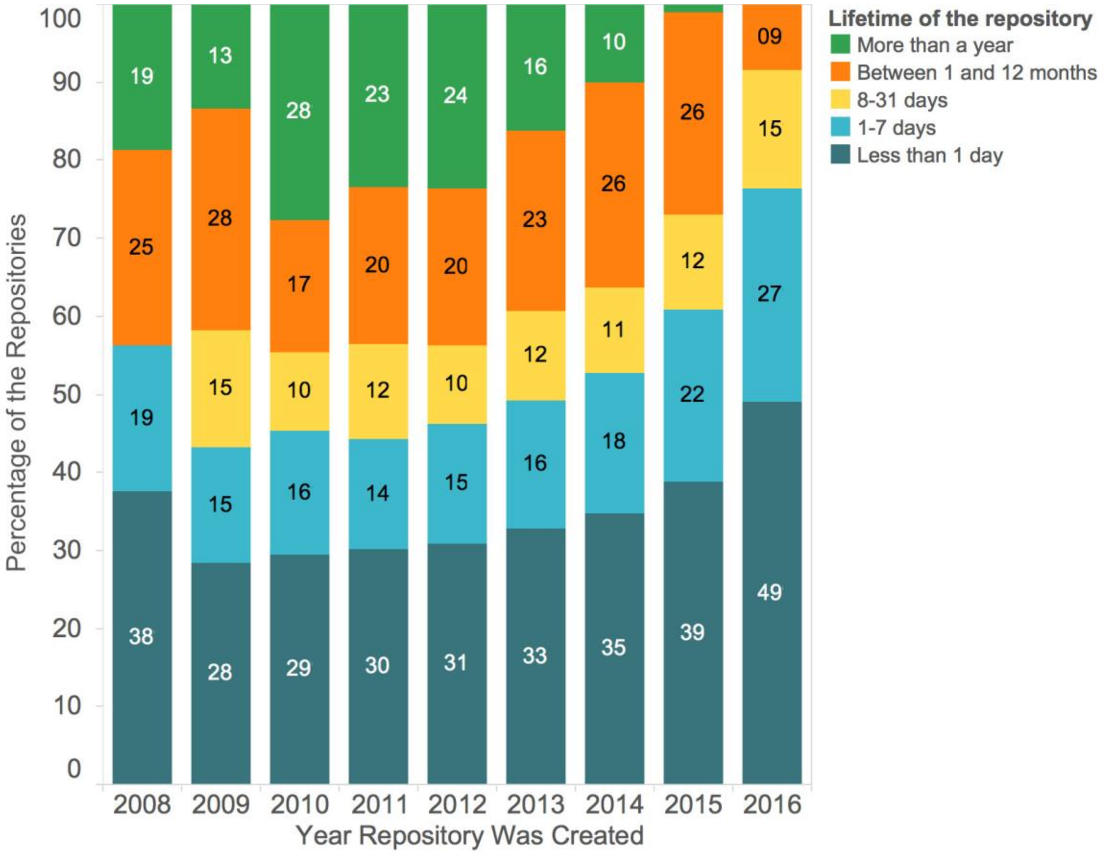

Figure 4. The life cycle of Twitter bot repositories on GitHub.

It is important to note that zero days does not mean that the user has only one version of the code online or that it was committed once. Doing some minor formatting after the first commit, uploading multiple files of the same repository in multiple commits, and changing the readme file of a repository are different from working on bugs or adding new features to a code weeks after its initial creation. Again, one could argue that developers who make changes soon after their initial upload only use GitHub for code dumping, and they do not work with the site for its social functions or for the version control features. Finally, it is important to note that many of the bot codes are so simple that a skilled developer can easily produce these codes in a single programming session.

Taking these considerations into account, it may be said that a large number of bot repositories on GitHub are either code dumps or one-time small projects, as these repositories are not developed or maintained for a long period of time. According to the data, 37% of the bot repositories were, for example, not updated after the first 24 hours. Another 20% of the bot repositories have a life cycle of no longer than a week.

