Towards Understanding the Public Gists on GitHub

by

Weiliang Wang
B.Sc., Southeast University, 2013

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

Towards Understanding the Public Gists on GitHub

by

Weiliang Wang
B.Sc., Southeast University, 2013

Supervisory Committee

Dr. Daniel M. German, Supervisor
(Department of Computer Science)

Dr. M. Member One, Departmental Member
(Department of Same As Candidate)

Dr. Member Two, Departmental Member
(Department of Same As Candidate)

Dr. Outside Member, Outside Member
(Department of Not Same As Candidate)

Supervisory Committee

Dr. Daniel M. German, Supervisor
(Department of Computer Science)

Dr. M. Member One, Departmental Member
(Department of Same As Candidate)

Dr. Member Two, Departmental Member
(Department of Same As Candidate)

Dr. Outside Member, Outside Member
(Department of Not Same As Candidate)

ABSTRACT

GitHub is a popular source code hosting site which serves as a collaborative coding platform. The many features of GitHub have greatly facilitated developers' collaboration, communication, and coordination. One of these features is Gists, which is defined as "a simple way to share snippets and pastes with others". This report presents a three-part study which explores how users are using Gists. The first part is a quantitative analysis against the metadata of 562,993 GitHub users and 618,393 Gists, as well as the 793,891 files contained in those Gists. The second part investigates the actual Gist content by manually labeling 400 randomly selected Gists. The third part of the study investigates users' comments about how they used Gists from the searched results of Websites and Twitter postings. Our study shows that Gists are used by a small portion of GitHub users, and those that use them typically only have a few Gists. We found that Gists are usually small snippets of source code and composed of a single file. However, Gists serve a wide variety of uses, from saving snippets of code or notes, sharing files, to drafting their writings.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

my supervisor, **Dr. Daniel M. German**, for mentoring, encouragement, patience and understanding.

German, **Evan** and **Daniel** for making this study published.

my family, **Rongyou**, **Yin'ai**, **Meiling** and **Tony** for your selfless love and understanding about all of my decisions I made in my life.

my best friends, **Yingsi** and **Hanbin**, for keeping our friendship always fresh and making me feel you are still by my side even though we are already in different countries fighting for our future.

the companies, **Isolation Network** and **CopperLeaf Technologies**, for giving me a chance of experiencing the industrial world before graduation.

and **True Detective**, **House of Cards**, **Breaking Bad** etc. for expanding my boundaries of thinking about everything.

DEDICATION

I dedicate this report to my parents.

Chapter 1

Introduction

GitHub[1] is a popular source code hosting site. It enhanced the version control tool Git[2] by combining the features of both software engineering and social networking, including event feeds, pull requests, code reviews, and an issue tracking mechanism[3], which has greatly facilitate developers' collaboration, communication and coordination.

Gist is one of the many features of GitHub, which allows users to instantly share code, notes, and snippets. GitHub illustrates[4] Gist as follows:

"Gists are a great way to share your work. You can share single files, parts of files, or full applications...Every gist is a Git repository, which means that it can be forked, cloned, and manipulated in every way."

Since how a technology is supposed to be used might differ from how it is actually used, it would be interesting to know how Gist is actually being used, and if it's exactly what GitHub expects or the users are finding innovative ways. However, Gist has been paid little attention by researchers, and there's no existing Gists dataset available for research. Thus, a lot of questions about Gists remain unanswered. For example, how many GitHub users are using gists? Do users collaborate on Gists? What are the contents of Gists about? How large is a Gist?

A noteworthy trend is that lots of third-party applications are developed to support creation, management and sharing of Gists. For example, both *GistBox*[5] and

[1] `https://www.github.com`
[2] `https://en.wikipedia.org/wiki/Git_(software)`
[3] `https://github.com/features`
[4] `https://help.github.com/articles/about-gists/`
[5] `http://www.gistboxapp.com/`

Gisto[6] are applications that help users better organize their Gists by adding additional features such as searching, tagging and sharing Gists; *GistBox Clipper* allows users to create Gists from any web page; many blog sites like *WordPress*[7] support Gists embedding either by its URL or its ID[8]; many popular IDEs or text editors support Gists creation through plugins, such as Visual Studio[9], *IntelliJ IDEA*[10], *sublime text*[11], etc. All these threads imply that gists are gaining popularity among GitHub users. It's worthy to explore Gists and how developers are using them.

In this report, an empirical study of GitHub Gists is presented. With the goal of getting a picture of what Gists are like and how they are being used, the following two research questions were addressed:

- **RQ1. What do Gists look like?**

  Although gists are gaining popularity among GitHub users, the picture of what Gists are like is still not clear.

- **RQ2. How are Gists being used?**

  GitHub Gists was initially developed for users to write and share code snippets, but gists could be used in unexpected ways. Thus it's worthy to find out various ways of using Gists which could provide good recommendations and suggestions on how GitHub Gists could be better used.

To answer **RQ1**, both qualitative and quantitative analysis were performed around Gists contents and metadata. For the quantitative part, 750k GitHub users were sampled and 762k Gists from these users were identified. For the qualitative part of the study, a manual inspection was performed on 398 Gists randomly sampled from the above 762k Gists. To answer **RQ2**, we searched the Web for evidence of how users described their use of Gists, which included a search of Websites (including blog posts) and Twitter feeds. Then these information was combined with the result of **RQ1** to get a full picture of how Gists are being used.

[6] `http://www.gistoapp.com/`
[7] `http://www.wordpress.com/`
[8] `http://en.support.wordpress.com/gist/`
[9] `https://marketplace.visualstudio.com/items/dbankier.vscode-gist`
[10] `https://www.jetbrains.com/idea/help/creating-gists.html`
[11] `https://github.com/bgreenlee/sublime-github`

Chapter 2

Related Works

2.1 Version Control History and GitHub

For the large and complex software projects, it is a challenge to well coordinate the collaboration between all the developers and to manage each developer's contribution. Based on this demand, Version Control System (VCS) was developed to record changes to a file or set of files over time so that a specific version can be recalled later. [1] The traditional Centralized Version Control System (CVCS) requires developers to commit a change to a central repository, merge and resolve the conflicts. This mechanism leads to the downside where the entire history of the project lies in the central database, and if the central database becomes corrupted, all the history data would be lost. [1][2]

Distributed Version Control System (DVCS) relaxes the requirement of CVCSs to have a central, master repository. In a DVCS, each developer owns a first-class repository of its own right and contains the entire history data,[2] thus making developing, releasing and coordinating large open source software projects much more flexible than traditional CVCS. [3]

Among all these DVCSs, Git has gained the most momentum. It began in 2005 as a revision management system used for coordinating the Linux kernel's development. Over the years, Git has evolved by "leaps and bounds" due to its functionality, portability, efficiency, and rich third-party adoption.[4]

Using Git as a back end to host open source projects, many Web-based applications enhanced project management functionality by adding rich user-friendly user interfaces, which provides a convenient way for developers to set up repositories, clone

existing projects and commit their contributions.[4] These applications also lay more emphasis on the social aspect of software engineering.

