Corvinus University of Budapest

Doctoral School of Sociology

Thesis

The automatization of social media communication

Exploring the development of bot codes on GitHub and the use of open-source bots on Twitter

Bence Kollányi

Tutor: Bence Ságvári, PhD

[Blank page]

STATEMENT ON THE ORIGINALITY OF THE THESIS WORK

Title of the thesis: The automatization of social media communication - Exploring the development of bot codes on GitHub and the use of open-source bots on Twitter
Length: 143 pages

This is to certify that the submitted doctoral thesis is solely my own work. This thesis has not been submitted for any degree or other purposes.

I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

Date: December 20, 2021

Bence Kollányi
signature

Table of Contents

List of figures and tables

Acronyms and abbreviations

AI - Artificial intelligence
API - Application Programming Interface
IFTTT - If This Than That
IT – Information technology
KMO - Kaiser-Meyer-Olkin
PCA - Principal component analysis

Acknowledgments

After completing the first part of my doctoral program, I had the opportunity to work and study in the United States of America for five years. These years were a formative time in my research career and helped me to find my true research interest. I first worked together with Prof. Michael C. Best from the Georgia Institute of Technology. His commitment to the use of ICT for development and his global network of non-governmental organizations and political activists allowed me to participate in a research project where his research team developed a tool that used social media, including Twitter and Facebook, to monitor and respond to outbreaks of violence during and after elections in various African countries. Later, also at Georgia Tech, I completed a Master of Science program in Public Policy. During my graduate studies, I worked as a research assistant under the supervision of Prof. Hans Klein. I was involved in a research project that looked at online misinformation and involved computer scientists from across the U.S. developing state-of-the-art bot detection systems. I learned a lot from Dr. Best about leading an international team and developing reliable systems that can be used in high-risk situations. Prof. Klein showed me how to work in an interdisciplinary research team and how to find a common language for social scientists and engineers. During my studies in the US, I learned how to program in R. After returning to Europe, I spent a year in Denmark where I learned how to code in Python from fellow students at the IT University of Denmark. Later, based on these new skills and my previous research on social media and misinformation, I found a job in a research project at the Oxford Internet Institute in the Computational Propaganda research team where I was mainly responsible for data collection from various social media platforms and data analysis. I am very grateful for the time I spent with a team of young and talented researchers led by Prof. Philip N. Howard. According to his vision of real-time social science, researchers should respond to social problems as they arise and communicate their findings to a wider audience instead of just publishing academic papers. Apart from finding

a warm and supportive research community, I also learned a lot professionally, such as working with the media, publishing my work in peer-reviewed journals and at conferences, and I was able to work in a truly international research group again. After finding my way back to Hungary, I redefined my research questions and conducted my PhD research with the continuous support and guidance of Dr. Bence Ságvári, my tutor. He was the right supervisor, because he knew how the Hungarian academic world worked, and he shared my passion for doing computation social science. He was always there for me during the time I was writing my dissertation, motivating me with his fresh research ideas and constructive criticism every time we discussed my project. I am also very grateful to my first and second PhD committees – Dr. Tamás Bartus who, in addition to his administrative role as the formal head of the committee, provided important assistance throughout the whole dissertation writing process, Dr. Tamás Bokor who helped me at an early phase of my research by asking the right questions, Dr. Péter Csigó who witnessed the dissertation taking shape from the first draft to the latest version, and who always supported me and gave me mindful comments, and to Dr. Zoltán Kmetty who carefully read my proposal and helped me immensely in completing the dissertation with his detailed feedback. Finally, on a personal note, I am also grateful to my supportive partner who always believed in me, even when completing my PhD research seemed to have no final date. As someone who has a great talent for visual work, she even helped me make some of my diagrams in the thesis look pretty.

1. Introduction

Social media bots are small programs that run on internet-connected servers and automatically post messages, follow or unfollow other users, set up polls, post pictures, send direct messages to specific users, and so on[1]. The typical bot acts based on predefined rules and uses an application programming interface (API)[2] to communicate with the servers of a given social media service. Social media bots often automate human actions: They do things on a social media platform that could have been done by a human user. However, a bot does these tasks on a larger scale, in less time and often without human intervention for longer periods of time.

As social media platforms become more open to automation, allowing businesses and government institutions to use bots to mediate online transactions and services, bots are becoming an increasingly important component in the power relations of the online communication sphere. The production and deployment of social media bots signal the emergence of new political economies that redistribute agency around new technological actors. This has implications for marketing, political action and even private lives. Yet, we have very little systematic knowledge about how bots are produced and the role of sharing code online and using a collaborative platform.

1.1. Purpose of the thesis

The goal of my PhD research project is to understand the practice of writing and deploying bots on Twitter. To understand current practices of bot development and use, I propose to bring together (1) an examination of bot codes available on GitHub, the largest online code repository, and (2) data on how human users communicate with the automatized social media accounts on Twitter, a platform where these bots are deployed. Thus, my research method combines data about automated accounts deployed on a social media platform with the source code behind those same bots – this provides a unique lens on bots and provides important insights about how they work.

Combining API-based, data driven research with a classical social science approach helps to understand current practices of writing for and deploying bots on online social

[1] Part of the introduction and the literature review is based on an earlier single author publication of mine, but the text was greatly extended and updated. (See. Kollanyi, 2016).
[2] An application programming interface (APA) is a "way for two computer applications to talk to each other over a network" (Jacobson, Brail & Woods, 2012). In practice, an API allows accessing data and services within the organization, by third-party developers or by partners of the company.

media platforms. As part of my PhD research, I examined bot repositories on GitHub and conducted a survey with the same bot developers to gain further insights into the nitty-gritty of writing bot code, including the most important challenges during the development phase and major barriers to deploying and operating bots on Twitter over an extensive period of time. The survey results also shed light on the motivations behind creating automated social media accounts. I also studied how programmers deploy their bots on Twitter and how other, mostly human, users react to the bot activity. The final part of my thesis contributes to a typology of open-source social media bots by systematically examining bot codes shared on GitHub and the activity of some of these bots on Twitter in tandem.

1.2. Research questions

While most of my research questions about the Twitter bots themselves can be answered by working with data and metadata obtained through the GitHub APIs and the various Twitter APIs, research questions about the motivations for developing a bot or questions about the bot developers themselves can only be partially answered by relying on these sources. To answer questions about this latter group of questions, developers have to be contacted directly, for example, through a survey or some type of interview.

The following section of this chapter provides a list of the five research questions that I intend to answer as part of the research project for my dissertation. These research questions are centered around two main themes. The first three research questions address the following more general questions: who are the developers behind the open-source Twitter bots and how do they develop these bots. The second set of questions focuses on the Twitter bots themselves and attempts to answer questions about how they work and how they produce tweet after tweet.

1.2.1. Research questions on bot developers

These questions are aimed at the bot developers both professional developers and non-professionals who are able to write code.

RQ 1.1 What are the practices for code developing and sharing code for open-source Twitter bots? What are the most import reasons behind using GitHub as a tool[3] for

[3] Some developers only host their code on GitHub and do not use the platform for aiding the code writing process by keeping track of changes or asking for contributions from other open-source developers. This

developing and sharing code?

This question focuses on how developers use GitHub, such as how often they update their code, how much information they include for other developers, whether they receive support from other developers, etc. To answer these questions, GitHub provides access to almost all metadata about its repositories. However, for the question about the reasons for using GitHub and the motivations for developing a Twitter bot, I relied on survey data.

RQ 1.2 How do developers acquire the skills needed to develop a Twitter bot? To what extent do these programming skills determine and facilitate the creation of Twitter bots?

I start from the assumption (based on my preliminary findings) that developing bots requires programming skills, and that bot codes are developed by a large and diverse group of developers, including programmers with computer science backgrounds, social scientists, journalists and artists. At the same time, the development of bots, like open-source software in general, is increasingly decentralized, with actors relying on reusable code that can be adapted to more specific needs, and sharing knowledge accordingly. This is the scene I would like to explore with my next research questions.

There are a handful of different sources of information available about creating Twitter bots, from blog posts to university courses to a look at the available bot codes on GitHub. What are the most important sources of information for bot development, and what does a bot developer do when an unexpected problem arises? To answer this research question, I rely mainly on survey data.

RQ 1.3 Is there a community of bot developers on GitHub? Or alternatively, is the code for various bots developed by lone developers?

Although GitHub provides a platform for collaboration on projects involving multiple developers, my previous research suggests that nearly 90 percent of the bot code available on GitHub was developed or at least published by only one developer.

The repositories on GitHub do not only keep track all the changes made by developers who have access to the code but also record who contributed to the project by either writing

practice is often called as code dumping. My study both includes developers who practice code dumping and developers who actively use the a wide range of functions of the platform.

code or simply reporting issues.

All of this data is publicly available for the repositories I examined and provides important insights into how widespread bot development collaboration is and whether there is an active community of developers on GitHub focused on bots.

1.2.2. Research questions on Twitter bots

RQ 2.1 How do open-source bots generate, process and publish content on Twitter?

What can we learn about Twitter bots by examining the source codes on GitHub and the activity of deployed bots on Twitter in tandem? By combining these two data sources, GitHub and Twitter, in a novel way, I can describe how open-source Twitter bots work (e.g., I can rely on the Twitter bios and account descriptions and the code published on GitHub). All of the above data can be collected computationally through the various APIs of GitHub and Twitter.

Do Twitter bots generate their own content? If so, how do these accounts generate their content, what are the main sources of information used? Some Twitter bots do disclose information about the sources they rely on, either in the bot's bio on Twitter or in the tweets themselves. A good example for the latter is a link included in the tweet. However, the exact way these sources are accessed, processed, and how the bot generates content is often difficult or impossible to understand without looking into the code running behind the bot.

To answer this research question, I am also trying to quantify how much of the traffic generated by open-source bots on Twitter is original content, and how much of the content is simply retweets or quoting other Twitter accounts.

RQ 2.2 What is the life cycle of an open-source bot, and how much traffic is generated by a bot on Twitter during that time? What are the challenges of running a Twitter bot for an extended period of time, and why do bots get banned or become inactive on Twitter?

The life cycle of a Twitter bot can be defined in several ways. We can look at the time between the first and last tweet generated by the account on Twitter or calculate the time spent developing and occasionally updating the bot's source code on GitHub. In most cases, these time periods overlap, but bot accounts are often suspended and sometime even redeployed with a different user handle by the developer. If the bot has a longer lifespan, code developers may need to address issues such as changes in how the Twitter APIs work

or how the bot can access its sources (outside of Twitter). I am also trying to understand why some bots not working anymore. To figure out the main challenges, I rely both on studying dysfunctional or inactive bots on Twitter and on asking the developers themselves in the survey.

1.3. Contribution to the field

It is relatively cheap to create or buy fake social media accounts and use them as bot accounts on Twitter. On the other hand, it is much harder to build a real social network and reach a large number of human users with a message, although previous research projects and media reports suggest that bots can succeed in building their own network based on simple algorithms. Furthermore, having followers does not necessarily mean that a bot was able to create a real audience. Real users (humans) in the bot's network will read a message posted by the automated account and spread it further. Through my research I hope to contribute to the understanding of social media automation in the broader context of audience engagement and to measure the impact of Twitter bots.

My research project will also help to better understand and conceptualize social media automation by creating a typology of Twitter bots. To create this typology, I will also going to investigate the technical background of bots and bot development. While creating the typology, I will explore the basic domains and functions of bots and look at the external tools or services that can be used for content creation or automation.

In academic journals and in the media, bots and bot networks are often equated with spam bots or large networks of fake user accounts created for the purpose of creating a false image of a grassroots movement (astroturfing). However, many bots perform important functions within the online social media ecosystem. For example, some bots gather information for journalists, facilitate often difficult access to open government data, or serve artistic purposes. In the literature review, I cited a few authors who have already written about these benign actors. I will also contribute to this "good bot" or "useful bot" literature.

Methodology, I propose a novel way to study Twitter bots by examining both the open-source bot codes on GitHub and the bots deployed on Twitter. Traditionally, research projects have examined bots as black boxes and attempted to "reverse engineer" the algorithms behind automated accounts. Instead, I will track bots to their source code on GitHub and connect the algorithm (code) running behind the bot to data about bot activity

on a social media platform. By connecting these two data sources, I will gain access to the inner “mechanics” of certain bots on Twitter, and I can test my findings about how algorithms built into the bots work on a large “real world” dataset.

Another important contribution of my research project is to examine the political economy of Twitter automation in terms of political communication. I plan to better understand the fundamentals of bot production in an open-source style development workflow. It is natural to hypothesize that bots are embedded in an open-source context of software code production and sharing. The open economy of bot codes is distinct from other areas of information manipulation and management in the political arena, such as state level filtering or online political marketing communications, which are supported by pre-built tools and IT services offered by companies. To put it more simply, bot writing could have a much lower barrier to entry in terms of resources. Meanwhile, it is also mediated by technological expertise. My study will explore these questions to examine the extent to which a restructuring of the political media sphere can be understood as a bottom-up phenomenon.

1.4. Overview of the dissertation

This work consists of four main chapters. The first and second chapters provide an introduction to the thesis by positioning the research within the relevant scholarly discourse and listing the central research questions of the thesis. The next chapter provides an overview of the methodology and research strategy I used to answer these research questions. The third chapter presents the findings in four long subchapters. The thesis ends with a discussion chapter that summarizes and reflects on the major findings of my PhD research; It also includes a systematic overview of the potential limitations and outlines interesting future research directions.

In the first chapter, after a brief introduction, I describe the novelty of my research strategy and my main contributions to the field. I then outline my research questions on bot developers and Twitter bots themselves.

The literature review in the second chapter provides an overview of the relevant literature on Twitter bots, including bot detection, as well as open-source software development using GitHub. This broad range of literature includes papers in computer security to journalism and political science. After a brief introduction to Twitter bots, I provide an overview of the major efforts to detection bots, discuss the literature on the

political use of Twitter bots, and present the existing typologies of Twitter bots. Because I am particularly interested in how developers gain the skills to develop bots, the GitHub section of the literature review focuses primarily on collaboration and knowledge sharing.

The methodology section presents the various computational methods I used to access and analyze the data. I have structured this section to provide an overview of each data source. The methodology section also provides an overview of the limitations of my PhD research project. Lastly, it includes a description of the programming work I did for the thesis.

The results chapter follows a similar structure, it has four subchapters that follow the path of bots from GitHub to Twitter and report the most important findings about bot developers and their attitudes towards GitHub usage, Twitter bot development and politics. The final section of the findings chapter provides a typology of the open-source Twitter bots based on a combination of the data I collected.

The thesis ends with a discussion chapter and a list of references. The appendices contain the materials that are important but not necessary for understanding the methodology of data collection or analysis.

1.5. Notes on computational social science

The empirical research in my doctoral project relies on both computational social science methods and traditional survey methods. These different methods are interwoven, as explained below, and can inform or support each other. For example, collecting and analyzing shared bot codes from an online code repository can allow me to ask more informed questions from bot developers. On the other hand, asking specific questions to bot developers or looking into bot codes can also help me "ask" the right questions from the dataset and contextualize my findings.

My use of social media data draws on the work of the Digital Methods movement and, in particular, its efforts to lay the conceptual groundwork for a digital methodology. As the media landscape has changed and the importance of online public discussion has increased, social scientists now have access to a new source of data to work with. In addition to familiarity with traditional quantitative methods and an understanding of the concept of sampling, representativeness, etc., this type of data often requires computer skills both to access and analyze the data.

Social scientists have long added computer-based tools to their research toolkit, using

software packages to analyze quantitative and qualitative data. However, the development and proliferation of social media platforms such as Facebook and Twitter, have given rise to a "new apparatus for researching social life" (Marres, 2012). This change did not happen overnight, and as Lazer et al. (2009) noted in their Science Magazine discussion of the emerging field of computational social science, the social sciences have been slow to respoond to the new situation. According to Lazer et al., other fields of science, particularly the hard sciences such as biology and physics, had already undergone fundamental changes by the late 2000s as they began to explore the possibilities of digital research methods, including big data and analysis. In the same paper, the authors also point out that in many cases, the new social data is being analyzed by computer scientists, not social scientists, and often in a corporate environments that serves the business interests of large IT companies like Google and Yahoo.

In the meantime, it is important for social scientists to understand that working with digital data requires a new approach and the development of new research methods. The first step is to understand the growing availability of data that are "born digital" as opposed to data that has been "digitized" (Rogers, 2014), and to recognize that this allows for a shift in our empirical use of data. Specifically, Rogers argues that reconceptualizing data as born digital paves the way for new methods. As online devices capture user input and actions, e.g. search and purchase history, location, etc., the "digital born" data can be repurposed for research (Rogers, 2013). Rogers also argues that the distinction between virtual and real has diminished, and that social scientists should use digital data to study "cultural change and societal conditions."

In addition, researchers need to "follow the medium" by "learning from and reapplying how digital objects (such as hyperlinks) are treated by devices" and using digital methods as opposed to digitized methods (Rogers, 2013). The inner workings of digital technologies have implications for the methodology social scientists can use to study the new digital platforms. For example, to understand the logic of working with Twitter data, I draw on both the documentation that Twitter developed for third party developers to show analysts and researchers how to interact with the platform through its APIs and on the extensive literature on conducting research based on Twitter data (see for example, Bruns & Burgess, 2015, Bruns & Stieglitz, 2013; Gaffney & Puschmann, 2013, 2014).

Digital platforms have their own internal politics. Besides understanding the nitty-gritty of API-based data collection and the best approaches to working with a particular platform, researchers need to think about platform politics and look critically at their new

data source. To take just one example: Social media companies have significant influence over what researchers can learn about platform users which can be achieved either through technological choices or through legal and policy decisions. Twitter, for example, can define and regulate how third party developers, analysts, data traders, or researchers can access data through its API by changing the architecture of the API, modifying its terms of services (ToS), and adding new documents to regulate how to use and interact with the platform (Puschmann & Burgess, 2013). These platform restrictions can severely limit who can access data that appears to be open and available to the public. Decisions about who can access data and interact with the platform through its APIs almost always depend on the commercial interests of the company behind the platform (Borra & Rieder, 2014). Borra and Rieder, who have conducted g social research with Twitter, caution social scientists that APIs are designed to help third-party developers create applications that Twitter can use to make money, and do not necessarily serve the needs of the research community. These are all important considerations when working with data collected from social media. The limitations of a research project must always be carefully analyzed and acknowledged, and researchers must think critically about their data.

2. Existing research on Twitter bots and open-source development

Automated social media accounts are often portrayed in the literature as actors that endanger social media platforms by spamming users, distributing malicious code or using fake profiles to create an artificial grassroots movement that support certain political goals. The growing phenomenon of social media automation and in particular the use of bots on Twitter triggered a strong response from the information security research community around 2010. Many of the early publications documented efforts to detect automated Twitter accounts to prevent commercial spam or the distribution of links pointing to malicious websites (Chu et al., 2010; Lee et al., 2011; Song et al., 2011). The literature has also addressed a new class of more sophisticated social bots that can be described as "software agents mimicking humans" and more difficult to detect (Ferrara et al., 2016). These social bots could still have various intentions, or more specifically, they could have been programmed or deployed with different motivations. Stieglitz et al. (2017) recognized these nuances and created a two-dimensional typology based on intentions (malicious / neutral / benign) and the degree which the bots mimic human actors (low to none / social bots).

Table 1 *Categorization Scheme of Social Media Bot Accounts by Stieglitz*

Intent / Imitation of humans	Malicious	Neutral	Benign
High / Social bots	• Astroturfing bot • Social botnets in political • conflicts • Infiltration of an organization • Influence bots • Sybils • Doppelgänger bots	• Humoristic bots	• Chat bots
Low to None	• Spam bots • Fake accounts used for • Botnet command & control • Pay bots	• Nonsense bots	• News bots • Recruitment bots • Public Dissemination Account • Earthquake warning bots • Editing Bots, Anti-Vandalism Bots on Wikipedia

Note. Source: Stieglitz, 2017

The table above is based on an extensive literature review. Table 1 shows that while the literature has focused extensively on malicious social bots, it also recognizes neutral and even benign actors that use elements of human-like communication - satire bots and chat bots respectively. Malicious actors include astroturfing and influence bots as well as a special type of bot, the political bot, which is often used on larger networks to change opinion on political and civic issues. Woolley (2016) provided an overview of incidents in which these types of social bots "deployed by powerful political elites during global political events."

2.1. Twitter bots

Kaplan and Haenlein (2010) define social media by user-generated content and describe how content creation and sharing are based on the "ideological and technological foundations" of Web 2.0. According to Tim O'Reilly, who popularized the term in the mid-2000s, the technical foundation of Web 2.0 is based on the "principles and practices" that enable the development of interoperable applications and the design of the Web as a platform (O'Reilly, 2005). Social media is a socio-technical construction, and the

development of social media is strongly influenced by the idea that users are not only consumers of media content, but also producers of the content and co-developers of the technology. This idea goes beyond what Toffler (1980) called prosumers (users who are both the customers and producers of the content), because users also have an active role in forming the technology that underpins content. In this way Web 2.0 is "harnessing the power of the users themselves" (O'Reilly, 2005).

According to Benkler (2006), the Internet has created a new networked public sphere in which individuals are no longer "passive readers and listeners" anymore but become potential "speakers and participants in a conversation." This has also transformed traditional media, allowing non-professional actors to not only produce content and publicly voice their opinions, but also influence the news by reporting about newsworthy events through social media. Following Turner's work, Murthy (2013) refers to this shift as the demotic turn which gives more visibility to "ordinary people." Indeed, Murthy cites examples of ordinary people's coverage of newsworthy events when analyzing the Twitter coverage of the 2009 U.S. Airways plane crash and the 2008 Mumbai bombing. On the other hand, some critical studies have questioned the role of citizen journalism in transforming traditional media. Based on content analysis of four major newspapers, Rebillard and Toubol (2010) concluded that these mainstream newspapers still rely on professional news sources, and as they noted, the "simplistic conception of a digital revolution in journalism does not stand up to empirical verification for all its cultural power."

With the development of software solutions for creating and maintaining automated accounts on social media and the widespread use of bots on platforms like Twitter, the definition of social media that focuses on "user generated content" is no longer valid. It is becoming increasingly important for users to help shape the agenda by sharing and disseminating messages on social media. The recent Brexit and the U.S. presidential election have shown that newsworthiness is increasingly determined by uptake on social media platforms like Twitter. This is where automation plays an important role; Various research projects have estimated the traffic generated by bots on Twitter at 19-24 percent (Bessi & Ferrara, 2016; Cashmore, 2009).

Recent studies have found that a bit more than half of the Web traffic is generated and disseminated by automated accounts (Zeifman, 2016 cited by Gilani et al., 2019). A simulation conducted by Gilani's research group based on real-world Twitter data also found that bots are involved in 54.59 percent of the conversations when information

exchanged between two or more Twitter accounts (Gilani et al., 2017). In a more recent study, Pew Research examined links posted on Twitter that pointed to popular news sites and found that 66 percent of such links were promoted or shared by bots (Wojcik et al., 2018). The research team at Pew Research relied on the popular Botometer service (also known as BotOrNot), but the results were also fine-tuned and tested against a human-coded dataset. Another recent study focused on a special segment of Twitter communication, the discussion of he Covid-19 pandemic, and found that 45 to 60 percent of the traffic was generated by bots (Hao, 2020). In Russian language, the contribution of bot accounts in politics exceeded 50 percent in 2014-2015 (Stukal et al., 2017).

The following section of the literature review draws in part on a paper (Kollanyi, 2016) I wrote earlier on the development of open-source Twitter bots but has been updated with more recent literature. This section gives an overview of the most important research projects dealing with bots on Twitter, as well as the literature on the social aspect of code development using online code repositories such as GitHub.

2.1.1. Bot detection

The growing phenomenon of social media automation, and in particular the use of bots on Twitter, triggered a strong response from the information security research community around 2010. Many of the early publications documented the efforts of detecting automated Twitter accounts to prevent commercial spam and filter out tweets with links pointing to malicious websites (Chu et al., 2010; Lee et al., 2011). To detect Twitter bots, researchers have focused on various aspects of Twitter activity, such as sender-receiver relationship analysis (Song et al., 2011) and behavioral patterns. The methods include supervised and unsupervised machine learning (Chen et al., 2012; Davis et al., 2016; Efthimion et al., 2018; Minnich et al., 2017; Varol et al., 2017; Wang, 2010), detection of highly correlated accounts (Chavoshi et al., 2016), neural networks that analyze both content and metadata (Kuduguntaa & Ferrara, 2018) and the use of honeypots to catch Twitter bots (Lee et al., 2010).

Karataş and Şahin (2017) provide a good overview of the detection methods that are currently used to detect automated accounts (bots) in online social networks. The authors distinguish between structure-based, crowdsourcing-based, and machine learning-based bot detection methods and briefly discuss the limitations of such efforts. Similarly, Cresci (2020) looks into a decade-long literature on bot detection. This review distinguishes

between early bot detection efforts that focused on analyzing individual bots, often using some form of supervised machine learning, and network-based approaches. The drawback of the earlier efforts was the lack of reliable baseline data, in other words, the lack of large, reliable, labeled bot and human data. Therefore, these methods were subject to some bias from the beginning. In addition, bot developers had become increasingly sophisticated in hiding the activities of automated accounts, and it became increasingly difficult to distinguish between bot and human users (Cresci, 2020). For example, headless browsers can completely avoid using Twitter's APIs, making them difficult to identify for both Twitter and third-party researchers (Gorwa and Guilbeault, 2020; Suchacka & Iwański, 2020). As a result, novel approaches employed some form of group detection (rather than attempting to evaluate or score individual accounts), and these bot detection methods more often used unsupervised or semi-supervised machine learning (Cresci, 2020).

On the other hand, Twitter itself has publicly questioned the accuracy of academic (and commercial) research on Twitter bots, claiming that most research projects overestimate the traffic generated by automated accounts (Roth & Pickles, 2020). Twitter's head of site integrity, along with the director responsible for the company's global public policy strategy, published a post on the company's official blog accusing research projects that rely on machine learning and a human-coded training dataset of strong human bias. According to the blog post, the training dataset was coded by people who only have access to a limited amount of metadata about the accounts and therefore have to incorporate their biases into the bot detection algorithm. Nevertheless, Twitter, just like any other major social media platform, has deployed sophisticated algorithms to combat spam. Lin and Huang (2013) studied the spam accounts that were able to survive for a long period of time without being caught by these algorithms.

In addition to spam bots that have survived for a long time, the literature has also looked at a new class of more sophisticated social bots that can be described as "software agents mimicking humans" and are more difficult to detect (Ferrara et al., 2016). Note: As Gorwa and Guilbeault (2020) note, the terms "social bot" and "socialbot" are used in two different ways in the social sciences and computer security literature, respectively. While some social scientists use the term social bot simply to refer to all automated accounts on social media (e.g., Hwang et al., 2012), the term is also used to refer to human-like algorithmic activities (see, e.g., the Ferrara paper mentioned above).

Twitter bots are often perceived as content polluters or a threat to security, and authors often use harsh language, such as claiming that bots weaponize the vaccine debate

(Broniatowski et al., 2018), armies of bots spread misinformation (Barners-Lee, 2017), and bots fought alongside humans in the Syrian civil war (Howard, 2015). However, even the early literature acknowledges that not all Twitter bots are evil. Chu et al. referred to the automated programs used on Twitter as “double-edged swords”: while some bots serve legitimate purposes, such as delivering news or automatically generating updates for a service, others just spread spam (Chu et al., 2010). A few years later, Ferrara et al. (2016) explicitly distinguished between benign and malicious Twitter bots. The latter category includes bots that aggregate content or produce automated responses for companies to provide customer support.

Twitter bots act automatically without direct human involvement. Code that requires constant monitoring and/or human action to perform a Twitter-based task is not a bot. The involvement of a human actor makes an important distinction between Twitter bots and accounts that are often referred to as cyborgs in the literature (Chu et al., 2010; Chu et al., 2012; Haustein et al., 2016). The term cyborg has two different meanings in the context of automated social media accounts and Twitter bots: The term can be used to describe a computer code that helps a user perform certain actions on Twitter, or it can be used to describe a Twitter account whose actions are partially human and partially code-based. A good example of the latter type is a Twitter user who, in addition to posting original tweets on Tweets, employs code to automatically retweet messages that originate from a specific user or contain specific hashtags. Another simple code can be used to automatically follow back users, and this code can also run in the background. This type of cyborg does not use automation to create content, but to perform simple but often tedious tasks, such as following other users or unfollowing hundreds of users at once or in small increments. This type of automation is often not visible on the platform, but if you track an account's activity over time, you can uncover or detect cyborgs. Researchers have also found that cyborg accounts tend to exhibit human-like tweet patterns. There are even “human spammers” who abuse the platform and post in an algorithmic manner (see, for example, Clark et al., 2016).

Communication scholars have also looked at the credibility of automated social media accounts. These findings suggest that social media users perceive Twitter bots as credible sources of information under certain circumstances (Edwards et al., 2014). Based on Japanese media reports, scholars have claimed that an automated Twitter account proved to be a reliable and critical source of information during a particularly severe earthquake in Japan when other forms of local communication were disrupted (Haustein et al., 2016). Other scholars examined so-called robotic journalism, the use of news bots in various

phases of content creation, from identifying newsworthy events to curating and analyzing data to writing (Lokot & Diakopoulos, 2016; Steiner, 2014).

2.1.2. Twitter bot typology

Twitter bots have been used for a variety of topics, from news to education to politics. Automated accounts can be set to tweet massively to promote a political ideology or commercial product, or they can tweet far less and even set to answer specific questions from interested users. Both the different genres (or tasks) and the different tweet patents have been covered in the literature on Twitter bots. The literature includes papers on spambots (Chen & Subramanian, 2018, Cresci et al., 2017), bots promoting academic papers (Haustein et al., 2014), newsbots and chatbots used by the media (Diakopoulos, 2019; Jones & Jones, 2019), bots used in crisis communication (Brachten et al, 2018; Hofeditz et al., 2019) and in politics (Bastos & Mercea, 2019; Caldarelli et al., 2020; Ferrara, 2017; Howard, Woolley & Calo, 2018; Wooley & Howard, 2016a, 2016b), as well as anti-harassment bots (Geiger, 2016).

In addition to focusing on specific types of Twitter bots, there are three articles in the literature that attempt to categorize social bots or Twitter bots. The remainder of this section describes each attempt. Later in my thesis, I introduce a typology developed specifically for open-source Twitter bots, and I provide an overview of the frequency of these bot types within the sample of open-source bots I worked with.

Oentaryo et al. (2016) introduced a framework for systematically profiling automated social media accounts in three broad categories - broadcaster, consumption, and spam bots. The authors noted that early literature on automated social media accounts focused almost exclusively on malicious bots, with the assumption that these bots should be removed from social media platforms, and that benign automated accounts were rarely discussed. This is even more true of the computer security literature and the literature on detecting and profiling bots.

Oentaryo's categorization focuses on who initiates the automated communication, regardless of who operates the account. The model assumes that benign bots are either operated by organizations and groups of people seeking to disseminate information (broadcaster bot) or used by individuals seeking to access information from multiple sources, often in an aggregated format (consumption bot). The latter category includes services that send automatic updates on request - for example, a personalized horoscope

service or a Twitter bot that sends a user a notification when another account unfollowed his or her account. Spam bots work the same way as broadcast bots, but the content is either malicious (e.g., links to malicious websites, viruses) or they simply aggressively promote content.

This categorization is useful because it recognizes that not all automated accounts are malicious. The article also provides important insights into bot detection by comparing tweet patterns and a large number of metadata fields across human and (both malicious and benign) bot actors' accounts. Another strength of the article is the discussion of social media automation tools in the context of automated social media accounts. Semi-automated or cyborg accounts often use social media management software to generate content, but fully automated accounts also often rely on social media management software (e.g., HootSuite) or web services (e.g., IFTTT). However, the categorization is oversimplified and mixes intent, tweet activity, and direction of communication. In addition, the exact method for distinguishing between consumption and broadcaster bots is not clear, and there could be many accounts that could be labeled as both.

Maus and Varol (2017) wrote an occasionally funny short paper on socialbots for the 2017 Web Science Conference that provided a comprehensive typology based on five dimensions, including the bot's goal, coordination, the operator behind the bot, and its transparency. (Onur Varol was instrumental in developing the bot detection platform Botometer at Indiana University, Bloomington.) Regardless of the funny tone, this is a much broader approach to bot categorization, and this paper addresses both benign (e.g., content curation, entertainment, motivator, etc.) and malicious (e.g., network graph or metrics manipulation, channel disruption, etc.) bots. A variety of examples from the literature are given for the different sub-dimensions.

Table 2 *Bot Typology Developed by Maus and Varol*

Dimension	Type
Alleged Humanity	Honest Bot Intentionally Ambiguous Fleshmask
Operator	Individual Government NGO Commercial
Bot Coordination	Hive Aspen Singleton
Operator Transparency	Covert Overt
Objectives	Novel Content Broadcasting Content Promotion Content Curation Information Acquisition Channel Disruption Network Graph Alteration Transaction Entertainment Proof of Concept Metric Manipulation Motivator

Note. This is an edited version of the typology originally developed by Maus and Varol.

