Do Developers Really Know How to Use Git Commands? A Large-scale Study Using Stack Overflow

WENHUA YANG, College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China
CHONG ZHANG and MINXUE PAN, State Key Laboratory for Novel Software Technology and the Software Institute, Nanjing University, China
CHANG XU, State Key Laboratory for Novel Software Technology and the Department of Computer Science and Technology, Nanjing University, China
YU ZHOU and ZHIQIU HUANG, College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China

Git, a cross-platform and open source distributed version control tool, provides strong support for non-linear development and is capable of handling everything from small to large projects with speed and efficiency. It has become an indispensable tool for millions of software developers and is the de facto standard of version control in software development nowadays. However, despite its widespread use, developers still frequently face difficulties when using various Git commands to manage projects and collaborate. To better help developers use Git, it is necessary to understand the issues and difficulties that they may encounter when using Git. Unfortunately, this problem has not yet been comprehensively studied. To fill this knowledge gap, in this article, we conduct a large-scale study on Stack Overflow, a popular Q&A forum for developers. We extracted and analyzed 80,370 relevant questions from Stack Overflow, and reported the increasing popularity of the Git command questions. By analyzing the questions, we identified the Git commands that are frequently asked and those that are associated with difficult questions on Stack Overflow to help understand the difficulties developers may encounter when using Git commands. In addition, we conducted a survey to understand how developers learn Git commands in practice, showing that self-learning is the primary learning approach. These findings provide a range of actionable implications for researchers, educators, and developers.

CCS Concepts: • **Software and its engineering → Software development process management**; **Software configuration management and version control systems**; • **General and reference → Empirical studies;**

Additional Key Words and Phrases: Git commands, Stack Overflow, user survey

This work was supported by the National Natural Science Foundation of China (No. 61802179), the Leading-edge Technology Program of Jiangsu Natural Science Foundation (No. BK20202001), the National Natural Science Foundation of China (Nos. 61932021, 61972197, 61972193), and the Natural Science Foundation of Jiangsu Province (No. BK20201292).
Authors' addresses: W. Yang (corresponding author), Y. Zhou, and Z. Huang, College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, 29 Jiangjun Ave., Nanjing, Jiangsu Province, China, 211106; emails: {ywh, zhouyu, zqhuang}@nuaa.edu.cn; C. Zhang and M. Pan (corresponding author), State Key Laboratory for Novel Software Technology and the Software Institute, Nanjing University, 22 Hankou Road, Nanjing, Jiangsu Province, China, 210093; emails: 171250587@smail.nju.edu.cn, mxp@nju.edu.cn; C. Xu, State Key Laboratory for Novel Software Technology and the Department of Computer Science and Technology, Nanjing University, 163 Xianlin Ave., Nanjing, Jiangsu Province, China, 210023; email: changxu@nju.edu.cn.

1049-331X/2022/04-ART44 $15.00
https://doi.org/10.1145/3494518

ACM Reference format:
Wenhua Yang, Chong Zhang, Minxue Pan, Chang Xu, Yu Zhou, and Zhiqiu Huang. 2022. Do Developers Really Know How to Use Git Commands? A Large-scale Study Using Stack Overflow. *ACM Trans. Softw. Eng. Methodol.* 31, 3, Article 44 (April 2022), 29 pages.
https://doi.org/10.1145/3494518

1 INTRODUCTION

The **version control system (VCS)** is an essential part of the modern software team's everyday professional practices. It helps developers move faster and allows software teams to preserve efficiency and agility as the team scales to include more developers. Git is a distributed VCS for tracking changes in any set of files, originally designed for coordinating work among programmers cooperating on source code during software development [13]. Since its first release in 2005, Git has grown into the most widely used modern VCS and become the de facto standard of VCS in software development. Its distributed nature incurs itself superior performance characteristics and allows developers the freedom to experiment locally and publish their changes only when they are ready for distribution to the team. As Git is a distributed version-control system, it could be used as a server out of the box. There are many offerings of Git repositories as a service, e.g., GitHub, GitLab, Bitbucket, and SourceForge. According to recent reports of these platforms [20, 21, 33, 43], the total number of users on them is about 100 million. Specifically, GitHub, GitLab, Bitbucket, and SourceForge has more than 56, 30, 10, and 3.7 million users, respectively.