Among all these applications, GitHub is the most popular one with more than 12.2 million users[1]. GitHub not only allows users to *star*[2] or *watch*[3] repositories to keep track of projects they find interesting, but also allows users to follow other users to see what other people are working on and who they are connecting with. GitHub also supports team management by *organizations*, projects discovery by *explore*, bugs tracing by *issue*, etc.[4] By integrating these social features into version control system, the communication and coordination among developers gets greatly enhanced.[5]

2.2 Recent Research on GitHub

GitHub introduces a new open source environment that has given rise to research from many different angles. GitHub is a huge data pool that contains not only numerous projects but also developers' profiles and developers' activities such as their contributions to projects and their interactions with other developers. More and more researchers have jumped into this data pool, trying to discover some interesting patterns or good stories in terms of either software engineering or social networking.

Some researchers attempt to help employers find technical experts by looking into developers' profile and their activities.[6][7][8][9] Others have focused on the source code in project repositories. For example, Bissyande et al. (2013) took advantage of rich data on GitHub by examining the "popularity", "interoperability", and "impact" of various programming languages measured in different ways, such as lines of code, development teams, issues, etc.[10] There are also some researchers who tried to discover patterns in developers' collaboration and interaction, such as how developers assess each other and find proper collaborators[11][12][13], the herding phenomena on GitHub[14], the relation between developers' behavior on GitHub and other Q&A websites like StackOverflow[5][15], etc.

However, no scientific studies focusing on GitHub Gists have been found, which makes it a brand new research spot.

[1] `https://github.com/about/press`
[2] `https://help.github.com/articles/about-stars/`
[3] `https://help.github.com/articles/watching-repositories/`
[4] `https://help.github.com/articles/be-social`
[5] `http://www.stackoverflow.com/`

2.3 GitHub Gists

2.3.1 Related Tools

There are some other snippets management tools similar to GitHub Gists such as *pastebin*[6], *snipt*[7], *codepen*[8], *dabblet*[9], etc. However, GitHub Gists is the only one that manages snippets using version control. Over these years, GitHub Gists keeps being updated to be more user friendly as is stated in their blogs[10] [11].

2.3.2 The Features of Gists

Gists are small snippets of code or text that utilizes the version control tool Git for creation or management. Wikipedia provides a good explanation of the benefits of Gists as follows[12]:

"Gist builds upon that idea by adding version control for code snippets, easy forking, and SSL encryption for private pastes. Because each 'gist' is its own Git repository, multiple code snippets can be contained in a single paste and they and be pushed and pulled using Git. Further, forked code can be pushed back to the original author in the form of a patch, so pastes can become more like mini-projects."

In addition, GitHub also provides a powerful Web-based editor to create or modify Gists, which makes it possible to work on Gists without Git. It also supports comments on Gists and provides a Web service that makes Gists embeddable in a Web page. All these features make Gists very flexible, functional and user friendly, helping users better manage their gists and share them.

[6] http://pastebin.com/
[7] https://snipt.net/
[8] http://codepen.io/
[9] http://dabblet.com/
[10] https://github.com/blog/1837-change-the-visibility-of-your-gists
[11] https://github.com/blog/1850-gist-design-update
[12] http://en.wikipedia.org/wiki/GitHub#Gist

Chapter 3

Methodology

3.1 Research Questions

There are two main research questions to be addressed in this study.

- **RQ1. What do Gists look like?**

  To get a whole picture of what Gists are like, we investigated the metadata and contents of Gists, and tried to answer this question using statistical results.

- **RQ2. How are Gists being used?**

  We used Google to search Web pages and tweets on Twitter for people's description about how they used Gists. We combined these information with the results of the previous question to answer this research question.

Our methodology was a mix of quantitative and qualitative analysis. The study involved three main parts with the first two parts to answer **RQ1** and the third to answer **RQ2**:

1. A quantitative analysis of the metadata and contents of sampled Gists that were created by a large sample of GitHub users.

2. A qualitative analysis of a small sample of Gists. The Gists files were manually inspected to infer the purpose of each Gist.

3. A qualitative analysis of searched results of people's description about how they used Gists.

3.2 Data Collection

3.2.1 GitHub REST API

The data on GitHub can be accessed through GitHub REST API[1]. It is based on the REST architecture[2] which uses pull strategy for data retrieval so that users should explicitly make requests to collect data. Despite the adaptability of GitHub API, there are some constraints and limitations of using it. GitHub describes the rate limiting as follows: "For requests using Basic Authentication or OAuth, you can make up to 5,000 requests per hour. For unauthenticated requests, the rate limit allows you to make up to 60 requests per hour."[3] Thus, in order to get a higher query speed, each request for data retrieval must be signed with valid GitHub user tokens.

3.2.2 GHTorrent

In order to make it convenient enough for developers or researchers to acquire and analyze GitHub data, Gousios et al. developed *GHTorrent*[4], which is a project that aims to offer a mirror of GitHub's data and event streams to the research community as a service in a scalable manner.[16] It provides data dumps of GitHub users, repositories, commits, comments, etc. in either MySQL or MongoDB format. Unfortunately, GHTorrent does not provide dataset of gists, so we had to retrieve Gists data by ourselves.

3.2.3 Data Retrieving

Since the goal is to get a whole picture of the use of Gists, any Gist from any GitHub user would of interest. The Gists data were collected by leveraging GHTorrent and GitHub REST API in the following approach.

We first used GHTorrent to obtain the most recent dataset of GitHub users at the time of this study (the MongoDB dump with date 2014-03-29[5]) which contained the metadata of all GitHub users then. We did some filtering to these users data because of two main concerns. First, there may exist duplicated users in the dataset.

[1] `https://developer.github.com/v3/`
[2] `https://en.wikipedia.org/wiki/Representational_state_transfer`
[3] `https://developer.github.com/v3/#rate/limiting`
[4] `http://www.ghtorrent.org/`
[5] `http://ghtorrent.org/downloads.html`

Second, there may exist unreal users in the dataset (Organizations are included in the GHTorrent user dataset[17]). We determined a user was unreal when the value of the *ext_ref_id* attribute for that user row in MySQL dump was NULL or empty. After filtering out the unreal users and duplicates, 2,572,370 valid users were left.

Then we could query these users' Gists. Since it would take too much time to query all Gists of all users, we finally managed to download the metadata and file content of 618,393 Gists which have 793,891 files in total.

Also, in order to know the percentage of GitHub users who used Gists, we randomly selected 562,993 users from all valid users. Among all these users, we found that 32,786 (5.8%) of them had at least one Gist and these 32,786 users had 144,073 Gists in total. Table 3.1 lists all the data used in this study.

Description	**Size**
Total valid GitHub users	2,572,370
Downloaded Gists	618,393
Files of downloaded Gists	793,891
Sampled users	562,993
Sampled users having Gists	32,786
Gists of sample users	144,073

Table 3.1: Data sample used in this study.

Table 3.2 lists the metadata we chose to download associated with each Gist.

Gist Metadata	**Description**
Gist identifier	The identifier of the Gist that is unique in the system
Description	How the owner described the Gist
Create date	The date when the Gist was created
Last update date	The most recent date when the Gists was updated
Forks count	Number of forks for the Gist made by other users
Commits count	Number of commits pushed to the Gist
Commits history	Number of additions and deletions in the commits
Comments count	Number of comments for the Gist
Files count	Number of files contained in the Gist

Table 3.2: Description of Gist metadata.