Gorwa and Guilbeault (2020) developed a quasi-typology for bot accounts. In the paper, the authors focus on the lack of clarity in defining bots and the political consequences of the ambiguity of the term. According to the authors, the term robot (or bot in a shorter form) has been used to refer to a variety of automated and semi-automated actors on the Internet and even to non-automated accounts. On the other hand, some activities of automated accounts are not visible to users at all, and other seemingly automated accounts are actually operated by humans, the authors argue. In the first part of the paper, they give examples of six types of bots. Four of them tend to be fully automated: 1) crawlers and scrapers that download data or follow web links, but these accounts are not visible to human users; 2) chatbots that interact with human users and engage in meaningful conversations; 3) Spambots, which may simply be infected computers that distribute unsolicited content, or social media accounts that usually spread spam messages in an automated manner; 4) Social bots, which are either just bots or social media platforms or social media accounts that mimic humans. The last two are either 5) human troll accounts or 6) cyborgs and hybrid accounts that are part human and part machine. This paper also raises the issue of accounts

that use available tools such as the popular service If This Than That (ITTT) or a social media or content management system (e.g., SocialFlow or Buffer) for automation.

In addition to the underlying mechanics (e.g., bot code vs. social media management tool), Gorwa and Guilbeault's paper suggests examining the function or end goal of the bot. However, this paper does not provide a complete typology, but argues against the broad use of the term and speculates on the possible dimensions that can be used to build a meaningful and relevant typology.

2.1.3. Twitter bots and high salience events

The use of automated accounts in social media to manipulate high-profile (political) events has been studied by researchers in the fields of information security, communication studies, and political science. This line of research has examined how political actors can gain political advantage through the use of bots. Ratkiewicz et al. (2011) described Twitter as a battleground of modern politics where political actors can use computer code to create the illusion of an artificial grassroots movement (astroturfing) or conduct smear campaigns. Metaxas and Mustafaraj (2010, 2011) gathered evidence of the use of automated accounts during the Massachusetts Senate race-the authors analyzed how messages from these Twitter accounts could influence search engine results, a technique known as Twitter bombing. Other early research projects also used Twitter data collected during U.S. elections to examine the use of automated accounts to retweet and report content associated with specific political groups (e.g., Lumezanu, Feamster & Klein, 2012). Woolley (2016) provided an overview of incidents in which social bots were "deployed by powerful political elites during global political events" in countries such as Argentina, Russia, Saudi Arabia, the United States, and 13 other countries.

Although the use of bots has been reported in the news around the world, including in many authoritarian regimes, the 2016 U.S. election and, to some extent, the 2018 midterm elections likely had the strongest impact on public discussion of Twitter bots (Bessi and Ferrara, 2016; Badawy, 2018; Luceri et al., 2019; Howard et al., 2017).

After careful review of the data collected from GitHub and Twitter, and a discussion with my committee, I decided not to include most of the analysis on Twitter bots and politics. Hence, this dissertation does not cover this topic in great detail, but I have suggested further research on this topic using a different methodology in the section on future research directions.

2.2. Open-source software development

2.2.1. Sharing code and the social use of GitHub

Previous research on Twitter bots has mainly focused on the outcomes of bot activities in order to regulate the bot ecosystem: Detection and categorization of bots into desirable and unwanted, or along similar simple distinctions. While this approach focuses heavily on detecting and describing bots already deployed on Twitter, these research projects have not addressed the development of the code and actors behind the bots. Therefore, to learn more about the creation of Twitter bots, I decided to investigate the process of writing and sharing bot code on GitHub.

GitHub is the largest online repository for shared computer code and for open-source developers it is the de facto solution for collaborating and sharing their work (Gousios et al., 2014). Therefore, GitHub is a good place to explore open-source codes for Twitter bots and the social arena around bot development.

GitHub has always advertised that it is an online repository hosting service with version control to support "collaborative development of software." In the early days of GitHub, one of the advantages of the platform over other code sharing and version control solutions was actually the social features of the site. For example, GitHub users can track the work of other users on the platform and acknowledge the programs written by other developers, which is called "staring a repository." Developers can also create a copy (fork) of another user's repository and work on that version independently. The changes in the forked repositories can later be integrated into the original repository.

Researchers have studied the social aspect of coding on GitHub using both qualitative and quantitative methods. These results are also relevant to the problem of how users learn through social sharing on Twitter bots. Margaret-Anne Storey et al. (2014) conducted a survey on GitHub by contacting 7,000 active users. According to the study, there is a new generation of developers characterized by heavy use of social media for communication, coordination, collaboration, and learning from others. By providing social tools for sharing code, GitHub has effectively lowered "the barriers to joining and contributing to development communities" (Storey et al., 2014).

Dabbish et al. (2012) provide a good example of the potential of qualitative research on GitHub. In this study, researchers used in-depth interviews to understand how social dynamics influence the way individual developers engage with code. They examined how users followed the work of other developers and whether this type of transparency, the

public nature of work on GitHub, led to learning and increased collaboration between developers (Dabbish et al., 2012). The researchers concluded that developers tended to produce cleaner code and committed changes less often when many people were watching their projects, but they also experimented less because of this social pressure. Still, GitHub users learned from each other by observing how others solved the same problem.

There is a growing body of literature based on GitHub data that examines how developers use the collaborative code repository. This growing interest is explained in part by the availability of data, the availability of big data technologies for collecting and distributing vast amounts of data, and in this particular case, the emergence of methods for overcoming the limitations on data collection imposed by GitHub as an organization.

Lima et al. investigated both the network-like features of GitHub and the collaborative aspects of the platform (Lima et al., 2014). This project was based on a very large dataset and showed that social ties have a relatively low level of reciprocity compared to other platforms, and active users do not necessarily have a large social network on the site. While there are some very visible cases of collaboration on GitHub, such as Linus Torvalds' Linux source code, the results also suggest that the site's strength lies in fostering a social learning process where users can view and appropriate the published code for their own purposes.

GitHub provides easy access to its repositories (i.e., shared code) and to metadata about the repositories through an application programming interface (API). Metadata available through the API includes the number of contributors, some measures of the repository's popularity (stars and forks), and information about what programming language the code was written in. Additional information is available about each user, such as location, when they joined the platform, number of shared repositories or programming code, and contributions to other repositories. The site has a more liberal API policy than many other sites (e.g., Google or Twitter), both in terms of the number of API calls a user can initiate and the level of detail provided. GitHub users generate a large amount of data: According to Gousios (2013), Github produces 200,000 events on an average day, which equates to just over 8,300 events per hour.

Data-driven research projects also investigated developers' choice of programming language (Sanatinia & Noubir, 2016), the geography of the open-source community (Takhteyev & Hilts, 2010), or the use of the platform in education (Zagalsky et al., 2015). For a more detailed overview of GitHub-related software research, see the review by Kalliamvokou et al. (2014).

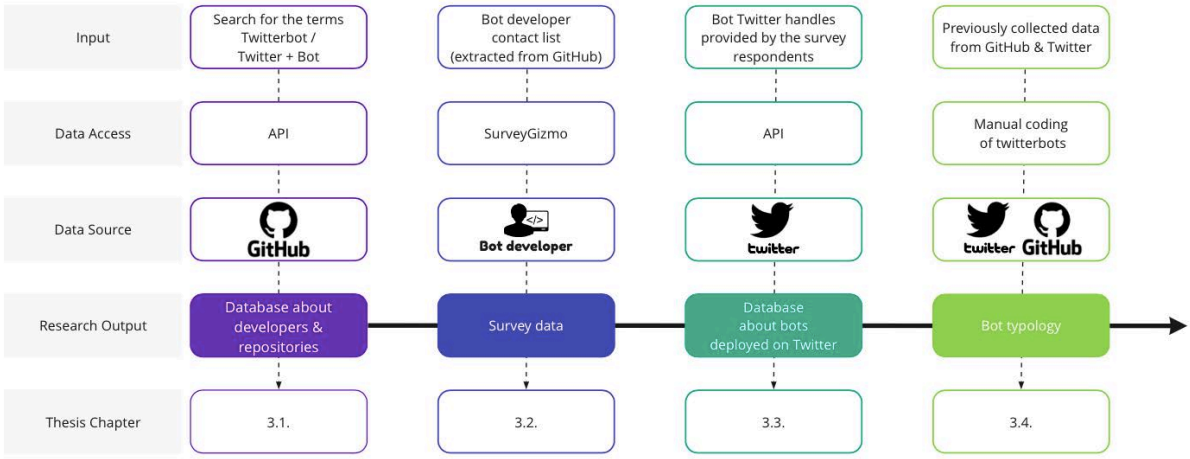
In addition to research projects investigating specific questions, the collection and curation of data on GitHub is an important research direction in its own right. There are at least three major attempts to download any GitHub repository for the purpose of archiving and research (Gousios, 2013; Grigorik, 2012; Sanatinia & Noubir, 2016). The researchers behind the GitHub Archive and GHTorent dataset decided to publish and share their datasets with the research community to facilitate further research, as they claim that the data available about GitHub users is still "largely underexplored in terms of academic publications" (Gousios et al., 2014). Eventually, GitHub itself decided to copy all active repositories on the site to archival film and store them 250 meters below ground in a decommissioned coal mine in Sweden near the North Pole (Metcalf, 2020).

3. Methodology

3.1. Research design and data collection strategy

Data collection for my dissertation is a four-step, often iterative process that includes initial data collection from GitHub via the website's APIs, a survey of bot developers, Twitter data collection, and more specific data collection from GitHub. These steps are designed to build on each other and therefore can only be done in a specific order. Figure 1 provides an overview of these four steps and how I accessed the data, as well as the research results generated after each step.

Figure 1 *Data Collection Strategy for the Thesis*



Since I was interested in open-source bot development, I began my data collection by exploring how to access data from GitHub, the largest online code repository. The website has an API that allows one to search for specific keywords in the name and description of the repositories. The next section of the methodology chapter (2.1.1.) provides a detailed overview of the data collection process, focusing on GitHub. The result of this data collection was a large database of metadata downloaded from the website about bot repositories and about all developers who had at least one bot repository available on the platform. Although the data collection was done solely through the API by writing a script in Python, I periodically visited the GitHub website and compared the raw data with the information available on the website to make sense of the collected data.

After collecting and analyzing the data from GitHub, I decided to contact developers to ask more questions about the motivation behind bot development, the skills required,

interaction with other developers, and the biggest challenges in developing a bot and deploying it on Twitter. The GitHub dataset collected during the first phase of data collection allowed me to extract the contact information of all developers who set their email address public on the platform. I used a paid online survey tool called Survey Gizmo, which allowed me to contact all developers by sending emails in mass. For a more detailed overview of the survey design and data collection, see Section 3.1.2.

After finding a large number of open-source bot codes on GitHub, I was curious to see how many of these bots were deployed on Twitter and what I could learn about the activity of such bots. Although some of the bot repositories include either a Twitter handle or a link to Twitter in the description or readme.md file uploaded to the repository, this is not always the case. Therefore, I decided to include a question in the survey asking developers to provide a list of the Twitter handles of the bot they use on Twitter. This way, I could link data about the developers on GitHub (including their bot repositories) to the bots deployed on Twitter. I also had survey data available for the bot developers. For more information about collecting data on Twitter and how I linked bot repositories to bots on Twitter, see Section 2.1.3.

During data collection, I made several small decisions that limited the generalizability of the results of my research, but my goal was never to collect as much data as possible. Instead, I wanted to develop a data collection strategy that was sound, easy to understand, and repeatable. (Note: The Limitations section in the Discussion chapter provides an overview of the limitations of the results in light of the above decisions.)

The following list provides an overview of the four major data sets collected during the data collection phase of my research project:

GitHub dataset: Metadata about 19K open-source bot repository;
Survey responses: Survey results from 860 bot developers;
Twitter dataset: Data and metadata about 321 open-source bot repositories with paired source code on GitHub (500K tweets and metadata, including user engagement);
Manual coding database: Data about 321 Twitter bots – manually labeled.

3.1.1. Bot repositories and bot developers on GitHub

The first step of my data collection focused on finding bot developers and bot codes by searching bot repositories on GitHub. GitHub has an API that allows users to get a list of repositories that contain certain search terms in their name, description, or readme file. Initially, an API call found GitHub repositories containing both the words "Twitter" and "bot." This initial search returned 17,722 results, and an additional search for the term "Twitterbot" as a word returned an additional 3,566 results. After combining the two datasets and removing the duplicates (the overlap between the two terms), the final dataset contained more than 19,000 repositories.[4]

Working with online social data, especially when accessing user-generated data and relevant metadata via multiple API calls, is in many ways an exploratory and iterative process. It is also a grounded process: the research questions are formulated as the researcher becomes familiar with the data. In other words, the initial collection and analysis is often designed to explore the questions a researcher may ask in a cursory examination of the data. A cyclical and systematic process of data interrogation creates the context for further analysis.

[4] 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 only using the term "Twitter bot" in itself would be higher than 26,000. This research proceeds with the initial dataset, which does not include forks.

Figure 2 *The Structure of the Data Collection*

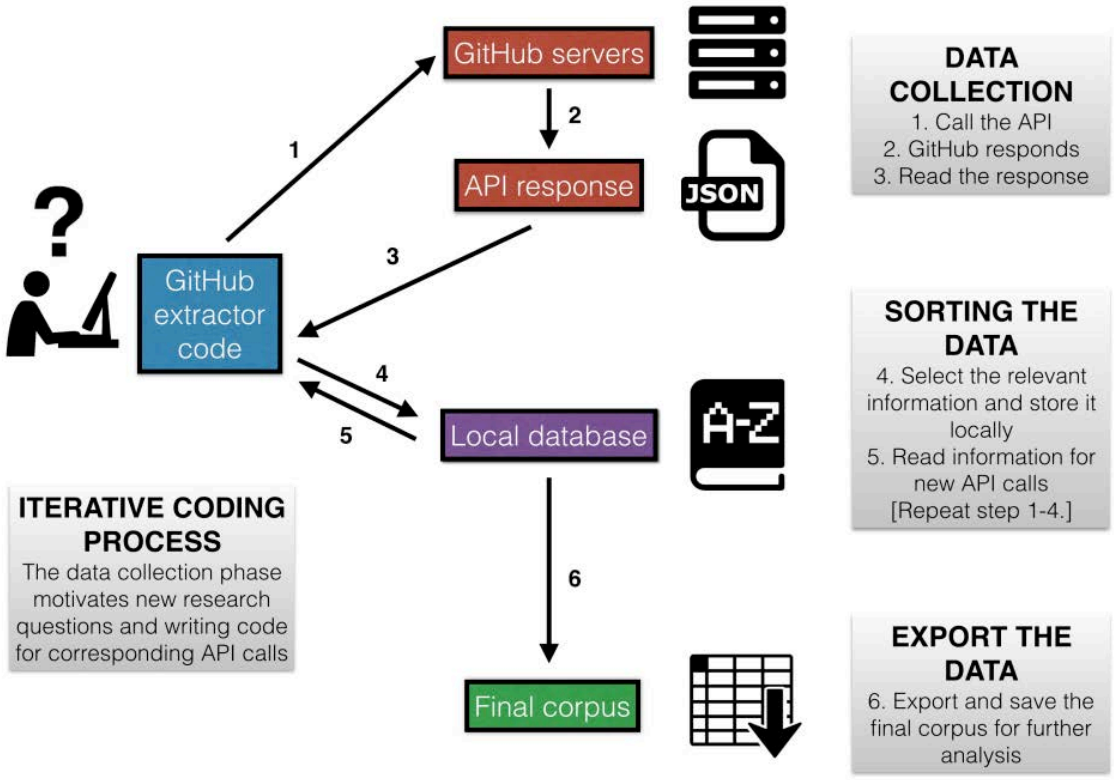


Figure 2 describes the steps of collecting data from GitHub. This approach to data collection is an iterative process. The researcher first queries the API and downloads the raw data. Then, all relevant information is processed (parsed and cleaned, if necessary) and stored in a local database. To make sense of this data and make new API calls, the researcher needs to think about the research questions as well as the structure, availability, and quality of the data.

Although the most important information about repository owners is readily available through the search API, additional information can only be obtained through separate API calls, one for each repository. These additional API calls are relatively easy to create, as they follow the same syntax for each repository. This additional step in the data collection process provided important insights about users, such as when they registered, their location, the number of public repositories, and the number of followers they have. Further requests could provide additional details, such as the readme file for each repository or the usernames of those who forked the repository on GitHub.

API calls must follow a specific syntax, for which there is extensive documentation

on GitHub (similar to APIs for other platforms). To learn the syntax and understand the scope and context of the available data, three different sources were used: the API documentation, the structure of the information on the website, and the data available by querying the API.

1. GitHub provides extensive documentation for its API. The language of the documentation is mainly geared towards developers and describes the proper syntax for accessing the data. It is also a useful tool for gaining insight into the kinds of questions a social scientist might ask of the data and for understanding the potential limitations of data collection.
2. Using the interface of GitHub was also helpful in understanding the mechanics of the platform. I regularly browsed through the various repositories created by bot developers and looked at the site's interface to contextualize the data. Because of the liberal API policy of GitHub, almost every piece of information available on the site is accessible through various API queries, so studying the website also helps develop ideas for the types of questions that can be asked through the API.
3. I started with a small, incomplete but easy-to-manage sample dataset of bot repositories. This approach allowed me to learn about the limitations of the API, the challenges of collecting data, and the possibilities of extending the scope of data collected.

Once we had developed a strategy for downloading the data, it was important to decide how to store the information. Figure 2 provides an overview of the iterative process of data collection and storage that led to the final dataset. This process began with the initial dataset of more than 19,000 repositories that contained the words "Twitter" and "bot" or "Twitterbot" in their name, description, or readme file. This dataset was then expanded by additional targeted API calls to GitHub.

Using the unique identifier that GitHub provides for each repository made it easy to add new entries and incorporate additional information into the dataset. It also makes it possible to update the dataset with newer bot repositories in the future. This was extremely useful in early 2021 when I updated the original dataset.

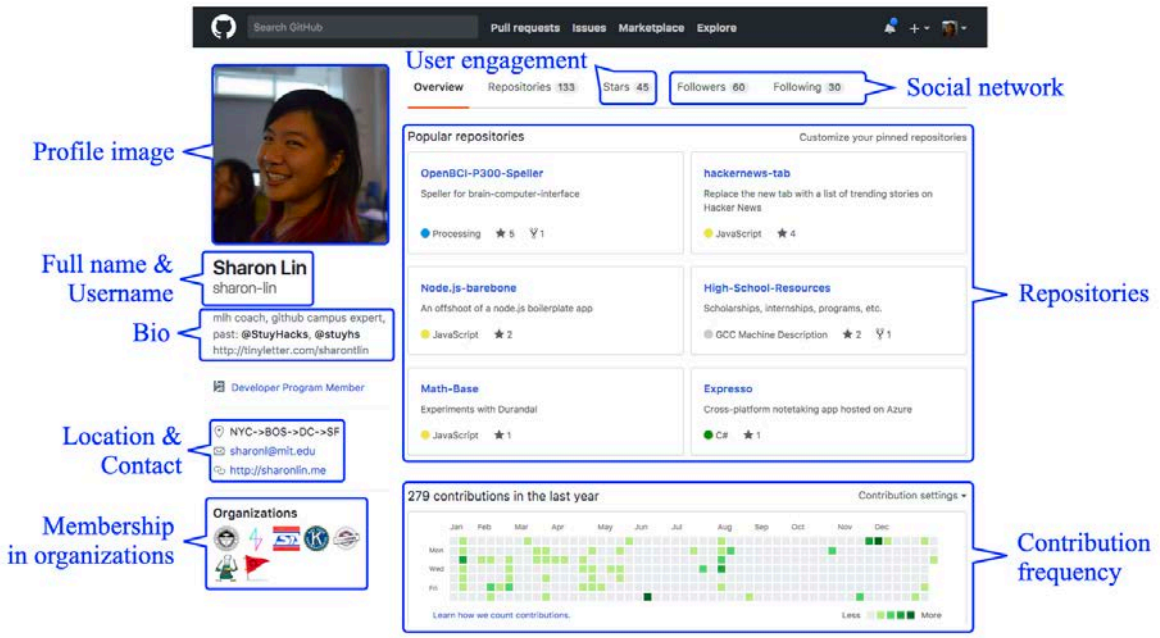
Interestingly, the methodology I developed and published in an article in 2016 had inspired a research group at the University of Münster (Germany), to investigate bots (Assenmacher et al., 2020). The researchers reused my methodology (and referenced

properly) and also extended it in a nice way: they focused on the top 5 code sharing platforms (including SourceForge, Bitbucket, GitLab, and Launchpad in addition to GitHub) and included other popular social media platforms like Telegram and WhatsApp in their analysis. This shows how the methodology can be generalized or used to collect data to answer a different research question. The paper also compared the available open-source bot repositories with commercial services or solutions available on both the public internet and the darknet[5].

3.1.2. Open-source bot developer survey

The Users API of GitHub provides access to a plethora of metadata fields about developers on the platform. The platform allows users to add a wide range of personal information to their profile, such as a biography or a location, links to social media platforms or websites, and contact information. The available information can be better understood if we take a look at a typical GitHub user profile, which is publicly available on GitHub's website. Figure 3 is an annotated screenshot from GitHub showing an example of a typical GitHub profile page.

Figure 3 *Annotated Screenshot of a Typical GitHub Profile*



[5] The darknet is the part of the internet that is only available through special configurations and/or software, e.g. peer-to-peer networks which can be used for downloading music and films.

Note. Source of the original screenshot: Sharon Lin, Medium.com. Annotated by the author.

A typical profile on GitHub includes a picture of the user (sometimes it's a photograph, but often it's just a cartoon or logo), a name (e.g., first name, last name), and a user handle, usually a short bio, often a company name, a link to a blog and/or a link to a personal Twitter profile, an email address, and some sort of location information (often a city and country). However, all of this information is optional, with the exception of providing a valid email address when registering and selecting a username.

Although providing an email address is mandatory for the registration process, it is completely optional to publish the email address in a user profile. Users have the option to either add their primary email address to their profile or use a secondary email address and hide the original email address. At the time of writing, the visibility of the email address associated with a GitHub user profile is set to hidden by default, and publishing an email address in a user profile is an opt-in feature. In other words: After registering with an email address, users must actively publish their email address in their profile if they want other users or site visitors to contact them. The decision to publish an email address could indicate a variety of attitudes, from encouraging openness and communication to creating transparency in the development of open-source software. Another potentially common reason for publishing an email address on a GitHub profile is so that professional recruiters can contact a developer on GitHub with a job offer more easily.

In addition to the above reasons, GitHub itself encourages users to publish their email address in their profile (Wanstrath, 2012). The platform offers only a fairly limited number of ways to communicate with other users. Since 2012, users have not been able to send messages to each other within the platform. One way to get around this limitation of the platform is to open an issue in the other GitHub user's repository or create a new repository and add an at mention (@username) to call the attention of the other user. (Users usually get a notification when someone mentions them). The other option is to contact the user through another channel - either Twitter (if a link is provided) or email. The above limitation could also encourage the practice of publishing an email address in a GitHub profile.

The Search API resulted in a list of 16,168 bot repositories in the summer of 2020 (note: by the end of the same year, the number of bot repositories exceeded 19,000). Because some users published more than one Twitter bot repository (that matched the keywords I used), the number of unique developers was slightly lower. In the end, I

retrieved the detailed user profiles of 14,302 unique GitHub developers via the User API. These users were mostly individual users, but a small number of profiles were institutional profiles. After processing this data, I was able to identify 5,591 valid email addresses in the user dataset.

After extracting the email addresses, I decided to use SurveyGizmo to host the survey and manage the survey request emails. (Recently, SurveyGizmo changed its name to Alchemer.) SurveyGizmo had a list of prohibited email addresses that included email addresses beginning with certain words, for example, info@, infomation@, tech@, and all@. No emails were sent to these addresses via the platform. Most of these addresses are either channeled to multiple people (e.g., all employees in the department IT) or at least not to a specific person, e.g., info@example.com. According to Survey Gizmo, sending surveys to these addresses results in a much lower response rate, as they are usually not answered and often result in some of the IP addresses used by the survey platform being blacklisted. However, the number of developers who used such email addresses was minimal in my GitHub sample. Nevertheless, all such email addresses were removed before the email campaign for the survey was set up.

The online data collection ran for 3 weeks, from July 20 to August 16, 2020, but most respondents answered the questionnaire during two peak periods within that time frame. The first data collection wave focused on users who had posted the word “twitterbot” either in the name of one of their repositories or in one of the repo descriptions or readme files in their repository. The second wave focused on developers who used both the words “twitter” and “bot” and were not included in the first wave. (There was significant overlap between these groups).

The main reason for using a commercial online survey tool instead of setting up my own open- source survey tool or using a simple online form was the ability to send invitations to more than 5000 users. This online survey tool also had the ability to send reminders to users who did not respond to the questionnaire. Any developer who did not respond received a reminder note about a week after the first emails were sent. Finally, SurveyGizmo also gave developers the option to opt out of any further communication, resulting in fewer unsolicited emails. SurveyGizmo also allowed me to pay for only one month instead of signing up for an annual subscription, and offered conditional logic in the survey as well as a pretty decent customizable design.

The invitation to participate was sent to 5,488 email addresses, and in the end 1,008 respondents completed all or part of the survey. After removing the incomplete responses,

I had 813 participants. Thus, the response rate for this survey was 14.8%, which compares favorably to, or in some cases slightly exceeds, similar online surveys focused on GitHub (Begel & Nagappan, 2007, Tao et al., 2012, Vasilescu et al., 2015).

The response rate of a survey depends on a relatively large number of factors. In the case of online surveys, reasons for lower response rates could include email checking habits, lack of interest in the study, and survey length (Saleh & Bista, 2017). Saleh and Bista recommend not only targeting an audience interested in the research topic, but also writing personalized invitations and sending reminders if respondents do not complete the survey. The online survey tool allowed me to add a personalized greeting to each email address and send a reminder to developers who did not respond and opt out of further communication. Most of the incomplete responses were from respondents who exited the survey very early, shortly after the first block of questions - this suggests that the length of the survey did not contribute significantly to the number of incomplete responses.

Another caveat to the survey: the language of the survey was English, and this fact may also introduce some bias in my sample. I was curious about how well GitHub users speak English. The best information on GitHub developers' language skills comes from a large-scale survey conducted in 2017 as part of a collaborative project with researchers from academia, industry, and the open-source community (Geiger, 2017; Zlotnick et al., 2017). According to this survey, 77% of open-source developers speak and write English very well, and another 19.94% reported moderately good speaking/writing skills reported by Geiger[6].

[6] The advantage of this (baseline) survey is that all the questions were available in 5 major languages - Spanish, Chinese, Japanese, and Russian, besides English. The survey was a large scale survey with respondents from a wide range of GitHub project. The final sample included 5,500 randomly selected developers from more than 3,800 open-source repositories on GitHub.com respondent to a survey, but the same survey also included 500 respondents from communities that work on open-source projects by using other platforms (cf. numbers presented above). Although the non-GitHub user sample is only about 9 percent of the GitHub sample, it is possible to filter out these respondents. The raw data of this 2017 survey is available on GitHub, so I could get an even more precise picture about the language skills of the developers on the platform. Within the sample of GitHub users, 76.5% speaks very well in English, while another 20.29% speaks moderately well.

3.1.3. Open-source bots deployed on Twitter

Bot developers who publish their bot's source code on GitHub and deploy the bot on Twitter can add a link pointing to the bot account to their GitHub bot repository. Similarly, developers can put a link in the profile information (bio) of a Twitter account that points to the bot's source code (a repository on GitHub). However, not every bot repository contains a link to Twitter and not all bots are transparent about their code, so it is not always easy to identify the creation of a bot developer (of a particular bot) on Twitter.

To identify bot accounts on Twitter and create a truly cross-platform dataset, I included a question in my online survey asking respondents to create a list of their bot accounts on Twitter. The goal was to create a complex dataset where I could analyze the following three sources together: 1) survey results, 2) bot repositories on GitHub, and 3) bot accounts on GitHub. The advantage of having a list of bot accounts linked to a specific GitHub developer (especially if it can be linked to a specific bot repository) is that the bot code can be studied together with the activity of the bots that were deployed on Twitter. It was often challenging to link the list of Twitter accounts shared by a particular developer to the repositories containing the code used to run the bot.

Participants in the bot developer survey collectively provided more than 800 automated accounts when I asked them to list the user handles (Twitter usernames) of their bots. Unfortunately, some of the accounts were no longer available on Twitter (for example, suspended bot accounts have no visible content on Twitter) or it was impossible to connect to a specific bot repository on GitHub.

There are at least three different situations that make it difficult or impossible to investigate a deployed bot and the corresponding bot repository on GitHub. 1) A bot can be suspended. Automated accounts are often caught violating Twitter's terms of service. These bot accounts can disappear from Twitter if the platform suspends the entire account. If the bot account is no longer available on Twitter, it is often difficult to establish a connection between a repository and the Twitter account unless there is a unique link, such as a link to a specific Twitter account in a repository's description. In addition, the content of these suspended accounts is no longer available. 2) A bot account can be deleted. If for some reason developers themselves decide to remove their bot activity from Twitter, the entire account can be deleted or all tweets posted from the account can be removed simply by making a few API calls. There can be several reasons for deleting the traces of an automated account, from trying to avoid detection by Twitter's algorithms to simply

rededicating the account to another task. 3) Twitter allows users of the platform, including bots, to change their usernames. Even if the account and its previous communications are available on Twitter, the user handle can be changed to (almost) any name not currently used on Twitter. (Every user registered with Twitter has a long, unique numerical ID that can be used to identify the account even if they change their username, but I did not have access to that information.)

To identify the Twitter accounts uniquely associated with a particular bot repository on GitHub, I manually linked the lists of accounts provided by the respondents and the repositories belonging to one of the developers who responded to the survey. I relied on a combination of manual and computational analysis of three different GitHub data fields. First, I compared the repository name to the Twitter handle provided - repository names often matched the user handle on Twitter. For example, both a repository on GitHub and an account on Twitter might have the name WeatherBot. I then downloaded all the metadata for the bot repositories on GitHub and the readme files for each repository that had a readme file. In this dataset, I computationally searched for specific text strings. The at symbol was often used to indicate a Twitter handle in the repository description. For example, a repository might contain text similar to this, “You can find the bot on Twitter at @WeatherBot.” Similarly, I looked for links pointing to the Twitter.com website. For example, “Check out my bot at https://www.twitter.com/weatherbot/.” Finally, I looked for the same patterns in the readme file uploaded to the repository (at symbol or a Twitter link). Note: Neither a meaningful description nor a readme file are mandatory parts of a GitHub repository, but users almost always add a short description to their repositories and the use of a readme file is also very common. A readme file is a simple file written in a markup language that is uploaded to a repository. It is automatically recognized by GitHub and can be read by users visiting the repository on the website.

In addition to looking for possible clues to identify a Twitter account on GitHub, I downloaded from Twitter all the user profile data for the bot accounts used. Remember that the bot developers have listed the user handles of their bot accounts. I simply downloaded the user profiles for all 800+ bots enlisted by survey participants. The profile information or bio of a Twitter account may also include a link to a specific GitHub repository, or the user profile may be linked to a website or web page. For example, a developer can link a Twitter profile to a specific GitHub repository using a web link.

3.2. Programming work for the dissertation

Open research brings transparency and reproducibility to science (Nosek et al., 2015). In addition to reproducibility, open research requires transparency in methodology and data collection, as well as publication of the code behind a research project. According to Nosek et al, all of this should be encouraged by scientific journals, but the current academic system does not adequately reward these open practices. The lack of open research has led to a credibility crisis in the hard sciences, such as biology, and also in the human sciences, such as psychology (on the "crisis of confidence in psychological science," see Pashler & Wagenmakers, 2012). It is not surprising that Science Magazine, one of the most established academic journals, has responded by introducing a specific policy on replicability, and the journal's editor-in-chief has emphasized the importance of producing peer-reviewed studies that are replicable in order to restore trust in science (McNutt, 2014). However, conducting open-ended research requires significant research effort from researchers working with vast, often unstructured and semi-structured datasets, including data collected from social media platforms, Internet of Things devices, and other forms of machine-generated data (Hashem et al., 2014).

Computational social science has its own challenges when it comes to open research. These include the difficulty of sharing massive datasets from social media platforms, problems working with "found" data, and the lack of research transparency. A recent journal article by Lazer et al. (2020) warns that these platforms were never designed to answer research questions and that platforms often change from one day to the next. As the authors put it, "The design, features, data recording, and data access strategy of platforms may change at any time because platform owners are not incentivized to maintain instrumentation consistency for the benefit of research" (Lazer et al., 2020). This means that working with platform data raises all kinds of issues with internal and external validity and often makes replication of research very difficult. Replication of research is also difficult because the "raw data are often unavailable to the research community owing to privacy and intellectual property concerns, or may become unavailable in the future" (Lazer et al., 2020).