The average difference between creating a repository and the latest commit is 92 days. Longer life cycles typically result from spurts of intense tinkering activity rather than a sustained interest for the entire time period. For example, one of the most popular Twitter bots has five commits from its very first day, but the next commit happened almost 10 months later. This commit was made in response to Twitter’s changing its process for handling API requests. Another popular repository had a period of 17 months, during which it was never updated. After editing some minor bugs, the owner of the repository left the code unchanged for a longer period, and she came back after 17 months to merge suggested changes from other users.

The Developer Behind the Bot Code

What can we learn about the developers who upload bot codes to GitHub? GitHub offers access to various pieces of information about the professional activity of the users both within the site and off the site. However, the information about off-site activities, such as a user’s workplace or a personal blog address, is rather limited and only collected in a free-text format. The data about the outside world are regulated only by the social norms and practices of the GitHub community. Thus, users’ professional activity outside of the GitHub universe is hard to quantify. In contrast, GitHub metrics are easy to read, because they are directly connected to the architecture of the website. Nevertheless, the on-site activity is important for understanding who are the developers behind the bots. In the following discussion, individual users rather than bot repositories are the unit of analysis, because some users may have several bot repositories listed.

GitHub differentiates between two types of accounts: regular users and organizations. An example for Twitter bots published by an organization is a school from San Francisco where the students published 37 bot codes to the account of the school, all written in Python. Browsing these repositories, it also became clear that the teacher looked at each bot and gave feedback in the form of comments directly inserted in the code.

The typical bot writer is an experienced programmer and a long-time GitHub user who has multiple repositories on GitHub. Users in this data set had 29 repositories on average at the time of the data collection, and on average they had registered two years before publishing their first bot code on GitHub.[3] Still, there are 155 bot repositories where the owner of the repository has no other repositories shared on GitHub. These data suggest that, for many developers, a Twitter bot is only a side project or a fun experiment. Less than 10% of Twitter bot writers have additional bot repositories.

Gist is a hosting service for code snippets with version control by GitHub. It is useful when a developer wants to share part of his or her code in a blog post or on a forum outside of GitHub. At the time of writing this article, there are more than 17.7 million gists on GitHub. The extensive use of gists is, however, not typical in the community of bot writers. The average bot writer has 10 public gists on GitHub, 1,739 bot writers have no public gists, and another 322 bot writers have only one public gist. In other words, bot code tends not to be publicized as much as other forms of code.

Most bot writers are well integrated into the GitHub community in terms of following other users or being followed by them. The average GitHub bot writer has 24 followers. Still, about every fifth GitHub bot writer—794 users out of the 3,591—had no followers at all at the time of the data collection. These codes, however, may still be found by users when they search for Twitter bots. The search interface of GitHub tends to give more weight to users who are socially active, so these repositories will be listed toward the end of the results page. Overall, it may be suggested that users with no followers have less influence on the community. Top developers represent the other end of the influence spectrum. There are eight users in the data set with more than 1,000 followers each. These popular users are called "programming rock stars" in the literature (Lima et al., 2014).

More About the Social Aspect of Developing Twitter Bots

The marketing of GitHub as well as the popular image of open-source projects suggest that shared bot codes attract large groups of developers who devote their free time to improving the code, and these programmers use GitHub as a social platform for both interacting with one another and working in a collaborative manner.

In contrast, the overwhelming majority of open-source Twitter bot projects on GitHub were developed by a single author. As Table 1 shows, almost 90% of Twitter bots shared on GitHub have been developed by only one person. These projects generate less social activity compared to Twitter bots developed by two or more developers. For example, the average number of forks, or copies of the original repository, for single-author projects is 0.3, while a project with multiple authors has on average more than three forks. Having forks is an indicator of both broader interest in a project and the possibility of receiving contributions from the wider open-source community around GitHub. About 87% of the single

[3] There were three users in the data set with more than 10,000 repositories. One of the accounts with 52,713 repositories hosts WordPress plug-ins. Users with an extremely large number of repositories were filtered out to obtain a more realistic picture of the typical bot writer.

authors have not generated this kind of interest, because these projects have zero forks. In contrast, about 50% of the bot projects with two or more authors have at least one fork.

Table 1. The Social Aspects of Bot Repositories on GitHub.