One Gist can contain one or more files. The metadata of each individual file is also included in the Gist metadata, and it's listed in Table 3.3.

Gist File Metadata	Description
Filename	The name of the Gist file which includes the extension.
Language	The languages used in the file
MIME Type	MIME Type of the file such as application/json etc.
File size	Size of the file in Bytes

Table 3.3: Description of Gist file metadata.

3.3 Quantitative Analysis of Gists Metadata and Content

Based on the obtained gists metadata and contents, we made a series of quantitative analysis using statistics, trying to find some patterns and characteristics of gists so as to get a overview of what gists look like. Not only did we analyze the metadata of the sampled 618,393 Gists and Gist files but also inspected the actual content of the 793,891 Gist files.

3.3.1 Analysis of Gists Metadata

We made statistical analysis for Gists metadata based on the a series of subquestions:

- How many Gists does each user own?
- How many files are there in a Gist?
- How large is a Gist usually?
- What languages are used in Gists?
- How many commits/forks/comments are there per Gist?
- How often do users update their Gists?

3.3.2 Analysis of Gists Files

As for Gist files, apart from looking into their size metadata, we also calculated the number of lines for each file of text or application MIME type. Then further

analysis was made by taking advantage of the source code analyzing tool *CLOC*[6]. *CLOC* is able to detect uniqueness of text files, as well as calculate source lines of code (SLOC[7]) for source files in many widely used programming languages[8].

3.4 Qualitative Analysis of Gists Content

In order to make qualitative analysis of actual contents of Gist files, we randomly chose 400 from those 618,393 Gists and tried to manually analyze them by coding them. For each of these 400 Gists, we started reading through it to get a general idea of its content and then labeled it. Since a Gist is composed of one or more files, we used two strategies for labeling a Gist: analyze the content of the Gist as a whole, and determine the relationships between all the files of the Gist.

- Strategy 1: labeling based on content of the whole Gist

  We manually assigned labels to each Gist based on its description, key words in Gist files and personal experience. A Gist could have more than one labels and these labels are not necessarily exclusive as are shown in Table 3.4.

- Strategy 2: labeling based on files relationships in a Gist

  Since a Gist may contain more than one file, we assumed there could exist some relationships between the files in each Gist. These relationships are listed in Table 3.5.

After finishing the data coding, we made statistical analysis of these labels.

3.5 Qualitative Analysis of Users' Discussion Regarding Gists

We also searched the Web pages and tweets on Twitter for users' description about how Gists were used. We considered all relevant Web pages, either official or unofficial, as well as the most recent comments at the time this study was conducted.

[6] `https://github.com/AlDanial/cloc`
[7] `https://en.wikipedia.org/wiki/Source_lines_of_code`
[8] `https://github.com/AlDanial/cloc#Languages`

Labels of Gists Content	Description
Code	Source code.
Test	Test code.
Class	Only a class is defined.
Template	Coding example/pattern.
Command	Commands used in shell.
Function	Only one or several functions are defined.
Fragment	Several lines of code without complete functions/classes.
Note	Script without source code.
Log	Log files.
Configuration	Configure files used for executing code.
Diff	Diff files used for visualizing the changes in a file.
Documentation	Text tutorial documentations.
Data	Data stored in json, csv or other forms.
Blog	Technical blog in narrative format.
Non-technical	Notes without any technical content.

Table 3.4: Description of Gist labels in terms of content.

Labels of Gist Files Relationships	Description
Single File	The Gist contains one file.
Independent	Files in a Gist are independent.
Reference	One file refers to another file.
Generation	One file is the input/output of another file.
Test	One file is to test another file.
Attachment	One file is the configuration or information of another file.

Table 3.5: Description of Gist labels in terms of files relationships.

1. Web Pages. In order to find relevant Web pages, we used Google Search Engine to perform the searching using the keyword "GitHub Gist". For each of the search results, we manually read through the Web page, extracted the information about how Gists were used, and took notes.

2. Twitter Postings. As is known, Twitter has become a big information gathering pool, providing much data for researchers. Recently, Twitter has come to be a more and more popular platform for developers to learn, share, discuss technical questions, so we chose Twitter as a information source to help understand how people are using gists. We performed the query using "GitHub Gist" as the

searching keywords on Twitter Web search interface, and read all the searched tweets one by one. While reading these data, we took notes in terms of how they used Gists suggested in the posting.

Chapter 4

Results

4.1 RQ1. What do Gists look like?

4.1.1 Users and Their Gists

As Table 4.1 shows, among the 562,993 GitHub users we randomly sampled, only 32,786 (5.82%) of them had at least one public Gist.

Users	**Count**	**Percentage**
Users having at least one public Gist	32,786	5.82%
Users having no public Gists	530,207	94.18%

Table 4.1: Comparison of GitHub users having at least one public Gist and those having not any.

For those 32,786 users having at least one public Gist, Figure 4.1 shows their distribution by the number of Gists per user. We can see that majority of users just have a small number of Gists. Specifically, 50% of the users have less than 5 public Gists. 70% have less than 10 public Gists. Only 4% of users have 30 or more public Gists.

4.1.2 Contents of Gists

The content of Gists can be analyzed in several different aspects, such as the number of files per Gist, the languages, the MIME type, the size. All the analysis results in this section are against the sampled 618,393 Gists or the 793,891 files in those Gists.

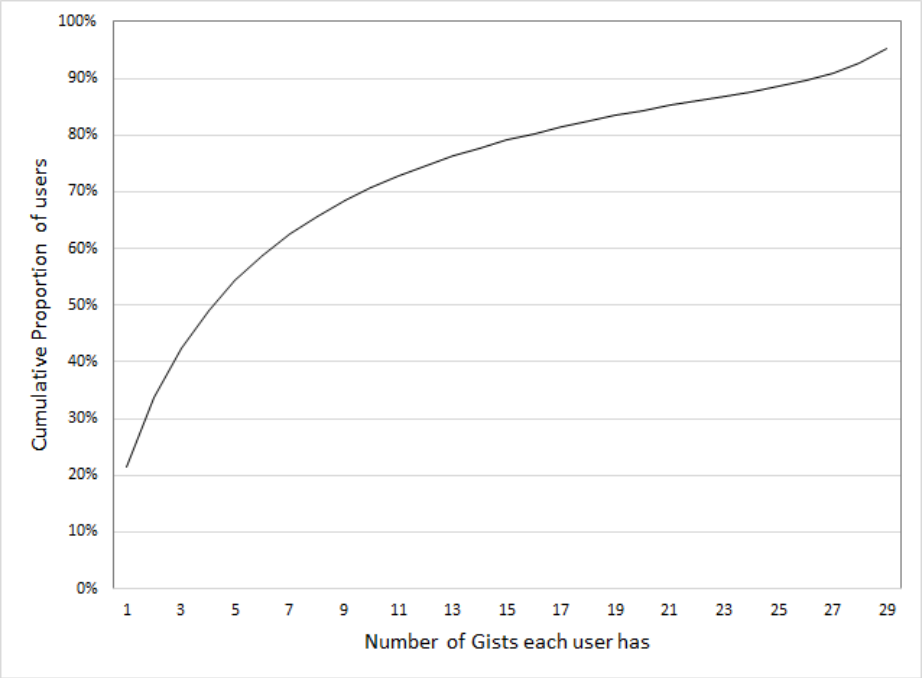

Figure 4.1: Users distribution by the number of Gists owned.