Using Git for version control can bring lots of benefits in software development. For example, a complete long-term change history of every file will be accessible; branching and merging can keep multiple streams of work independent from each other while also providing the facility to merge that work back together and enabling developers to verify that the changes on each branch do not conflict; it is possible to trace each change made to the software and connect it to project management and bug tracking software, and annotate each change with a message describing the purpose and intent of the change. Individual software developers who are accustomed to working with Git in their teams typically recognize the incredible value that Git brings, even on small solo projects. All these functions are fulfilled through using Git commands, such as `git add` and `git commit`.

Though Git has a tiny footprint, fast performance, and features that include cheap local branching, convenient staging areas, and multiple workflows, it comes with an often disputed and discussed issue that whether Git is easy to learn [9, 17]. We have seen Git elicits negative reactions from developers who complain that Git is difficult to learn, and they have to memorize a list of commands and apply them repeatedly without fully understanding their meaning [15]. On the one hand, the complexity of Git is an inevitable consequence of its greater power and flexibility compared to conventional, centralized version control systems. It leads to many difficulties that are experienced by developers. For example, there are many concepts in Git's information model, including files, working tree, index, local and remote repositories, remotes (pointers to remote repositories), commits, branches, stash, and so on, and developers need to understand them (at least some) before using Git. There are also complex dependencies among files, the branches, the repositories, and so on, which, even with those Git graphical user interface tools (e.g., SourceTree[1] and TortoiseGit[2]) that can make some of the dependencies visible, can still be challenging to

[1] https://www.sourcetreeapp.com/.
[2] https://tortoisegit.org/.

comprehend. Different operations on these different dependent concepts require different (combinations of) Git commands. On the other hand, due to time and resource constraints, developers in many cases can only learn to master a few basic commands to get by. However, they can barely survive for a while with basic commands, e.g., `git clone`, `git add`, `git commit`, and `git checkout`. Very soon they would need more advanced commands, e.g., `git rebase`, `git fetch`, `git merge`, and `git pull-request`. Therefore, many developers could encounter difficulties when using Git.

As it happens, we noticed that numerous Git command-related questions are being asked on Q&A forums. Take Stack Overflow, a large and popular Q&A site, for example, as of this writing, among all the questions on Stack Overflow, three of the top-five questions that have the most votes are about Git commands. Each of the three questions has received more than 10,000 votes. Moreover, they have been viewed for millions of times. The top question, "How do I undo the most recent local commits in Git," [36] has obtained more than 22k votes and 9m views. This clearly reveals an understudied issue that developers still have a lot of confusion about the use of Git commands.

It is imperative to understand the problems and difficulties that developers may have when using Git in developing software, since Git has a huge base of users and is used frequently by developers in their daily software development. Unfortunately, this topic has not been well studied by any prior work. Considering that developers often ask questions about their problems and confusions on professional Q&A forums, understanding Q&A characteristics related to Git commands is of substantial help. Hence, this article presents the first comprehensive empirical study on the questions about Git commands raised by developers on Stack Overflow. Specifically, to understand what struggles that developers may face when they use Git commands, we analyze the relevant posts from developers on Stack Overflow. The forum allows developers to seek technical advice from other developers and experts. Thus, it has been a common practice for researchers to understand the interests and difficulties of developers when dealing with different engineering tasks from Stack Overflow posts, as shown in recent work [4, 6, 14, 35, 45, 50]. This study fills the knowledge gap in the literature with a large-scale investigation of posted questions, identifying Git command-related problems and difficulties reported by developers. Given the widespread use of Git and its importance, this study can aid developers to prepare themselves for similar difficulties and make educators and researchers better positioned to help developers use Git in a more targeted way. For example, it can help developers know which commands are commonly used but challenging to others so they can adjust their study plans accordingly, help educators better understand students' educational needs and find ways to best support them through the learning process in practice (e.g., provide a clearer explanation about the frequently-asked commands), and help software engineering researchers determine avenues for future research (e.g., propose assistance for developers when they have trouble choosing Git commands).