	N	Percentage	Average number of stars	Average number of forks	Average lifetime (days)
Single author	3,567	89.6	1.1	0.3	80
Two or more authors	415	10.4	9.8	3.2	201
Total	3,982	100	2.0	0.6	93

The average number of stars, an indicator of a repository's popularity, is also more modest for single-author projects. A bot project with two or more developers has on average 10 stars, whereas the average single-author project has earned only one star. This does not mean that single-author projects cannot be popular or gain support from the open-source community; the most acknowledged bot code developed by a single author has gained an impressive 162 stars, while the most widely forked single-author project had 52 forks.

Finally, projects with two or more developers are maintained or updated for a longer time on average. While an average single-author project is "closed down" in less than three months, projects with multiple authors are maintained for more than 200 days on average.

In summary, only 1 out of 10 Twitter bot repositories shared on GitHub has utilized the collaborative or social aspects of the platform. About 400 projects were co-developed by two or more programmers; these projects have received more acknowledgment from other GitHub users and generated more forked repositories. Co-development and forking could result in better-quality codes by having contributions from more than one programmer.

Political Bots and General-Purpose Bots

The great majority of Twitter bots codes shared on GitHub do not serve any direct political purpose. Here, the term *political bots* is used in a broad sense to include (a) bots designed to support democratic processes, such as spreading information about elections or referendums; (b) bots that use data from public institutions, such as governmental agencies, municipalities, and police departments; and (c) bots that simply focus on issue politics, such as climate change or gun control.

On GitHub, most repositories have a short description of the code. These descriptions are usually five- to 10-word summaries of the project. Most of them are too general to provide a full picture of the bot's goal and functions. Authors may, for example, specify that the code is "a tool kit for the creation of Twitter bots" or the specific repository is "my first Twitter bot," which does not reveal much about the function of the bot. Other bot descriptions describe the function of the bots in more detail, but they do not disclose any details about the possible fields of application for the bot or additional information about the way it was deployed. Similarly, the description rarely provides clues about the intention behind writing the code. By manually going through the description of the more than 4,000 repositories, 41 projects with direct references to politics were identified. This is about 1% of all bot repositories on GitHub.

A readme file, a special information file added to the repository, offers more background about bots. These files usually contain a quick introduction to the project that often has more information than the description. These introductions also often include practical guidelines for making sense of the code and instructions for installing, fine-tuning, and deploying the bot on Twitter. These readme files are also accessible through the API, and it is easy to analyze the text of the readme files alongside the description. However, thousands of readme files are too long to read. So a keyword-based search looking for political terms in the readme files was conducted. After this search, every result was double-checked to remove irrelevant bots. Based on this search, 50 projects developed with explicit political purposes. There was a significant overlap with the previous approach. The second strategy did not result in significantly more than the 41 bots identified by reading the descriptions, so most of the political bots on GitHub have enough information about their purpose in their short descriptions. An important caveat is that the searches were done using English terms in the data set. However, the corpus contained very few projects with a non-English description. These results indicate that most of the Twitter bot codes shared on GitHub were not developed with political purposes in mind.

The collection of political bot repositories on GitHub is neither representative nor complete enough to serve as the foundation for a typology of Twitter bots. This analysis can best be pursued on Twitter itself. Nevertheless, the political bot codes on GitHub showcase a wide range of functionalities and distinct purposes, from providing useful information about upcoming elections, through access to large but often hard to search public databases, to satirical bots making fun of political issues.

There are multiple examples of bots that draw on public data and transform this information to a regularly updated stream of tweets. One of the political bots on GitHub will tweet the waiting time at the San Ysidro-Tijuana border between the United States and Mexico based on data scraped from the website of the California Institute for Telecommunications and Information Technology at the University of California. Other examples include bots searching in public databases maintained by the Environmental Agency in the United Kingdom and the U.S. Geological Survey and creating tweets about environmental issues.

A larger group of bots have been coded to provide information about elections. In 2014, a user called Schwad wrote a bot that scraped live election results and posted regular updates on Twitter during the Montana State House races. Another election-themed bot was created to use the Election API provided by Google to answer direct messages about the location of polling stations. During the ongoing U.S. presidential election, the code was published for a pair of bots that tweet information respectively about Republican and Democratic campaign contributions based on a government-maintained database.

Another bot type can be labeled as awareness-raising bots: These automated accounts focus the attention of the Twitter community to certain public issues, such as police violence, racism, and feminism.