Number of Files per Gist

GitHub doesn't restrain the number of files in each Gist. Table 4.2 shows the distribution of all sampled Gists based on the number of files they contain. It shows that majority of Gists (86.79%) contain only one file.

Number of Files Each Gist contains	Percentage of Gists
0	0.02%
1	86.79%
2	7.16%
3	3.16%
4	1.53%
≥ 5	1.35%

Table 4.2: Distribution of Gists by number of files contained.

It is expected that majority of Gists contain only a few files, since Gists are supposed to record and share small code snippets. However, one surprising observation we obtained was that there also existed Gists containing a lot of files (1001 files in a Gist at most).

Gist Files Types

We analyzed the types of Gist files by their MIME types and their languages.

Figure 4.2 shows the distribution of top 20 MIME types of all Gist files (there are other 93 falling into the "Others" category which is 0.4%). It's obvious that text/plain dominates with 53.5% which actually includes a variety of file types. Some programming languages are very common indicated by the MIME type, such as application/javascript (9.7%), application/x-ruby (9.4%), application/python (5.6%), but we need to note that lots of programming languages fall into the text/plain category. Other common types include images (png 0.5%, gif 0.4%, jpeg 0.1%), markup language text (html 3.7%, xml 1.4%, yaml 0.5%), JSON data (1.7%), etc.

The Gist file metadata provides the language attribute for each file. We discovered 257 languages in total from our sample. Figure 4.3 shows the distribution of top 30. We can see 27.9% of Gist files don't have any languages. The most popular programming languages are Ruby (10.2%), JavaScript (9.7%), Python (5.6%), Shell script (5.3%). Lots of markup languages are very common too, which includes Markdown (4.6%), HTML (3.7%), JSON (1.9%), XML (1.5%), Diff (1.3%). The "Others" category represents the other 227 languages that are not shown in the chart and they account for 5.8%.

Size of Gist Files

As for the size of Gist files, we looked into three metrics: size in bytes, number of lines for files of text or application MIME type, and SLOCs for source files.

Figure 4.4 shows the distribution of Gist files in terms of their size in bytes. We can see majority of Gist files are around 1KB. The median value is 714 Bytes (quartiles are 287 and 1830 bytes). There exist outliers though: 0.17% of Gist files are over 1MB.

We also counted the number of lines for all the text files (783,326 in total). Figure 4.5 shows the distribution of these files by their number of lines. 87.1% of them have less than 100 lines. The median value is 23 lines (quartiles are 10 and 55 lines).

We ran *CLOC* script against all Gist files to calculate their SLOC (binary or empty files would be ignored by *CLOC*). *CLOC* recognizes a source file by its extension or its first line of code. If the file does not have a recognized extension or is not a recognized scripting language, it would be ignored.[1]. Finally 778,806 text files (98.1%

[1] `https://github.com/AlDanial/cloc#How_it_works`

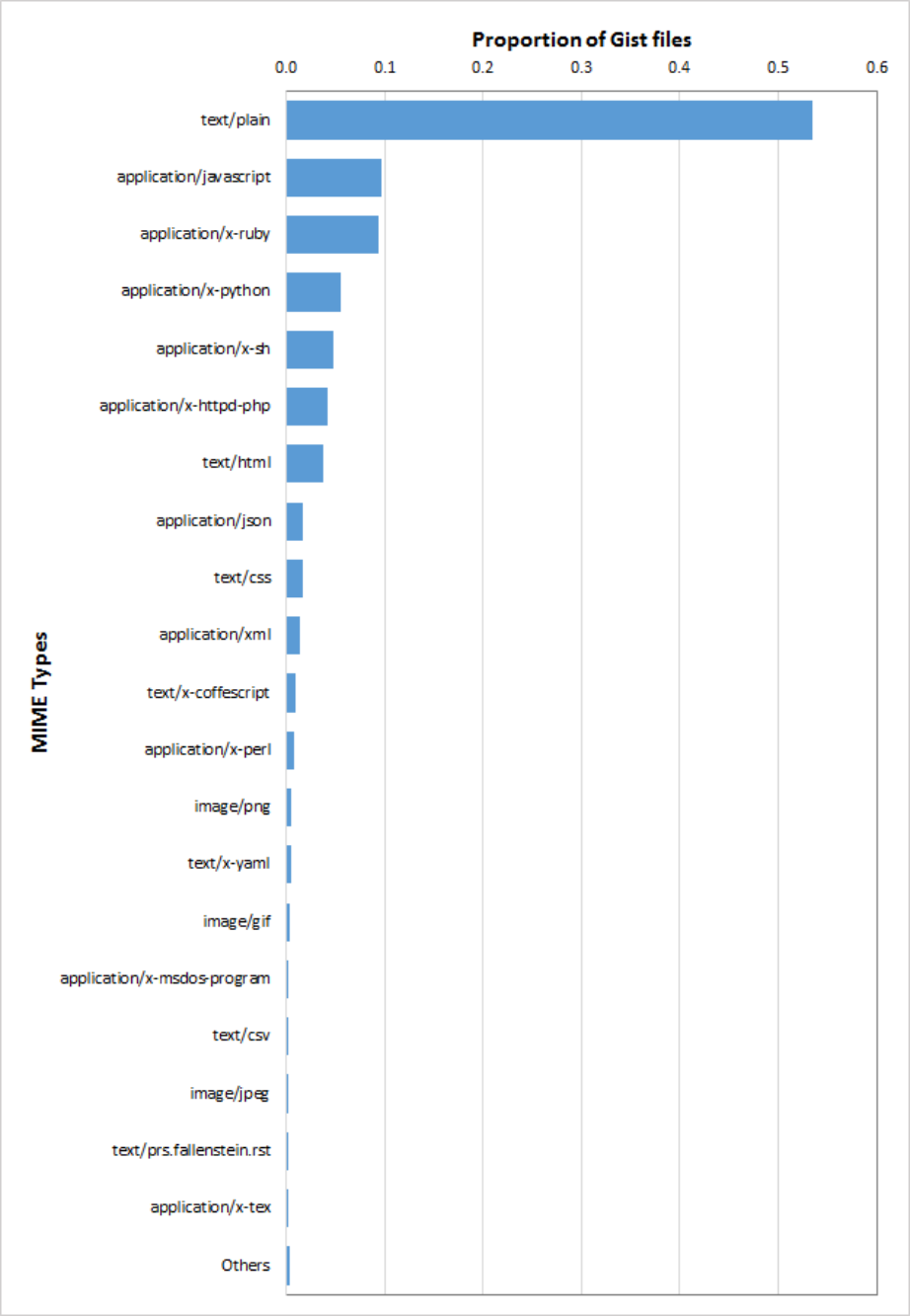

Figure 4.2: Distribution of Gist files by MIME types.

