GitterCom - A Dataset of Open Source Developer Communications in Gitter

Esteban Parra
parrarod@cs.fsu.edu
Florida State University
Tallahassee, Florida

Ashley Ellis
ake17@my.fsu.edu
Florida State University
Tallahassee, Florida

Sonia Haiduc
shaiduc@cs.fsu.edu
Florida State University
Tallahassee, Florida

Abstract

Team communication is essential for the development of modern software systems. For distributed software development teams, such as those found in many open source projects, this communication usually takes place using electronic tools. Among these, modern chat platforms such as Gitter are becoming the de facto choice for many software projects due to their advanced features geared towards software development and effective team communication. Gitter channels contain numerous messages exchanged by developers regarding the state of the project, issues and features of the system, team logistics, etc. These messages can contain important information to researchers studying open source software systems, developers new to a particular project and trying to get familiar with the software, etc. Therefore, uncovering what developers are communicating about through Gitter is an essential first step towards successfully understanding and leveraging this information.

We present a new dataset, called *GitterCom*, which aims to enable research in this direction and represents the largest manually labeled and curated dataset of Gitter developer messages. The dataset is comprised of 10,000 messages collected from 10 Gitter communities associated with the development of open source software. Each message was manually annotated and verified by two of the authors, capturing the purpose of the communication expressed by the message. While the dataset has not yet been used in any publication, we discuss how it can enable interesting research opportunities.

CCS Concepts

• **Software and its engineering → Collaboration in software development**; *Open source model*; *Documentation*;

Keywords

datasets, chat, social media, team communication

ACM Reference Format:
Esteban Parra, Ashley Ellis, and Sonia Haiduc. 2020. GitterCom - A Dataset of Open Source Developer Communications in Gitter. In *17th International Conference on Mining Software Repositories (MSR '20), October 5–6, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3379597.3387494

MSR '20, October 5–6, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7517-7/20/05…$15.00
https://doi.org/10.1145/3379597.3387494

1 Introduction

Modern, complex open source software systems often require large teams in order to be developed. The teams are usually geographically distributed across different locations, countries and even continents. In order to collaborate, communicate, and coordinate, these teams make use of electronic tools such as instant messaging, email, etc. [14, 15, 18, 20]. Recently, modern messaging and collaboration platforms such as Gitter[4] and Slack[5] have revolutionized team communications and project coordination by providing a user-friendly way of managing and organizing conversations, facilitating knowledge sharing, and by integrating with external software development tools such as GitHub, Asana, and Jira[19]. Given their features and the support for software development, many open source projects have adopted Gitter and Slack as their preferred communication means [15]. In particular, Gitter is currently the most popular instant messaging platform in open source development teams [15]. It also presents some advantages over Slack, such as:

- *Open access to communications*: in Slack, communities are controlled by the administrators, whereas in Gitter, access to the user-generated data is public. In particular, public messages and user-generated content in Gitter are subject to the Creative Commons license: Attribution + Non-Commercial + ShareAlike (BY-NC-SA)[6]
- *Free access to historical data*: in Slack communities, only the latest 10,000 messages are accessible without paying. Since most public Slack channels use the free tier [13], their historical data is unavailable. Conversely, messages posted to public Gitter channels are preserved and accessible indefinitely in chat room logs.

Despite its advantages over Slack, its greater popularity among open source developers, and the availability of tens of thousands of message exchanges between developers of open source software, there have been no papers so far investigating developer communications in Gitter. Rather, existing works analyzing developer communications in modern instant messaging platforms have so far focused solely on Slack [11–13, 16].

We argue that Gitter developer communications are an untapped information resource that could be leveraged by researchers in order to get a deeper understanding about the nature of developer communications in open source software. With this paper, we aim to encourage research in this direction by introducing GitterCom, the first manually labeled dataset of Gitter instant message histories in open source systems. GitterCom contains 10,000 messages labeled based on the communication purpose they express. This dataset is the largest manually labeled dataset of developer instant messages;

[4] https://gitter.im/
[5] https://slack.com/
[6] https://creativecommons.org/licenses/by-nc-sa/3.0/us/