A simple but smart Twitter bot uses an existing public database of police-related civilian deaths and shows the location where the death took place by posting an image from Google Street View. Another bot, deployed on Twitter as @YourRepsOnGuns, simply retweets any member of Congress who tweets about firearms and related words.

Bots can also act as watchdogs and document certain online activities. GitHub has the code for Wikiedit bots that tweet warnings if someone edits a page on Wikipedia from an IP address linked to the government. Another version of this bot tweets every time a student from an American university edits a wiki page about another, competing university.

Finally, many satirical bot codes are published on GitHub. Donald Trump, for instance, has attracted multiple parody bots. One of these bot codes is called "Donaldo Trumpilini" and mixes the text of tweets from the realDonaldTrump Twitter account with quotes from the autobiography of Benito Mussolini, the Italian fascist leader. Another project called "wikitrump" picks a random term from Wikipedia and uses derogatory words to describe it, parodying the style of Donald Trump.

Regulation and Norms on the Platform

Whereas Twitter has a policy against abusive bots and developed sophisticated algorithms to detect these bots on its platform, GitHub does not screen the content of its repositories and does not remove spam bots. GitHub's terms of service limit the uploading of materials infringing copyrights and using the service for any illegal activity or for activities that violate any laws in a jurisdiction. In most countries, however, Twitter bots are not regulated.

GitHub does not censor the codes shared on the website, but GitHub has been censored many times based on the content shared on the site. Site-wide blocking of GitHub has occurred in the past due to the posting of partly ironic suicide guides in Russian (Lunden, 2014), anti-India propaganda from ISIS in a repository (Russell, 2014), and some controversial and unconfirmed reasons in the case of China (Kan, 2013). Although there are many known political bot cases reported by the media, there is no information about any action against GitHub or request to developers using the site to remove bot codes.

At the time of data collection, GitHub has hosted several self-disclosed spam bot repositories on its site. GitHub offers some level of anonymity to its users, as the site does not enforce a "real name" policy, but it is reasonable to suggest that the majority of developers do not want to be associated with malicious bot codes. Hence, the number of open-source spam bot codes on GitHub is actually rather small.

Self-regulation through shared values and norms could also play an important role in limiting the spread of malicious Twitter bot codes on GitHub. Many developers, for example, request that other users only use the bot for good instead of evil. Some project descriptions include warnings related to the bot. Many authors ask that other users not use the script for spamming or harassment.

Discussion and Future Research

There are more than 4,000 Twitter bot code repositories shared on GitHub. Based on the number of available repositories, the diverse functions these bots can be programmed to perform, and the relative ease with which these bot codes may be repurposed or adapted to new tasks, we may see a rise of Twitter bot usage in political contexts. Today, individuals and groups without high-level programming skills or resources to pay for expensive IT services could easily deploy bots on Twitter, which suggests a democratization of these powerful social media tools in the realm of politics.

The findings presented here suggest that the overwhelming majority of bot codes on GitHub were developed by a single author in a relatively short time. Almost 40% of the bot repositories were single-day projects, or in some cases these bots were developed outside of GitHub and the author only uploaded the final code to the platform. Also, only 1 in 10 bot repositories were developed by multiple authors using the site's collaborative and version tracking functions. Nevertheless, GitHub users may learn from one another by looking at how a programming challenge, such as producing a Twitter bot, has been solved by others.

By skimming through the descriptions of bot repositories and searching for various keywords in their readme files, a small number of Twitter political bot codes were identified. Most of these political bots can be labeled as "good bots," because they have been designed with the intention of providing useful information to the Twitter community, providing information about polling locations during election times, tweeting about the results of the election, or translating complicated public databases to the format of easily digestible 140-character tweets.

Studying nonpolitical bots, including the methods of development and some information about the authors behind the bots, can still be relevant for understanding the creation of political bots. Bot writers gain significant experience in working with the Twitter API and writing bots, so they can later decide to write a political bot or can be hired to do so. Second, software development often relies and builds on existing code. Hence, the nonpolitical or general-purpose Twitter bot codes available on GitHub today can serve as the foundation for future political bots. Functions developed for nonpolitical bots can be easily repurposed to serve political purposes.