API-based data collection for this work has focused entirely on public data, yet data sharing itself presents its own privacy issues. Both intellectual property and privacy reasons are indeed relevant here, especially for the data collected by Twitter. The platform's "Developer Agreement and Policy" explicitly restricts the redistribution of Twitter content,

allowing only the sharing of tweet IDs and user IDs even for academic purposes (Twitter, 2020). These unique identifiers allow other researchers to download the original tweets, but only if they are still publicly available on the platform. This means that the company retains its right to withheld information-for example, tweets that have been removed by Twitter or posted by accounts that have been suspended by Twitter are no longer available. This makes it difficult to reproduce research. Publishing data collected through GitHub's API also raises privacy issues, even if the data published on the platform was routinely archived and made available online.

Releasing the code used to access the data is another way to increase both the reproducibility and transparency of my research method. After cleaning and annotating the code produced for this thesis, the code will be published on GitHub with a brief description.

Work with the GitHub data began in early 2016, although all data analyzed for this article was collected in 2020 and early 2021. To access the above data via the GitHub API and parse the information from the API response, the Python programming language was used. Most of the data processing and analysis was also done in Python, while most of the data visualization was done in R.

4. Findings

4.1. Open-source bot repositories on GitHub

The first steps of my empirical study involved finding Twitter bots on GitHub and collecting metadata about both the repositories (the bot code stored on GitHub) and the bot developers. The following subsection focuses on the open-source Twitter bots published on GitHub and the user data accessed through the platform's APIs.

4.1.1. The exponential growth of Twitter bot repositories on GitHub

GitHub was launched on April 10, 2008, and Twitter bots have been present since the beginning of the site's history. The oldest bot repository found was created just four days after GitHub's official launch date. The number of Twitter bots has grown rapidly. In GitHub's first two years, 2008 and 2009, nearly 100 different bot codes were published. Since then, the number has grown steadily, reaching 1,000 by 2013, and by 2016, the number of bots had quadrupled (about 4000 repositories). By the end of 2020, there were more than 19,000 repositories available on GitHub. Note that deleted bot repositories no longer show up in the search, so the total number of bot codes published on GitHub might be slightly larger.

4.1.2. The location of the bot developer

The analysis in this sub-chapter is largely based on the analysis of bot codes from 2016. The 2016 data was analyzed using Tableau software, and I decided to include this earlier version originally published in a paper (Kollanyi, 2016). However, this sub-chapter has been updated with new data and based on all bot codes available by the end of 2020, and a new, updated version of the geographic distribution of the Twitter bot repositories was added to the thesis.

Some online services automatically collect location information based on the IP address of their users or by geotagging data with location reported by a phone, computer, or tablet. GitHub uses a free-text question in user profiles, so location data is based on self-declared information. This decision has numerous implications for the quality, reliability, and usability of the location information on GitHub. First, users have the option of not providing any location information. In this dataset, there are 7,081 bot repositories that do

contain location information. Second, even if a user provides a location, there is no guarantee that it is truthful. Finally, location information can 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.

Asides from missing, incorrect, false, and outdated information, location data in the user profiles does not follow a strict syntax of street address, city, zip code, and country name. A developer living in New York could add this information in a variety of 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 encoding geographic information, and can be used for free with some limitations to geocode data. Google Places provided accurate geospatial data with latitude and longitude coordinates for almost every valid location. Some of the original location data on GitHub included street-level details, while others specified only the country. Another geocoding API was used to obtain the address in a structured format that allowed the identification of the country name for each repository. The results of the geocoding process are shown in Figure 4 (from 2016) and Figure 5 (updated data from 2020).

Figure 4 *Global Map of Bot Repositories in 2016*

Note. N = 2,615, Original publication: Kollanyi, 2016

According to the location information in the dataset, the largest number of Twitter bots

within the geocoded sample was created by a developer from the United States in both 2016 and 2020. Developers from the United States have dominated the open-source bot landscape since 2008. In fact, exactly 50% of the bots in 2018 came from the United States, with the rest being uploaded by European and Chinese developers. The first Japanese bot was added to GitHub in 2009, and in 2010, Japanese bot writers published more bots than developers from the United State. Bot developers from Japan have been remarkably active since then. However, the share (relative contribution) of Japanese bot developers has declined in the past five years. While Japan had the second largest number of bot repositories on GitHub in 2016, Indian developers has published more bot repositories than Japanese developers by 2020. Similarly, developers from other developing countries such as Brazil, Nigeria and Indonesia have published significantly more Twitter bot repositories.

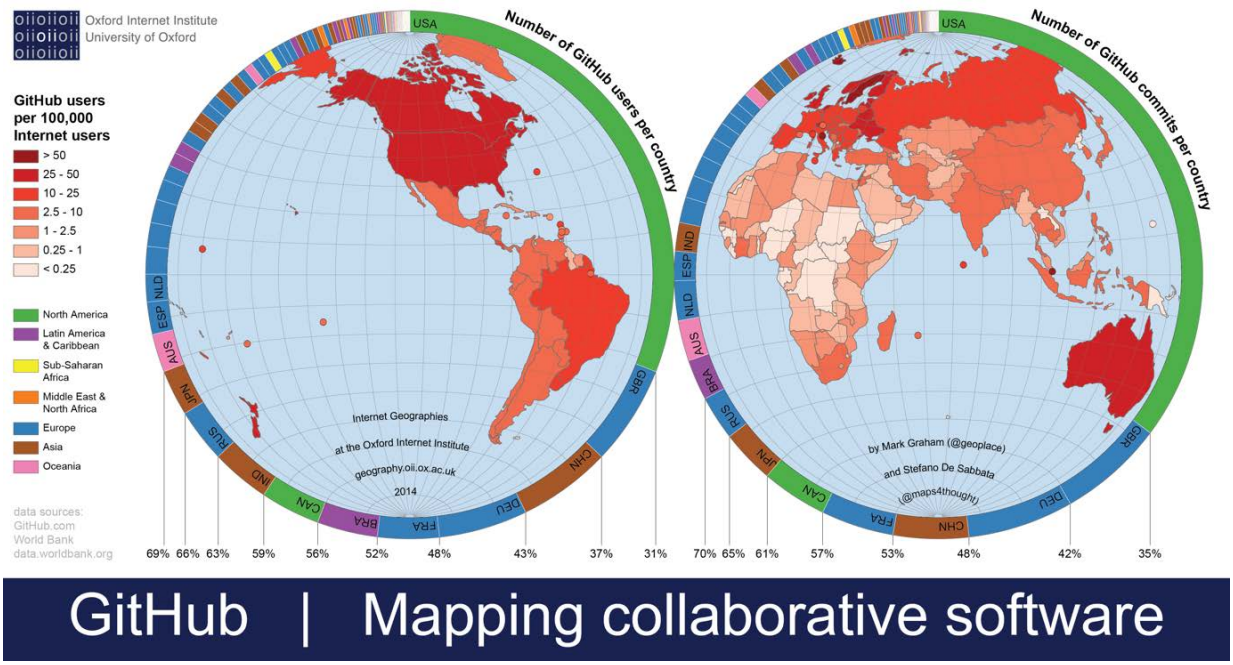
Figure 5 *Global Map of Bot Repositories in 2020*

Note. N = 11,180

Graham and De Sabbata (2015) mapped the number of users and the number of commits per country based on the GitHub Archive dataset with 65 million commits and 1.1 million active users. The authors were able to geocode about 26 percent of the active GitHub users by geocoding the self-declared location information in the profiles of the GitHub users by using the University of Edinburgh's Unlock Places web service. (This is similar to the success rate I had using the Google Places API for my own GitHub data.) Graham and De

Sabbata (ibid.) found a large dominance of North American and European developers in both user base and number of commits (see Figure 6). However, the authors claim that the dominance in code production on GitHub and in other fields of content creation, such as the number of Wikipedia edits, is higher in Europe and the U.S. than in the rest of the world not consistent with other ICT-related statistics. For example, the number of internet-users per country shows a different picture. The number of Internet users from Asia was significantly higher than the number of users from the U.S. and Europe.

Figure 6 *Global Map of GitHub Users and Contributions*



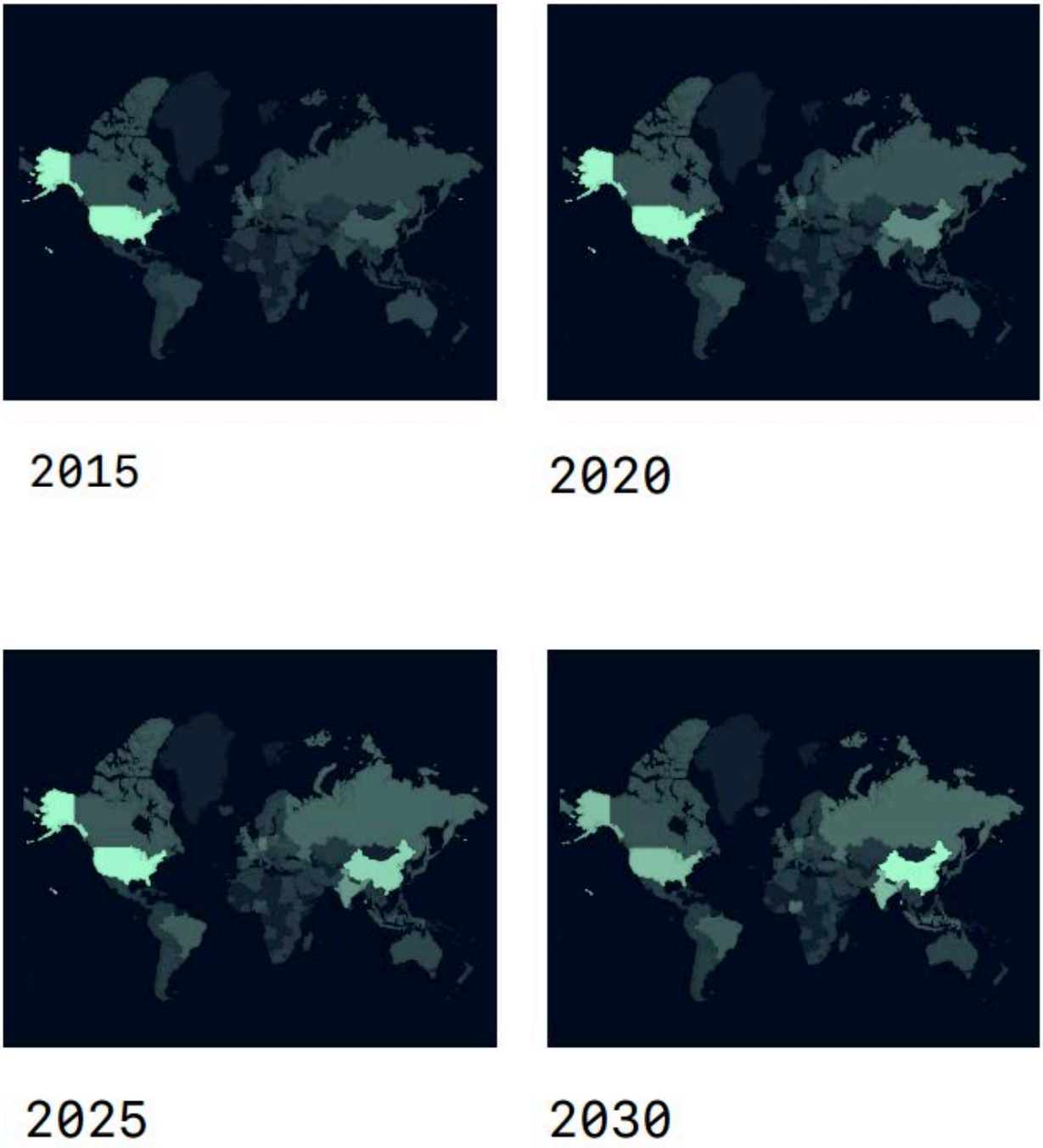
Note. Source: Graham and De Sabbata, 2015

What can we expect in terms of GitHub usage, given that ICT usage figures have changed been dramatically in many developing countries? Unfortunately, GitHub has not released any detailed report (or raw data) on the geographic distribution of its users, but the 2020 GitHub Octoverse report, an internal study by GitHub, suggests that the open-source contribution from U.S.-based developers has decreased but is still high (22.7 percent). At the same time, the contribution from China (9.8 percent) and India (5.2 percent) has increased significantly (GitHub, 2020). It is also worth noting that the company behind GitHub should have better location data than the academic community that studies the platform. Self-described location has its own limitations, including the addition of non-

existent or ambiguous locations, and the fact that in some cases no location information is provided at all. On the other hand, GitHub has access to the IP-address for every user action, including registration, repository creation, commits, etc. Geolocating users by their IP-addresses has its own limitations. For example, users can use proxy servers, but the databases that can be used to geocode IP-addresses are limited, especially in the case of less developed countries (Poese et al., 2015).

GitHub has also published its forecasts based on a model using past data (see figure 7). The company predicts that the contribution of Chinese and Indian developers will increase this decade (between 2010 and 2020).

Figure 7 *Forecast of Global GitHub Usage*



Note. Source: GitHub Octoverse, 2020

When we compare the geographic distribution of Twitter bot codes with global Twitter usage, we have to work with similar limitations. Twitter does not publish official statistics on the number of users per country, but statistics from research projects on worldwide Twitter usage are available (Lipman, 2014). According to Sysomos, approximatly 62 percent of users are from the United States (Sysomos, 2014), while the United Kingdom

and Canada represent 8 percent and 6 percent, respectively. The global map of Twitter bot repositories (Figure 5) largely coincides with these statistics, and the dominance of the United States is also evident on my map. Twitter had reported in 2016 that the number of monthly active users in Japan has reached 35 million (Purnell, 2016). This number can be compared to the 320 million active users Twitter reported globally in the fourth quarter of 2015. The relatively high number of bot repositories for Japan is also in line with the country's importance in the Twitter universe. As for the sudden surge in bot repositories from India, the country was one of the fastest growing markets for the company in 2020 (Hariharan, 2021). This is in line with GitHub's usage forecasts, which suggest that the number of bot repositories in India will continue to rise.

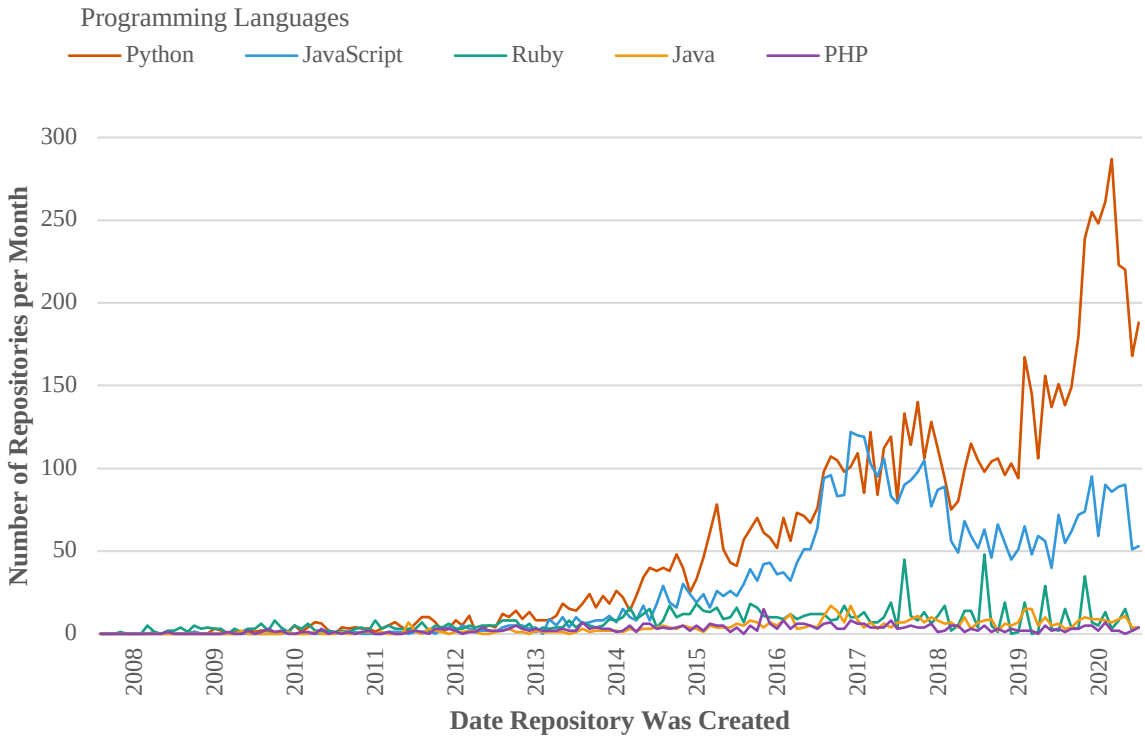
4.1.3. The code behind the Twitter bots

As Dubbin (2013) notes, a Twitter bot can be written in almost any modern programming language. This GitHub corpus supports his observation by showing the diversity of Twitter bots in terms of technical implementation.

The first bot code still available on GitHub was written in Ruby, and the developer worked on the project for two months. This bot used the then-popular instant messaging system Jabber to control a Twitter account. Ruby, the language used for this first bot, was the most popular Twitter bot language in 2008 and 2009, as more than half of the bot code was written in Ruby. Around 2012, Python became popular and by mid-2013, the number of bots written in Python surpassed the number of bots written in Ruby. Today, Python is by far the most popular language for writing bots: 8,085 of the 19,085 available bot codes on GitHub use this language. For an overview of how the programming language has evolved over time, see Figure 8.

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

Figure 8 *Bot Programming Languages Used in GitHub, 2008–2020.*



Python is known as a relatively simple language and is often the first programming language someone 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). Therefore, it is also easier to take a Python code from GitHub and apply it to a new problem. Finally, Python is particularly well-suited for building Twitter bots because it includes a number of packages that support various aspects of bot code, from network access to text analysis and manipulation.

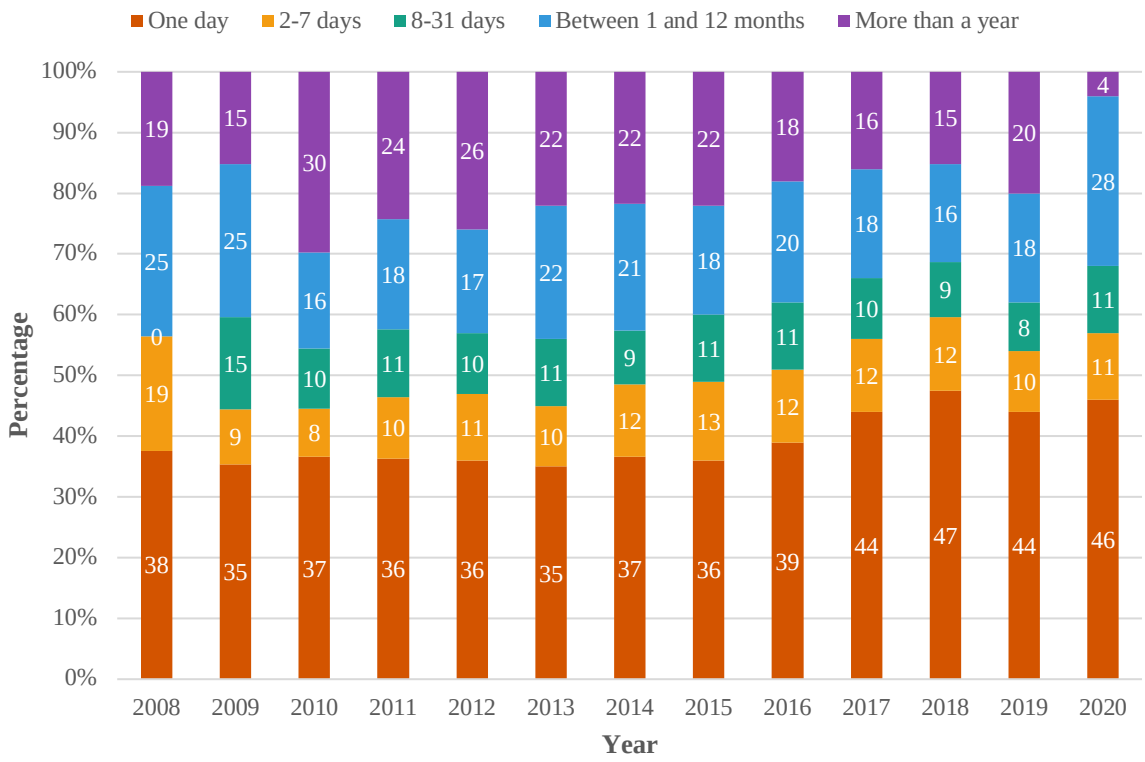
The size of the repositories is a good indicator of the complexity of the code. Interestingly, the average repository size is very similar for all major bot writing languages (between 1 MB and 10 MB). The only exception is R, which tends to have larger repositories on average (around 100 MB). Unfortunately, the size of the repository is not a perfect indicator of the complexity of the code, since it contains all the images and data files. The length of the code in lines of code would be a much better indicator of the complexity of the bots, but this information is not readily available.

4.1.4. 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 releasing programs they do not want to work on - rather than actively working on code and committing changes.

The lifetime of repositories on GitHub can be measured as the difference between the creation date and the last version of the repository plus one day (see Figure 9). The lifetime of a Twitter bot repository can include periods of time when code is actively being worked on, as well as longer gaps between maintenance and code updates. In most cases, however, a longer lifetime indicates that the project has been actively used and supported by one or more developers over a longer period of time.

Figure 9 *The Life Cycle of Twitter Bot Repositories on GitHub Between 2008 and 2020.*



It is important to note that one day does not mean that the user only has one version of the code or that it has only been committed once. Minor formatting after the initial commit, uploading multiple files from the same repository in multiple commits, and changing a repository's readme file are different than working on bugs or adding new features to a code

weeks after its initial creation. Again, one could argue that developers who make changes shortly after their initial upload are only using GitHub for code dumping. Finally, it is important to note that many of the bot codes are so simple that an experienced developer can easily create these codes in a single programming session.

Taking these considerations into account, it can be said that a large number of bot repositories on GitHub are either code dumps or one-off small projects, as these repositories are not developed or maintained over a long period of time. For example, according to the data, 43 percent of bot repositories were not updated after the first 24 hours. Another 11 percent of bot repositories have a life cycle of no more than a week.

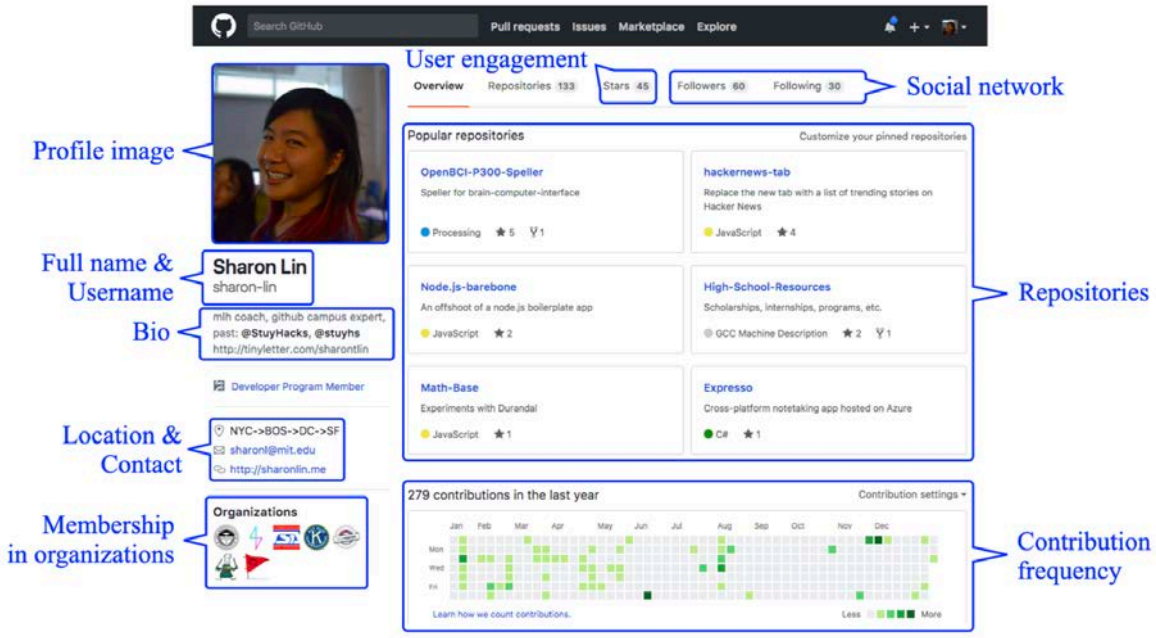
The average time between the creation of a repository and the last commit is 160 days.[7] Longer life cycles usually result from intense tinkering activity rather than sustained interest throughout the period. For example, one of the most popular Twitter bots has five commits from day one, but the next commit occurred almost 10 months later. This commit was in response to Twitter changing its process for handling API requests. Another popular repository was not updated for a period of 17 months. After handling some minor bugs, the repository owner left the code unchanged for an extended period of time and came back after 17 months to merge change suggestions from other users.

4.1.5. The developer behind the bot code

What can we learn about developers uploading bot codes to GitHub? GitHub provides access to various information about users' professional activities both inside and outside the site. However, the information about off-site activities, such as a user's workplace or a personal blog address, is rather limited and is only collected in a free-text format. The data about the outside world is only governed by the social norms and practices of the GitHub community. Therefore, the professional activities of users outside the GitHub universe are difficult to quantify. In contrast, GitHub metrics are easy to read because they are directly linked to the site's architecture. Still, site activity is important to understand who the developers behind the bots are. In the following discussion, individual users rather than bot repositories are the unit of analysis, as some users may have multiple bot repositories listed.

[7] The lifetime of the repository depends on the time of the data collection. The latest data collection happened about 150-160 days after the end of 2020 calendar year. This approach allowed me to calculate a more realistic lifetime for the bots that were added to GitHub at the last quarter of 2020, but it has slightly changed the average lifetime compared to an earlier (or later) data collection.

Figure 10 *Annotated Screenshot of a Typical GitHub Profile*



GitHub distinguishes between two types of accounts: regular users and organizations. An example of Twitter bots published by an organization is a San Francisco school whose students published 37 bot codes to the school's account, all written in Python. While browsing these repositories, it also became clear that the teacher looked at each bot and provided feedback in the form of comments inserted directly into the code.

The typical bot author is an experienced programmer and a long-time GitHub user who has multiple repositories on GitHub. The users in this dataset had an average of 48 repositories at the time of data collection. Yet, there are 655 bot repositories where the repository owner has no other repositories on GitHub. This data suggests that a Twitter bot is just a side project or fun experiment for many developers. Less than 4 percent of Twitter bot authors (588 developers) have additional bot repositories.

Gist is a hosting service for code snippets with version control through GitHub. It is useful when a developer wants to share some of their code in a blog post or forum outside of GitHub. As of 2016, there are more than 17.7 million gists on GitHub. However, the extensive use of gists is not typical of the community of bot writers. The average bot author has 5.9 public gists on GitHub, 10,385 bot authors have no public gists, and another 1,638 bot authors have only one public Gist. In other words, bot code is generally not as highly publicized as other forms of code.

Most bot writers are well integrated into the GitHub community by following or being

followed by other users. The average GitHub bot writer has 30 followers and follows 21 developers on the platform. Yet about one in four GitHub bot writers - 4,318 users out of 19,085 developers - had no followers at all at the time of data collection. Code developed by developers without social connections can still be found by users searching for Twitter bots. GitHub's search interface tends to give more weight to socially active users, so these repositories are listed at the bottom of the results page. Overall, it's safe to assume that users without followers have less influence on the community. The top developers represent the other end of the influence spectrum. There are 61 users in the dataset with more than 1,000 followers each. These popular users are referred to as “programming rock stars” in the literature (Lima et al., 2014).

4.1.6. More about the social aspect of developing Twitter bots

GitHub's marketing and 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. In contrast, the overwhelming majority of open-source Twitter bot projects on GitHub were developed by a single author. As Table 3 shows, nearly 90 percent of the Twitter bots shared on GitHub were 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 (copies of the original repository) for projects with one author is 0.3, while a project with multiple authors has more than three forks on average. The presence of forks is an indicator of both broader interest in a project and the ability to receive contributions from the broader open-source community around GitHub. About 87 percent of single authors did not attract this type of interest because these projects do not have forks. In contrast, about 50 percent of bot projects with two or more authors have at least one fork.

Table 3 *The Social Aspects of Bot Repositories on GitHub.*

	N	Percentage	Average number of stars	Average number of forks	Average lifetime (days)
Single author	16,428	89.2	1.3	0.4	133.9
Two or more authors	1,992	10.8	9.0	3.0	427.5
Total	18,420	100	2.2	0.7	165.6

The average number of stars, an indicator of a repository's popularity, is also rather modest for projects with only one author. A bot project with two or more developers has an average of 11 stars, while the average single-author project has received only one star. This does not mean that single-author projects are not popular or cannot receive support from the open-source community. The most recognized bot code developed by a single author has received an impressive 5,962 stars, while the most forked project with only one author has had 849 forks.

Finally, projects with two or more developers are maintained or updated longer on average. While the average project with a single author is “closed” in less than five months, projects with multiple authors are maintained for more than 400 days on average.

In summary, only 1 in 10 Twitter bot repositories shared on GitHub took advantage of the collaborative or social aspects of the platform. About 2,000 projects were co-developed by two or more programmers. These projects received more recognition from other GitHub users and generated more forking repositories. Collaborative development and forking could lead to better quality code because more than one programmer contributes.

4.1.7. Regulation and norms on the platform

While Twitter has a policy against abusive bots and has developed sophisticated algorithms to detect these bots on its platform, GitHub does not review the content of its repositories and does not remove spam bots, for example. GitHub's terms of service restrict the uploading of materials that infringe copyrights and the use of the service for illegal

activities or activities that violate a country's laws. However, in most countries, Twitter bots are not regulated, so users can upload all kinds of bot code.

GitHub does not censor code shared on the site, but GitHub has been censored several times due to content shared on the site. In the past, GitHub was blocked due to the publication of partially ironic suicide instructions in Russian (Lunden, 2014), anti-Indian 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 that have been reported in the media, there is no information about any actions against GitHub or requests to developers using the site to remove bot codes.

At the time of data collection, GitHub hosted a number of self-declared spam bot repositories on its site. GitHub provides some level of anonymity to its users because the site does not enforce a “real name” policy, but it is reasonable to assume that the majority of developers do not want to be associated with malicious bot codes. As a result, 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. For example, many developers require that other users use the bot only for good and not for evil. Some project descriptions include warnings regarding the bot. Many authors specifically ask that other users not use the script for spamming or harassment.

4.1.8. Most important research findings

In the first section of the results chapter, I presented the results of an analysis focusing on the bot codes available on GitHub and the authors who created the repositories. The bulk of this section addresses the first research question (RQ 1.1) and discusses how the bot was developed. Bot developers are often experienced programmers and have published a large number of other repositories on GitHub. For many developers, writing a Twitter bot seems like a one-day project. In fact, more than 40 percent of bot repositories were not updated after the first 24 hours. However, some of these repositories were developed without using GitHub, and the code-sharing platform was only used as a code dump - developers simply uploaded their finished and polished bot code without using the system's version control. Although most bot developers developed only one bot for Twitter, about 4% of bot authors (588 developers) had at least two Twitter bot repositories at the time of data collection. Often developers develop bots for multiple platforms, but that is not part of this analysis.

The section also addressed RQ 1.3 and analyzed the contribution to each bot repository to determine if the bot code was developed by one author or if multiple developers contributed. The results indicate that 9 out of 10 bot codes were developed by one author. Bot repositories developed by multiple authors might be more complex, and developers tend to develop the code themselves over a longer period of time. In addition, these repositories are maintained longer. One possible explanation for this is the higher attention these repositories receive - while repositories developed by one author receive only one star (a form of user appreciation), bot codes developed by multiple authors receive an average of 11 stars. Similarly, these repositories are on average more often forked (copied in GitHub's terminology) as a separate repository (3 forks vs. 0.4 forks).

4.2. The results of the bot developer survey

To learn more about developers publishing open-source bot codes on GitHub, I conducted a survey and asked developers on GitHub to answer a series of questions about how they use GitHub and about their projects. To identify developers on GitHub who have published code to run Twitter bots and to find their contact information, I turned to a specific GitHub API. As I described in more detail in Chapter 1, GitHub's Search API allows you to get a list of repositories that contain specific search terms either in their name, description, or in the readme file attached to the repository. After creating a database of all repositories that matched a set of keywords related to Twitter bots, I used a separate GitHub API to get more information about the developers behind the repositories. These API calls revealed the contact information (email address) for a large number of developers. The following section explains the structure of the survey and the main themes within it.

4.2.1. The structure of the survey

The survey itself consists of 41 questions in 4 larger blocks. The first block focuses on GitHub usage, the second block covers topics around bot development in general and includes specific questions about how the respondent's own bot was developed and deployed on Twitter, while the next block includes questions about Twitter bots and politics. The questionnaire also includes a demographic section with some questions about education in IT or programming. The full questionnaire can be found in Appendix A, at the end of the thesis.