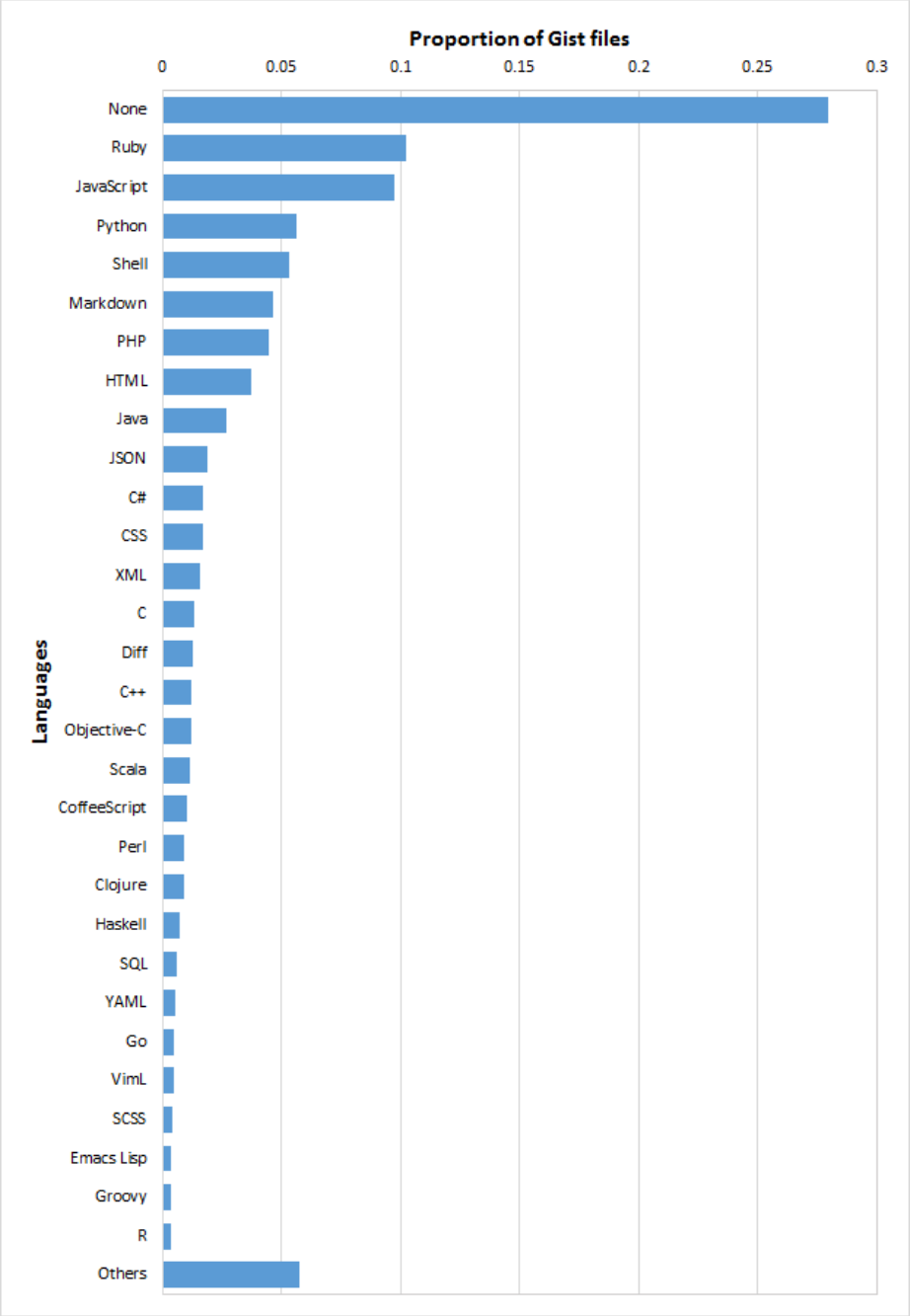

Figure 4.3: Distribution of Gist files by language.

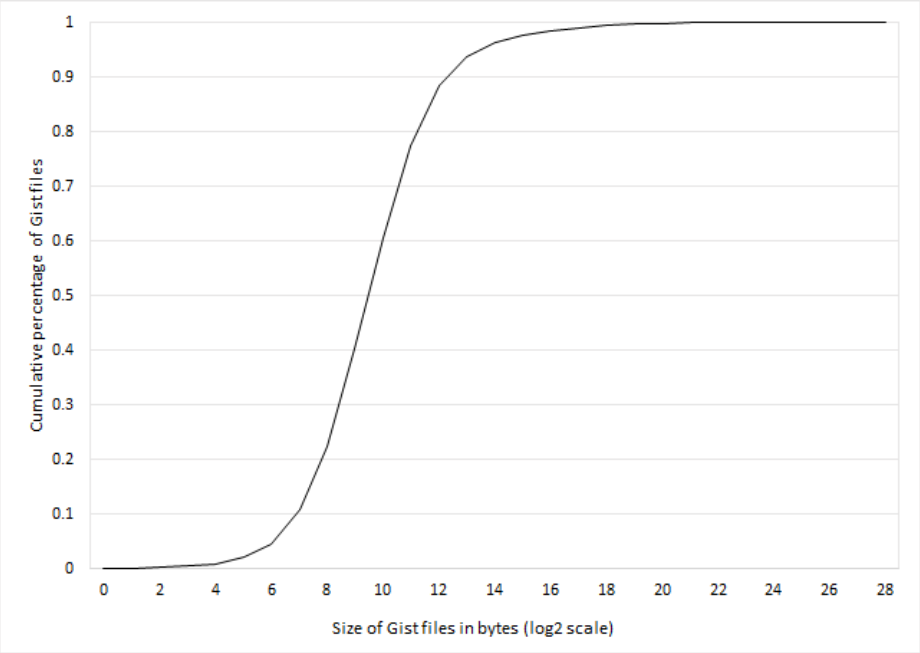

Figure 4.4: Distribution of Gist files by size.

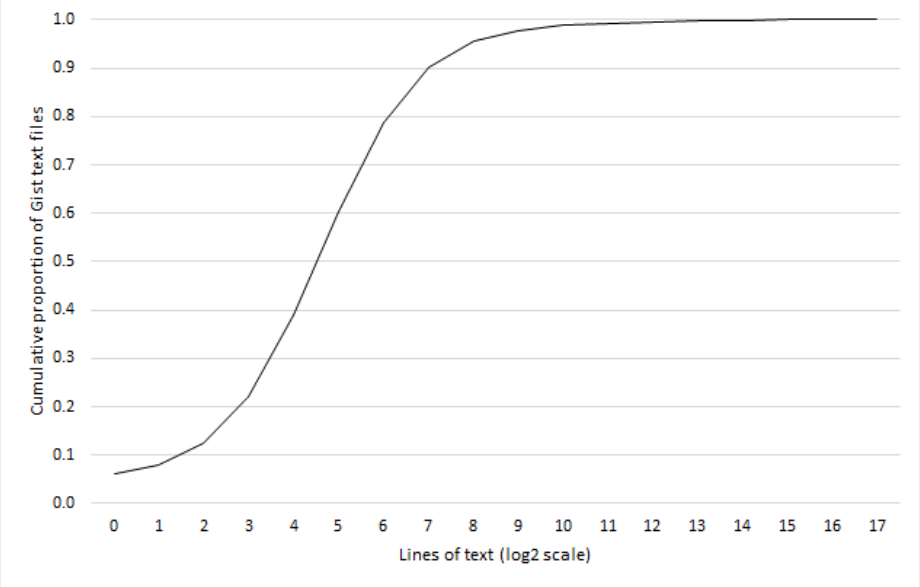

Figure 4.5: Distribution of Gist text files by number of lines.

of all Gist files) were detected, from which 502,242 source files (64.5% of all Gist text files, 63.3% of all Gist files) were recognized as Table 4.3 shows. The distribution of source files by SLOC as is shown in Figure 4.6. todo% of them have less than todo lines. The median value is todo lines (quartiles are todo and todo lines).