Besides, we are also curious about how developers learn to use Git commands, since from the results of studying Stack Overflow posts, many developers seem to have no systematic training on how to use Git. Understanding the learning approaches taken by developers can have many benefits. It enables potential new developers to make systematic learning decisions and also researchers and educators to improve the state-of-the-art in software engineering and teaching approaches. To that aim, we survey developers that have various experiences in using Git, since a survey is an ideal instrument as developers have first-hand experiences of their learning process for Git commands.

In summary, our study collects and analyzes 80,370 Stack Overflow posts regarding Git commands and surveys 92 developers with experiences of using Git commands both from academia and industry. To understand the problems and difficulties that developers may have in using Git commands, based on the collected posts and the survey, we focus our study on the following research questions:

- *RQ1: Question popularity trend.* How many questions and questioners are related to Git commands, and what are the trends in these numbers over the years? This RQ aims to answer whether it is common for developers to encounter problems in using Git commands.
- *RQ2: Questioner distribution.* What is the distribution of questioners regarding their numbers of years of registration on Stack Overflow? This RQ is to answer whether the questioners asking about Git commands are novices or developers with several years of development experience.
- *RQ3: Command popularity.* What Git commands are more popular among the questions asked by developers? The purpose of this RQ is to find those Git commands that developers are more likely to have questions about.
- *RQ4: Command difficulty.* What Git commands are more difficult to find answers to their questions? This RQ aims to answer which Git commands are more difficult to the developers.
- *RQ5: Learning approaches.* How do developers learn to use Git commands? This RQ is designed to understand the approaches developers adopt to learn Git commands.

A few findings of our study are the following. ***Question popularity trend:*** (1) There are in total more than 80,000 Git command-related questions on Stack Overflow, and over the years, the percentage of Git command-related questions and questioners has remained relatively stable as the number of questions on Stack Overflow has been growing. ***Questioner distribution:*** (2) Many questioners of Git command-related questions have been registered on Stack Overflow for quite a long time: About 40% of the questioners had been registered for more than four years when asking the questions since 2017. ***Command popularity:*** (3) Git commands (e.g., *git revert* and *git reflog*) about recovery are among the most viewed commands. Git commands are often used together in combinations, with the combination of five commands accounting for the largest share. ***Command difficulty:*** (4) Some of the seldom-used Git commands (e.g., *git pack-redundant* and *git http-push*) have the highest percentage of no accepted answers, while for the more frequently-used commands, *git credential* and *git submodule* are among the most difficult ones. ***Learning approaches:*** (5) Among all the learning approaches used by our respondents, 81.7% are self-learning ones, e.g., learning from the documentation and the internet. In general, our study leads to the following primary contributions:

(a) We conduct an empirical study on Stack Overflow to investigate questions related to Git commands. To the best of our knowledge, it is the first large-scale study to investigate Q&A characteristics related to Git commands on Stack Overflow.
(b) We report several interesting and valuable conclusions concerning the popularity, questioner distribution, and difficulty of Git command-related questions and analyze the results of a survey that explores how developers learn Git commands in practice.
(c) We provide actionable implications derived from our results for researchers, educators, and developers, which can help them better decide when and where to focus their efforts on learning or building supports for Git commands.

Furthermore, this article offers a dataset of posts related to Git commands[3] as an additional contribution to the research community for other researchers to replicate and build upon. Other supplemental materials including the collected data and complete analyzed results are also available online.[3]

Organization of the article. Section 2 describes our research methodology. The experimental results are presented and discussed in Sections 3. Section 4 discusses the implications of our findings.