In order to get a richer dataset about the developers who completed the survey, I decided to add to the survey some additional metadata about the users that was previously collected. This includes metadata about the user, such as the number of public repositories uploaded by the user, the time of registration, the number of programming languages used by the user in public repositories, etc. In addition to this user-specific information, I was also able to enrich the survey results with information about the specific bot project, e.g., social engagement with the repository (stars, watching), programming language used for the code, when the repository was created and when the repository was last updated, etc.

In my questionnaire, I asked GitHub users to provide a list of Twitter handles for the bot(s) they developed and deployed. In addition to this source of information, many bot repositories included a link to a Twitter profile if the bot was deployed on Twitter. After compiling a list of open-source bot accounts available on Twitter, I collected additional

data from Twitter itself about these bots to provide an even richer dataset about bot developers and their projects. This data includes the time of registration, followers, number of posts by the bot, and much other information.

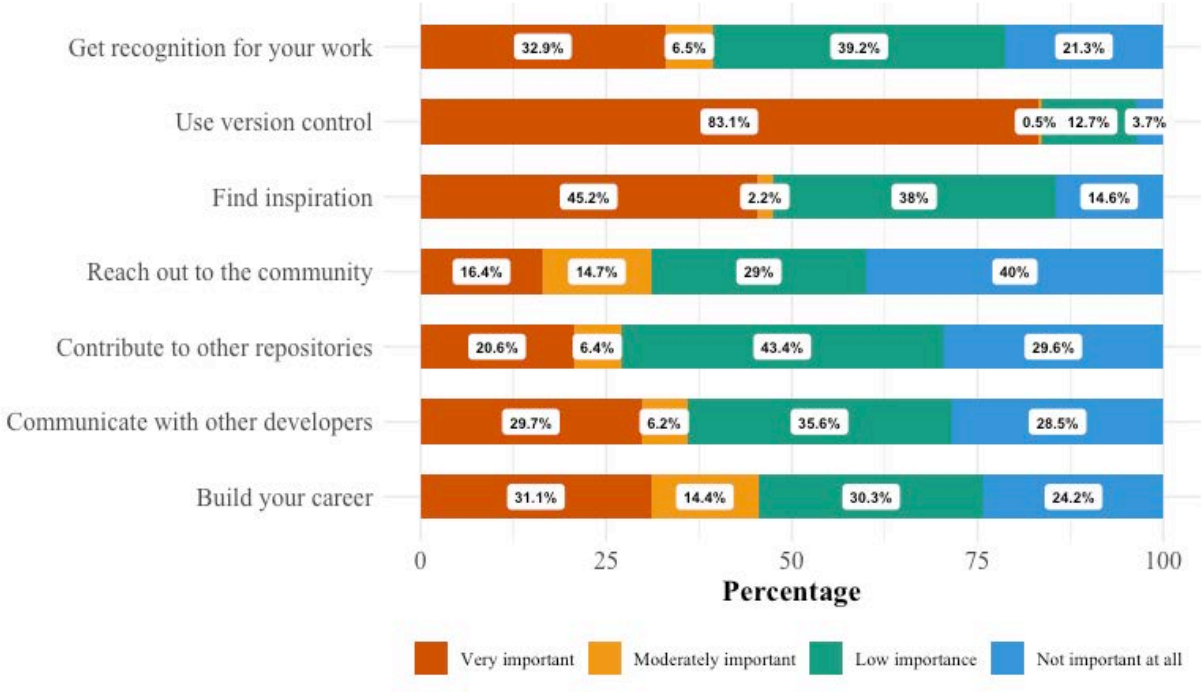
4.2.2. The social aspect of developing code over GitHub

GitHub is a professional web service that allows developers to use version control, host code online, and manage projects from development to release. GitHub also offers some social features, but these are limited and mostly focused on three main topics: 1) discussing code and software issues (e.g., requesting features, reporting bugs/problems); 2) getting inspired by others, expressing support or respect, and engaging with other developers (following other users, star repositories, etc.); and 3) collaborating on software projects (e.g., managing teams).

The above social features of the site are limited. Therefore, developers often use other platforms to communicate with each other. For example, a developer may use GitHub for code sharing and version control, but rely on Slack for coordination or turn to LinkedIn to advance their career and Twitter to gain broader recognition. Wu et al. therefore suggest that GitHub is part of a larger ecosystem for developers (Wu et al., 2014).

This is also reflected in the results of my survey of bot developers. The most important feature of GitHub for bot developers is the ability to control versions-more than 83 percent of respondents indicated that version control is a very important reason for using GitHub.

Figure 11 *Reason for Using GitHub*



Note. N=811

Of the three social aspects (or use) of GitHub mentioned above, being inspired and finding interesting projects on GitHub is the most important to bot developers - about one in two respondents find this feature important when using GitHub. Collaborative software development is significantly less important for bot developers. Even when combining the Very important and Moderately important response categories, only about one-third of developers find it important to give back to the community and contribute to the work of other developers. Similarly, only one-third of developers find it important to reach out to other developers and ask them to contribute to a project they manage or host. This limited social use is consistent with the results of the analysis of the GitHub repositories themselves - bot codes are often developed by a single author rather than in collaboration with others.

Although the platform offers limited communication options, e.g., users cannot message each other directly, communication with other developers was still an important feature of the platform - about 35 percent of bot developers find this social aspect very important, and another 29 percent find it moderately important.

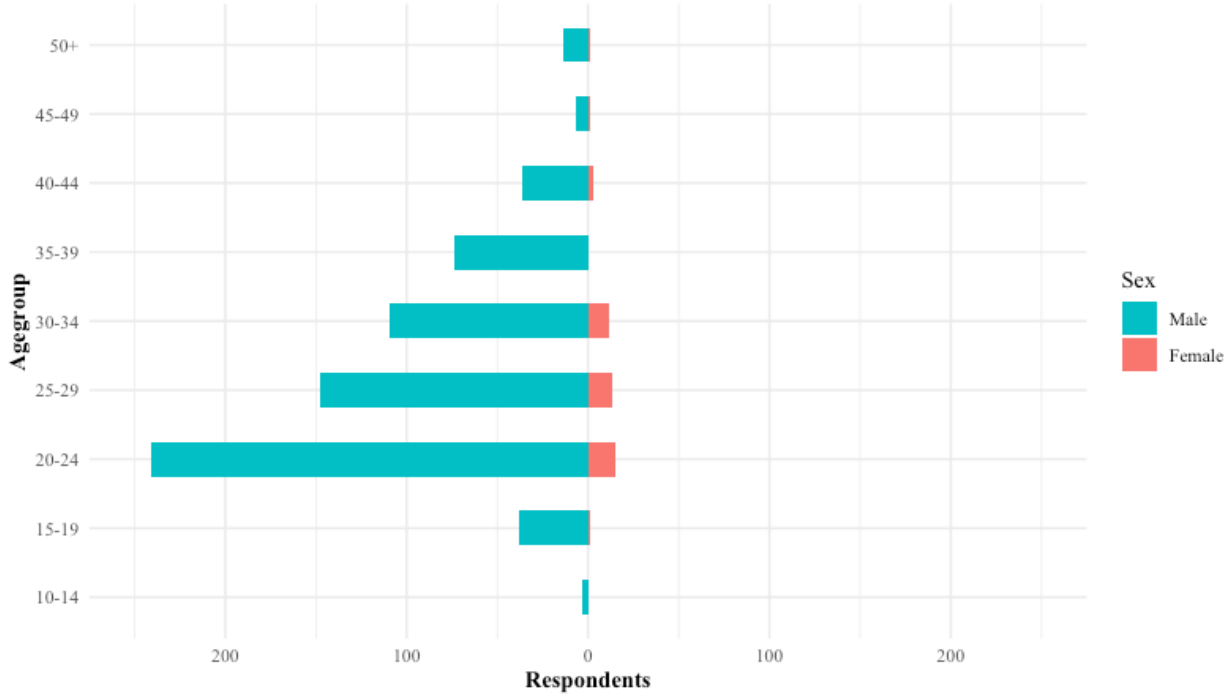
There are some gender differences in GitHub usage, at least among bot developers. Of the seven possible reasons for using GitHub, four showed significant differences by

gender. These characteristics are all related to social use of the platform - either to contribute, such as finding programmers to contribute to someone's work or writing code for other repos, or to contribute, such as giving feedback to other programmers, answering questions, etc.

Note: I decided to simplify the results and compare only male and female respondents because only 2.8 percent of respondents chose the other gender identity categories. Another 5.9 percent did not answer the gender identity question. These respondents were filtered out along with the other category to compare male and female respondents. After these groups were filtered out, the number of valid responses dropped to 740 (694 male respondents and 47 female respondents).

The following graph shows the age and gender distribution of respondents in the form of a population pyramid.

Figure 12 *Bot Developer Population Pyramid*



Note. The age and the gender of the respondents. N=811

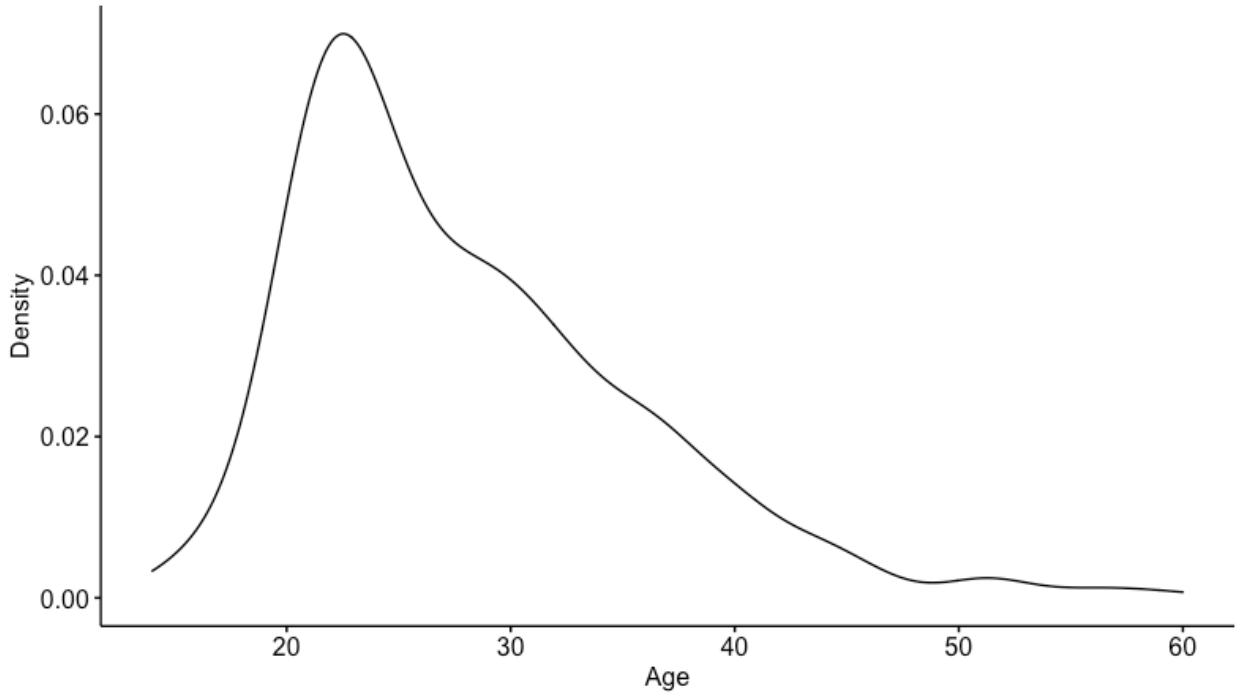
To put the relatively high percentage of male respondents in context: GitHub and software development in general has a strong gender-bias (Geiger, 2017). The study mentioned above focused on open-source development and GitHub and concluded that 91 percent of respondents were male and only 3.4 percent identified their gender as female, while another

1.1 percent chose the “non-binary or other” category and 4.7 percent preferred not to specify. (My study found that 85.6 percent of bot developers are male and 5.8 percent are female).

More than 75 percent of respondents have formal training in programming or computer science (77.9 percent). Programming background only had a significant impact on career-related use of GitHub and none of the other reasons for using the platform. While 62 percent of developers with a programming background (formal education in IT or programming) selected the Very important or Moderately important response, only 54 percent of developers without a programming background selected similar responses. This difference was significant according to the chi-square test.

The importance of some of GitHub's features showed a significant relationship with bot developer age groups. Based on a chi-square test to detect independence, career building and contacting other developers on the platform have a p-value of less than 0.05. However, some of the counts per cell in the corresponding contingency tables are too low, even after using larger age group categories. Therefore, I decided to find another method to examine the relationship between GitHub usage (more specifically, the importance of various platform features) and age.

Figure 13 *Density Plot of Age*



Note. N=811.

Anova, T-tests, certain types of regression models, and many other statistical tests all rely on the assumption that the continuous data used for the test follows a normal distribution. These methods are called parametric tests. A quick inspection of the normal distribution of the age variable used in the GitHub bot developer dataset shows that age does not follow a normal distribution. Both visual inspection of the distribution of the age variable in Chart 3 and a quick Shapiro-Wilk normality test confirmed that the data does not follow a normal distribution. The results of the Shapiro-Wilk normality test ($W = 0.92514$, p-value $< .001$) indicate that the data deviate significantly from the normal distribution. Again, an ANOVA or t-test cannot be used without assuming normality.

For nonparametric categorical independent variables and data at least ordinal level, or as in this case for continuous dependent variables, a Kruskal-Wallis test can be used to detect differences between population means (Kruskal & Wallis, 1952). This test can be used to compare the means of a variable between subgroups formed along one or more categorical independent variables. It is the best nonparametric alternative to a one-way ANOVA test. The results of the Kruskal-Wallis test can be seen in Table 4.

Table 4 *The Result of the Kruskal-Wallis Test*

Nr	Kruskal-Wallis rank sum test	Mean	chi-squared	p-value
1	Recognition	3.0	36.1	**0.00**
2	Version control	3.8	1.9	0.59
3	Inspiration	3.3	2.7	0.44
4	Reach out	2.5	21.5	**0.00**
5	Contribution	2.8	8.8	**0.03**
6	Communicate	2.9	15.9	**0.00**
8	Building career	2.8	63.0	**0.00**

Note. N=766.

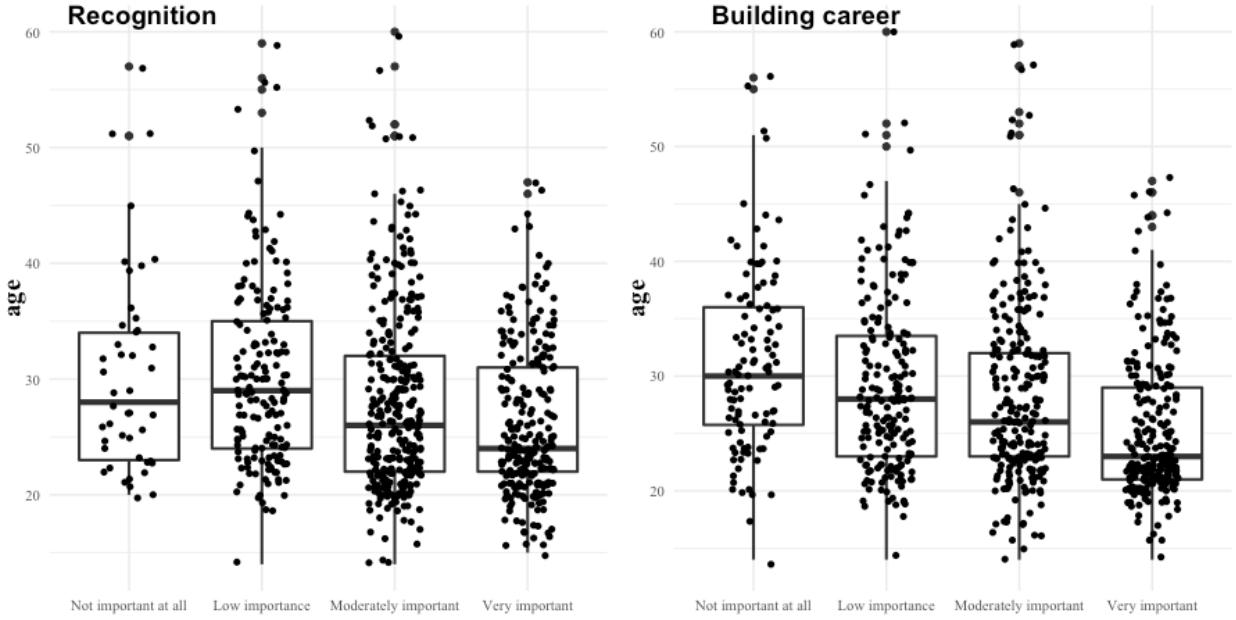
The table shows that the rating of the importance of 5 features of GitHub has a significant difference in the mean age. The null hypothesis of the Kruskal-Wallis test is that the groups have the same mean. Based on the p-values ($p < 0.05$), we can reject the null hypothesis for five of the seven variables. However, this test does not tell us if there are significant differences between the groups of developers who chose certain response categories.

To compare each response category, I decided to use the pairwise Wilcoxon test. This test is basically the nonparametric version of the t-test. (The t-test is based on an assumption of normality, as described above.) The Wilcoxon tests showed that for all five important questions, the mean age for the response category 'Very important' was significantly different from all the other groups based on the other responses - 'Moderately important' 'Low importance' and 'Not important at all' In the case of recognition and communication with other developers, the 'Moderately Important' group showed a significantly different mean than the 'Low importance' group. In the case of making contacts with other developers and making a career, the 'Moderately important' group has a significantly different mean than the 'Not important at all' group.

Although I found significant differences between the age of the respondents and the importance of the various features of the platform, a look at the actual means and standard deviations of the groups formed around the specific response categories suggests that there are only minor differences between the groups. These minor differences can be easily seen

in the visual representation of the groups shown below.

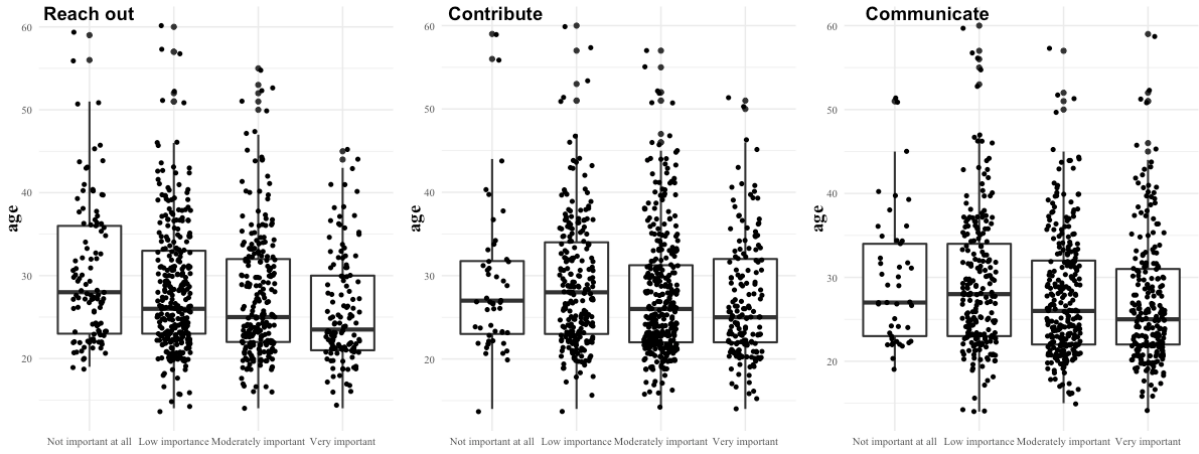
Figure 14 *Using GitHub for Career Purposes*



Note. Mean, standard deviation and age-distribution, N=766.

Bot developers on GitHub tend to be young: 68.7 percent of respondents who provided an age are younger than 30. The average age of developers is 27 years, with a standard deviation of 7.6 years. Among the respondents, those who are under 25 seem to be more focused on starting their careers - they find it more important to gain recognition by publishing code on GitHub and to find a job through the platform than the older developers.

Figure 15 Using GitHub for Contributing to the Open-source Community



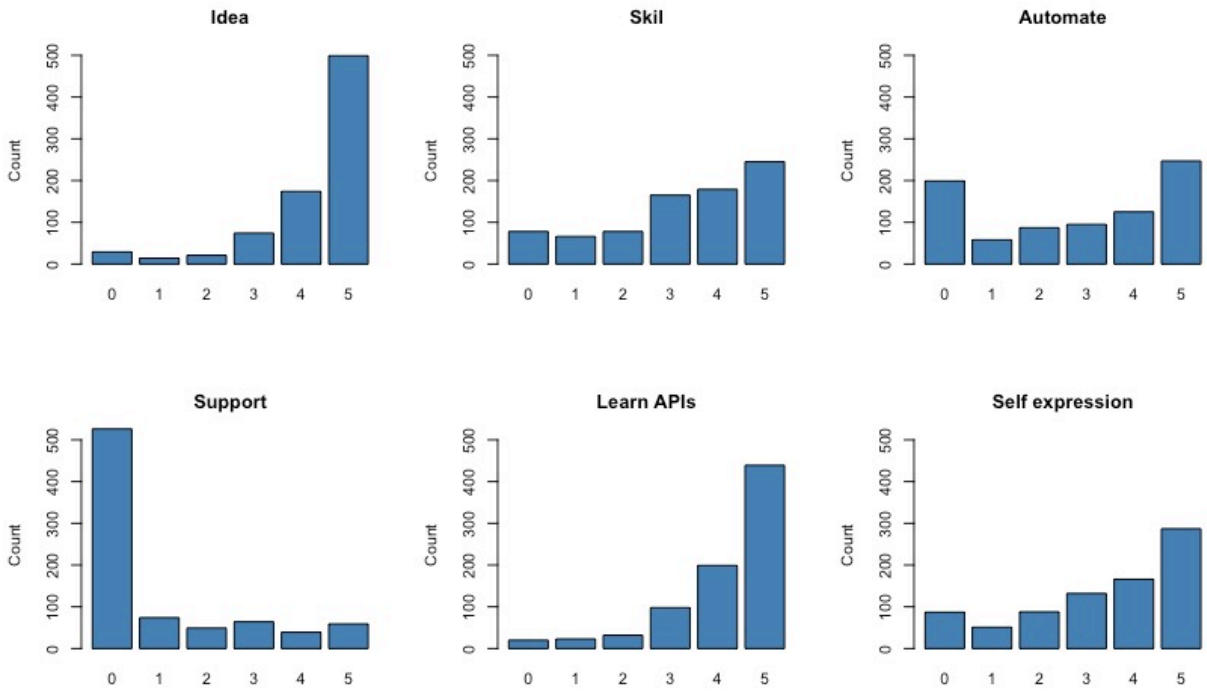
Note: Mean, standard deviation and age-distribution, N=766.

The social aspects of using GitHub - contributing to others' code, finding developers to help write code, and communicating with other developers - are more important to younger developers. Respondents who selected “Very important” for these questions are, on average, slightly younger than all other groups made up of other response categories, and these differences are all significant.

4.2.3. Reasons for developing a bot

Developing bots is often about having a quick, simple idea and seeing if that idea works in practice. When I asked developers about their top reasons for developing a bot, testing an idea received the highest average score. More than nine out of ten bot developers said that this was indeed a very important reason for starting to develop a bot. On a scale of zero to five, where five means the reason is very important and zero means the reason is not important at all, 61.5 percent chose the highest score, 5, and another 21.5 percent chose 4.

Figure 16 *The Reason for Developing a Bot Rated on a 0 to 5 Scale*



Note. 0: Not important at all; 5: Very important, N=811.

The second most important reason for developing a bot is to practice writing code and learn more about Twitter's APIs. This suggests that writing bots is often about learning. In fact, a qualitative analysis of bot repositories suggests that bot codes are often the result of a school project. There are many bots that were developed for a class or put together as part of a hackathon.

Interestingly, the aspect of showing off one's skills as a developer was rated significantly lower, suggesting that bot development is often not considered a highly technical task.

As for the purpose of bot development, in addition to testing an idea, I also asked about the importance of automating some tedious and boring tasks. For one in three developers (30.2 percent), automation is a primary reason for developing a bot. On the other hand, every fourth developer (24.5 percent) thinks that automation is not important at all.

Finally, we can note that supporting a cause or a political agenda is not the main reason for developing a bot. In fact, 64.9 percent of respondents indicated that a political

agenda or cause was not important at all when they developed their bots. Only 59 respondents selected the highest value for political support as a reason for developing a bot. This suggests that the vast majority of bots are non-political bots. This was confirmed by analyzing the bot codes and reading all descriptions and readme files downloaded from the GitHub repositories. (For more details, see the chapter on open-source bots).

The difference between male and female respondents in terms of reasons for developing a bot is negligible. However, the number of non-male respondents is very small, and I often got too few cases in cells when comparing the different response categories for these six questions.

Although a Kruskal test shows that age is significantly related to four of the six reasons for developing a bot, these differences are only statistically significant and cannot be meaningfully interpreted. After forming age groups, we can see that the age of the respondents has no significant relationship with the importance of the six reasons for developing a bot. In this case, there was no reason to pair the different response categories.

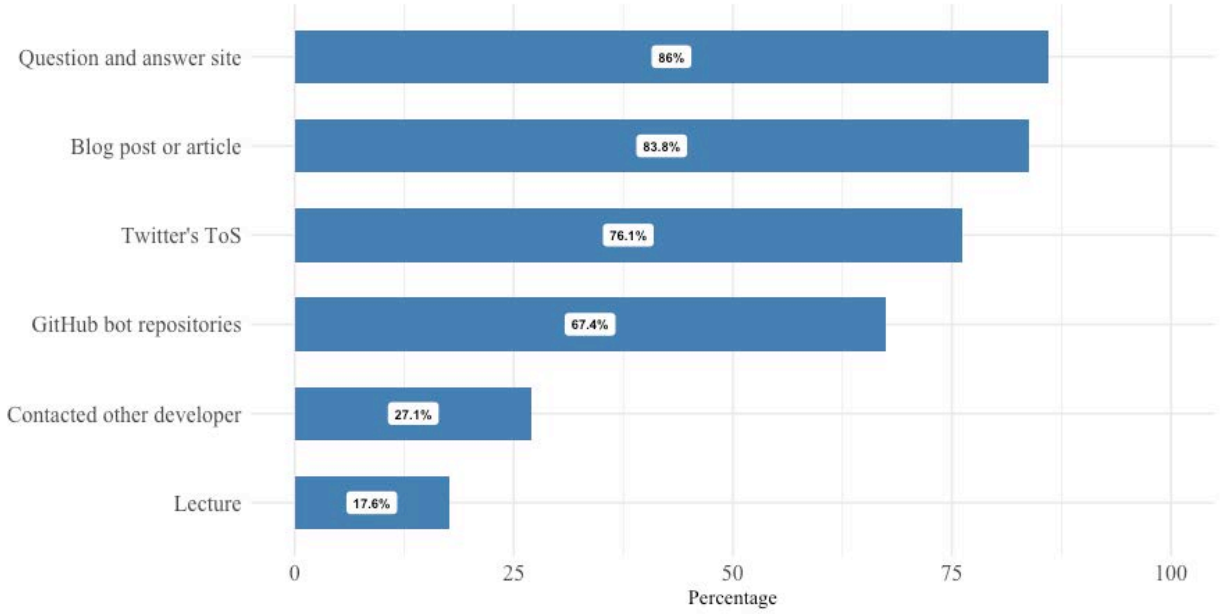
Programming background (formal education in computer science or programming) is not significantly related to the importance of the different options for developing a Twitter bot.

4.2.4. Skills for developing Twitter bots

One of the central questions of my dissertation is to understand where the skills for developing a bot come from and what kind of skills are needed to develop bots and deploy them on Twitter. This question is important, because it tells us a lot about the democratization of this technology. If the knowledge about automating accounts is easily accessible and available to everyone, even someone without much technical knowledge can develop and deploy a bot on Twitter.

To learn more about the issues developers faced either in developing their bot or deploying it on Twitter, the questionnaire included two relevant questions. The first question lists online sources of information that may be useful for developing a bot - it includes options such as studying Twitter's published guidelines for developers, using one of its popular question and answer sites, or contacting other developers who have more experience developing Twitter bots. The other question is an open-ended question where I asked respondents to describe the most difficult aspects of bot development. These two questions are complemented by manual analysis of available bot code on GitHub.

Figure 17 *Sources of Information Used for Developing Bots*



Note. N=811.

The top resources for bot developers are Q&A sites, such as Stack Overflow and Quora, as well as blog posts or articles about developing a Twitter bot. More than 86 percent of respondents relied in part on question and answer sites. The keyword “twitterbot” for example, leads to 461 results on Stack Overflow, covering topics from working with Twitter's APIs to hosting a bot code on a free hosting platform like Heroku. The same platform returns 478 results for the search phrase “twitter bot”, although there could be some overlap between these results. Similarly, people ask questions like “How can I make a twitterbot using Python?” and “How do I create a Twitter Bot that retweets and likes tweets?” on Quora, and the site's users usually give good advice on how to create a bot. Blog posts also often provide step-by-step instructions for developing a bot (e.g., Spence, 2017).

Female respondents tend to rely at least partially on blog posts and articles as a source of information for bot development. There is a significant difference between male and female respondents who partially rely on a blog post or article. As mentioned earlier, the sample of females in our database is rather small, but the cell counts in the contingency table are large enough to conclude that female respondents rely heavily on blog posts and articles. In fact, 46 of the 47 female developers cited blog posts and articles as one of the

sources of information that helped them develop a Twitter bot.

The average developer relies on more than 3 of the 6 sources of information offered in the survey. Female respondents use a greater number of sources on average (3.79 for women vs. 3.30 for men).

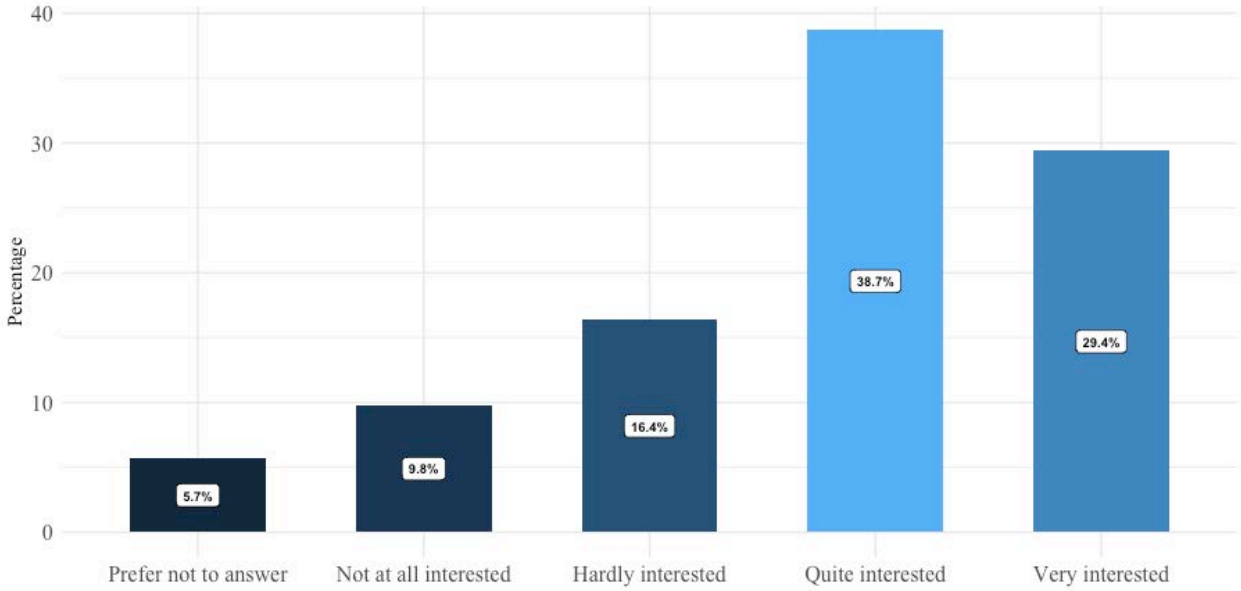
Interestingly, the source of information has almost no significant relationship with age. The appropriate analytical approach for a binary variable and a continuous variable that does not meet the normality assumption is to use the Mann-Whitney U test (Corder & Foreman, 2009). This nonparametric test allows us to compare two groups and test the null hypothesis that the two groups come from identical populations. We can reject this null hypothesis only in the case of using question and answer sites. The average age of developers who used a question and answer site is 27.6 years, while the non-user group has an average age of 30.8 years. One can only speculate about the reason for this age difference. Either the projects of the older respondents are simpler or the older users are more experienced, so they do not need to respond to this popular source of information. Another possibility is that older developers have different heuristics for finding a solution to a programming problem than younger respondents.

Although the survey did not give me an opportunity to test the skill level of the respondents, the question about formal education in computer science or programming could serve as an indicator of skill level. Unfortunately, there is no significant relationship between programming background and use of a question and answer site.

4.2.5. Twitter bot and politics

Bot developers who publish their code on GitHub are generally interested in politics-29.2 percent of developers are very interested and another 38.9 percent are fairly interested in politics, the two highest scores on a 4-point scale. Less than 10 percent indicated that he or she is (or are) not at all interested in politics. Another 5.7 percent chose the option "Prefer not to answer this question."