Files	Count
All files of 618,393 Gists downloaded	793,891
— Gist text files	778,806
——— Gist source files	502,242

Table 4.3: Gist files recognized by CLOC

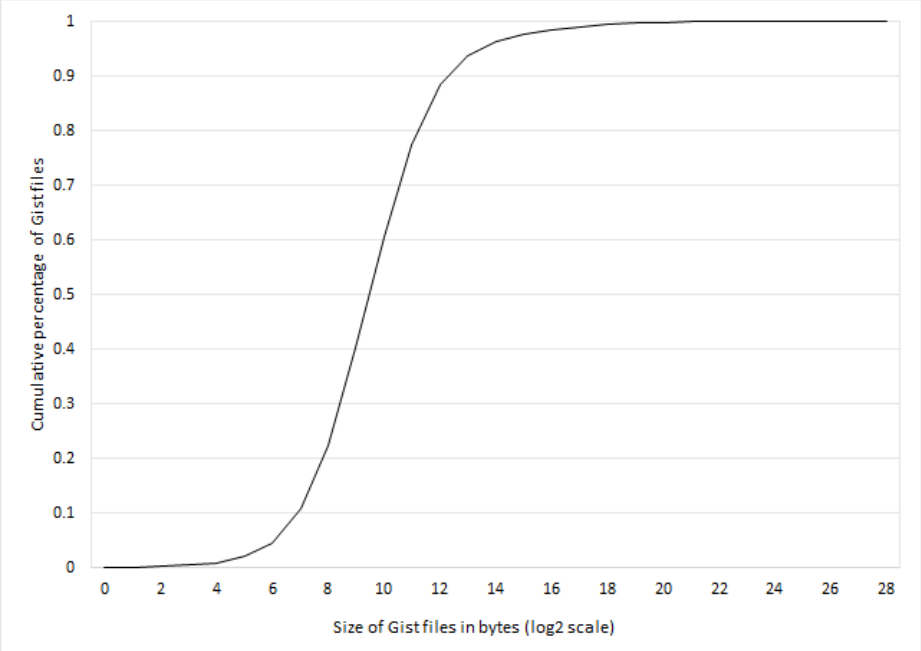

Figure 4.6: Distribution of Gist source files by SLOC.

Uniqueness of Gist Files

CLOC is able to detect uniqueness for text files based on *Digest:MD5*[2]. By taking advantage of that, We performed duplicate detection against all Gist files. Since *CLOC* cannot detect duplicates for binary files, it only analyzed all the 778,806 text files from which 759,586 (97.5% of all text files) files were detected as unique. It means that 2.5% of text Gist files were duplicates.

4.1.3 Activity

Since Gists are like a GitHub repositories, we can trace users' activities against them, such as number of commits, number of comments, times of being forked, etc.

[2] `https://github.com/gisle/digest-md5`

As Figure 4.7 shows, 62.9% of Gists had a single commit. 32.1% had 2 to 5 commits, and only 1.85% of Gists had more than 10 commits. This result is also verified by the distribution of days between the creation the latest update for each Gist, which is shown in Figure 4.8. We can see the creation and the latest update of 83.7% of Gists happened at the same day.

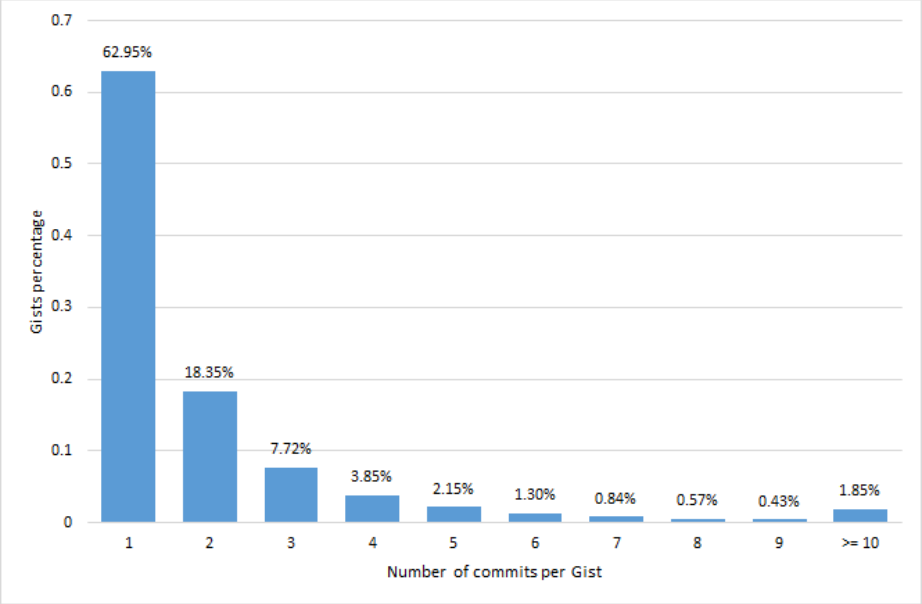

Figure 4.7: Distribution of Gists by number of commits.

In terms of forks, 94.6% of Gists have never been forked, and 4.0% of Gists have only been forked once. Only 0.7% of Gists have been forked more than 3 times (shown in Table 4.4). Table 4.5 shows the distribution of comments per Gist, from which we can see only 7.3% of Gists have been commented once or more. All of these observations show that majority of Gists are not active.

Number of forks	**Gists proportion**
0	94.6%
1	4.0%
2	0.7%
≥ 3	0.7%

Table 4.4: Distribution of Gists by number of forks.

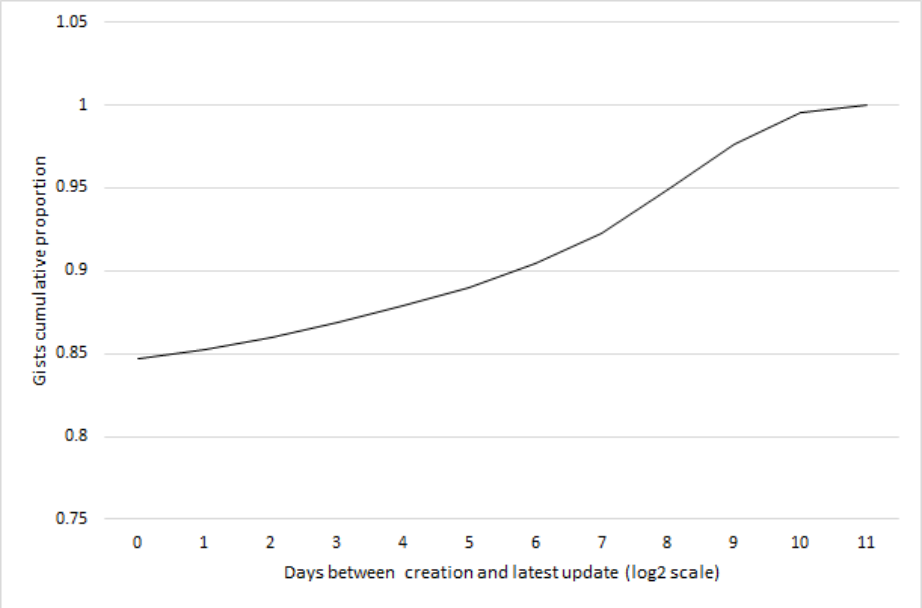

Figure 4.8: Distribution of of Gists by days between creation and latest update.