Table 1: Distribution of messages per purpose category

Category	Cucumber	Freezing	ImageJ	Jhipster	JSPM	MJS[1]	Sklearn[2]	THW[3]	UIKit	Xenko	Overall (%)
Communication	325	794	490	446	635	695	506	583	321	480	5275 (52.75%)
Customer support	442	0	150	239	0	0	4	145	451	0	1431 (14.31%)
Dev-Ops	198	183	308	269	305	235	464	240	190	383	2775 (27.75%)
Discovery and news	13	1	10	9	7	2	3	15	5	32	97 (0.97%)
Fun	0	2	0	0	0	39	0	0	1	0	42 (0.42%)
Networking and social activities	0	0	1	3	0	3	0	0	0	32	39 (0.39%)
Participation in Communities of Practice	4	2	9	15	21	13	13	10	12	54	153 (1.53%)
Team Collaboration	18	18	32	19	32	13	10	7	20	19	188 (1.88%)

the only other manually labeled dataset available is comprised of 500 developer Slack messages in one software company [20].

The rest of the paper is structured as follows: section 2 presents an overview of the dataset, section 3 outlines the data collection process, section 4 discusses potential research directions using this dataset, section 5 presents limitations and future improvements that could be made to the data set and lastly, section 6 concludes the paper.

2 Dataset Description

GitterCom includes data about 10,000 messages collected from 10 open source software development Gitter communities (1,000 messages per community). Each message was manually labeled with information about the purpose of the communication it expresses, based on the categories identified by Lin *et al.* [16].

GitterCom is available in both CSV and Microsoft Excel Open XML Spreadsheet (XLSX) file formats online[7]. In the CSV file, each line is a data record. Each record contains the information for a single message and consists of seven information fields, separated by comma and using quotes as the text delimiter. In particular, each row contains: (i) the channel/system the message belongs to, (ii) a unique messageID, (iii) the date and time at which the message was posted, (iv) the author of the message, (v) the content of the message in plain text, (vi) the corresponding purpose category (manual label), and (vii) the purpose subcategory (manual label).

Next, we present brief descriptions of the different purposes, their categories and subcategories we used to manually label the messages in GitterCom. These were first identified by Lin *et al.* [16], who surveyed software developers about their use of Slack.

The first purpose, called **Personal benefits**, includes messages in which the developer's main purpose is to fulfill personal needs. Messages within this purpose can be further divided into three categories: *discovery and aggregation of news and information*, where developers post reliable, interesting, and relevant blogs or other sources of information; *networking and social activities*, where developers interact with other developers who share similar interests or jobs; and *fun*, which are messages sharing gifs and memes or meant for participating in gaming activities.

The second purpose relates to **Team-wide** activities and includes messages aimed towards carrying out software development activities related to the system being developed. Messages within this purpose can be further divided into four categories: *communication* messages in which the developers engage in activities such as communication with teammates (*e.g.*, members of a distributed team) during meetings and note-taking, communication with other stakeholders, or discussing non-work topics; *team collaboration* messages in which the developers engage in activities such as team management, file, and code sharing; *Dev-Ops* messages in which the developers engage in activities such as communicating updates regarding the status of the project (*e.g.*, development operation notifications about recent changes to the system, commits, bug fixes, pushes to the repository, merges), software deployments, and team Q&As; and finally, *customer support* messages in which the developers assist new or existing users of the system on how to perform certain tasks, identify bugs, and troubleshoot errors.

The last purpose is represented by **Community support** messages, where developers participate in communities of practice or special interest groups. These messages are characterized by developers aiming to keep up with specific frameworks/communities, to learn about new tools and frameworks for developing applications, or to brainstorm ideas with other people in the community.

Table 1 shows the number of messages per category in GitterCom, for each of the 10 open source systems/communities we considered and the overall distribution of messages associated with each category across all the communities in GitterCom.

Based on the hierarchy presented above, we notice that the majority of Gitter messages in GitterCom belong to Team-wide purposes. Table 1 shows that the distribution of messages varies significantly across categories. In particular, 83% of the messages are meant to support activities directly associated with the development of the system. On the other hand, 14.31% of the messages are related to community support and engagement with communities of practice,

[4]MarionneteJS
[5]SciKit-Learn
[6]TheHollyWaffle
[7]https://figshare.com/s/9b3df36e22a8a8f77169

Table 2: Subset of Gitter communities included in GitterCom

Community	Users	Messages	Application domain
Marionette [6]	3014	181108	Javascript framework
jspm [5]	1103	27245	Package manager
scikit-learn [7]	3188	9844	Machine Learning
Xenko3d [10]	103	2890	Game engine
FreezingMoon [2]	109	207925	Video game
UIkit [9]	2155	41265	Front-end framework
jHipster [4]	2575	39418	Application generator
Cucumber [1]	337	2030	Testing framework
Imagej [3]	209	8149	Image processing
TheHolyWaffle [8]	196	15046	VoIP communication

and only 2.69% of the messages are linked to personal benefits. Moreover, 53% of the messages involve communication between the developers and stakeholders, 28% of the messages communicate updates regarding the status of the system, and 15% of the messages involve customer support.