Figure 18 *How Interested are the Developers in Politics*

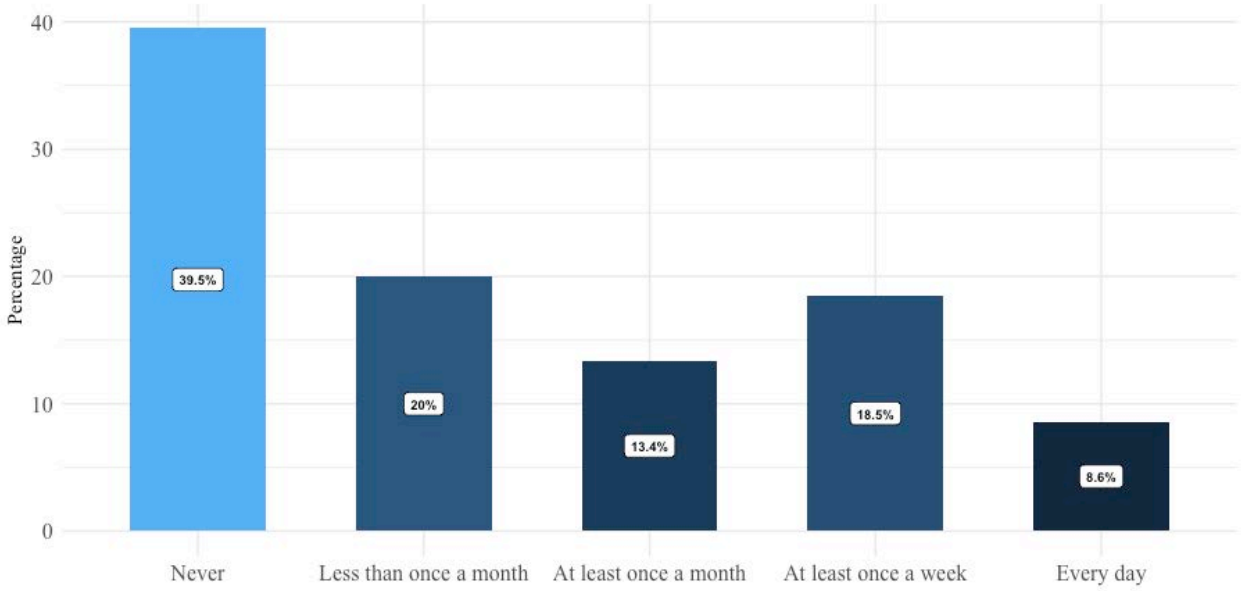


Note. N=765

Younger developers tend to be less interested in politics than older developers. The Kruskal-Wallis test for age and political interest is significant (Kruskal-Wallis chi-squared = 34.383, df = 3, p-value < .001). The pairwise comparisons based on a Wilcoxon rank sum test indicate that the age composition of the groups selecting the “Very interested” and “Quite interested” response categories are significantly different from all other groups. The above two groups also differ from each other, as shown by the result of the test presented below.

There is no significant difference between female and male respondents in terms of their interest in politics. There is also no significant relationship between the level of education and political interest. We know from the literature that people with higher levels of education tend to be more interested in politics (Emler and Frazer, 2010). However, due to the relatively young age of the bot developers on GitHub, we can assume that many of the respondents are still students. For example, the Octoverse GitHub survey suggests that at least 18.7 percent of programmers who regularly contribute to projects on GitHub and some other major open-source projects are still full-time students (Geiger, 2017).

Figure 19 *Frequency of Using Social Media to Discuss Politics*



The bot developers are not only interested in politics, but 40.2 percent of them use social media to discuss politics and public issues at least once a week. This type of political social media use includes writing or responding to a political Twitter post and writing or commenting on a political Facebook post.

Similar to interest in politics, younger respondents tend to be less active in discussing politics online. While the average age among developers who discuss politics online daily is 33.3 years and among those who do so at least once a week is 29.3 years, respondents who discuss politics on social media less frequently (or not at all) are on average only 27 years old. The Kruskall-Wallis test found significant differences between the average age

of the different groups formed around the frequency of online political activity. The pairwise Wilcoxon tests showed that “At least once a week” and “Every day” were the two response categories that divided developers into groups that were significantly different (based on age) from the groups formed around any other response category.

I asked respondents to tell me if their bot had ever been used for political purposes and whether it would bother them if it had. While the first question helps identify political bots, the second question provides important information about bot developers' attitudes toward political use of automated social media accounts.

About 11.8 percent of developers reported that one or more of the bots they developed were used for political purposes. Not surprisingly, these developers are significantly more active in politics than developers who have only non-political bots (X-squared = 16.031, df = 3, p-value < 0.001). Two out of ten developers who are themselves very interested in politics reported that their bot had been used by someone for political purposes (18.6 percent). In the group of developers only quite interested in politics, one in ten reported the same (10.8 percent), while the rest of the developers less interested in politics reported a lower percentage of use (6.3 - 6.8 percent).

While 39.1 percent would mind if the bot they created was used for political purposes, the majority of developers (60.9 percent) would have no problem if their bot codes were used politically in at least some form. Still, most of these developers indicated that whether or not they would object to their bot being used politically would depend on the political issue.

4.2.6. Most important research findings

The bot developer survey primarily addressed the first two research questions - RQ 1.1 focuses on the practices of using GitHub and the reasons for using GitHub for development, while RQ 1.2 focuses on the skills required for bot development and how GitHub users learn how to write bot code. When I asked bot developers why they use GitHub, the overwhelming majority of respondents rated the site's main feature, version control, as a very important reason for using the platform. In addition to tracking changes to their code, developers use GitHub to advance their careers, gain recognition and find inspiration. Interestingly, contributing to the work of other developers or finding other developers who can complement their work was much less important to developers. Younger developers find both career advancement and social opportunities on the platform more important.

Although the majority of respondents built the bot to realize an idea, building a bot also seems to be a way for many developers to learn how to work programmatically with social media (learning APIs). Self-expression was much less important, according to the survey results. (These results are related to RQ 1.1 and RQ 1.2.) I specifically asked about how developers deal with problems they face during bot development, and it is partially consistent with other findings in my thesis that developers tend to learn bot creation (or how to solve specific problems) by visiting Q&A pages and reading blog posts rather than turning to other developers within the platform. Most social media platforms have documentation for developers, and in the case of Twitter bot development, this source of information was important to more than 76 percent of developers. So developers needed to learn not only about APIs in general, but also about platform-specific rules and ways to deal with the limitations imposed by Twitter.

Bot developers are generally interested in politics, but most of them do not frequently participate in political discussions on social media. The bot developer survey addressed other political issues related to the last research question. The results indicate that the vast majority of bot codes were not developed or used for political purposes. Only 11.8 percent of respondents indicated that they had developed a bot that was used for political purposes. These bots were generally developed by developers who are themselves interested in politics. Nevertheless, most developers would not mind if the bot they developed was used for political purposes.

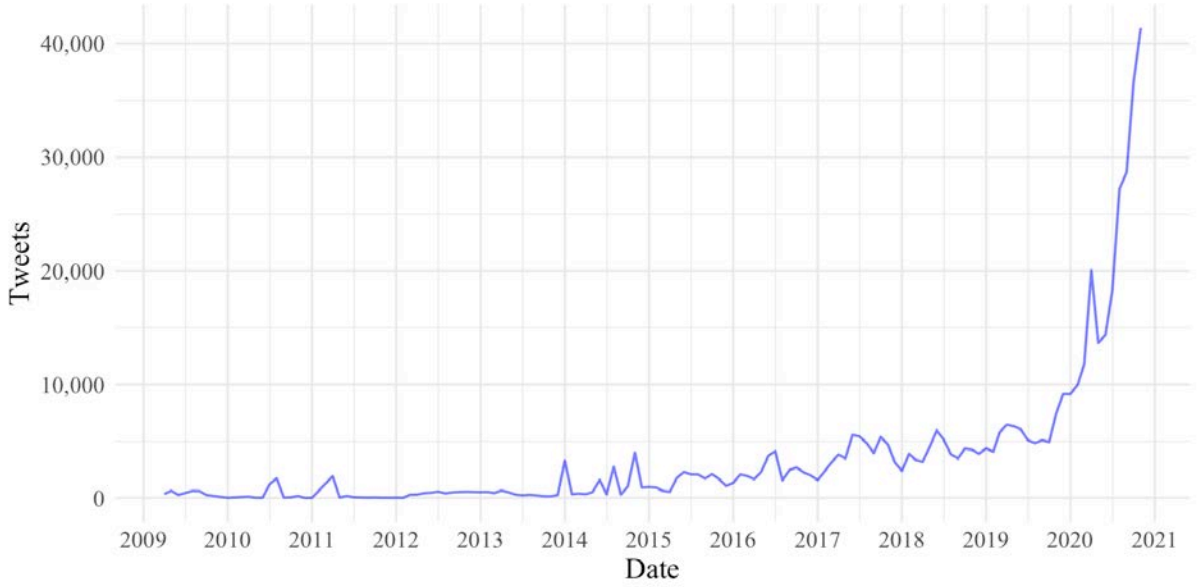
4.3. Open-source bots deployed on Twitter

After carefully analyzing the list of GitHub repositories associated with the developers who responded to the survey and the Twitter profiles of the bot accounts that were either specified in the survey or included in a repository in the form of a link pointing to Twitter, I identified 381 bot repositories with a corresponding Twitter profile. After querying these 381 accounts via Twitter's REST API, I was able to access partial or full timelines (tweets either produced or retweeted/quoted by the account) of 321 bot accounts. The remaining accounts were either suspended by Twitter, deleted by the bot master (the developer who controls the account), or simply had no activity on Twitter, i.e., no tweet or retweet was available in the timeline at the time of data collection. Within the data collected, 17 accounts had posted 10 or fewer tweets - this either indicates that the account was only tested on Twitter, or the account deleted most of its content. The most active automated accounts had posted well over 100,000 tweets by the time of my data collection.

4.3.1. The bots' tweeting activity - volume of traffic

Twitter sets a limit on access to a user's timeline: a maximum of 3,200 tweets can be downloaded via the free Twitter API. The remaining tweets can either be scraped through the website, which violates Twitter's terms of services (ToS) of Twitter, or are only available if the unique IDs for each message posted to the timeline are known. The following graph shows the traffic captured by querying the timelines of automated Twitter accounts.

Figure 20 *Monthly Activity of the Automated Accounts on Twitter*

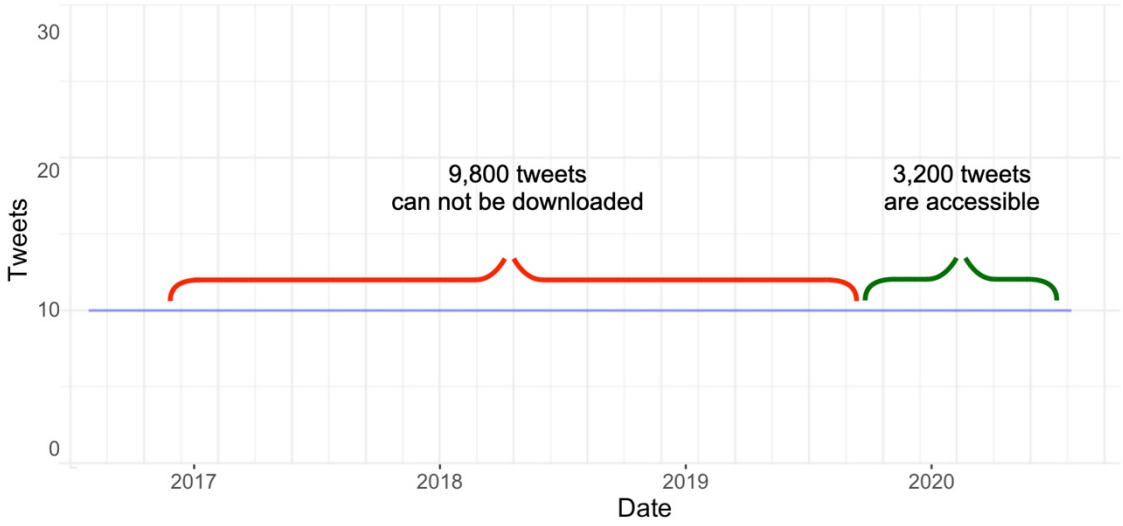


Note. Based on the data collected from Twitter. The maximum number of tweets per account available through the API is 3,200. The graph above shows the number of tweets per day produced by all the active bot accounts studied.

Figure 20, the line chart above, shows the distribution of captured tweets over time. However, due to the 3,200 tweet limit imposed by the Twitter API, the activity of the highly active accounts is not well represented in this plot. This is especially true if an account has been active for a long period of time and has exceeded the 3,200 limit. Let us take the example of an account that tweets at a moderately high frequency, 10 tweets per day, and was registered in 2017. API-based data collection would not be able to capture tweets from the period between 2017 and 2019 because the account had already posted more than 3,200 tweets in 2020. (Note: The 3,200 tweet limit includes recent tweets).

To overcome this limitation of the Twitter API, I looked at the total number of tweets posted (the status count metadata filed) available for each active Twitter account. If we take the example above, this account has posted well over 10,000 tweets since 2017. This simplified example of an account that tweets 10 times per day can be seen in Figure 21.

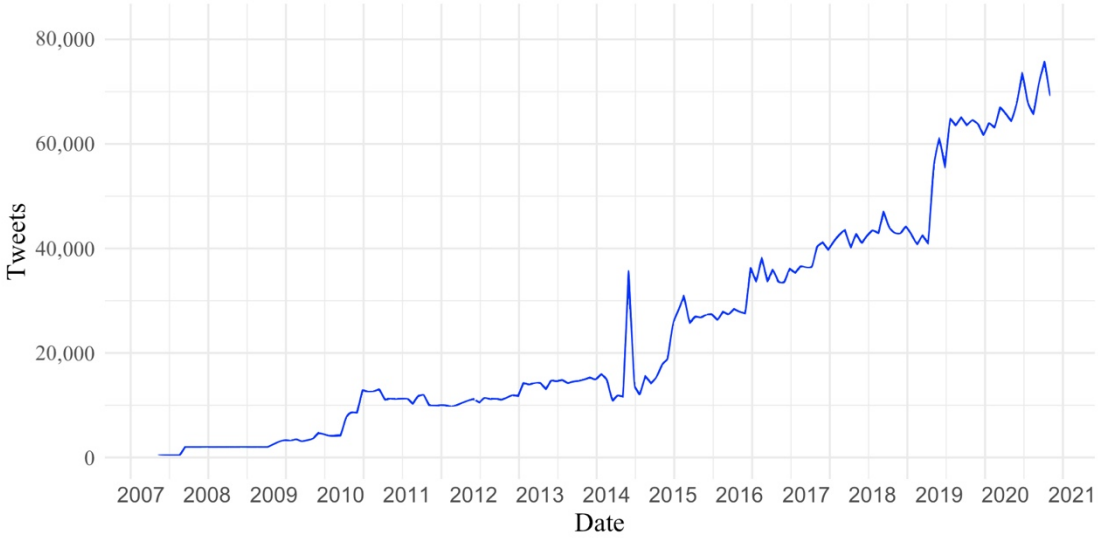
Figure 21 *Limitation of the API-Based Data Collection*



Note. This graph is a simple example with a bot account that tweeted 13,000 tweets or 10 tweets per day constantly for a roughly 4 year-long period. The API only allows to download the latest 3,200 tweets as it is shown with green.

If the total number of tweets posted by the account in the example shown in Figure 21 is 13,000, I can simply subtract the number of tweets captured by that account (about 3,200 tweets according to the API limit) and recreate the account's timeline by evenly distributing the number of missing (uncaptured) tweets between the time of registration and the oldest tweet captured by the account. The following chart (Figure 22) shows the reconstructed activity of the automated accounts over time. Again, any tweets that were unavailable to an account were evenly distributed between the time the account registered with Twitter and the earliest tweet downloaded from its timeline via the API. Therefore, the chart retains the granularity of the last 3,200 tweets captured per account and provides estimated simplified data for the period that was not available through the API.

Figure 22 *Monthly Activity of the Automated Accounts on Twitter*

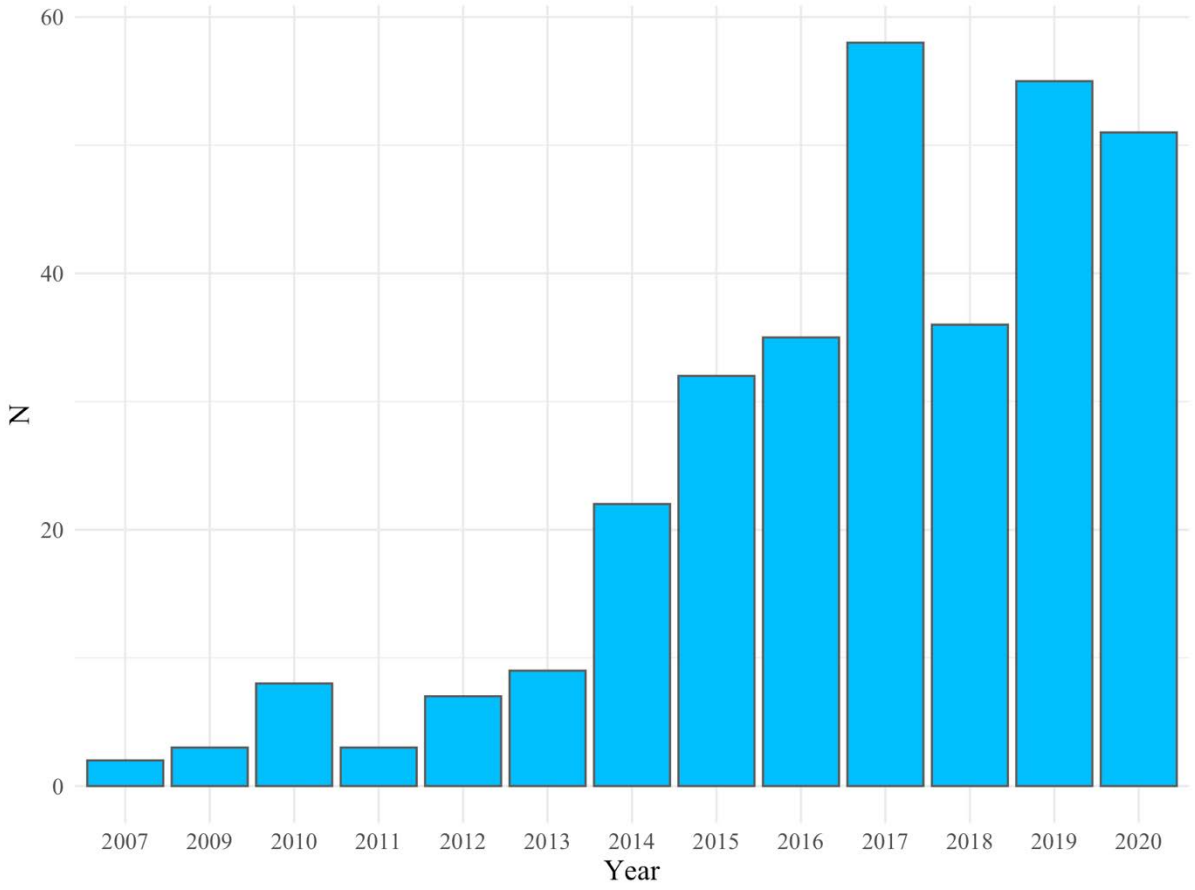


Note. Author's calculation based on the data collected from Twitter (max 3,200 tweets per account) and an estimation of the remaining traffic based on the total number of tweets per account and the time of registration for the accounts. The graph shows the number of tweets per month produced by all the active bot accounts studied.

This chart represents the traffic generated on Twitter by the known automated open-source accounts developed by survey participants. On the one hand, this chart gives a good estimate of the Twitter bot traffic that is still available through the platform. The distribution of traffic is consistent with the registration time of the 321 GitHub bot repositories analyzed here, as you can see in Figure 23. On the other hand, accounts that were active on Twitter but were suspended over time are not shown in this chart.

4.3.2. Temporal trends – how old are the Twitter bots

Figure 23 *Number of Known Bot Accounts Registered by Survey Respondents on Twitter*



Note. Author's calculation based on data collected from GitHub and Twitter. These are all open-source bots with a matching repository on GitHub and a bot account on Twitter.

The time trend or temporal distribution of registration for the 321 known GitHub bot repositories is consistent with the changes in the number of bot repositories registered annually on GitHub. The latter statistic has already been presented in Subsection 3.1, but we can quickly compare the changes in the number of repositories in our sample and the changes in the total number of Twitter bots registered. According to the analysis of GitHub repositories, bots have gained significant popularity over the last five or six years. Table 5 shows a similar trend for the repositories created on GitHub and the Twitter bot accounts registered by respondents to my survey.

Table 5 *Annual Changes in the Number of Bot Accounts Registered on Twitter*

	Twitter accounts		GitHub repositories	
Year	N	%	N	%
2020	51	15.9	2,423	15.0
2019	55	17.1	2,911	18.0
2018	36	11.2	2,996	18.5
2017	58	18.1	3,166	19.6
2016	35	10.9	1,846	11.4
2015	32	10.0	1,201	7.4
2014	22	6.9	693	4.3
2013	9	2.8	377	2.3
2012	7	2.2	223	1.4
2011	3	0.9	135	0.8
2010	8	2.5	117	0.7
2009	3	0.9	64	0.4
2008	0	0.0	16	0.1
2007	2	0.6	0	0.0

For both survey respondents (with a matching bot deployed over Twitter) and the full database of Twitter bot repositories available on GitHub, the number of bots created was highest in the last five years, and 2017 was the most active year of bot development/deployment. Although the survey sample includes Twitter accounts registered before 2015, most survey respondents created their bots in the last 5 years. A quick look at

the repositories opened by the 321 survey participants with a paired bot account suggests that the vast majority (73.2 percent) of Twitter bots were registered in the last 5 years - this number was even higher (82.5 percent) in GitHub's full Twitter bot repository database. This suggests that developers who have developed a Twitter bot more recently were not more likely to respond to a survey about bots. The differences in responses between developers with older and newer bot repositories can be explained by the distribution of bot repositories on GitHub over time.

To reconstruct the activity of open-source bot accounts on Twitter, I assumed that an automated account tweets more or less the same number of tweets per day, since its activity is controlled by a predefined schedule. While this is certainly the case for many accounts, it depends heavily on the algorithm the Twitter bot uses to find or generate content. Accounts that generate content by either retweeting specific accounts or listening for a set of keywords or hashtags via Twitter's Streaming API, for example, tend to follow a more hectic tweeting pattern. In other words: If an account's activity depends on content posted by another Twitter user or content updated by a person or organization outside of Twitter, the account could follow a more diverse, human-like tweeting pattern. This could also lead to challenges in detecting such bots, as their tweeting behavior follows real human communication patterns. Nevertheless, these accounts could follow certain guidelines, e.g., the bot's developer could limit the number of daily tweets for various reasons to avoid flooding the bot's followers or to avoid hitting Twitter's API limits.

The 321 Twitter accounts analyzed in this chapter of my dissertation collectively posted 3,922,879 tweets, but there is a large difference between the volume of tweets posted by the different accounts. While the most active account posted more than 500,000 times, half of the accounts posted less than 1,000 times and about 1 in 5 accounts posted less than 100 times. Even among very active accounts that posted several thousand messages, there is wide variation in the number of average daily tweets.

4.3.3. Life-time of a Twitter bot

To calculate the number of average daily tweets, I first calculated the amount of time being active on Twitter. This time span was based on the number of days between registration and the last posted tweet. The total number of tweets was simply divided by the number of days being active. This method was better than calculating the age of the account, i.e., the time from registration to the time of data collection, because there were accounts that were

inactive for a significant amount of time, e.g., an account that tweeted a lot during the 2016 U.S. presidential election campaign period but had been inactive since the election.

Accounts that posted more than 10,000 times posted between 5 times and more than 500 times on an average day, according to data downloaded from Twitter. A good example of the accounts with a high average number of daily tweets is the account called First Issues (@first_issues), which tweets about new issues on GitHub that can be easily solved by inexperienced developers so that people new to open-source development can try to solve them. This bot has been active since 2018 and has posted more than 300K tweets, averaging 433 tweets per day.

Twitter bots can tirelessly produce tweets or complete tedious tasks over an extended period of time. However, bots often stop their activity. There can be many reasons for a bot to become inactive, and even the fact that the bot is inactive is not always obvious. In some cases, the artifacts produced by the bot, i.e., the tweets posted by the account, remain available on Twitter. An account may stop tweeting for at least the following three reasons: 1) A developer's API access may be suspended, either for the activity of the bot under investigation or for the activity of another bot developed by the same developer if the accounts use the same developer account. If the bot has not been banned by Twitter, the account's tweets will remain online. 2) Hosting and running the bot could become either cumbersome (too much maintenance) or too expensive (cost of hosting the bot). If the code needs to be updated or there are issues with the hardware used to run the bot, the bot developers will have to put in extra effort to keep the bot running. If the capacity of free hosting services is exhausted, the cost of hosting the bot can also become expensive. 3) There are bots that focus on a specific event, such as the example bot that focuses on a presidential campaign or other specific time-bound political event. These bots are often abandoned, becoming inactive, but the content is not usually removed from Twitter.

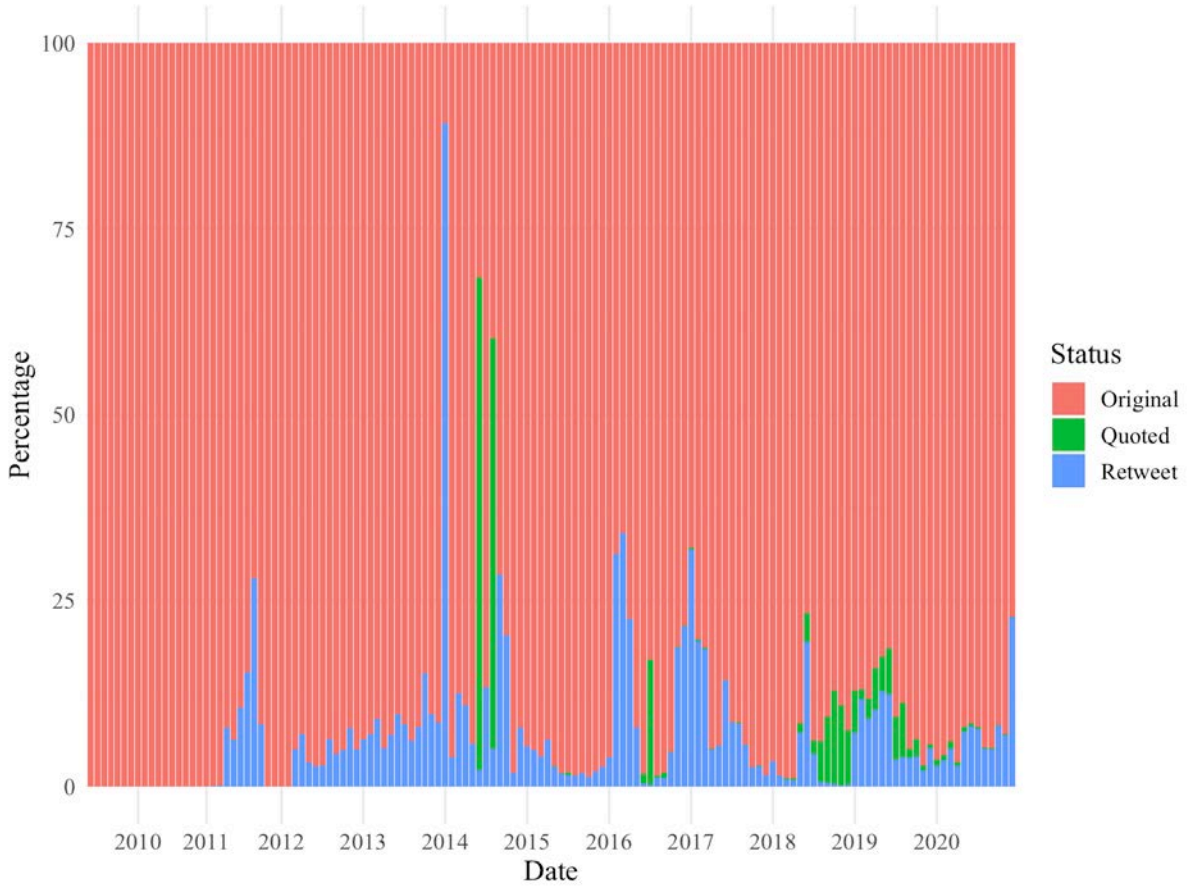
The Twitter bots examined in this chapter tend to be active for a long period of time. The average lifespan, or time in days from the account's registration with Twitter to the last available tweet from the account, was 836 days. About 30 percent of accounts are or were active for at least 3 years, while only 36 accounts (about 10 percent) were active for a period of less than 2 months. The oldest longstanding account, named Library 1, 2, 3... (@library) has been active on the platform for more than 13 years as it was registered on Twitter in April 2007. The account posts short think pieces derived from famous quotes or memorable lines from literature. The account currently has 674 followers and has been posted more than 3,500 times.

4.3.4. Where does the content posted by Twitter bots come from?

Bot accounts can post original content by generating content based on code, using a list of pre-written messages, or relying on third-party sources. For example, the aforementioned library account relies on content created outside of Twitter - the bot generates original content based on an algorithm and a text file. These tweets, regardless of the source of the information or the actor behind the tweets, are called original tweets. In contrast, Twitter users (including bot accounts) have the ability to retweet or quote content from other users. These tweets are circulated within the platform. Accounts that exclusively retweet other accounts are sometimes referred to in the literature as signal boosters or amplifier accounts. Bot accounts often retweet either specific accounts or any content based on specific keywords or hashtags. The Twitter API also makes it easy for bots to quote tweets by adding a bot-generated (or pre-written) comment to content posted by another Twitter account.

My Twitter bot sample has a good mix of original content, retweets, and quote tweets. The following chart (Figure 24) shows that open-source bots generate a lot of original content on Twitter.

Figure 24 *Original, Retweet and Quote Tweets Generated by Bot Accounts*



Note. Ratio of Original, Retweet and Quote Tweets Generated by Bot Accounts by Month.

During the Twitter phase of my research project, I was able to download 486,859 tweets from the 321 accounts that had public tweets available on Twitter at the time of data collection, including original tweets, retweets, and quoted tweets. Of these tweets, 443,171 (about 91 percent) were original tweets, meaning that the bot generated the content based on an algorithm or source outside of Twitter. It is important to note that this does not mean that the only source of the content was the algorithm itself. Bots often draw on public datasets, pre-written text sources, or Twitter itself to create content. In fact, open-source bots often act as a new interface to an existing online service or information source. On the other hand, there are bots that generate their own content from geometric shapes to a unique combination of different text sources. Among the non-original content generated by the bots, the vast majority of tweets were retweets. The 34,854 retweets accounted for 7.2 percent of the total traffic captured, while the 8,834 quotes accounted for less than 2 percent

(1.8 percent) of the total traffic.

An original tweet is, very simply and technically, a tweet written or generated by a Twitter user. In other words, these tweets are not generated by Twitter's retweet feature or its quote feature. I'll define the latter two tweet formats in more detail later, but first let us take a quick look at the source of content published on Twitter. Each tweet can contain textual and (audio) visual elements. The source of these elements can be any content (audiovisual or textual) that is 1) user-generated (this can include bot-generated content); 2) stored locally by the user, such as the text of a digitized book; 3) retrieved over the internet, such as content dynamically downloaded from a public database or website; and 4) Twitter itself.

Retweets have been available on Twitter since the earliest days of the platform. Interestingly, the retweet feature was developed by a user of the platform-the term first appeared on Twitter in April 2007 (Seward, 2013). Prior to that, another user, Narendra Rocherolle (@narendra), used the term “echo” to refer to reposting a tweet (Tarte et al., 2015). The retweet button (RT) was not introduced until 2009. This information is important only because there is now a computational mode to retweet a content, and this feature is often used by bot accounts.

When an account retweets another tweet, the original tweet is shared with all followers of the retweeting account. The platform's retweet feature can also be defined as a computerized method of re-positing an entire tweet on Twitter by either using the retweet command through the Twitter API (the preferred way for bots) or clicking the retweet button on Twitter's website or a Twitter client. (Technically, Twitter clients also use the API to get or put content to Twitter.)

There are two specific characteristics of a retweet: 1) Retweets are directly linked to the original tweet and appear in the timeline of the user who retweeted a tweet as content generated by another user. 2) Retweets contain a standard RT @useraneme element at the beginning of the tweet. This last element of a retweet allows users to manually “retweet” a message by adding the RT @username element and simply pasting the content of the original (“retweeted”) tweet. However, this tweet would show up as an original tweet in our database because it was not generated by the website's retweet function or by retweeting via the API. Consequently, this tweet is not directly linked to the original tweet. This also means that this manual retweet does not contribute to the retweet number displayed under the original message. The poster of the original tweet still receives a notification, because the username controlled by the author of the tweet is mentioned in the tweet (@username).