Number of comments	Gists proportion
0	92.7%
1	4.6%
2	1.3%
3	0.5%
≥ 4	1.0%

Table 4.5: Distribution of Gists by number of comments.

4.1.4 Manual Inspection of Gists Content

We manually inspected 400 randomly sampled Gists by assigning non-exclusive labels to each of them based on two aspects: the content of a Gist as a whole, and the relationships between files within a Gist.

Gists Content

During categorizing these 400 Gists based on content with the labels shown in Table 3.4, we observed that the contents of Gists were usually very short and demonstrated a big variety of uses. Code is expectedly the most dominant use, but we also found lots of other uses, such as blogs, configuration information, logs, data, letters, or even a restaurant menu. We also found Gist written in several other languages

besides English. Moreover, we also observed that the MIME type or language of a Gist file might not be enough to understand a Gist. For example, source code and plain text are often mixed together.

The result of categorizing is shown in Table 4.6. Note that the sum of all labels are more than 100% because a Gist can be assigned several labels. It's not surprising that most of Gists (79.25%) contain code, including complete programs, functions, code fragments, classes, test code. However, other types of Gists content cannot be dismissed: Log (7.5%), Blog (1.5%), Non-technical (1.5%), etc.

Labels in terms of content	**Percentage of Gists**
Code	79.25%
Note	24.5%
Function	10.0%
Log	7.5%
Class	6.5%
Data	5.25%
Fragment	4.75%
Command	4.75%
Template	4.0%
Test	3.25%
Configuration	3.25%
Diff	1.75%
Blog	1.5%
Non-technical	1.5%
Documentation	0.5%
Empty	0.5%

Table 4.6: Distribution of Gists by content labels.

Files Relationships Within One Gist

A Gist can have more than one file. When a Gist has multiple files, we observed that they were usually related. The 400 Gists were categorized with labels shown in Table 3.5 to show the relationships between the files within each Gist, and the result is shown in Table 4.7. We can see most Gists contained only one file which is consistent with *FINDING 3*. For those having multiple files, they spread approximately uniformly among different categories. Only 3.25% of Gists have multiple files that are not related to each other.

Relationship between files of each Gist	Percentage
Single File	87.25%
Reference	4.25%
Generation	3.25%
Independent	3.25%
Test	1.25%
Attachment	0.75%

Table 4.7: Distribution of Gists by relationships between files

4.2 RQ2. How are Gists being used?

We answered this question based on the searched result of two sources: Websites and Twitter. We used "Github Gist" as keywords to search for related Web pages or tweets. While reading these content, we took notes in terms of suggested Gist uses. Finally we combined the results from Web pages and tweets, and classified them into several categories. Where appropriate, some quotations from the sampled Websites and Tweets are provided to illustrate our findings which are as follows.

Being embedded in blogs

One of the biggest blog websites, WordPress[3] has supported embedding Gists in the blogs[4]. Apart from that, Many other blogging platforms like Medium[5] also supporting such embedding soon.[6] This makes it easy to manage the code in blogs by saving it in Gists and simply referencing it in blogs by ID or URL.

"I think I might start using GitHub Gists in my blog posts. Thinking about how to integrate it smoothly."[7]

Version-controlled list

Gists can serve as version-controlled lists by taking advantage of the markdown rendering. Also the version control system can help to keep track of the change history of the lists, adding functionality at no cost. One good example is a popular blog that

[3] `https://wordpress.com/`
[4] `http://crunchify.com/how-to-embed-and-share-github-gists-on-your-wordpress-blog`
[5] `https://medium.com/`
[6] `https://medium.com/the-story/yes-we-get-the-gist-1c2a27cdfc22`
[7] `https://twitter.com/BenNadel/status/157495925231714304`

has been widely tweeted on Twitter. The blog illustrates how to maintain a to-do list using a private GitHub Gist authored by Carl Sednaoui[8].

"GitHub Gists are a great way to keep version-controlled lists (in this case, US states I've visited) `https://gist.github.com/dliggat/11003570`*"*[9]

"Im a HUGE fan of todo lists. They help me stay organized, prioritize my day and add structure to an otherwise chaotic day. I recently discovered what appears to be the best yet simplest way to keep a to-do list: a GitHub Gist."[10]

Saving notes/tips

Some people use Gists to save notes or tips such as technical information, learning outcomes tips, etc.

"I've been saving my learning as gists because blogging is a barrier: `https://gist.github.com/Greg-Boggs`*"*[11]

"Useful fiddles and gists collected from #AngularJS forum discussions `https://github.com/angular/angular.js/wiki/JSFiddle-Examples...`*"*[12]

Sharing files

Gists can also be used to share files with other people. Even if a Gist is private, it can still be visited by whoever has its URL. Sometimes private Gists can be a perfect way to share files. The following conversation on Twitter is a good example of this use.

User A to user B: *"Can you show me what your application.xml looks like for your ios+android game? Need to know what info is needed to compile."*

User B replied user A: *"I can put it on GitHub Gists sometime soon, remind me in a few days if I forget..."*[13]

---

[8] `http://carlsednaoui.com/post/70299468325/the-best-to-do-list-a-private-gist`

[9] `https://twitter.com/dliggat/status/458090816930848768`

[10] `http://www.carlsednaoui.com/post/70299468325/the-best-to-do-list-a-private-gist`

[11] `https://twitter.com/gregory_boggs/status/483111455550877697`

[12] `https://twitter.com/AppsHybrid/status/481901294563901440`

[13] `https://twitter.com/stvr_tweets/status/483092598639185920`

Drafting

Gists are also a perfect place to write a draft. For example, a user uses a Gist[14] to draft the wiki for a Google project[15]. This use can also be suggested by the following tweet:

"Every once in a while I think I wish I could "draft" a pull-request, issue, or comment on GitHub, then I remember that private Gists exist." [16]

4.3 Summary

Based on all the analysis above, our observations can be summarized as the following themes.

Majority of GitHub users don't use public Gists.

For those GitHub users we randomly sampled, only 5.82% of them have at least one public Gist. For those who have public Gists, half of them have less than 5 Gists. Also, 86.79% of those sampled Gists only contain 1 file. All these results show that GitHub users generally don't use public Gists much.

Gists usually contain small files.