3 Data Collection

This section presents in detail the data collection and curation procedure we used to create the GitterCom dataset. First, we gathered the list of all the communities listed in Gitter's Explore interface[8] on April 1, 2019. We then excluded the channels in which the conversations were not in English, resulting in a list of 139 Gitter communities. Afterwards, using the Gitter API[9], we extracted all of the messages in the *main channels* of these communities, from their inception until April 1, 2019. This data collection resulted in a set of 2,939,335 messages across all 139 channels.

To extract the raw data for GitterCom, we used a custom python script, which uses pycurl to connect to Gitter's REST API and obtain all the messages' raw text and their corresponding metadata. Any code snippets included in the messages are also extracted as raw text. Afterwards, to facilitate the labeling process, we ran a custom script in Java to convert the messages from the JSON format provided by Gitter's API to CSV format. The data collection scripts and instructions on their usage are found in our replication package [17].

The 139 channels collected as raw data vary in three main ways: by *membership* - the channels contain between 100 and 17,000 members per channel, by *level of activity* - the smallest channel contains 21 messages, whereas the largest channel contains over 423,000 messages, and by *type* - channels can be made for the development of a particular software system, where the developers communicate with each other and with the system's stakeholders, or made for building communities of practice in which the members' discussion revolves around particular topics, frameworks, or programming languages, but does not involve discussion about the active development of a system.

While we make the entire data we extracted for all the 139 channels available for download to other researchers[10], our main goal for GitterCom was to manually curate and label a subset of the messages, based on the purposes/intents identified by Lin *et al.* [16] (as described in Section 2). We therefore selected the first ten channels, out of 100 qualifying channels which met the following criteria: (i) they are linked to an active GitHub repository (commits have been made within the past year), (ii) they are used as a communication tool for the active development of an open-source software system, (iii) they cover different application domains, (iv) they have been active in the past year, and (v) they contain at least 1,000 messages. Table 2 shows the details of the selected systems/channels.

From each of the ten selected channels we then collected the 1,000 most recent consecutive messages up to April 1, 2019, for a total of 10,000 messages. The first two authors then carried out a coding procedure to label these messages, using the categories and subcategories identified by Lin *et al.* [16] as labels. More specifically, each message was assigned a category describing the main purpose of the message and a subcategory describing the specific activity the message relates to. If a message did not provide any meaningful information by itself (*e.g.*, a single emoji, "ok", "great", ""), it was classified as "Uninformative". After the individual coding, the two authors met, discussed, and resolved any coding conflicts. The messages for which a classification of "Uninformative" was agreed upon were discarded and replaced by an equal number of messages from the same channel. Then, the coding process was applied on these new messages. This procedure was repeated until 1,000 messages were obtained for each channel, all having a label other than "Uninformative". Across all channels, a total of 1,061 messages were labeled as "Uninformative" during the labeling process.

During the coding process, when the content of a message was insufficient to determine a category, we used the list of contributors to the system's repository as a source of additional information that could give an insight into the nature of the message. One example of such ambiguous messages were questions which could be interpreted as either a customer asking about the system (Customer Support) or a developer of the system asking about a part of the system they are unfamiliar with (Team Q&A). In this particular case, if a question was made by a contributor to the system, it was classified as Team Q&A, and Customer Support otherwise.

The manual coding procedure took the two authors about 100 hours per person to complete (200 hours total), spread across three weeks. After completing the manual labeling, we obtained GitterCom, a dataset comprised of 10,000 Gitter messages, 1,000 per Gitter channel, classified according to their purpose.