Finally, quotes have a similar function to replies, but here tweets are re-posted with the option to add a comment to the original tweet. The original user can reply to a quoted tweet. This type of tweet overcomes an important limitation of retweeting, which is that it is not clear whether a retweet means an acknowledgement or not. In other words, it is not clear whether the user agrees with the original message or not. The quoting user has the option to add text or visual information to the quoted tweet. Just like retweets, quoted tweets are directly linked to the original tweet, and this quote contributes to the engagement metrics of the original tweet.

Twitter recently announced that the platform will encourage quotes to combat the rapid spread of misinformation. The following Twitter message was posted on the company's official blog in connection with the 2020 U.S. presidential election:

"First, we will encourage people to add their own commentary prior to amplifying content by prompting them to Quote Tweet instead of Retweet. People who go to Retweet will be brought to the Quote Tweet composer where they'll be encouraged to comment before sending their Tweet. Though this adds some extra friction for those who simply want to Retweet, we hope it will encourage everyone to not only consider why they are amplifying a Tweet, but also increase the likelihood that people add their own thoughts, reactions and perspectives to the conversation."

4.3.5. Metadata about bot repositories and the deployed bots

There is a large amount of metadata, both about the bot repository and the actual bot deployed over Twitter. For each repository, GitHub provides access to the number of stars, watchers, and forks. Users can star a repository on the GitHub website by clicking a button, similar to the "Like" button on Facebook. Watchers are developers who want to keep an eye on a particular repository and subscribe to notifications about changes in the code or readme file of the repository. The number of forks is another type of metric that is most relevant in the context of social coding or open-source development. On GitHub, users can make a copy of the content (code and files) of a repository. Then, these users can work on the copy of the repository and later have the opportunity to propose changes to the original

code. The latter must be accepted by the owner of the original repository.

However, there are problems with almost all of these GitHub metrics. Stars are not used too often in the bot developer community. Of the 321 paired GitHub repositories (these are the repositories created by survey participants and include a known Twitter bot "pair"), 186 repositories have zero stars. The vast majority of the remaining bot repositories have five or fewer stars. A relatively small number of repositories (28) have more than five stars. The most popular bot repository has 98 stars. The ten most popular repositories have about the same number of stars (364 combined) as all other bot repositories (393 combined). This long tail distribution of stars and the high number of zero stars make it difficult to use this metric in more sophisticated, multi-level analyses, such as a cluster analysis. Still, this is probably the best internal indicator of success on GitHub.

The watch feature on GitHub is even less used compared to staring, at least the total number of watchers (361) is smaller than the total number of stars (757) in the case of the 321 paired bot repositories. Moreover, the number of repositories without watchers is relatively high (58). A careful analysis of the watch metrics also shows that users often watch their own repositories. Subtracting all self-watchers would inflate the metrics even more.

The final internal GitHub metric for paired bots is the number of forks. Only 67 bot repositories have a fork, and the total number of forks is only 237, indicating that, with the exception of a few less popular bot repositories, developers are not interested in directly reusing other developers' bot code or contributing individually by working on a copy of the repository.[8]

Twitter also provides a set of metrics to describe an account's activity, source of information, and user engagement. I described information sources and bot activity above, so I'll focus on user engagement metrics here. The standard Twitter APIs provide access to two important user engagement metrics: the number of retweets for a tweet and the number of likes, formerly known as favorites. However, retweets can be misleading. When a bot retweets a tweet from another Twitter user, the retweet counts available for that tweet are based on the retweet count of the original tweet. The retweet count indicates the popularity of the original tweet, regardless of how many retweets were generated by the bot or its followers. This means that retweeting a tweet that has already been retweeted (for example,

[8] A fork allows another developer to improve a code on a parallel copy, and the author of the original code can later merge these changes or improvements to the original repository.

when you see a retweet in a bot's feed) is counted in the retweet count for the original tweet. In other words, the bot's contribution to the spread of a tweet cannot be measured using public data, as Twitter does not provide access to this information.

Because of the difficulties in measuring the bot contribution to the spread of a retweeted message mentioned above, and because the metrics provided by Twitter are misleading, I decided to set the RT number for retweets generated by bot accounts to zero. Interestingly, the favorites for a retweeted tweet is still relevant and accurate, as these numbers show how many users favorited the (retweeted) tweet that appeared in the bot account's timeline.

Querying the timeline of each bot account that has a paired Twitter handle and a GitHub repository yielded a database of about 490K tweets. This dataset is limited by Twitter's imposed limit of 3,200 tweets per account, so the dataset contains at most about 3,200 tweets per account, even if the account tweeted significantly more. The original or quoted tweets from the downloaded tweets were retweeted a combined 93,078 times. This suggests that a bot-generated tweet is retweeted 0.19 times on average, or to put it more simply, for every 10 tweets generated by an average bot, there are 2 retweets. However, not every bot account achieves the same retweet ratio. One would assume that bots that produce a very large number of tweets would be retweeted less often, but this is usually not the case.

4.3.6. Most important research findings

In this part of the thesis, I examined 321 Twitter bots based on both their Twitter activity (tweets, retweets, etc.) and their corresponding GitHub repositories. This analysis mainly addresses the second set of research questions, more specifically the research questions about how bots generate content (RQ 2.1), how long they are active on Twitter, and how much they post (RQ 2.2).

One of the most important and less expected findings of this chapter is that open-source bots are indeed posting new, original content on the platform. Due to Twitter's API limitations, I was only able to access the most recent content (3,2000 tweets per account) in the bots' timelines. For the 321 bots, I was able to download 443,171 tweets, and more than 90 percent of that content was original, mostly bot-generated content as opposed to simply retweeting existing tweets on the platform. Together, the 321 accounts posted nearly 4 million times. The time between registration (usually a bot's first tweet) and the bot's last

communication averaged 836 days. About one in three bots was active for at least 3 years. This suggests that a large number of Twitter bots from the bot repositories identified on GitHub are or have been active for longer.

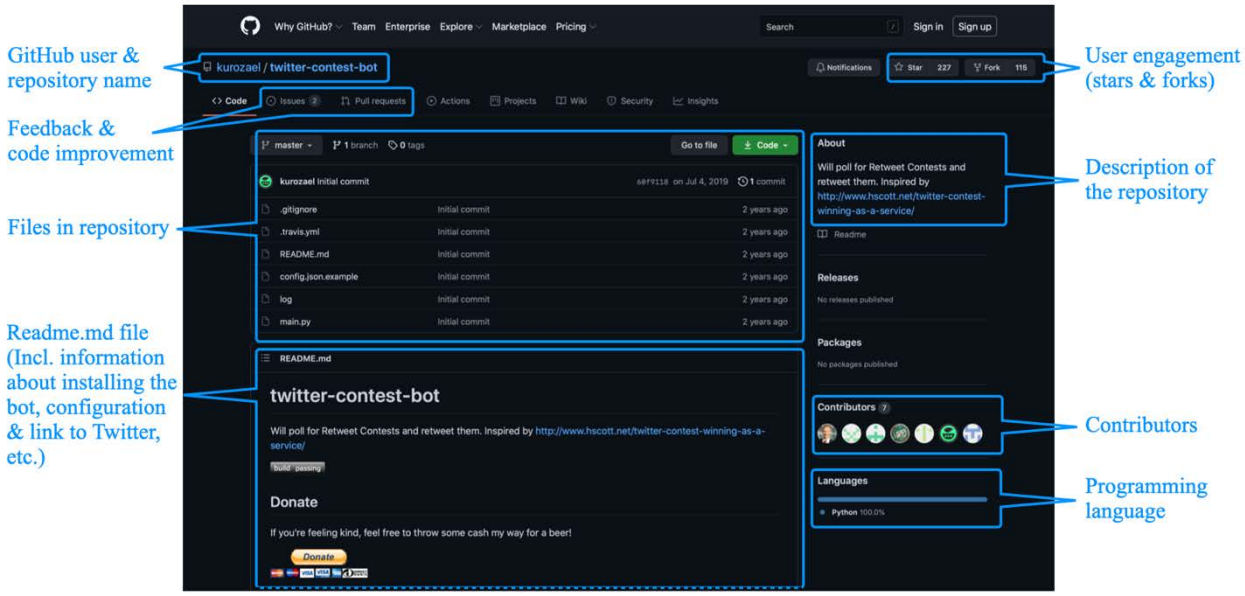
4.4. An open-source Twitter bot typology

The goal of developing an open-source bot typology, or typology in general, is twofold: on the one hand, we need to determine the key characteristics of our subjects that can distinguish them from one another; on the other hand, the subjects in our research that are similar along these variables can be grouped and quantified. According to Babbie (2016), a typology is a way to summarize our data along two or more variables. A typology is often based on qualitative data. To obtain reliable qualitative data about our subjects, the response categories (values) for a variable should be mutually exclusive. Otherwise, it is not possible to consistently apply coding decisions to a large dataset.

4.4.1. Code review and content labelling

First, I linked the bot repositories on GitHub and deployed bots on Twitter (as described earlier in the results chapter and methodology section of this thesis). Then, for each bot account, I manually reviewed the code on GitHub and visited the bot profile on Twitter in a browser. On GitHub, in addition to the already downloaded metadata about the bot repository and bot developer, I also checked the following information: 1) the description of the bot repository, 2) the Readme.md file uploaded to the repository, 3) the file(s) containing the code to run the bot, 4) the content used by the bot if it is also stored on GitHub, and in some cases the 5) description added to a commit (a commit is a change to the code uploaded to GitHub). The annotated screenshot below shows an example of a typical bot repository on the GitHub website.

Figure 25 *Annotated Screenshot of a Typical GitHub Bot Repository*



Note. Screen shot and annotation by the author.

Almost all bot repositories have a brief description, but the information is often limited and does not provide enough detail about the bot. For example, a description “A simple Python twitterbot” reveals nothing about the bot's function(s) or the data source it uses. Although adding a Readme.md file to a repository is a common practice on GitHub, not every bot repository has a Readme.md file. The Readme.md file is a short text file written in a markup language that must follow a simple syntax. When the file is present, GitHub automatically loads the contents of the file and displays it among the files available in the repository, as shown in Figure 25. Depending on the common practices of the programming language used for the bot, the repository contains one or more files with code. In addition to the code, these files often contain annotations and comments that help both other programmers understand the code and the developer make changes to the code later. Most bot repositories did not contain content because usually repositories used online data sources as input for content creation. Even bot repositories that relied on locally stored content often contained only a limited version of the database or text or audiovisual files used by the bot. (Note: I define locally stored data as data stored on a computer (server) connected to the internet that hosts the code itself. In other words: If the developer is using cloud computing, store locally means that the data is stored in the cloud on the same server or a connected online storage). Finally, commits, updates, or changes made by a developer often include useful comments, such as adding a specific function to the bot or connecting the bot to a web

source or API.

The remainder of this chapter is organized as follows: I categorized open-source Twitter bots based on three dimensions - the source of content used by the bot (see Section 4.4.2.), how bots process and publish content (see Section 4.4.3), and whether or not the bot is interactive (see Section 4.4.4). The remaining sections of this chapter focus on the technical details of the bot - first listing the Twitter features that the bot can access, and then examining how the bot generates its content (generative vs. found content). The bot typology presented in this thesis is based on the above three dimensions, but the technical details provide further insight into the actual programming of the bots.

4.4.2. Finding or generating content for bots

Question for dimension 1 - Where does the content for bot accounts come from?

The first variable that can be used to create a typology of open-source Twitter bots is the data source used by the bot. To quantify bot accounts that rely on static information, dynamic information posted on Twitter itself, or content that comes from somewhere else, including generative bots that generate content based on code, I provide a description for each type of content generation or distribution, along with statistics based on the relatively small sample of 321 paired bot accounts I worked with - those accounts where I was able to pair source code (data from GitHub) with a bot deployed on Twitter (data from Twitter).

Although some bot accounts rely solely on a set of previously prepared, static, often locally stored content, most open-source bots use dynamic sources of information. Bots that work with static, local content, such as a folder full of images or a digital version of a book, are often used only to illustrate the capabilities of a Twitter bot. These bots are deployed on Twitter only to showcase a bot. It should also be noted that these bots are often skeletons, meaning they provide the mechanics of a bot, but they must be connected to a data source and set up properly before they can be deployed on Twitter. These skeleton bots are simple, designed for common tasks, and can usually be easily customized.

Of the 321 bot repositories, 20 bot codes either use only pre-built or previously collected static content, or rely mainly on locally stored content. A good example of the former is an automated account that tweets a picture of a fox every Friday, or a project where the bot tweets 140-character long sections of Pi, a mathematical constant. Both are very simple bots that use only local content. Of course, locally stored content can change

as well. For example, there is a bot that tweets a random album cover from the bot master's phone with the folder title - the content on the phone can change often.

Another source of bot content is in the code itself. For example, code can generate an image or use an algorithm to manipulate text that is either stored in a local file or dynamically scraped or downloaded from an internet source. 134 bots use code-generated content or content that has been previously specified in the bot code. A good and simple example of code-generated content is a bot that plays with the concept of normal distribution and constantly tweets random numbers generated by an algorithm. The difference with the Pi bot is simple: the normal distribution bot generates the random numbers dynamically based on its code, while the Pi bot reads the numbers from a very long text file. Code-based bots can even produce art, as one of these bots produces computer-generated drawings with seemingly random, made-up (also code-generated) titles. This bot refers to its self-generated images as “procedural art” and uses hashtags such as #ComputerArt or #CreativeCoding in the text of its tweets.

The next important source is Twitter itself. There are a large number (38) of bots in my dataset that use tweets from other Twitter users. The simplest version of this type of bots simply retweets a specific source (e.g., a Twitter user) or hashtag. These bots can also do some curatorial work in the sense that they retweet only certain messages based on an algorithm, e.g., tweets that contain a hashtag (i.e., belong to a particular thematic discussion on Twitter) and have already received a high level of user interest (e.g., a message that has been liked more than 100 times). Although retweeted tweets disappear when the original message is deleted, retweeting specific tweets creates an archive of tweets in the bot's timeline that can later be searched or simply browsed.

More complex bots, which are still among those that rely on Twitter as a source, use a combination of a Twitter source and either local content, another web source, or the code itself. The bot called @WuTrump, for example, used the former U.S. president's recent tweets and lyrics from the Wu-Tang Clan, the popular New York hip-hop band from the 1990s. The result was an often funny version of the already not very presidential tweets in a jargon used by the hip-hop subculture.

The last type of primary source is the web outside of Twitter. To access dynamic, reliable, well-formatted (machine-readable) content, many bot accounts rely on the application programming interface (API) of other web services.

Often, the content published on a website can be accessed without using an API, namely by scraping a website, i.e., directly accessing and downloading the content of a

website through a computational method. The reasons for not using an API are many and range from the cost of using a commercial API to the limitations of accessing data through the API. Lack of skills/technical knowledge about a particular API can also contribute to not using an API - it is often easier and sometimes simpler to access data without using an API. In addition, not every possible web resource has a public API.

Nevertheless, the availability of well-formatted, reliable data via the API is an important source of data for open-source Twitter bots - 39 Twitter bots used an API to publish content to Twitter. While some of these bots simply published content retrieved via the API, others mixed it with code-based content or data from other web sources. There are a large number of free and open APIs that can provide content to a bot - Google Maps, HERE, and OpenStreetMap all provide access to maps and various geographic data, Wikipedia has a free, well-documented API, Dark Sky used to provide access to high-quality weather data, and of course all the major social media platforms have APIs, from Instagram to Reddit. There are also (often fun) niche APIs for accessing free pictures of dogs (DogCEO API), facts about the Star Wars movies (SWAPI), and all sorts of data about Pokémon (Poké API) - all of which is good material for building a Twitter bot. Microsoft, IBM, Amazon and other big tech companies also provide access to image recognition, online computing and artificial intelligence that can be used for more complex Twitter bots.

Using the well-formatted, reliable data and web services available through APIs, which are often free and open, developers have created bots that generate content by accessing and processing data from a web service. For example, a bot was developed that documented an imaginary walk on the world map (@DailyWalkBot). The bot posted more than 21,700 short tracks that were visualized on Google Maps. At the time of data collection, the bot was walking on an unnamed road in the middle of the Mexican state of Coahuila. The code behind the bot generates random locations within walking distance of each other, and Google Maps, accessed through its API, calculates the best route between the locations and creates a map with the route's track highlighted. Another developer has launched a Star Wars quiz bot that posts trivia questions about the popular science fiction film (@starwarsbot1) - the content here comes from a Star Wars API. In addition to fun side projects, developers have also created useful services based on music recognition services (ACRCloud API and Shazam API) that can name the music used in a video posted on Twitter or elsewhere by simply mentioning the bot in a tweet (@nomemusica). These bots post dynamic content, they are reliable and require less effort to maintain.

4.4.3. Processing and publishing content

Question for dimension 2 - What functions do the bots perform?

The second variable that can be used to create a typology of open-source Twitter bots is the function that these bots perform, which is directly related to how these bots process and publish content. Bots are often designed to automate tedious and tedious tasks, such as creating similar messages or automatically retweeting certain content based on simple rules. Although the categories are designed to be mutually exclusive, individual bots can and in some cases do combine them. In other words, the majority of bots fall into one of these categories, primarily because bots are often programmed to perform a single task. However, some bots are more complex and can perform multiple or complex tasks.

Amplifier. These bots take a specific signal on Twitter and repeat the content, either from an account or from content that matches certain keywords or hashtags. Technically, this often takes the form of a retweet, but the account may also simply copy the content and repost it as an original tweet, or use the platform's quote feature, where the bot adds a short comment to the original tweet. Either way, these accounts can duplicate the content, make it available to other users, or simply drive up retweet counts for a particular account, keyword, or hashtag.

Curator. These accounts also take content that is available on Twitter, but the method of selecting the content is more sophisticated and the curation itself creates a new filtered view of the information previously published on Twitter. For example, a bot that retweets every tweet from the U.S. president that mentions a particular country creates a stream of such tweets, making it easier to either follow or look up those tweets later, effectively creating an archive on Twitter. Similarly, retweeting only tweets that have been liked above a certain threshold creates a new filter for content available on Twitter.

Content mash-up. Mash-up bots still frequently use tweets posted on the platform. The bot either combines tweets from multiple sources on Twitter into single mash-up tweets or uses a combination of tweets posted on Twitter and a third-party source. A good example of the latter type is the bot that created mashups of Donald Trump's tweets and the lyrics of the Wu-Tang Clan, the famous New York hip-hop band.

Transmitter. These bots rely on content posted on another social media platform or website. The bots either use an API designed for computational access to web services (including content published on those websites), or they simply scrape data from a specific website. A Twitter bot that posts customized weather reports may access weather data (content) through the API of a weather service or website. Another example is a bot that automatically generates a tweet when content is posted on another social media platform, such as Reddit.

Content creator. These are bots that produce their own content without using data (text or visual content) from Twitter. Generative bots produce and publish content based solely on the code running behind the bot - this category includes artbots that produce computer-generated images or, to take another example, computer-generated poetry. Other bots creatively use multiple sources outside of Twitter or mix and match them with content created based on rules set in the bot's source code.

Service bot. Service bots do not have public/visible content. Instead, these accounts perform tasks in the background, such as collecting data, following specific accounts, or liking content posted by other accounts. Liking content is not invisible per se. Users who visit the account's profile can see this activity, but it is not displayed in the account's timeline, and the bot account's followers are not notified about it either. So, the invisibility of this type of actions depends on the design of the platform. It is also important to note that bot accounts are often able to perform some of these invisible tasks, even if they belong to one of the other types. For example, they can follow back a user who starts following them, save Twitter data, or like certain types of tweets.

4.4.4. Interactivity and Twitter bots

Question for dimension 3 - How bots interact with users?

Media theory has identified interactivity as a key feature of new media, especially when compared to traditional electronic and print media (Lister et al., 2008). This literature focuses on both the role of users in content creation (contribution) and how audiences can control access to content. Contribution and interactivity are linked together, and the latter concept plays a central role in defining social media, a platform where users, both

individuals and communities, “share, co-create, discuss, and modify” content (Kietzmann et al., 2011). The role of interactivity in the changing media landscape has also been discussed extensively in the literature on digital journalism (e.g., Schultz, 2006; Spyridou and Veglis, 2008; Sundar, 2004). Similarly, political science and political communication studies have devoted a large body of literature to the interactivity of politics, which primarily includes papers on the interaction between politicians and their constituents (Bright et al., 2020; Jacobs and Spierings, 2019; Tromble, 2018).

Much of the media studies literature on interactivity draws on Jan Jensen's (1998) early definition. Jensen defines interactivity as “a measure of a medium's potential ability to let the user exert an amount of influence on the content and/or form of the mediated communication” In addition to content creation, control plays a central role in theories of interactivity. Both Jensen (1998) and McQuail's (1994) classic textbook on mass communication draw on the media typology developed by Bordewijk and Kaam. For Bordewijk and Kaam (1986), the two most important dimensions of interactivity are ownership (who owns or produces the content) and control (who decides the subject and timing of the content accessed).

Everett M. Rogers (1986) uses a simple, two-dimensional interactivity scale that can be used to distinguish between low and high interactivity media. For example, TV, radio, and the press, often referred to as traditional media, can be positioned at the low end of this spectrum, while new media represent the other end. This not only highlights interactivity as the main difference between these media types, but also shows that interactivity is a continuum and that there are different levels between high and low interactivity. Furthermore, according to Rogers, the level of interactivity depends not only on the medium or the technology behind it, but also on the human users of that technology and the context of use.

Jensen (1998) introduces a more complex, 3-dimensional representation of the different types and levels of interactivity - he calls it the cube of interactivity. The 3 dimensions are derived from conversational interactivity, selective interactivity, the freedom to choose content, and registering interactivity, the ability to adjust the response based on information provided by the user. This 3-dimensional model can also be divided into four sub-concepts that may be useful for understanding interactivity. The following section provides brief definitions or descriptions for these four sub-themes:

“Transmissional interactivity – a measure of a media’s potential ability to let the user

choose from a continuous stream of information in a one way media system without a return channel and therefore without a possibility for making requests

Consultational interactivity – a measure of a media's potential ability to let the user choose, by request, from an existing selection of preproduced information in a two way media system with a return channel

Conversational interactivity – a measure of a media's potential ability to let the user produce and input his/her own information in a two way media system, be it stored or in real time

Registrational interactivity – – a measure of a media's potential ability to register information from and thereby also adapt and/or respond to a given user's needs and actions, whether they be the user's explicit choice of communication method or the system's built-in ability to automatically 'sense' and adapt"

Transmissional and constitutional interactivity, i.e., selection from preproduced content in a one or two-way media system, yield the dimension of selective interactivity, while the other two subthemes refer to low or high levels of conversational and registrational interactivity.

Jensen also used his 3-dimensional model to categorize different media or technological platforms when he wrote his article, which was in the late 1990s, corresponding to the early days of the internet and predating Twitter. Television, for example, can be at the low end of the selective interactivity scale, but it can also offer the ability to choose between different content streams (multichannel) and achieve a high level of selectivity through the introduction of on-demand content. Both Rogers and Jensen's two-dimensional and three-dimensional models mark precise positions for each medium, but one can also find examples of media that typically have a low level of interactivity but can also be used in more interactive ways. Radio, for example, is typically a one-way medium with only limited interactivity, but listeners can easily dial in and join the radio program to voice their opinions or even change the stream by asking for specific content.

Twitter itself is much more flexible than most of the media categorized by Rogers and Jensen. The technology itself provides various means of interactivity, from replying to a user's mention to quoting a tweet and adding a comment. In addition, the platform can be used as a one-way communication tool when users follow information sources and consume

content without producing content themselves or interacting with the information sources in any way (from retweets to other forms of user engagement). Similarly, Twitter bots can be programmed to exhibit varying levels of interactivity, from sharing information to having a human conversation with chat capabilities.

From a technical perspective, interactivity can mean choosing between simple, predefined options and giving the user much finer control. For example, Lev Manovich (2001) distinguishes between closed interactivity and open interactivity - while the first option provides the user with a branch like decision tree and they can only choose from predefined options, the second allows much more freedom in customizing the retrieved content. However, open interactivity also requires more sophisticated technological solutions such as artificial intelligence, neural networks, etc. compared to the simpler closed interactivity. This is also true for Twitter bots - more sophisticated conversational bots tend to rely heavily on natural language processing and AI.

To understand interactivity in the context of Twitter bots, it is necessary to consider media-specific features (interaction capabilities offered by Twitter) and some technical aspects. Twitter offers the following options for interaction between users: retweet, quote, at mention, reply, and direct message.

To be an interactive bot, a Twitter bot must use one of the above options to interact directly with users. While some bots use at mentions to interact with users but the communication is still one-way, other bots respond to either input, requests, or commands from Twitter users, so the communication is truly two-way.

Based on the existing media studies literature on interactivity and the technical characteristics of Twitter as the medium used by bots, I propose to measure the following three dimensions of interactivity:

Communication – The first dimension is communication, more specifically, how does communication take place between users and the bot? This dimension can take 4 forms, partly due to the technical characteristics of the platform. Bots can be designed to avoid communication and generate content that is only visible to the bot's followers or users searching for that content. Users may communicate with the bot intentionally and without intent or knowledge of the communication. Bot accounts can listen to the stream of public communications on the platform or search for specific keywords or hashtags that cause the bot to tweet. Similarly, bot accounts can automatically retweet certain tweets. The user who originally posted the tweet is notified, but this type of notification can be turned off by the

user. Intentional communication (from the user's perspective) can be done in a closed interaction format, e.g., a user can simply include an at mention (@username_of_bot) in a tweet to summon or command the bot. More advanced chatbots can allow users to interact openly (following Manovich's notion), but this type of communication is extremely rare among the studied open-source Twitter bots.

Control – This dimension is about who determines the topic and timing of the content generated. This is one of the two dimensions of interactivity defined in the Bordewijk and Kaam (1986) article cited earlier. In the context of Twitter bots, an automated account can tweet or respond to user-posted content or direct messages from users without user intervention. To be an interactive Twitter bot, users must be able to tweet or message the bot account and instruct it to respond with a specific piece of content (e.g., a local politician's contact information or the location of a voting booth) or a service (e.g., identifying a song used in a video uploaded to Twitter). It is also important that the bot responds automatically and in a timely manner. Aside from being interactive and responding to user requests, automated accounts often tweet by their own accord, to produce content for their followers or simply to become more visible.

Authorship – Because of Twitter's design, tweets in which the user mentions the bot account are visible to the bot's followers or to people searching for the bot's username. My review of more than 300 Twitter bot source codes and Twitter accounts found that almost all communication originating from Twitter bots is public and visible on the platform. Users can contribute to bot-published content in three ways: 1) users can post content that is picked up by the bot and either retweeted, quoted, or otherwise reused (e.g., by partially or fully re-posting it to the bot's timeline); 2) users can allow bots to access previous public posts by them (e.g., a bot can create a word cloud from the most frequently used words that a user has previously tweeted); 3) users can ask the bot for specific information or services (e.g., a user sends an image to a bot, whereupon the bot creates a deep dream version of the same image).

Considering the above dimensions, I have defined the following three levels of interactivity:

No interactivity: the bot generates content without mentioning any other Twitter

user. In this way, there is no interaction between the bot and other users on the platform, and the content generated by the bot is only visible to accounts that follow the bot or to users searching for a specific keyword or hashtag that matches the content. Also, the timing and content of the tweets produced by the bot do not change when a user contacts the bot.

One-way interactivity: the bot mentions other users as it generates posts, or reposts content, but it does not respond directly to communications from users who write to (or mention) the bot. Depending on the user's notification settings, a user may be notified of their mention on Twitter. This usually happens when the bot retweets or quotes a user's tweet, likes a tweet posted by someone else, or mentions a user in their tweet. In this case, there is indeed an interaction between the bot and the user, but it is a one-way communication - the bot does not respond to a message or mention from a user.

Two-way interactivity: the bot account can respond to some form of direct communication with a Twitter user, and in response to a direct message or an at mention, the bot generates tailor-made content. This is a two-way, albeit sometimes limited, communication (cf. chatbots that can maintain longer exchange of messages). It is important that users begin communication with the bot with the expectation of receiving a response. In other words, bot accounts that automatically respond to content posted by a user (e.g., when the bot is triggered by a keyword or hashtag) but do not respond to the same user's message, or at least not within a short period of time (e.g., within a day), are not two-way interactive. Table 6 provides examples of the different levels of interactivity defined above.

Table 6 *Examples for Various Levels of Interactivity*

Interactivity	Bot handle	Description
Not interactive	@rainbow_a_day	A simple Twitter bot that only tweets colorful emojis. It generates a tweet from six, randomly selected emojis to form a rainbow.
	@nswbushfires	This bot puts any new bush fires on a Google map and tweets about them based on the RSS feed of the Rural Fire Service of New South Wales, Australia.
One-way	@Its_Cold_Bo	This bot uses the Twitter Search API to find tweets where users complain about cold weather. Then, the bot selects a random tweet, and it generates a short tweet based on the report and at mention the original user as a source.
	@DoggoTheBot	The bot identifies sad tweets based on a sentiment analysis and sends a cute puppy picture in a reply tweet to the users who tweeted.
Two-way	@paranoiabot1	The bot responds to users based on a trained AI. It can also take videos from YouTube based on a query from a user and play a video backwards to find "clues" that can "support" conspiracy theories.
	@domaincheckbot	This bot can react to at mentions (mentioning its user name in a tweet) and look up any domains included via the *Domains API* of GoDaddy, a popular domain registration and website hosting company.

The distinction between one-way and two-way interactivity is not always straightforward, and there are indeed borderline cases. The bot @puppersplz, for example, communicates only with its followers and every now and then selects a random follower who receives a picture with a cute dog. Even though following a bot is a form of permission for the bot to contact a user, it is still not a two-way communication because the message is not a direct

(and immediate) response.

Similarly, the account called @you_are_bot contacts users who make a grammar mistake and confuse the spelling of your and you're. The bot responds to each reply from the trolled user by retweeting the user's reply. In this case, retweeting does not generate custom content in response, but simply amplifies the message or changes the transparency of the communication. Just like amplifiers bots that retweet content based on the source of the tweet or a matching keyword or hashtag, this bot is only interactive in one direction.

The account @AttentiveBot advertises itself as “A Twitter bot that finds easily overlooked tweets.” This account sends its followers links to tweets from their own followers (followers of the user who fallows the bot) who tweet the least often - these are tweets that are easy to overlook according to the bot developer behind this bot. This bot calculates the most overlooked tweet posted recently by an account you follow and sends a link to that tweet every day. It is a subscription-based model. Instead of notifying the bot, users simply have to follow or unfollow it to receive or stop receiving messages. The messages are customized for each user. I have classified this as a two-way interaction account, but it responds to following or unfollowing rather than responding to a specific mention or message.

It is important to note that there is no difference in quality between bots based on their interactivity. While it's true that chatbots often require a certain level of sophistication, there are readily available solutions and numerous examples that can help create a simple interactive Twitter bot that responds to users. On the other hand, a Twitter bot can be quite complex and sophisticated, even if it is not designed to interact with other users at all.

The goal of creating a well-defined, simple categorization that can be applied to any Twitter bot is to quantify the different types of bots in our dataset. The results presented below are not representative of all open-source Twitter bots, but give an idea of the use of different levels of interaction in the creation of bots. Since the method for identifying open-source bots with a matching repository and a deployed bot on Twitter is transparent (I describe it in detail in my dissertation), I feel comfortable to include this statistics.

About one in three bot codes contain some level of automation, while the majority of bots (236 accounts) offer no interaction - these accounts simply tweet on their own and do not respond to content or direct communication from users. Within the interactive bots, 50 accounts provide one-way interaction, meaning these accounts communicate directly with other users, such as by mentioning them in a tweet, but they do not provide tailored content and are not capable of any form of two-way interaction. Another 49 bots contain elements

that make them capable of two-way interaction-these accounts respond to intentional communications from Twitter users and provide tailored content, e.g., users can request specific information via mentions or direct messages, and the bot responds with the requested information.

Table 7 *Overview of the Open-source Twitter Bot Typology*

Dimension	Data source	Function	Interactivity
Main question	Where does the content for bots come from?	What functions do the bots perform?	How bots interact with users?
Types	• Locally stored (static, dynamic) content • Code-generated content • Content from Twitter • Web source (API, scraping)	• Amplifier • Curator • Content mash-up • Transmitter • Content creator • Service bot	• No interaction • One-way interaction • Two-way interaction

The remaining sections are discussing the technical details of the open-source bots and the content they distribute, but they are not part of the typology.

4.4.5. Themes in open-source Twitter bots

In addition to creating a typology, I also looked for common themes in the development of open-source bots. The following sections provide an overview of two major themes in bot development - bots that provide access to information, with a focus on information, and bots that focus on art, often visual art, with less emphasis on the information itself. These are additional insights about open-source bots, but I chose not to include them in a bot typology because even though there are definitions for information and art, these categories are less precise and may have some overlap (cf. mutually exclusive categories).