[3]https://github.com/gitcommandstudy/gitcommands.

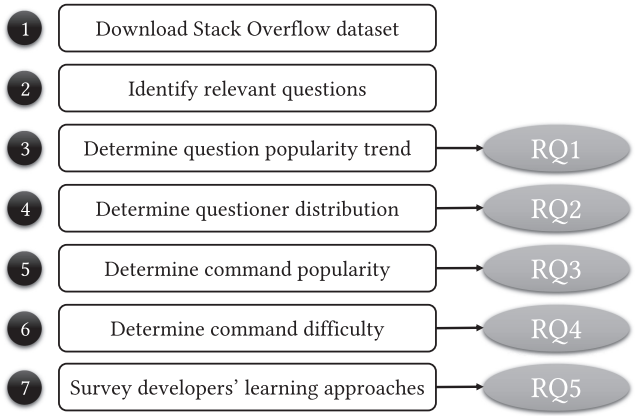


Fig. 1. An overview of the methodology.

Section 5 talks about the threats to the validity of our results. Section 6 discusses the related work. Finally, Section 7 concludes the article.

2 METHODOLOGY

In this section, we describe the methodology used in the study. We analyze the relevant questions posted on Stack Overflow, where developers seek technological advice from other developers and experts and survey developers that have various experiences in using Git command. Therefore, our research methodology consists of a quantitative study of Stack Overflow posts and a survey of developers. We show an overview of the methodology of our study in Figure 1 and detail each step in the following.

2.1 Quantitative Study

In this section, we first introduce the quantitative study design that consists of the following six steps.

Step ❶: Download the Stack Overflow dataset. In the first step of our study, we download the publicly available Stack Overflow dataset from Stack Exchange Data Dump [22] dated to December 7, 2020, which covers the Stack Overflow posts generated from July 31, 2008, to December 7, 2020. The dataset, which we denote as $\mathcal{S}$, includes a large set of "question and answer" posts. Each post has a set of data, including its identifier, its type (i.e., question or answer), creation date, tags, title, body, view count, score, favorite count, and the identifier of the accepted answer for the post if the post is a question. Besides, one to five tags can be attached to a post specifying its topics. The contributor who posted a question can mark an answer as accepted to the question. There are 51,296,931 posts in $\mathcal{S}$, among which 20,611,833 (40.2%) are questions and 30,685,098 (59.8%) are answers.

Step ❷: Identify relevant questions. To identify questions related to Git commands from Stack Overflow, we leverage the tags of questions. We traverse the dataset $\mathcal{S}$ to find the questions whose tags contain the term "git." In total, we extract 198,626 questions and denote them as the set $\mathcal{T}$. However, since "git" covers broad topics, there are some questions tagged with "git" but not related to Git commands. For instance, question "*Where can I report a GitHub bug*" [40] is tagged with "git" and "github," but it has nothing to do with Git commands. Therefore, we refine $\mathcal{T}$ by identifying questions that are actually relevant to Git commands. The refinement idea is to further determine whether the titles, bodies, or the accepted answers of those questions with the "git" tag contain Git

commands. Specifically, we collect all the Git commands from Chacon and Straub's Pro Git book [13]. There are 13 classes of Git commands (e.g., setup and config, branching and merging, and patching), including about 140 Git commands,[4] with some differences in the number of commands depending on the Git version. We use the Git commands in the more recent Git version 2.30.0 as keywords to identify relevant questions. Then, we perform a case-insensitive search of the keywords within the title and body of each question and its accepted answer in $\mathcal{T}$ and denote the questions that contain at least one of the keywords as the set $\mathcal{P}$. During the search, we use exact string matching, which means that we require the question's title, the body, or the accepted answer contains strings that exactly match the Git command (e.g., `git add`), but we do not distinguish between the case of the strings. Finally, we have 80,370 questions relevant to Git commands in the set $\mathcal{P}$.