4 Potential Research Applications

Previous studies have investigated the growing use of alternative communication means by developers [15, 16, 20]. The results of these studies show the rise of instant messaging tools and the impact they have on reshaping team dynamics and the communication landscape in increasingly distributed software development environments. Future studies could make use of GitterCom to study the relationship between open source development activity and communication trends. In particular, GitterCom enables further research to analyze and understand patterns in developer communications and to address important questions such as: How do software teams use tools like Gitter to communicate among themselves and with

[8] https://gitter.im/explore
[9] https://developer.gitter.im
[10] https://figshare.com/s/3fd5af0b869b8fd010bb

Channel	MessageId	Date-Time	Author	Message	Category	Subcategory
Cucumber	5551fd48f853	2015-05-12	amit007	Hi Team, I just recently upgraded our cucumber-jvm version ever since then I am getting following error	Dev-Ops	Development Operation Notifications
Cucumber	5551fd4a00ec	2015-05-12	amit007	Exception in thread \main\ java.lang.NoSuchMethodError: cucumber.runtime.Glue.removeScenarioScopedGlue()V\n\tat cucumber.runtime.java.JavaBackend.buildWorld(JavaBackend.java:131)	Dev-Ops	Development Operation Notifications
Cucumber	5551feedf853	2015-05-12	aslakhellesoy	@amit007 looks like you have inconsistent cucumber-* jar versions	Communication	Communication with Teammates
Cucumber	5552032a181	2015-05-12	LiohAu	So github is trying to replace irc :P	Communication	Communication with Teammates
Cucumber	55520e5900e	2015-05-12	amit007	@aslakhellesoy Thanks , seems like I was using older version of cucumber-core , updated the maven with specific version and things started working	Communication	Communication with Teammates
Cucumber	55637ddfbd9b	2015-05-25	Danon9111	Hi all! I have one question. With friends we are starting some project on studies. And we would like to write tests in cucumber. Can anybody say how we should start?	Dev-Ops	Team Q&A
Cucumber	55649744220	2015-05-26	DEllster	@Danon9111 You should integrate Cucumber with another testing framework which can allow you use some APIs to write Test scripts and execute them on the System Under test (Like Appium or Calabash) ... see this video : https://www.youtube.com/watch?v=fDV1p-byVeE	Dev-Ops	Team Q&A
Cucumber	556498018f55	2015-05-26	DEllster	@aslakhellesoy ... When reading \ The Cucumber Book\ i found the code below in Ruby . there is any way to convert it to Java ? \n----------\nCode : \nGiven /^an activated customer (\\w+) exists$/ do |name|\nsteps %{\nGiven I create a customer with login #{name}\nAnd I register the customer with login #{name}\nAnd I activate the customer with login #{name}\n}\nend	Dev-Ops	Team Q&A

Figure 1: GitterCom sample

other stakeholders? How do team dynamics reflect in team communications? Do developers exchange different types of messages at different times in the software life cycle? Do developers new to a project post different types of messages than the more senior developers?

GitterCom could also be used as a training dataset for machine learning approaches for automatically classifying new developer messages based on their purpose. This could, in turn, be useful to automatically organize messages into threads or to create summaries of developer conversations based on their purpose, such that developers that were away for a while or newcomers to a project could quickly catch up on important conversations they missed.

Another avenue for future work would be to use GitterCom in order to perform large scale replications of previous studies that analyzed developer communications in Slack [11, 12, 20], but used much smaller or restricted datasets (*e.g.*, communications in student projects or a particular software company). These replications on GitterCom could help corroborate previous findings or uncover new information about how developers communicate through instant messaging tools. One example of such work that could benefit from a large scale replication is work on the identification of messages that contain rationale for the decisions made by developers throughout the software life cycle [12]. Thus far, work on rationale has been limited to analyzing the chat messages of three student teams working on a multi-project capstone course.

5 Limitations and Future Improvements

Although GitterCom is *the largest data set of curated and manually labeled developer instant messages*, it still encompasses a small subset of all the existing Gitter developer communications. Therefore, one limitation to GitterCom could be that the collected projects are not representative of all open-source projects and that the most recent 1,000 messages for a project are not representative of all the messages exchanged by developers in a project. Improvements that would help increase the generalizability of the results of future studies analyzing this dataset include the expansion of the labeled data in GitterCom to include more messages from more projects.

For this purpose we also release the raw, unlabeled data extracted by our crawler script, containing over 2 million messages from 139 open source projects at https://figshare.com/s/3fd5af0b869b8fd010bb. We therefore hope other researchers will join our effort and will select more of this raw data to label and contribute to GitterCom.

6 Conclusions

The rapid adoption of instant messaging tools in open source development communities indicates a strong need to study the nature of these developer communications and their implications for open source software. However, such analysis is not possible without data to explore.