The first issue, which also accounts for the largest proportion of open-source bots, is access to information. Luciano Floridi, known for his work on the ethics of information, has devoted a short introductory book to information. In his book, Floridi (2010) gives a general definition of information that is useful for defining information bots. According to Floridi, information consists of data that is well-formed and meaningful. Data are well-formed and have meaning if they follow the rules that govern "the chosen system, code or language" (syntax). To be meaningful, they must "comply with the meanings (semantics) of the chosen system, code, or language in question."

Automated accounts that focus on information rely on data that are well-formed and have meaning. Floridi also distinguishes between factual data that is true (information) and factual data that is either intentionally misleading or false (disinformation) or simply not true (misinformation). Based on a manual review of hundreds of open-source bots that rely on external data sources, bot developers also look for well-formed data sources that provide access to meaningful and factually correct data. On the other hand, the very same code can be used to disseminate factually incorrect information.

Empirical evidence from other research projects suggests that not all bot accounts are created to disseminate factually correct information. However, I have not found any open-source Twitter bot that has produced disinformation, that is, false information that is not factually correct and intentionally produced to mislead or manipulate people. It is quite possible that bot developers who produce bots that are used for manipulation or disinformation campaigns are less proud of their creations and do not publish the code. On the other hand, as I mentioned earlier, almost exactly the same bot code can be used to

spread factually correct and false or misleading information.

Floridi also describes the typical phases of information that start with discovering, designing or authoring the information (1 - occurrence), this information is then often disseminated in a network (2 - transmission), the next phase is usually some kind of organization or classification and storage of the information (3 - processing and management) and finally the use of the information for education, decision making, monitoring, etc. (4 - usage). The phases described above are useful to describe the activities of open-source Twitter bots. These automated accounts are not only involved in transmission, but can also take an active role in processing and managing the information.

I have deliberately avoided calling these accounts broadcasters, although many of them simply relay information collected from a source. Broadcasting does not take interactivity into account. Twitter bots can only provide information when contacted (e.g., by a direct message) or triggered by a keyword, hashtag or at mention.

The second theme is concept, visual or art. These automated accounts tweet text-based or visual content, including short poems, text that is not information, images, and graphs.

There are several definitions of art (for the difficulty of defining the term, see footnote). [9] Without going into the intricacies of defining art or classifying bots as art bots, I have three important reasons to talk about art bots. 1) Bot developers regularly refer to the content generated by their bots as literature, digital art, or computer-generated art. Literature usually means that the bot is text-based, and the latter categories are most often used for visual bots. 2) Bot accounts often use artwork as input (free images from a museum) or use artistic methods to produce content (e.g., writing haikus). For the procedural definition of art, see footnote). I want to distinguish between bots that produce/acquire content that meets the definition of information and bots where the form of a tweet is more important than the information conveyed.

It is important to note that there are bots where both the content and the information

[9] Davies distinguish between functionalist and procedural approaches to defining art (Davies 1991, 2010). The functionalist approach, according to Davies claims that art has a function, usually providing a "rewarding aesthetic experience." However, not all art works are centered around traditional aesthetics or beauty, or made to provide a pleasant impression at all. Proceduralists believes that art is created by following certain rules and procedures (Davies, 1991). In another piece, Davies explain that the function of art changes over time (Davies, 2010). There are less-theoretical definition, for example Davies in his chapter on the definition of art cites George Dickie who defined art as an any kind of artifacts that was created and presented for the public interested in art.

are important. For example, in a background conversation, a bot developer who created a bot that tweets the schedule of Barack Obama, the 44th president of the United States, intentionally used the colors and style that has been used by the White House for a long time for official communications. Still, the information itself is important. Even if the format is visual (textual information on a blue and white background about the former U.S. president's schedule), the bot would fall under the information category. In other cases, the format is more important than the content: a bot that translates the probability of rain and other information from a weather report into a series of emoticons is an art/visualization bot. I admit that there are borderline cases or hard-to-decide examples.

In our sample of 321 labeled open-source bots, 118 accounts were created to provide access to information. Of these, 19 bots are two-way interactive and provide information after a user contacts the bot, meaning the user can ask the bot questions or request specific information. Another third of the bots fall into the art category - of the 321 tagged open-source bots, 97 focus on visual (i.e., images or moving pictures) or text-based art.

4.4.6. Twitter functions used by the bot

The Twitter API is defined by Twitter as a "set of programmatic tools that can be used to learn from and engage with the conversation on Twitter." To systematically examine and compare Twitter bots, I looked up the code published on GitHub under the repositories linked to the open-source Twitter bots and listed the Twitter functions implemented in the code.

Technically, there are at least four different ways to automate Twitter communication. The bots can access almost all of Twitter's basic functions, such as tweeting or posting an image, directly (type post or get) via the Twitter API request. See the documentation for the various Twitter APIs for detailed information on how to formulate these requests and how to make sense of the responses provided by the API.

The same API functions can be accessed through readily available open-source packages and libraries. These libraries are often referred to as API wrappers, because the software wraps the API and often provides a much easier way to interact with Twitter. This can reduce the time and knowledge required to develop a Twitter bot. In addition, these API wrappers are usually well maintained and help maintain the code even if the syntax for accessing the API or the structure of the API response changes.

The third way to access Twitter's features is to use a third-party tool that can

authenticate and interact with Twitter APIs. For example, the popular service If This Than That (IFTTT) can be set to post content to Twitter based on simple rules. IFTTT has its own API for programmatic access to the service. Developers can thus rely on IFTTT or a similar service to post content to Twitter or use other features of the microblogging platform.

Finally, developers can simulate a browser and access Twitter's website by pretending to be a real user who would access the platform through a browser. This is often referred to as headless browsing and can be used to overcome the limitations and restrictions of the Twitter API. For example, only the 1200 most recent tweets are available through the API for a Twitter account. However, this limitation does not apply when the same user profile is accessed through twitter.com in a browser. Headless browsers are reportedly used to create automated bots (Gorwa and Guilbeault, 2020; Suchacka & Iwański, 2020).

Most open-source Twitter bots access Twitter's functionalities either directly through the API or through an API wrapper. Both types of programmatic access to Twitter are well documented and easy to find in the code published on GitHub. The following table provides a list of the most commonly used Twitter functions based on the 321 bot source codes examined on GitHub and on Twitter.

Table 8 *The Most Used Twitter Functions by Automated Accounts*

Function	Nr of bots
Tweet	290
Image	66
Reply	40
Retweet	35
Mention	33
Link	25
Search	18
DM	10
Follow	9
Timeline	9
Video	8
Emoji	6
Stream	5
Quote	4
Unfollow	3
Trend	2
GIF	2

The vast majority of Twitter bots studied are set to post content to Twitter in either text and/or audiovisual format. Text is the most common format, with 290 bots posting tweets that contain text, while 66 bots can post images, 8 post videos, and 2 post animated GIFs.

A smaller portion of Twitter bots are set to reuse, amplify, or comment on content from Twitter. These bots use the retweet (35) and quote (4) functions. To find the right content, bots search for tweets (18), access the timelines of specific users (9), and use Twitter's streaming API (5) - the latter feature allows bots to “listen” for filtered content (based on filter keywords, hashtags, or at mentions) and immediately respond to content posted on Twitter (e.g., retweet, reply to a user, etc.).

While retweeted and quoted tweets automatically mention the original user and link to the original tweet respectively, Twitter bots can also mention specific users in their tweets or listen for specific at mentions. This helps either link content to specific users or create interactive bots that can respond to users when they mention the bot in a tweet. Of the 321 Twitter bots, 33 (about 10 percent) can recognize or include at mentions in tweets.

4.4.7. Examples for political bots

The collection of political bot repositories on GitHub is neither representative nor complete enough to serve as the basis for a typology of Twitter bots. Nonetheless, the political bot codes on GitHub show a wide range of functionalities and different purposes, from providing useful information about upcoming elections to accessing large but often difficult-to-search public databases to satirical bots that poke fun at political issues.

There are numerous examples of bots that access public data and turn that information into a regularly updated stream of tweets. One of the political bots on GitHub tweets the wait time at the U.S.-Mexico border in San Ysidro-Tijuana based on data retrieved from the California Institute for Telecommunications and Information Technology website at the University of California. Other examples include bots that search public databases of the Environmental Protection Agency in the United Kingdom and the U.S. Geological Survey and compose tweets on environmental topics.

A larger group of bots were programmed to provide information about elections. In 2014, a user named Schwad wrote a bot that queried live election results and posted regular updates on Twitter during the House of Representatives elections in Montana. Another election bot was created to use the Election API provided by Google to respond to direct

messages about the location of polling places. During the U.S. presidential election, code was released for two bots that tweeted information about Republican and Democratic campaign contributions based on a government-maintained database.

Another bot type can be referred to as awareness bots: These automated accounts draw the attention of the Twitter community to specific public issues, such as police violence, racism, and feminism.

One simple but intelligent Twitter bot uses an existing public database of police-related civilian deaths and shows the location where the death occurred by posting a Google Street View image. 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, documenting certain online activities. On GitHub, you can find code for Wikiedit bots that tweet alerts when someone edits a page on Wikipedia from an IP address associated with the government. Another version of this bot tweets every time a student from an American college edits a wiki page about another, rival college.

Finally, many satirical bot codes are published on GitHub. Donald Trump, for example, has inspired several parody bots. One of these bot codes is called “Donaldo Trumpilini” and mixes the text of tweets from the (now defunct) 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 describes it in derogatory terms to parody Donald Trump's style.

4.4.8. Most important research findings

In this chapter, Twitter bots from four different angels were examined. Most of the results are directly related to RQ 2.1, as they discuss how open-source bots generate, process, and publish content. After identifying common themes in the content published by Twitter bots, the chapter first examines interactivity and then how open-source Twitter bots produce content. These sections serve to create a typology, and whenever possible, I quantified these results.

The typology I created is based on three dimensions: 1) the source of information and how the data is accessed and processed, 2) the level of interactivity built into the bot, 3) the list of Twitter features used by the bot. In other words, the bot is described by what Twitter features are included in the bot code, how the bot produces the content, and whether users

have control over the content or the timing of the bot's tweet behavior.

First, I identified three main sources of content used by the bot. Open-source Twitter bots use content published by other Twitter users (38 bots) by retweeting their tweets, algorithmic content generated by code written for the bot (134 bots). Other bots collect content from other websites or platforms, including other social networks or social media sites (39 bots access their content through APIs). Another 20 bots simply rely on (locally stored) pre-selected or pre-written content - examples include bots built to tweet images from a folder/directory or lines of text from a large text file.

Next, I examined how much control users have over the content posted by a bot, whether a bot attempts to communicate with other Twitter users, and whether the bot is able to respond to user communications. While the majority of bots (236 bots) cannot directly engage with users or respond to users' communications, 50 bots are capable of one-way interactions (typically mentioning users in their posts) and another 49 bots can respond to users' communications and provide tailored content (two-way interaction) in response.

Lastly, 290 bots can post tweets using Twitter's API, while visual communication was much less common as a feature - this includes posting images (60 bots), video files (8 bots), and animated GIF files (2 bots). Consistent with the findings about interactivity, 40 open-source Twitter bots can reply to tweets and 33 can mention other users.

5. Discussion

5.1. Reflections on the findings

For this thesis, I collected data from the following three different sources: Repository data from GitHub, survey data from developers, and Twitter data. At each new step of the data collection process, the previously collected data was used, often in a processed and analyzed form. For example, relevant Twitter bot profiles were selected based on survey responses and the references (or links) provided in the repositories. Although I chose to structure the results around a particular data source (or step in the data collection process), comparing, contrasting, or amplifying some of the results across different data sources can make the results more robust.

In my dissertation, I first investigated open-source bot development practices by examining metadata about bot codes shared on GitHub, the largest online code repository. After analyzing the available metadata about more than 19,000 Twitter bot repositories, I contacted developers who set their email address public on GitHub and asked about their motivations for developing a bot for Twitter and using the platform. In the same survey, I also asked questions about the challenges of developing and deploying a bot on Twitter and the ways in which they acquired the skills necessary to develop a bot. A large number of bot developers shared the Twitter username of their deployed bots - this gave me a unique opportunity to examine the source code of open-source Twitter bots along with the activity of deployed bots on Twitter.

Much of this thesis is concerned with developing a methodology for investigating the development of open-source software on GitHub, so I think it is important to explain the rationale for contacting developers directly. Working with digital platform data has its own limitations. There are certain questions that you cannot ask based on this type of the data. In the case of GitHub repository data, the data contains very little information about the motivation behind a particular project, the challenges during development, or the communication related to the project outside of GitHub. On the other hand, answers to direct questions about the challenges during development based on surveys or interviews combine well with automatically collected metadata about the project.

Another example of comparing and contrasting different results based on different data sources is the complexity of source codes published on GitHub and the level of skills required to develop or implement such bot codes. From the repositories analyzed, it

appears that Twitter bots can be programmed to perform a variety of functions and that is relatively easy to repurpose or adapt these bots to perform new tasks on Twitter. It can be argued that individuals and groups who do not have high-level programming skills or the means to pay for expensive IT services can easily use bots on Twitter, suggesting a democratization of these powerful social media tools. On the other hand, the survey results suggest that the vast majority of bot developers are programmers, as nearly 78 percent of bot developers have formal training in programming or computer science. This suggests that while developing a bot is not a super complex task, writing code and understanding other people's code still requires some level of programming knowledge.

The results about the programming background of bot developers bring us to the limitations of this study. On the one hand, we can conclude that the majority of the developers are well educated and have a professional programming background. This could be related to the fact that GitHub is mainly used by professional programmers or students of computer science or in other IT fields. On the other hand, we can assume that big players, such as marketing companies and political groups, do not necessarily publish their source code on GitHub.

GitHub itself is changing according to the literature and to internal studies within GitHub. For example, the 2020 Octoverse report suggests that over the past 5 years, the share of developers in the platform's user-base has decreased from nearly 60 percent to 54 percent, while the share of education, for example, has increased from 17 percent to 23 percent (Forsgren, 2020).

The results presented here suggest that the overwhelming majority of bot code on GitHub was developed by a single author in a relatively short period of time. More than 40% of bot repositories were one-day projects, or in some cases these bots were developed outside of GitHub and the author only uploaded the final code to the platform (using GitHub as a code dump site). This suggests that GitHub is not being used to bring together different skill sets, connect developers from different geographic locations, or help existing teams share tasks, compared to more complex software projects. Moreover, only 1 in 10 bot repositories was developed by multiple authors using the site's collaboration and version tracking features. Nevertheless, GitHub users can learn from each other by looking at how a particular bot-related problem was solved by other developers.

Interestingly, projects developed by more than one developer are maintained or developed longer, and these projects receive more engagement on average, such as stars from other GitHub users. Similarly, about 9 out of 10 bot repositories developed by one

author were not forked on GitHub – meaning that these repositories were not copied to enable further work on the code by using the platform. On the other hand, half of the projects developed by multiple authors had at least one fork. Single-author projects received about 1 star on average, while multi-author repositories received 9 stars on average. Finally, single-author projects were maintained for an average 134 days, while repositories developed by more than one author were maintained for 428 days. This could be explained by both the complexity of the code (more complex codes can break more easily) and the increased attention paid to these repositories.

Bot developers on GitHub, on the other hand, take advantage of the social features of the site: the average developer has 30 followers and follows 21 developers on the platform. Of course, there are developers who are more connected and there are developers who are less socially engaged – one in four developers has no followers at all, for example. Still, bot source codes uploaded to the site by less-connected developers can be easily found by other developers by searching for a keyword on the GitHub site and serve as a model for other projects.

The typical bot developer is young (the vast majority of developers were 20-29 years old) has a degree in computer science, or is studying programming. The typical bot developer is male, just like the majority of the developers on the platform. Only 3.4 percent of bot developers identified herself as a woman. As I mentioned earlier in this thesis, the 2017 GitHub Open Source Survey had almost exactly the same gender distribution, as the percentage of woman in their data was 3.36 (Geiger, 2017). The age distribution of respondents was also very similar.

In addition to programming classes in school and looking up bot code on GitHub, the typical developer consults multiple sources when developing a bot for Twitter. The main sources of information were Q&A (question and answer) sites (e.g., a solution to a specific problem) and blog posts (e.g., a step-by-step description of how to develop a bot). The fact that they prefer to solve their problems individually rather than turning to the developer community on GitHub is partly related to the structure of GitHub. Although the platform is designed to encourage collaboration and teamwork, individual developers for one-person projects do not have good opportunities to get input or help from other developers. The survey of bot developer on GitHub also found that bot developers often do not use the platform to contribute to other people's repositories or reach out to developers. Using the site's version control system, for example, is much more important.

The survey results also suggest that bot developers use GitHub to gain exposure and

build their careers. Especially for young developers, recognition and career building are among the top motivations for using the platform. Aside from having a good idea and implementing it, creating a bot seems to be a learning project for many developers. When I asked developers about the most important reason for creating a bot, the majority of survey respondents indicated that learning how the APIs work was a very important or important reason for starting a bot project. Interestingly, self-expression was much less important, and the vast majority rejected the idea that they were developing the bot to support a political cause. From background conversations with developers, it was clear that developers often post bot codes to their GitHub profiles to show that they have personal projects and are interested in programming outside of work.

Using the Twitter usernames provided by survey participants and the bot repositories owned by developers on GitHub, I was able to compile a list of 321 paired bot repositories. By querying each Twitter account via Twitter's REST API, I was able to download and analyze nearly 500,000 tweets. This methodological step was important, because it allowed me to connect repositories identified on GitHub with the deployed bots on GitHub. In this way, I was able to not only examine the traces of the bots on Twitter but also understand the exact working mechanism of a particular bot.

The registration time for bot accounts on Twitter and the estimated distribution (see Chapter 4.3.1.) of traffic generated by such accounts coincide with the creation of bot repositories on GitHub. The changes in the number of bot repositories on GitHub and the bot traffic studied on Twitter both suggest that there has been a dynamic expansion of bot activity over the past 6 years. This also suggests that the survey respondents represent the entirety of open-source bot developer quite well.

Although Twitter data has severe limitations, I was able to overcome the lack of access to older tweets for high-frequency tweeting profile in my dissertation work. By modelling the temporal distribution of tweets, I was able to provide estimations about the volume of traffic generated by a sample of open-source bots. This is a methodological novelty and can be used in similar settings for analyzing the contribution from high-frequency tweeting accounts.

The vast majority of content posted to Twitter by the open-source bots studied is original content as opposed to retweets or quotes of existing content. This is one of the most interesting findings of the study of bot activity on Twitter. Although the review of bot repositories revealed bot accounts that either only amplify other accounts (retweeting them automatically) or curate content posted on Twitter (e.g., only retweet content that contains

certain keywords and has certain popularity level), most accounts would fall into to the generative and transmitter bots categories. While generative bots generate their content based on code (e.g., images rendered based on an algorithm), transmitters use widely available web APIs to access content from other platforms, and then post the information (or content) to Twitter.

The literature, including my own previous research at the Oxford Internet Institute, has discussed the use of bots in political communication. However, the sample of open-source Twitter bots was not designed to study political bots. This is both an important limitation of my study (see Limitations), and an interesting new challenge for future research (see Directions for future research). Nonetheless, studying non-political bots, including the methods of development and some information about the authors behind the bots, may be relevant to understanding the how political bots are created. Bot authors gain important skills when working with the Twitter API, so they can later decide to write a political bot or be hired to do so. Secondly, software development often relies on and builds upon existing code. Therefore, the non-political or general-purpose Twitter bot codes available on GitHub today can serve as the foundation for future political bots. Moreover, features developed for non-political bots can easily be repurposed for political purposes.

5.2. Generalizability and limitations of findings

When investigating digital platforms, and especially when working with API-based data collection, it is important to both report on the platform-specific limitations and think critically about the data collected. Due to the cross-platform data collection strategy of my research, the results presented in this thesis have a complex set of limitations. These limitations can be attributed to at least three different sources: 1) each platform has its own data collection logic, both GitHub and Twitter limit the range and scope of data that can be accessed through the free and public APIs; 2) on GitHub developers can also restrict access to certain data (e.g., their email address or location); and 3) the researcher also has to make certain decision during the data collection process that can further limit the generalizability of the results and, in worst case, compromise the internal validity of the research. Therefore, in the following section, I will reflect and think about the process of data collection. While reporting the details of the data collection itself is important for understanding the data and for reproducibility, discussing the limitations can create a more robust research and help with the understanding the generalizability of the findings. In addition, reporting the

limitations opens the way for suggestions for future research and new ways to study a problem.

PLATFORM SELECTION. These limitations stem from the fact that my thesis focuses only on developers who publish Twitter bot codes on GitHub. By only examining bots that have an accessible GitHub repository, developers that do not use GitHub are not part of this study. This was a reasonable decision on my part, as GitHub is not only the most used code sharing platform, but bot developers prefer GitHub compared to other platforms, and significantly less bot code is published on other platforms (see Assenmacher et al., 2020). Similarly, the decision to focus exclusively on Twitter bots limits the generalizability of the results.

OPEN-SOURCE CODE. It is important to note that my research focuses on open-source bots. The results of the thesis cannot be fully applied to bots that are either based on closed source code or use a third party social media content management tool. On the one hand, the study investigated how to automate social media communication, more specifically, activities on Twitter. The technical solutions and logic of automation can be used in any kind of bot. On the other hand, it can be argued that certain activities that are controversial, unethical, or against the Twitter's terms of services are not published on GitHub. For example, bots used in information manipulation campaigns or bots developed by black hat marketing companies are not available on GitHub.

SEARCH TERMS. GitHub had more than 56 million registered developers, and the number of repositories on the platform had reached 100 million by the end of 2020 (GitHub, 2020). Only repositories containing the term Twitter bot or twitterbot (written in two words and one word) are part of my study. This method resulted in very few false positives (i.e. repositories that are not Twitter bots), while making it easy to explain the search criteria. However, this decision could exclude a large number of bot repositories, something that I tried to address in my initial data collection in 2016. I collected smaller samples of datasets based on other potentially relevant keywords, such as automated social media and other combinations of relevant words, but the results almost every time included either Twitter and bot or twitterbot either in the name or short description of the repository or in the Read.me file.

LANGUAGE. Twitter bot (or Twitterbot), the combination of the words Twitter and robot, is an English term, and it is possible that some languages use local terms for Twitter bots. Although the language of the repository Read.me files and the geographic information in the dataset suggest that this general search term could identify Twitter bots from a large number of countries, identifying other terms for Twitter bots in different languages, especially in the case of languages that use a non-Latin alphabet, could further expand the scope of this study.

API LIMITATIONS. Although GitHub has a fairly open API policy, the platform still places certain limits on the data that can be accessed. For example, the Search API can only return a maximum of 1,000 results for each search. One strategy to deal with this limitation is to use narrower search terms or parameters. By limiting the search to a specific type of repositories (e.g., only repositories with a Python code) – a combination of such search results can overcome the 1,000 result limitation. Another strategy is to use time windows, and collect repositories in smaller batches. The GitHub Search API does have a limit on the number of queries per minute, but this can be easily worked around programmatically.

Twitter, on the other hand, has a variety of limitations that are not easy to overcome. Both the academic literature and the API documentation on Twitter's website describe these limitations in great detail. For example, the number of tweets per account that can be accessed is 3,600. Bot accounts are often programmed to produce content in large numbers, some communication from these high-frequency tweeting accounts were not available through the Twitter API. To mitigate this limitation, I used the total number of tweets per account and the time of registration to reconstruct the distribution of the traffic generated by such accounts.

DELETED OR PRIVATE CONTENT. When working with platform data, it is often important to consider the temporal dimension of data availability. In the case of both GitHub and Twitter, deleted data is not accessible through the API. GitHub, on the other hand, provides access to any changes made to source code or documents, provided the user uses version control and upload commits to the system. Content can be deleted (or changed from public to private) by the user on both platforms. GitHub allows its users to both use public and private repositories to work on projects. Since 2019, GitHub's free-tier users have unlimited private repositories (Friedman, 2019), and we can assume that, especially in the recent years, many bots have been developed in a private repository and never made

public. The reliance on the public GitHub API means that bot codes where the bot developer decides to use a private repository to work on a bot code are not part of this study.

REMOVED CONTENT. GitHub and Twitter have a set of rules about how to behave on the platform and how to use their APIs. Moderators and algorithms on both platforms remove content. GitHub has only a loose community guideline about behavior that is "not tolerated" on the platform, including harassment, incitement to violence, hate speech, doxing, and so forth. Twitter, on the other hand, has more stringent regulations in place, and its algorithms regularly monitor (and police) the platform and remove content that violates the terms of service (ToS). Deleted or removed content is not part of my analysis unless the content was removed or deleted after the data collection was complete.

PLATFORM USE. In addition to examining the API and retrieved data, the literature points to other limitations when working with GitHub data. For example, Kalliamvakou et al. (2016) note that repositories on GitHub are often "used for purposes other than strictly software development." For example, the authors report repositories that were used solely for archiving project data and did not contain a single line of code. In addition, GitHub can be used to store notes, host CV-s, and so on. By manually reviewing the repositories, I have found a limited number of projects where researchers or companies uploaded algorithms to detect bots.

STRUCTURE OF THE PLATFORM. While developing the data collection strategy for this thesis, I decided not to include forks to my GitHub dataset. This decision was made to avoid serious bias in the dataset from a few heavily forked repositories. One of the most popular Twitter bots, a bot repository by Randy Olson, has 415 forks. The number of contributors to the original project is only 13. Hence it has more than 400 forks by developers who have not contributed to the original repository - this would have distorted the dataset. Also, it did not seem like a good idea to contact developers who did not develop their own bots (only forked one). On the other hand, a separate analysis of all forked repositories and their forked versions would be a great addition to my research project and a possible new research project.

CONTACT INFORMATION. It is also important to note that the limited availability of email addresses could skew data collection, especially if certain types of users tend to add

an email address to their profile. For example, users developing politically sensitive software, especially in a repressive regime, might choose to remain anonymous, even though there are email services that provide (some degree of) anonymity. Also, certain ways of using GitHub might reduce the likelihood an email address being made public. For example, one could imagine that users who use the platform more as a code dump (a place where they keep a records of past projects), rather than actively using the social coding and version tracking features and/or collaborating with other users, might be less likely to publish an email address. Finally, the platform has changed the way it handles personal data and privacy in the past, and this could affect the rate of developers publishing their email address. For example, a change from an opt-out policy on publishing email addresses to an opt-in policy, could result in developers who joined the platform later being less likely to publish their email address than earlier members.

5.3. Directions for future research

Compare and contrast. While the data on bot developers is interesting in its own right, it would be interesting to compare some of the results to baseline numbers. For example, one could compare the number of contributors or the time spent maintaining a repository to the "average" GitHub repository. Unfortunately, GitHub does not provide an easy way to create a random sample of all GitHub repositories. Each repository has a unique numeric ID - it can be a single number or a nine-digit number, but there is no available index for the numbers used. There is a project called GHTorrent (Gousios et al., 2014), which is an archive for data available through the GitHub API, maintained by Georgios Gousios, a computer scientist at TU Delft and a research engineer at Facebook. The dataset includes 18 TB data and can be used for research purposes. By creating a random sample of all archived GitHub data, many of the results from studying open-source bots on GitHub can be compared to the average of a small random sample of GitHub.[10] Using a similar approach, researchers examined computer music repositories (Burlet & Hindle, 2015). In this study, the authors selected 819 random repositories from GitHub and compared them to 819 repositories in which musicians used visual music-oriented languages to generate music. Results indicated, for example, that musicians used fewer commits in general and

[10] A possible extension of the GitHub data collection came up during the internal defense of the thesis. Dr. Zoltán Kmetty suggested to compare some of my own findings with a random sample of GitHub repositories.

updated their repositories more frequently on weekends, but the number of posts and the number of bug reports did not differ significantly between the two groups.

More data on question-and-answer (Q&A) sites. Survey responses indicate that Q&A sites are the most important source of information when it comes to developing bots (and likely in other areas of software development as well). More than 86 percent of respondents used these sites while developing their Twitter bot, much more than contacting another developer (27 percent) or searching for another GitHub bot repository (67 percent). So another extension of my thesis could be to analyze the problems associated with bot development and the proposed solutions available on Stack Overflow, the largest Q&A site. Stack Exchange, the company behind Stack Overflow and a number of similar websites, has a free and open API for accessing the data on its platforms. In addition, the platform introduced the Stack Overflow Creative Commons Data Dump, database dumps of public data published on the site, back in 2009. A year later, Stack Exchange also introduced a web-based data explorer, making it possible to write queries and search for specific data in that database. As of early December 2021, Stack Overflow had 16 million users and 33 million answers to 22 million queries. Searches for both “Twitter bot” and “twitterbot” on the website returned up to 500 results. Since the website only displays 500 results, the number of results could be much higher. In addition to the types of questions developers ask during development, it would be interesting to learn more about who the users are who answer these questions and what other questions they typically answer.

A review of bot repository descriptions identified a small number of political Twitter bots. Most of these political bots can be considered “good bots” because they are designed with the intent of providing useful information to the Twitter community, providing information about polling locations during election times, tweeting about election results, or translating complicated public databases into the format of easily digestible 140- or 280-character tweets. The small number of political bots suggests that studying the political use of bots requires a different methodology. One avenue might be a case study-based approach, similar to the studies of WikiEdit bots (Ford et al., 2016), in which the authors examined open-source bots that report edits made on Wikipedia from a government IP address. Another approach might be to examine datasets recorded during elections, referendums, and other political events of major significance (or monitor ongoing political events) and try to find bots that reference a code repository in their profile information.

On the other hand, the potential use of these bot codes to influence social media discussions for political purposes raises serious ethical questions. Transparency of bot

activity may be central to the inclusion of bots in human conversations in the online public sphere. Currently, Twitter does not provide a standardized way to distinguish between human and bot activities. Publishing the source code of Twitter bots on a code-sharing platform like GitHub and providing a link to the appropriate repository could be a great way to increase transparency, as we can gain insight into the process of automating social media-based communications.

6. References

Assenmacher, Dennis, Lena Clever, Lena Frischlich, Thorsten Quandt, Heike Trautmann, and Christian Grimme. 'Demystifying Social Bots: On the Intelligence of Automated Social Media Actors'. *Social Media + Society* 6, no. 3 (July 2020): 205630512093926. https://doi.org/10.1177/2056305120939264.

Babbie, Earl R. *The Practice of Social Research*. Fourteenth edition. Boston, MA: Cengage Learning, 2016.

Badawy, Adam, Emilio Ferrara, & Kristina Lerman. 'Analyzing the Digital Traces of Political Manipulation: The 2016 Russian Interference Twitter Campaign'. *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, August 2018, 258–65. https://doi.org/10.1109/ASONAM.2018.8508646.

Benkler, Yochai. *The Wealth of Networks: How Social Production Transforms Markets and Freedom*. New Haven [Conn.]: Yale University Press, 2006.

Berners-Lee, Tim. 'I Invented the Web. Here Are Three Things We Need to Change to Save It'. the Guardian, 12 March 2017. http://www.theguardian.com/technology/2017/mar/11/tim-berners-lee-web-inventor-save-internet.

Bastos, Marco T., & Dan Mercea. 'The Brexit Botnet and User-Generated Hyperpartisan News'. *Social Science Computer Review* 37, no. 1 (February 2019): 38–54. https://doi.org/10.1177/0894439317734157.

Begel, A., & Nagappan, N. (2007, September). Usage and perceptions of agile software development in an industrial context: An exploratory study. In First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007) (pp. 255-264). IEEE.

Bessi, Alessandro, & Emilio Ferrara. 'Social Bots Distort the 2016 US Presidential Election Online Discussion'. *First Monday* 21, no. 11–7 (2016). https://doi.org/10.5210/fm.v21i11.7090.

Bordewijk, Jan, & Ben Van Kaam. ‘Towards a New Classification of Tele-Information Services’. *McQuail’s Reader in Mass Communication Theory*, 1986, 113–24.

Borra, Erik, & Bernhard Rieder. ‘Programmed Method: Developing a Toolset for Capturing and Analyzing Tweets’. Edited by Dr Axel Bruns and Dr Katrin Weller. *Aslib Journal of Information Management* 66, no. 3 (19 May 2014): 262–78. https://doi.org/10.1108/AJIM-09-2013-0094.

Brachten, Florian, Milad Mirbabaie, Stefan Stieglitz, Olivia Berger, Sarah Bludau, & Kristina Schrickel. ‘Threat or Opportunity? - Examining Social Bots in Social Media Crisis Communication’. In *Australasian Conference on Information Systems 2018*. University of Technology, Sydney, 2018. https://doi.org/10.5130/acis2018.bo.

Bright, Jonathan, Scott Hale, Bharath Ganesh, Andrew Bulovsky, Helen Margetts, & Phil Howard. ‘Does Campaigning on Social Media Make a Difference? Evidence From Candidate Use of Twitter During the 2015 and 2017 U.K. Elections’. *Communication Research* 47, no. 7 (October 2020): 988–1009. https://doi.org/10.1177/0093650219872394.