According to the metadata provided by GitHub, we found that majority of Gist files were around 1KB. The median size was 714 Bytes. As for all the text files, 87.1% of them have less than 100 lines, and the median number of lines is 23. We also calculated the source lines of code for all the source files, and todo of them have less than todo lines with median as todo lines.

Gists content shows a great variety.

There are 113 MIME types and 257 programming languages discovered from all the Gist files we sampled. Although majority of Gist files are source files, we also observed data files (such as JSON, CSV, XML) and binary files (such as images, audios and even videos) in Gists. During our manual inspection against Gists content, we also found blogs, letters, and even restaurant menus.

[14] `https://gist.github.com/danbri/58db1297f3b488da9f86`
[15] `https://github.com/google/cayley/wiki`
[16] `https://twitter.com/nuclearsandwich/status/249213040610910209`

Users don't collaborate on Gists.

We found Gists are mostly personal artifacts. Among all of the sampled Gists, 62.9% of them had a single commit (by the owner obviously), and only 1.85% had more than 10 commits. The creation and the latest update of 83.7% of Gists happened at the same day. 94.6% of Gists have never been forked by other users, and only 7.3% of Gists have been commented once or more.

Gists are being used for a variety of purposes.

Based on our searched results from Websites and Twitter, a great variety was discovered in terms of what people use Gists for: some people use it to create components that can be embedded in their blogs; some use it to keep track of some activities by a version controlled list; some save their notes such as learning outcomes in Gists; some share content with others simply by Gist URLs; some use it to draft their writings.

Chapter 5

Limitations and Future Work

5.1 Limitations

We have to admit there do exist threats to the validity of our results.

First of all, though private Gists are available to whoever has the Gist URL, they cannot be queried through the APIs GitHub provides. Therefore, we can only perform the analysis against public Gists, which introduces limitations to our results. The answer to our research questions for private Gists might differ from public ones.

Besides, the manual labeling in our qualitative analysis may introduce errors into our results. One researcher conducted the labeling by manually reading and interpreting the Gist content. The misinterpretation can introduce errors.

Also, since we have no control of how the search engines of Google and Twitter work, our searched results really depend on when we performed the search. This dependence makes the results transient and likely to change.

5.2 Future Work

This study is exploratory. We are just scratching the surface of GitHub Gists. One aspect we didn't reach is what users think about Gists. In the future, interviews of users should be involved. Some interview questions could be: what motivates a user to create a Gist? When do users choose to use Gists instead of GitHub repositories? What determines if a user create a public Gist or a private one? Do they collaborate much on Gists?

Lots of users use Gists to save reusable components, but where do these reusable

components from? Are they from other people's blogs? Or the answers of Q&A websites like StackOverflow? Or GitHub repositories?

There also exist other tools for users to save and share code snippets. A comparison between Gists and these tools could also be an interesting research topic.

Chapter 6

Conclusion

In this study, with the goal of understanding what GitHub gists look like and how they are used, we performed quantitative analysis against 618k Gists and 562k GitHub user, manual coding for the content of 400 randomly selected Gists, and manual analysis against the searched results from Google and Twitter about users' description of how they use Gists.

Our results can be summarized as follows:

What do Gists look like? Only a small portion of GitHub users use Gists. Gists usually contain small snippets of source code, but a wide variety of Gists content was also discovered including data files (such as JSON, XML), binary files (such as images, audios, vidoes), and text files of various kinds (such as blogs, logs, letters, restaurant menus). Majority of Gists were not updated frequently, and users rarely fork or comment other users' Gists.

How are Gists being used? Although GitHub expects users to use Gists to instantly share code snippets, people are actually using Gists for various purposes: some people use it to create components that can be embedded in their blogs; some use it to keep track of some activities by a version controlled list; some save their notes such as learning outcomes in Gists; some share content with others simply by Gist URLs; some use it to draft their writings.

Gists can serve for lots of purposes by taking advantage of either its convenient editor or version control support. We hope it will become more and more popular in the future..

Bibliography

[1] Scott Chacon and Junio C Hamano. *Pro git*, volume 288. Springer, 2009.

[2] B. de Alwis and J. Sillito. Why are software projects moving from centralized to decentralized version control systems? In *Cooperative and Human Aspects on Software Engineering, 2009. CHASE '09. ICSE Workshop on*, pages 36–39, May 2009.

[3] P.C. Rigby, E.T. Barr, C. Bird, P. Devanbu, and D.M. German. What effect does distributed version control have on oss project organization? In *Release Engineering (RELENG), 2013 1st International Workshop on*, pages 29–32, May 2013.

[4] D. Spinellis. Git. *Software, IEEE*, 29(3):100–101, May 2012.

[5] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb. Leveraging transparency. *Software, IEEE*, 30(1):37–43, Jan 2013.

[6] A. Capiluppi, A. Serebrenik, and L. Singer. **Assessing Technical Candidates on the Social Web**. *Software, IEEE*, 30(1):45–51, Jan 2013.

[7] Jennifer Marlow and Laura Dabbish. Activity traces and signals in software developer recruitment and hiring. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 145–156, New York, NY, USA, 2013. ACM.

[8] Rahul Venkataramani, Atul Gupta, Allahbaksh Asadullah, Basavaraju Muddu, and Vasudev Bhat. Discovery of technical expertise from open source code repositories. In *Proceedings of the 22Nd International Conference on World Wide Web Companion*, WWW '13 Companion, pages 97–98, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[9] C. Teyton, J.-R. Falleri, F. Morandat, and X. Blanc. Find your library experts. In *Reverse Engineering (WCRE), 2013 20th Working Conference on*, pages 202–211, Oct 2013.

[10] T.F. Bissyande, F. Thung, D. Lo, Lingxiao Jiang, and L. Reveillere. Popularity, interoperability, and impact of programming languages in 100,000 open source projects. In *Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual*, pages 303–312, July 2013.

[11] Anirban Majumder, Samik Datta, and K.V.M. Naidu. Capacitated team formation problem on social networks. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 1005–1013, New York, NY, USA, 2012. ACM.

[12] Jennifer Marlow, Laura Dabbish, and Jim Herbsleb. Impression formation in online peer production: Activity traces and personal profiles in github. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 117–128, New York, NY, USA, 2013. ACM.

[13] Leif Singer, Fernando Figueira Filho, Brendan Cleary, Christoph Treude, Margaret-Anne Storey, and Kurt Schneider. Mutual assessment in the social programmer ecosystem: An empirical investigation of developer profile aggregators. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 103–116, New York, NY, USA, 2013. ACM.

[14] Joohee Choi, Junghong Choi, Jae Yun Moon, Jungpil Hahn, and Jinwoo Kim. Herding in open source software development: An exploratory study. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work Companion*, CSCW '13, pages 129–134, New York, NY, USA, 2013. ACM.

[15] Bogdan Vasilescu, Vladimir Filkov, and Alexander Serebrenik. Stackoverflow and github: Associations between software development and crowdsourced knowledge. In *Social Computing (SocialCom), 2013 International Conference on*, pages 188–195, 2013.

[16] G. Gousios and D. Spinellis. Ghtorrent: Github's data from a firehose. In *Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on*, pages 12–21, 2012.

[17] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. The promises and perils of mining github. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, pages 92–101, New York, NY, USA, 2014. ACM.