We introduced GitterCom, the largest manually labeled and curated dataset of Gitter developer messages. It comprises 10,000 messages and their corresponding purpose labels across multiple open source Gitter channels, corresponding to systems covering a wide range of application domains. We believe that our dataset provides immense opportunities for researchers to perform large scale empirical research and further analysis on developer discussions, communication with stakeholders, and team dynamics in open source systems. Our hope is nevertheless that the initial data set in this paper will spur interest for the continuing collection and analysis of developer instant communications.

7 Acknowledgments

Sonia Haiduc and Esteban Parra were supported in part by the National Science Foundation grants CCF-1846142 and CCF-1644285.

References

[1] [n.d.]. Cucumber. https://github.com/cucumber/cucumber
[2] [n.d.]. FreezingMoon. https://github.com/FreezingMoon
[3] [n.d.]. Imagej. https://github.com/imagej/imagej
[4] [n.d.]. jHipster. https://github.com/jhipster/jhipster/
[5] [n.d.]. jspm. https://github.com/jspm
[6] [n.d.]. Marionette. https://github.com/marionettejs/backbone.marionette
[7] [n.d.]. scikit-learn. https://github.com/scikit-learn/scikit-learn
[8] [n.d.]. TheHolyWaffle. https://github.com/TheHolyWaffle
[9] [n.d.]. UIkit. https://github.com/uikit/uikit
[10] [n.d.]. Xenko3d. https://gitter.im/xenko3d/xenko
[11] Rana Alkadhi, Jan Ole Johanssen, Emitza Guzman, and Bernd Bruegge. 2017. REACT: An Approach for Capturing Rationale in Chat Messages. In *Proceedings of the 11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'17)*. IEEE, Toronto, ON, Canada, 175–180.
[12] R. Alkadhi, T. Lata, E. Guzmany, and B. Bruegge. 2017. Rationale in Development Chat Messages: An Exploratory Study. In *Proceedings of the 14th IEEE/ACM International Conference on Mining Software Repositories (MSR'17)*. 436–446.
[13] Preetha Chatterjee, Kostadin Damevski, Lori Pollock, Vinay Augustine, and Nicholas A Kraft. 2019. Exploratory Study of Slack Q&A Chats as a Mining Source for Software Engineering Tools. In *Proceedings of the 16th IEEE International Conference on Mining Software Repositories (MSR'19)*. IEEE, Montreal, Canada, 490–501.
[14] Shaiful Alam Chowdhury and Abram Hindle. 2015. Mining StackOverflow to Filter out Off-topic IRC Discussion. In *Proceedings of the 12th IEEE Working Conference on Mining Software Repositories (MSR'15)*. IEEE, Florence, Italy, 422–425.
[15] Verena Käfer, Daniel Graziotin, Ivan Bogicevic, Stefan Wagner, and Jasmin Ramadani. 2018. Communication in Open-Source Projects-End of the E-mail Era?. In *Proceedings of the 40th IEEE/ACM International Conference on Software Engineering(ICSE'18)*. IEEE, Gothenburg, Sweden, 242–243.
[16] Bin Lin, Alexey Zagalsky, Margaret-Anne Storey, and Alexander Serebrenik. 2016. Why Developers Are Slacking Off: Understanding How Software Teams Use Slack. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW'16)*. ACM, 333–336.
[17] Esteban Parra. 2020. GitterCom, dataset. https://figshare.com/s/9b3df36e22a8a8f77169
[18] M. Storey, A. Zagalsky, F. F. Filho, L. Singer, and D. M. German. 2017. How Social and Communication Channels Shape and Challenge a Participatory Culture in Software Development. *IEEE Transactions on Software Engineering* 43, 2 (Feb. 2017), 185–204.
[19] Margaret-Anne Storey, Leif Singer, Brendan Cleary, Fernando Figueira Filho, and Alexey Zagalsky. 2014. The (R) Evolution of Social Media in Software Engineering. In *Proceedings of the 36th ACM/IEEE International Conference in Software Engineering, Future of Software Engineering (FOSE'14)*. ACM, Hyderabad, India, 100–116.
[20] Viktoria Stray, Nils Brede Moe, and Mehdi Noroozi. 2019. Slack Me if You Can!: Using Enterprise Social Networking Tools in Virtual Agile Teams. In *Proceedings of the 14th International Conference on Global Software Engineering (ICGSE'19)*. IEEE, Montreal, Quebec, Canada, 101–111.