Broniatowski, David A., Amelia M. Jamison, SiHua Qi, Lulwah AlKulaib, Tao Chen, Adrian Benton, Sandra C. Quinn, and Mark Dredze. ‘Weaponized Health Communication: Twitter Bots and Russian Trolls Amplify the Vaccine Debate’. *American Journal of Public Health* 108, no. 10 (October 2018): 1378–84. https://doi.org/10.2105/AJPH.2018.304567.

Bruns, Axel, & Jean Burgess. ‘Twitter Hashtags from Ad Hoc to Calculated Publics’. *Hashtag Publics: The Power and Politics of Discursive Networks*, 2015, 13–28.

Bruns, Axel, & Stefan Stieglitz. ‘Towards More Systematic *Twitter* Analysis: Metrics for Tweeting Activities’. *International Journal of Social Research Methodology* 16, no. 2 (March 2013): 91–108. https://doi.org/10.1080/13645579.2012.756095.

Caldarelli, Guido, Rocco De Nicola, Fabio Del Vigna, Marinella Petrocchi, & Fabio Saracco. ‘The Role of Bot Squads in the Political Propaganda on Twitter’. *Communications Physics* 3, no. 1 (December 2020): 81. https://doi.org/10.1038/s42005-020-0340-4.

Cashmore, Pete. ‘Twitter Zombies: 24% of Tweets Created by Bots’. *Mashable*, 6 August 2009. https://mashable.com/2009/08/06/twitter-bots/.

Chavoshi, Nikan, Hossein Hamooni, & Abdullah Mueen. ‘DeBot: Twitter Bot Detection via Warped Correlation’. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 817–22. Barcelona, Spain: IEEE, 2016. https://doi.org/10.1109/ICDM.2016.0096.

Chen, Zhouhan, & Devika Subramanian. ‘An Unsupervised Approach to Detect Spam Campaigns That Use Botnets on Twitter’. *ArXiv:1804.05232 [Cs]*, 14 April 2018. http://arxiv.org/abs/1804.05232.

Chen, Zhouhan, Rima S. Tanash, Richard Stoll, & Devika Subramanian. ‘Hunting Malicious Bots on Twitter: An Unsupervised Approach’. In *Social Informatics*, edited by Giovanni Luca Ciampaglia, Afra Mashhadi, & Taha Yasseri, 10540:501–10. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017. https://doi.org/10.1007/978-3-319-67256-4_40.

Chu, Zi, Steven Gianvecchio, Haining Wang, & Sushil Jajodia. ‘Who Is Tweeting on Twitter: Human, Bot, or Cyborg?’ In *Proceedings of the 26th Annual Computer Security Applications Conference on - ACSAC ’10*, 21. Austin, Texas: ACM Press, 2010. https://doi.org/10.1145/1920261.1920265.

Chu, Zi, Steven Gianvecchio, Haining Wang, & Sushil Jajodia. ‘Detecting Automation of Twitter Accounts: Are You a Human, Bot, or Cyborg?’ *IEEE Transactions on Dependable and Secure Computing* 9, no. 6 (November 2012): 811–24. https://doi.org/10.1109/TDSC.2012.75.

Clark, Eric M., Jake Ryland Williams, Chris A. Jones, Richard A. Galbraith, Christopher M. Danforth, & Peter Sheridan Dodds. ‘Sifting Robotic from Organic Text: A Natural Language Approach for Detecting Automation on Twitter’. *ArXiv:1505.04342 [Cs]*, 14 June 2016. http://arxiv.org/abs/1505.04342.

Corder, Gregory W, & Dale I Foreman. *Nonparametric Statistics for Non-Statisticians*. Hoboken, New Jersey: John Wiley & Sons, 2009.

Cresci, Stefano. ‘A Decade of Social Bot Detection’. *Communications of the ACM 63*, no. 10 (23 September 2020): 72–83. https://doi.org/10.1145/3409116.

Cresci, Stefano, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, & Maurizio Tesconi. ‘The Paradigm-Shift of Social Spambots: Evidence, Theories, and Tools for the Arms Race’. *Proceedings of the 26th International Conference on World Wide Web Companion - WWW ’17 Companion*, 2017, 963–72. https://doi.org/10.1145/3041021.3055135.

Dabbish, Laura, Colleen Stuart, Jason Tsay, & Jim Herbsleb. ‘Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository’. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work - CSCW ’12*, 1277. Seattle, Washington, USA: ACM Press, 2012. https://doi.org/10.1145/2145204.2145396.

Davies, Stephen. *Definitions of Art*. Cornell University Press, 1991.

Davies, Stephen. ‘Definitions of Art’. *The Routledge Companion to Aesthetics*, 2010, 169–79.

Davis, Clayton Allen, Onur Varol, Emilio Ferrara, Alessandro Flammini, & Filippo Menczer. ‘BotOrNot: A System to Evaluate Social Bots’. In *Proceedings of the 25th International Conference Companion on World Wide Web - WWW ’16 Companion*, 273–74. Montréal, Québec, Canada: ACM Press, 2016. https://doi.org/10.1145/2872518.2889302.

Diakopoulos, Nicholas. *Automating the News: How Algorithms Are Rewriting the Media*. Cambridge, Massachusetts: Harvard University Press, 2019.

Dubbin, Rob. ‘The Rise of Twitter Bots’. *The New Yorker*, 2013.

Edwards, Chad, Autumn Edwards, Patric R. Spence, & Ashleigh K. Shelton. ‘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 (April 2014): 372–76. https://doi.org/10.1016/j.chb.2013.08.013.

Efthimion, Phillip George, Scott Payne, & Nicholas Proferes. ‘Supervised Machine Learning Bot Detection Techniques to Identify Social Twitter Bots’ 1, no. 2 (2018): 71.

Emler, N., and Elizabeth Frazer. ‘Politics: The Education Effect’. *Oxford Review of Education* 25 (1999): 251–73. https://doi.org/10.1080/030549899104242.

Ferrara, Emilio, Onur Varol, Clayton Davis, Filippo Menczer, & Alessandro Flammini. ‘The Rise of Social Bots’. *Communications of the ACM* 59, no. 7 (24 June 2016): 96–104. https://doi.org/10.1145/2818717.

Ferrara, Emilio. ‘Disinformation and Social Bot Operations in the Run up to the 2017 French Presidential Election’, 2017, 33.

Forsgren, N., G., Cecarelli, D., Fordi, V., Gennarell, Y., Huang, & Zimmerman, T. (2020). 2020 State of the Octoverse: Empowering healthy communities. https://octoverse.github.com/.

Floridi, Luciano. *Information: A Very Short Introduction*. OUP Oxford, 2010.

Gaffney, Devin, & Cornelius Puschmann. ‘Data Collection on Twitter’. *Twitter and Society* 55 (2014): 67.

Geiger, R. Stuart. ‘Bot-Based Collective Blocklists in Twitter: The Counterpublic Moderation of Harassment in a Networked Public Space’. *Information, Communication & Society* 19, no. 6 (2 June 2016): 787–803. https://doi.org/10.1080/1369118X.2016.1153700.

Geiger, R. Stuart. ‘Summary Analysis of the 2017 GitHub Open Source Survey’. *ArXiv:1706.02777 [Cs]*, 2017. https://doi.org/10.17605/OSF.IO/ENRQ5.

Gilani, Zafar, Reza Farahbakhsh, Gareth Tyson, Liang Wang, & Jon Crowcroft. ‘Of Bots and Humans (on Twitter)’. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, 349–54. Sydney Australia: ACM, 2017. https://doi.org/10.1145/3110025.3110090.

Gilani, Zafar, Reza Farahbakhsh, Gareth Tyson, and Jon Crowcroft. ‘A Large-Scale Behavioural Analysis of Bots and Humans on Twitter’. *ACM Transactions on the Web* 13, no. 1 (20 February 2019): 1–23. https://doi.org/10.1145/3298789.

GitHub. ‘The State of the Octoverse’. The State of the Octoverse. Accessed 18 June 2021. https://octoverse.github.com/.

Gorwa, Robert, & Douglas Guilbeault. ‘Unpacking the Social Media Bot: A Typology to Guide Research and Policy’. *Policy & Internet* 12, no. 2 (June 2020): 225–48. https://doi.org/10.1002/poi3.184.

Gousios, Georgios, Bogdan Vasilescu, Alexander Serebrenik, & Andy Zaidman. ‘Lean GHTorrent: GitHub Data on Demand’. In *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*, 384–87. Hyderabad, India: ACM Press, 2014. https://doi.org/10.1145/2597073.2597126.

Gousios, Georgios. ‘The GHTorent Dataset and Tool Suite’. In *2013 10th Working Conference on Mining Software Repositories (MSR)*, 233–36. San Francisco, CA, USA: IEEE, 2013. https://doi.org/10.1109/MSR.2013.6624034.

Graham, Mark, Stefano De Sabbata, & Matthew A. Zook. ‘Towards a Study of Information Geographies: (Im)Mutable Augmentations and a Mapping of the Geographies of Information: Towards a Study of Information Geographies’. *Geo: Geography and Environment* 2, no. 1 (June 2015): 88–105. https://doi.org/10.1002/geo2.8.

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

Hamilton, Naomi. (2008, August). The A–Z of programming languages: Python. Techworld. Retrieved from https://www2.computerworld.com.au/article/255835/a-z_programming_languages_python/

Hao, Karen. ‘Nearly Half of Twitter Accounts Pushing to Reopen America May Be Bots’. *MIT Technology Review*, 21 May 2020.

Hashem, Ibrahim Abaker Targio, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, and Samee Ullah Khan. 'The Rise of "Big Data" on Cloud Computing: Review and Open Research Issues'. *Information Systems* 47 (January 2015): 98–115. https://doi.org/10.1016/j.is.2014.07.006.

Haustein, Stefanie, Timothy D Bowman, Benoît Macaluso, Cassidy R Sugimoto, & Vincent Larivière. 'Measuring Twitter Activity of ArXiv E-Prints and Published Papers', 3, 2014. https://doi.org/10.6084/m9.figshare.1041514.

Haustein, Stefanie, Timothy D. Bowman, Kim Holmberg, Andrew Tsou, Cassidy R. Sugimoto, and Vincent Larivière. 'Tweets as Impact Indicators: Examining the Implications of Automated "Bot" Accounts on Twitter: Tweets as Impact Indicators: Examining the Implications of Automated "Bot" Accounts on Twitter'. *Journal of the Association for Information Science and Technology* 67, no. 1 (January 2016): 232–38. https://doi.org/10.1002/asi.23456.

Hofeditz, Lennart, Christian Ehnis, Deborah Bunker, Florian Brachten, & Stefan Stieglitz. 'Meaningful Use of Social Bots? Possible Applications in Crisis Communication during Disasters', 17. Stockholm & Uppsala, Sweden, 2019.

Howard, Philip N., Gillian Bolsover, Bence Kollanyi, Samantha Bradshaw, and Lisa-Maria Neudert. "Junk news and bots during the US election: What were Michigan voters sharing over Twitter." *CompProp, OII, Data Memo* (2017).

Howard, Philip N. *Pax Technica: How the Internet of things may set us free or lock us up*. Yale University Press, 2015.

Howard, Philip N., Samuel Woolley, & Ryan Calo. 'Algorithms, Bots, and Political Communication in the US 2016 Election: The Challenge of Automated Political Communication for Election Law and Administration'. *Journal of Information Technology & Politics* 15, no. 2 (3 April 2018): 81–93. https://doi.org/10.1080/19331681.2018.1448735.

Hwang, Tim, Ian Pearce, & Max Nanis. 'Socialbots: Voices from the Fronts'. *Interactions* 19, no. 2 (March 2012): 38–45. https://doi.org/10.1145/2090150.2090161.

Jacobs, Kristof, & Niels Spierings. ‘A Populist Paradise? Examining Populists’ Twitter Adoption and Use’. *Information, Communication & Society* 22, no. 12 (15 October 2019): 1681–96. https://doi.org/10.1080/1369118X.2018.1449883.

Jacobson, Daniel, Dan Woods, & Gregory Brail. *APIs: A Strategy Guide*. Sebastopol, CA: O’Reilly, 2012.

Jensen, Jens F. ‘Interactivity: Tracking a New Concept in Media and Communication Studies’ 12 (1998): 20.

Jones, Bronwyn, & Rhianne Jones. ‘Public Service Chatbots: Automating Conversation with BBC News’. *Digital Journalism* 7, no. 8 (2019): 1032–53. https://doi.org/10.1080/21670811.2019.1609371.

Kalliamvakou, Eirini, Kelly Blincoe, Leif Singer, Daniel M German, & Daniela Damian. ‘The Promises and Perils of Mining GitHub (Extended Version)’, 39, 2014.

Kan, Michael. (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

Kaplan, Andreas M., & Michael Haenlein. ‘Users of the World, Unite! The Challenges and Opportunities of Social Media’. *Business Horizons* 53, no. 1 (January 2010): 59–68. https://doi.org/10.1016/j.bushor.2009.09.003.

Karataş, Arzum, & Serap Şahin. ‘A Review on Social Bot Detection Techniques and Research Directions’, 7. Turkey, 2017.

Kietzmann, Jan H., Kristopher Hermkens, Ian P. McCarthy, & Bruno S. Silvestre. ‘Social Media? Get Serious! Understanding the Functional Building Blocks of Social Media’. *Business Horizons* 54, no. 3 (May 2011): 241–51. https://doi.org/10.1016/j.bushor.2011.01.005.

Kollanyi, Bence. ‘Where Do Bots Come from? An Analysis of Bot Codes Shared on GitHub’. *International Journal of Communication* 10 (2016): 4932–51.

Kruskal, William H, & W Allen Wallis. ‘Use of Ranks in One-Criterion Variance Analysis’, 2021, 40.

Kudugunta, Sneha, & Emilio Ferrara. ‘Deep Neural Networks for Bot Detection’. *Information Sciences* 467 (October 2018): 312–22. https://doi.org/10.1016/j.ins.2018.08.019.

Kumar, Sangeet. ‘Richard Rogers: Digital Methods (Review)’. *International Journal of Communication* 8 (2014).

Lazer, D., A. Pentland, L. Adamic, S. Aral, A.-L. Barabasi, D. Brewer, N. Christakis, et al. ‘SOCIAL SCIENCE: Computational Social Science’. *Science* 323, no. 5915 (6 February 2009): 721–23. https://doi.org/10.1126/science.1167742.

Lazer, David M. J., Alex Pentland, Duncan J. Watts, Sinan Aral, Susan Athey, Noshir Contractor, Deen Freelon, et al. ‘Computational Social Science: Obstacles and Opportunities’. *Science* 369, no. 6507 (28 August 2020): 1060–62. https://doi.org/10.1126/science.aaz8170.

Lee, Kyumin, James Caverlee, & Steve Webb. ‘The Social Honeypot Project: Protecting Online Communities from Spammers’. In *Proceedings of the 19th International Conference on World Wide Web - WWW ’10*, 1139. Raleigh, North Carolina, USA: ACM Press, 2010. https://doi.org/10.1145/1772690.1772843.

Lee, Kyumin, Brian David Eoff, & James Caverlee. ‘Seven Months with the Devils: A Long-Term Study of Content Polluters on Twitter’. In *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*, 8, 2011.

Lima, Antonio, Luca Rossi, & Mirco Musolesi. ‘Coding Together at Scale: GitHub as a Collaborative Social Network’. In *Proceedings of the Eighth International AAAI Conference on Weblogs and Social Media*, 10, 2014.

Lin, Sharon. ‘Cleaning Up Your GitHub’. Medium, 30 September 2020. https://medium.com/@sharonlin/cleaning-up-your-github-fedaf9e7cef2.

Lin, Po-Ching, & Po-Min Huang. (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, Victor. (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/

Lister, Martin, Jon Dovey, Seth Giddings, Iain Grant, & Kieran Kelly. *New Media: A Critical Introduction*. 0 ed. Routledge, 2008. https://doi.org/10.4324/9780203884829.

Lokot, Tetyana, & Nicholas Diakopoulos. 'News Bots: Automating News and Information Dissemination on Twitter'. *Digital Journalism* 4, no. 6 (17 August 2016): 682–99. https://doi.org/10.1080/21670811.2015.1081822.

Luceri, Luca, Ashok Deb, Silvia Giordano, and Emilio Ferrara. 'Evolution of Bot and Human Behavior during Elections'. *First Monday*, 31 August 2019. https://doi.org/10.5210/fm.v24i9.10213.

Lumezanu, Cristian, Nick Feamster, & Hans Klein. '#bias: Measuring the Tweeting Behavior of Propagandists'. In *Proceedings of the 6th International AAAI Conference on Weblogs and Social Media*, 8. Dublin, Ireland, 2012.

Lunden, Ingrid. (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/

Manovich, Lev. *The Language of New Media*. MIT press, 2002.

Marres, Noortje. 'Foreword by Noortje Marres'. In Snee, Helene, Christine Hine, Yvette Morey, Steven Roberts, and Hayley Watson, eds. *Digital Methods for Social Science*. London: Palgrave Macmillan UK, 2016. https://doi.org/10.1057/9781137453662.

Maus, Gregory, & Onur Varol. 'A Typology of Socialbots', 2017, 8.

McNutt, Marcia. ‘Reproducibility’. *Science* 343, no. 6168 (17 January 2014): 229–229. https://doi.org/10.1126/science.1250475.

McQuail, Denis. *Mass Communication Theory: An Introduction*. 3rd ed. Sage Publications, Inc, 1994.

Metcalf, Julia. (2020, July). GitHub Archive Program: the journey of the world’s open source code to the Arctic. Retrieved from https://github.blog/2020-07-16-github-archive-program-the-journey-of-the-worlds-open-source-code-to-the-arctic/

Metaxas, Panagiotis Takis, & Eni Mustafaraj. ‘From Obscurity to Prominence in Minutes: Political Speech and Real-Time Search’, 7. Raleigh, NC, USA, 2010.

Minnich, Amanda, Nikan Chavoshi, Danai Koutra, & Abdullah Mueen. ‘BotWalk: Efficient Adaptive Exploration of Twitter Bot Networks’. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, 467–74. Sydney Australia: ACM, 2017. https://doi.org/10.1145/3110025.3110163.

Murthy, Dhiraj. *Social Communication in the Twitter Age (Digital Media and Society)*. Cambridge: Polity Press, 2013.

Mustafaraj, Eni, & Panagiotis Takis Metaxas. ‘What Edited Retweets Reveal about Online Political Discourse’. In *Conference: Analyzing Microtext, Papers from the 2011 AAAI Workshop*, 6. San Francisco, California, USA, 2011.

Nosek, B. A., G. Alter, G. C. Banks, D. Borsboom, S. D. Bowman, S. J. Breckler, S. Buck, et al. ‘Promoting an Open Research Culture’. *Science* 348, no. 6242 (26 June 2015): 1422–25. https://doi.org/10.1126/science.aab2374.

O’Reilly, Tim. ‘What Is Web 2.0’. Tim.oreilly.com, 2005.

Oentaryo, Richard J., Arinto Murdopo, Philips K. Prasetyo, & Ee-Peng Lim. ‘On Profiling Bots in Social Media’. In *Social Informatics*, edited by Emma Spiro and Yong-Yeol Ahn, 10046:92–109. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016. https://doi.org/10.1007/978-3-319-47880-7_6.

Pashler, Harold, & Eric–Jan Wagenmakers. ‘Editors’ Introduction to the Special Section on Replicability in Psychological Science: A Crisis of Confidence?’ *Perspectives on Psychological Science* 7, no. 6 (November 2012): 528–30. https://doi.org/10.1177/1745691612465253.

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

Purnell, Newley. (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/

Puschmann, Cornelius, & Jean Burgess. ‘The Politics of Twitter Data’. *SSRN Electronic Journal*, 2013. https://doi.org/10.2139/ssrn.2206225.

Ratkiewicz, J, M D Conover, M Meiss, B Goncalves, A Flammini, & F Menczer. ‘Detecting and Tracking Political Abuse in Social Media’. In *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*, 8, 2011.

Rebillard, Franck, & Annelise Touboul. ‘Promises Unfulfilled? “Journalism 2.0”, User Participation and Editorial Policy on Newspaper Websites’. *Media, Culture & Society* 32, no. 2 (March 2010): 323–34. https://doi.org/10.1177/0163443709356142.

Rogers, Everett M. *Communication Technology*. Simon and Schuster, 1986.

Rogers, Richard. *Digital Methods*. Cambridge, Massachusetts: The MIT Press, 2013.

Rogers. ‘Political Research in the Digital Age’. *International Public Policy Review* 8, no. 1 (2014): 73-87.

Roth, Yoel, & Nick Pickles. ‘Bot or Not? The Facts about Platform Manipulation on Twitter’. Accessed 29 April 2021. https://blog.twitter.com/en_us/topics/company/2020/bot-or-not.html.

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, Amirali, & Guevara Noubir. ‘On GitHub’s Programming Languages’. *ArXiv:1603.00431 [Cs]*, 1 March 2016. http://arxiv.org/abs/1603.00431.

Schultz, Tanjev. ‘Interactive Options in Online Journalism: A Content Analysis of 100 U.S. Newspapers’. *Journal of Computer-Mediated Communication* 5, no. 1 (23 June 2006): 0–0. https://doi.org/10.1111/j.1083-6101.1999.tb00331.x.

Seward, Zachary M. (2013, October 15). The first-ever hashtag, @-reply and retweet, as Twitter users invented them. Quartz. Retrieved December 14, 2020, from https://qz.com/135149/the-first-ever-hashtag-reply-and-retweet-as-twitter-users-invented-them/

Song, Jonghyuk, Sangho Lee, & Jong Kim. ‘Spam Filtering in Twitter Using Sender-Receiver Relationship’. In *Recent Advances in Intrusion Detection*, edited by Robin Sommer, Davide Balzarotti, & Gregor Maier, 6961:301–17. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. https://doi.org/10.1007/978-3-642-23644-0_16.

Spence, Scott. ‘Why You Should Have Your Own Twitter Bot, and How to Build One in Less than 30 Minutes’. freeCodeCamp.org, 28 January 2017. https://www.freecodecamp.org/news/easily-set-up-your-own-twitter-bot-4aeed5e61f7f/.

Spyridou, Paschalia-Lia, & Andreas Veglis. ‘Exploring Structural Interactivity in Online Newspapers: A Look at the Greek Web Landscape’. *First Monday*, 2008.

Steiner, Thomas. ‘Telling Breaking News Stories from Wikipedia with Social Multimedia’, 2014, 7.

Stieglitz, Stefan, Florian Brachten, Björn Ross, & Anna-Katharina Jung. ‘Do Social Bots Dream of Electric Sheep? A Categorisation of Social Media Bot Accounts’, 2017, 11.

Storey, Margaret-Anne, Leif Singer, Brendan Cleary, Fernando Figueira Filho, & Alexey Zagalsky. ‘The (R) Evolution of Social Media in Software Engineering’. In *Future of Software Engineering Proceedings*, 100–116. Hyderabad India: ACM, 2014. https://doi.org/10.1145/2593882.2593887.

Stukal, Denis, Sergey Sanovich, Richard Bonneau, & Joshua A. Tucker. 'Detecting Bots on Russian Political Twitter'. *Big Data* 5, no. 4 (December 2017): 310–24. https://doi.org/10.1089/big.2017.0038.

Suchacka, Grażyna, & Jacek Iwański. 'Identifying Legitimate Web Users and Bots with Different Traffic Profiles — an Information Bottleneck Approach'. *Knowledge-Based Systems* 197 (June 2020): 105875. https://doi.org/10.1016/j.knosys.2020.105875.

Sundar, S. Shyam. 'Theorizing Interactivity's Effects'. *The Information Society* 20, no. 5 (November 2004): 385–89. https://doi.org/10.1080/01972240490508072.

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, Yuri, & Andrew Hilts. 'Investigating the Geography of Open Source Software through GitHub', 2010, 10.

Tao, Y., Dang, Y., Xie, T., Zhang, D., & Kim, S. (2012, November). How do software engineers understand code changes? An exploratory study in industry. In Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering (pp. 1-11).

Tarte, S., Willcox, P., Glaser, H., & De Roure, D. (2015, June). Archetypal narratives in social machines: approaching sociality through prosopography. In Proceedings of the ACM web science conference (pp. 1-10).

Toffler, Alvin. *The Third Wave*. A Bantam Book. New York Toronto London Sydney Auckland: Bantam Books, 1990.

Tromble, Rebekah. 'Thanks for (Actually) Responding! How Citizen Demand Shapes Politicians' Interactive Practices on Twitter'. *New Media & Society* 20, no. 2 (February 2018): 676–97. https://doi.org/10.1177/1461444816669158.

Twitter. 'Developer Agreement and Policy – Twitter Developers'. Accessed 1 June 2020. https://developer.twitter.com/en/developer-terms/agreement-and-policy.

Varol, Onur, Emilio Ferrara, Clayton A Davis, Filippo Menczer, & Alessandro Flammini. 'Online Human-Bot Interactions: Detection, Estimation, and Characterization'. In *Proceedings of the Eleventh International AAAI Conference on Web and Social Media*, 10, 2017. https://arxiv.org/abs/1703.03107.

Vasilescu, Bogdan, Daryl Posnett, Baishakhi Ray, Mark G.J. van den Brand, Alexander Serebrenik, Premkumar Devanbu, & Vladimir Filkov. 'Gender and Tenure Diversity in GitHub Teams'. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 3789–98. Seoul Republic of Korea: ACM, 2015. https://doi.org/10.1145/2702123.2702549.

Wang, Alex Hai. 'Don't Follow Me: Spam Detection in Twitter'. In *2010 International Conference on Security and Cryptography (SECRYPT)*, 10. Athens, 2010.

Wanstrath, Chris. (2012, April 4). Spring Cleaning. The GitHub Blog. https://github.blog/2012-04-03-spring-cleaning/

Wojcik, Stefan, Solomon Messing, Aaron Smith, Lee Rainie, & Paul Hitlin. 'Bots in the Twittersphere'. Pew Research Center, April 2018.

Woolley, Samuel C., & Philip N. Howard. (2016). 'Political Communication, Computational Propaganda, and Autonomous Agents - Introduction', 2016, 9.

Woolley, Samuel C., & Philip N. Howard. (2016). Social media, revolution, and the rise of the political bot. *Routledge handbook of media, conflict, and security. New York, NY: Routledge*, 282-292.

Woolley, Samuel C. 'Automating Power: Social Bot Interference in Global Politics'. *First Monday*, 21, no. 4 (2016). https://doi.org/10.5210/fm.v21i4.6161.

Wu, Yu, Jessica Kropczynski, Patrick C. Shih, & John M. Carroll. 'Exploring the Ecosystem of Software Developers on GitHub and Other Platforms'. In *Proceedings of the Companion Publication of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing - CSCW Companion '14*, 265–68. Baltimore, Maryland, USA: ACM Press, 2014. https://doi.org/10.1145/2556420.2556483.

Zagalsky, Alexey, Joseph Feliciano, Margaret-Anne Storey, Yiyun Zhao, & Weiliang Wang. ‘The Emergence of GitHub as a Collaborative Platform for Education’. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, 1906–17. Vancouver BC Canada: ACM, 2015. https://doi.org/10.1145/2675133.2675284.

Zlotnick, F., Smith, A., Linksvayer, M., & Filippova, A. The Open Source Initiative. (2017). The open source survey website. GitHub, Inc. https://opensourcesurvey.org/2017/

7. List of relevant publications

7.1. Peer-reviewed journal articles

Marchal, Nahema, Lisa-Maria Neudert, Bence Kollanyi, and Philip N. Howard. 'Investigating Visual Content Shared over Twitter during the 2019 EU Parliamentary Election Campaign'. *Media and Communication* 9, no. 1 (3 February 2021): 158–70. https://doi.org/10.17645/mac.v9i1.3421.

Bradshaw, Samantha, Philip N. Howard, Bence Kollanyi, and Lisa-Maria Neudert. 'Sourcing and Automation of Political News and Information over Social Media in the United States, 2016-2018'. *Political Communication* 37, no. 2 (3 March 2020): 173–93. https://doi.org/10.1080/10584609.2019.1663322.

Neudert, Lisa-Maria, Philip Howard, and Bence Kollanyi. 'Sourcing and Automation of Political News and Information During Three European Elections'. *Social Media + Society* 5, no. 3 (July 2019): 205630511986314. https://doi.org/10.1177/2056305119863147.

Kollanyi, Bence. 'Where Do Bots Come from? An Analysis of Bot Codes Shared on GitHub'. *International Journal of Communication* 10 (2016): 4932–51.

7.2. Conference presentations and publications

Machado, C., Kira, B., Narayanan, V., Kollanyi, B., & Howard, P. (2019, May). A Study of Misinformation in WhatsApp groups with a focus on the Brazilian Presidential Elections. In Companion Proceedings of The 2019 World Wide Web Conference (WWW '19), Ling Liu and Ryen White (Eds.). ACM, New York, NY, USA.

Gorwa, R., Kollanyi, B., & Howard, P. (2018, May). A critical analysis of bot detection methodologies In: Methodological Challenges to Studying Misinformation and Disinformation in Data-Driven Politics: Fake News, Bots, and Digital Campaigns, Panel Discussion at the ICA 2018, Prague.

Kollanyi, B. (2016, June). How to write a Twitter bot? Bot Codes Shared on GitHub. Algorithms, Automation and Politics, Preconference of the ICA 2016, Fukuoka.

8. Appendix

GitHub Questionnaire

Using GitHub

1 How IMPORTANT are the following aspect of GitHub for you?
[Likert scale from Very important to Not important at all]

- Get recognition for your work - using public repositories to show your work to others
- Use version control - track changes in your own code
- Find inspiration - learn from others by looking in to their code
- Reach out to the community - find other programmers who can contribute to your work
- Contribute to other repositories - write code to someone else's repository
- Communicate with other developers - describe your own work, give feedback to others, answer questions
- Give back to the community - contribute to other programmers work (besides writing code)
- Build your career - find a job through the platform

Setting up a Twitter bot

2 Have you ever deployed or set-up a bot on Twitter?
(The bot can use your own code or any other tool or software.)
- Yes
- No

Your bots on Twitter

3.1. Have you ever done the following?
[Yes or No answer]
- Set-up a bot on Twitter based on your OWN bot code?
- Set up a bot on Twitter that was NOT DEVELOPED by you?
- Used a DEDICATED ACCOUNT for the bot (ie. An account primarily used by the bot)?
- DISCLOSED (communicated clearly) that some or all of the COMMUNICATION coming from this account is AUTOMATED?

3.2. Please add the Twitter handles / usernames of the bot accounts here.
(You can add it even if the bot account is inactive at the moment.)
[Open text field]

Attitudes toward bots

4.1. People have different thoughts on automated accounts on Twitter. How much do you agree with the following statements?
[Mostly agree / Somewhat agree / Somewhat disagree / Mostly disagree]

- It is harder and harder to distinguish between human and automated communication.
- Twitter bots are too often used to spread misinformation.
- The communication from bots should be protected by free speech.
- Bots are a way of self-expression.
- During an election campaign, candidates and other politicians should not use bots.

Developing a Twitter bot

5.1 Think about your own Twitter bot project(s)! Have you reached out to other developers or used online sources of information when you developed your code? Have you contacted/checked the following sources:
[Yes or No answer]

- Twitter terms of service, developer guidelines published by Twitter
- Other GitHub bot repositories checked for inspiration / finding a solution to a specific problem
- Contacted another developer by using GitHub (e.g. sending a direct message via the platform)
- Contacted another developer outside GitHub (e.g. messages over email, tweet, etc.)
- Checked any question and answer site, e.g. Stack Overflow, Quora, etc.
- Read a blog post or an article about developing Twitter bots
- Participated in a lecture (online / offline) that involved / discussed bot development
- Other:

5.2 When you think about your own Twitter bot code hosted on GitHub, how important were the following reasons to develop your bot?
[5 - Very important, 0 - Not important at all]

- Test an idea
- Show your skills as a developer
- Automate some tedious and boring tasks
- Support a cause / political agenda
- Practice writing code and working with API-s
- Self-expression or art
- Other reason:

5.3 If there were challenging aspects of developing a Twitter bot or deploying it over Twitter, please describe these challenges. You can mention multiple challenges.
[Open text field]

Twitter bots and politics

6.1. Have you or someone who you know ever used any of the bot codes you developed for political purposes, including civil issues, party politics, activism, etc.?

- Yes
- No

6.2. Would you mind if someone would use your bot code for political purposes?
- Yes
- No
- It depends on political issue

6.3 How interested would you say you are in politics in general?
[Very interested / Quite interested / Hardly interested / Not at all interested]

6.4 How often do you discuss politics and public affairs with others ONLINE – such as by writing or responding to a political Twitter post, writing or commenting on a political Facebook post?
Would you say:
Every day, At least once a week, At least once a month, Less than once a month, or Never?

Demography and education

7.1. To which gender identity do you most identify with?
Male
Female
Variant/Non-confirming
Not listed:
Prefer not to answer

7.2. When did you born?
Month / Year

7.3. What is your highest education?
No schooling completed / Less than high school degree
High school degree or equivalent
Completed vocational training
Bachelor's degree or equivalent
Master's degree or equivalent
Doctorate (PhD, DLA, etc.)

7.4. Do you have any formal education in computer science or programming?
Yes / No