The potential use of these bot codes for influencing social media discussions for political purposes, on the other hand, raises serious ethical questions. Transparency of the bot activity may be central to incorporating bots in human conversations in the online public sphere. Currently, Twitter does not offer a standard way to differentiate between human and bot activity. Making the source code of Twitter bots public on a code-sharing platform, such as GitHub, and providing a link to the relevant repository may be a great way to increase transparency, because it allows us to peek into the process of automating social media–based communication.

References

Chu, Z., Gianvecchio, S., Wang, H., & Jajodia, S. (2010, December). Who is tweeting on Twitter: Human, bot, or cyborg? In *Proceedings of the 26th Annual Computer Security Applications Conference* (pp. 21–30). Austin, TX: ACM.

Dabbish, L., Stuart, C., Tsay, J., & Herbsleb, J. (2012, February). Social coding in GitHub: Transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work* (pp. 1277–1286). Seattle, WA: ACM.

Dubbin, R. (2013, November). The rise of Twitter bots. *The New Yorker*. Retrieved from http://www.newyorker.com/tech/elements/the-rise-of-twitter-bots

Edwards, C., Edwards, A., Spence, P. R., & Shelton, A. K. (2014). Is that a bot running the social media feed? Testing the differences in perceptions of communication quality for a human agent and a bot agent on Twitter. *Computers in Human Behavior, 33,* 372–376. Retrieved from http://doi.org/10.1016/j.chb.2013.08.013

Ferrara, E., Varol, O., Davis, C., Menczer, F., & Flammini, A. (2014). The rise of social bots. arXiv:1407.5225. Retrieved from http://arxiv.org/abs/1407.5225

Gousios, G. (2013, May). The GHTorent dataset and tool suite. In *Proceedings of the 10th Working Conference on Mining Software Repositories* (pp. 233–236). San Francisco, CA: IEEE. Retrieved from http://dl.acm.org/citation.cfm?id=2487132

Gousios, G., Vasilescu, B., Serebrenik, A., & Zaidman, A. (2014, May). Lean GHTorrent: GitHub data on demand. In *Proceedings of the 11th Working Conference on Mining Software Repositories* (pp. 384–387). Hyderabad, India: ACM. Retrieved from http://dl.acm.org/citation.cfm?id=2597126

Graham, M., & De Sabbata, S. (2014). *GitHub: Mapping collaborative software.* Retrieved from http://geography.oii.ox.ac.uk/?page=github

Grigorik, I. (2012). *The GitHub archive*. Retrieved from http://www.githubarchive.org/

Hamilton, N. (2008, August). The A–Z of programming languages: Python. *Techworld*. Retrieved from http://www.techworld.com.au/article/255835/a-z_programming_languages_python/?pp=4

Kalliamvakou, E., Gousios, G., Blincoe, K., Singer, L., German, D. M., & Damian, D. (2014, May). The promises and perils of mining GitHub. In *Proceedings of the 11th Working Conference on Mining Software Repositories* (pp. 92–101). Hyderabad, India: ACM.

Kan, M. (2013, January 23). GitHub unblocked in China after former Google head slams its censorship. *Computerworld.* Retrieved from http://www.computerworld.com/article/2493478/internet/github-unblocked-in-china-after-former-google-head-slams-its-censorship.html

Lee, K., Caverlee, J., & Webb, S. (2010, April). The social honeypot project: Protecting online communities from spammers. In *Proceedings of the 19th International Conference on World Wide Web* (pp. 1139–1140). Raleigh, NC: ACM.

Lee, K., Eoff, B. D., & Caverlee, J. (2011, July). Seven months with the devils: A long-term study of content polluters on Twitter. In *Proceedings of the International Conference on Web and Social Media* (pp. 185–192). Barcelona, Spain: ACM.

Lima, A., Rossi, L., & Musolesi, M. (2014). Coding together at scale: GitHub as a collaborative social network. arXiv:1407.2535. Retrieved from http://arxiv.org/abs/1407.2535

Lin, P. C., & Huang, P. M. (2013, January). A study of effective features for detecting long-surviving Twitter spam accounts. In *Proceedings of the 15th International Conference on Advanced Communication Technology* (pp. 841–846). Pyeonchang, Korea: IEEE.

Lipman, V. (2014, May). Top Twitter trends: What countries are most active? Who's most popular? *Forbes.* Retrieved from http://www.forbes.com/sites/victorlipman/2014/05/24/top-twitter-trends-what-countries-are-most-active-whos-most-popular/

Lokot, T., & Diakopoulos, N. (2016). News bots: Automating news and information dissemination on Twitter. *Digital Journalism, 4*(6), 682–699. Retrieved from http://doi.org/10.1080/21670811.2015.1081822

Lumezanu, C., Feamster, N., & Klein, H. (2012). # bias: Measuring the tweeting behavior of propagandists. In *Proceedings of the Sixth International AAAI Conference on Weblogs and Social Media* (pp. 210–217). Dublin, Ireland: AAAI.

Lunden, I. (2014, December). To get off Russia's blacklist, GitHub has blocked access to pages that highlight suicide. *TechCrunch.* Retrieved from http://social.techcrunch.com/2014/12/05/to-get-off-russias-blacklist-github-has-blocked-access-to-pages-that-highlight-suicide/

Makice, K. (2009). *Twitter API: Up and running*. Beijing, China: O'Reilly.

Mustafaraj, E., & Metaxas, P. T. (2010). *From obscurity to prominence in minutes: Political speech and real-time search*. Retrieved from http://repository.wellesley.edu/computersciencefaculty/9/

Peters, T. (2004). *The zen of Python*. Retrieved from https://www.python.org/dev/peps/pep-0020/

Purnell, N. (2016, Feb). *Twitter has more users than Facebook—in Japan.* Retrieved from http://blogs.wsj.com/digits/2016/02/18/twitter-has-more-users-than-facebook-in-japan/

Ratkiewicz, J., Conover, M., Meiss, M., Gonçalves, B., Patil, S., Flammini, A., & Menczer, F. (2011, March). Truthy: Mapping the spread of astroturf in microblog streams. In *Proceedings of the 20th International Conference Companion on World Wide Web* (pp. 249–252). Hyderabad, India: ACM.

Russell, J. (2014, December). India's government asks ISPs to block GitHub, Vimeo and 30 other websites. *TechCrunch.* Retrieved from http://social.techcrunch.com/2014/12/31/indian-government-censorsht/

Sanatinia, A., & Noubir, G. (2016). On GitHub's programming languages. arXiv:1603.00431. Retrieved from https://arxiv.org/pdf/1603.00431.pdf

Song, J., Lee, S., & Kim, J. (2011, September). Spam filtering in Twitter using sender−receiver relationship. In *Proceedings of the 14th International Symposium on Recent Advances in Intrusion Detection* (pp. 301–317). Menlo Park, CA: Springer.

Steiner, T. (2014). Telling breaking news stories from Wikipedia with social multimedia: A case study of the 2014 Winter Olympics. arXiv:1403.4289. Retrieved from https://arxiv.org/abs/1403.4289

Storey, M. A., Singer, L., Cleary, B., Figueira Filho, F., & Zagalsky, A. (2014, May). The (r)evolution of social media in software engineering. In *Proceedings of the Conference on Future of Software Engineering* (pp. 100–116). Hyderabad, IN: ACM.

Sysomos. (2014, May). *Inside Twitter: An in-depth look inside the Twitter world.* Retrieved from http://sysomos.com/sites/default/files/Inside-Twitter-BySysomos.pdf

Takhteyev, Y., & Hilts, A. (2010). *Investigating the geography of open source software through GitHub* (Working paper). Retrieved from http://takhteyev.org/papers/Takhteyev-Hilts-2010.pdf

Wang, A. H. (2010, July). Don't follow me: Spam detection in Twitter. In *Proceedings of the 2010 International Conference on Security and Cryptography (SECRYPT)* (pp. 1–10). Athens, Greece: IEEE.

Woolley, S. C. (2016). Automating power: Social bot interference in global politics. *First Monday, 21*(4).

Zagalsky, A., Feliciano, J., Storey, M. A., Zhao, Y., & Wang, W. (2015, February). The emergence of GitHub as a collaborative platform for education. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work and Social Computing* (pp. 1906–1917). Vancouver, Canada*:* ACM. Retrieved from http://doi.org/10.1145/2675133.2675284