To further confirm that the identified questions are related to Git commands, we randomly sample 300 questions from the set $\mathcal{P}$ in which the questions are identified as relevant and 300 questions from the questions in the set $\mathcal{T}$ but not in the set $\mathcal{P}$ (i.e., identified as irrelevant). Then, two of the authors of this article independently check whether these sampled questions are relevant to Git commands or not and then mutually agreed on a final result. All 300 questions classified as relevant and 299 questions classified as irrelevant were determined as correctly classified by manual analysis. The one incorrectly classified question is due to the reason that although a Git command was mentioned in the question, the full Git command did not appear. The level of inter-rater agreement using Cohen's Kappa score is 0.844, indicating almost perfect agreement and demonstrating the reliability of our identification schema and procedure.

Step ❸: Determine question popularity trend. To illustrate the popularity trend of Git commands, following previous work [4, 6, 35, 45, 50], we measure the question popularity trend using five metrics. The first is the number of questions related to Git commands per year. The second metric is the number of users who raised Git command-related questions per year. The third metric is the average number of views by registered users and visitors of the Git command-related questions asked in each year. The fourth metric is the average number of favorites marked by users of the Git command-related questions asked in each year. The fifth metric is the average score of the Git command-related questions asked in each year. These metrics can represent how much attention Git command-related questions are getting on Stack Overflow and thus can measure the question popularity trend. Intuitively, a topic with a higher number of views, favorites, a higher score, and more related posts is more popular. The metrics are calculated based on the post set $\mathcal{P}$ for each of the past 13 years, i.e., from 2008 to 2020. Specifically, we traverse all the questions in $\mathcal{P}$ and count the number of questions and the number of questioners for each year from 2008 to 2020 based on the "CreationDate" and "OwnerUserId" fields in the post, respectively. Afterward, for each year's questions, we extract the number of views, favorites, and scores of each question from the "ViewCount," "FavoriteCount," and "Score" fields of the post, and then calculate the average number of views, average number of favorites and average scores of each year's questions. To better understand the trend of Git command-related questions on Stack Overflow, we also compare the global Stack Overflow question popularity trend (i.e., based on the post set $\mathcal{S}$) with the Git command trend in terms of the above metrics. This step answers the first research question RQ1.

Step ❹: Determine questioner distribution. Git is a set of command-line utility programs that are designed to execute on a command-line environment. Git is claimed to be easy to learn according to its webpage,[4] but this claim has been in dispute [9, 17]. We have seen complaints that Git is difficult to learn and use even for developers with years of development experience [15]. Similar descriptions can also be found on Stack Overflow, in which developers mentioned that they

[4] https://git-scm.com/.

had doubts about using Git even though they had years of development experience [37, 41]. Yet, these need to be further confirmed with more evidence from Stack Overflow. Considering that Stack Overflow is a forum specifically for developers to seek technological advice, its registers must have already been involved in software development for some time when they make the registration. The distribution of developers' length of registration time can help understand the relationship between the developers' software development experience and their proficiency in using Git commands. Hence, in this step, we analyze the questions related to Git commands to determine the distribution of questioners regarding their years of registration based on the set $\mathcal{P}$. Specifically, we count the number of years that developers had been registered on Stack Overflow when they asked the Git command-related questions. To achieve this, we first extract the ID of the questioner from the "OwnerUserId" of the post and then obtain the questioner's registration time from his/her profile on Stack Overflow to find out how long he/she had been registered when the question was asked. Furthermore, we compare the distribution for all questioners on Stack Overflow with questioners raising Git command-related questions to allow readers to interpret these numbers in a broader context. That is, we also count how long the questioners in $\mathcal{S}$ had been registered on Stack Overflow at the time they asked the question by following the above procedure. Step 4 answers the second research question RQ2.

Step ❺: Determine command popularity. In this step, we measure the popularity of a Git command among the Stack Overflow posts using four metrics that have been used by existing work [3, 6, 35, 45, 50] based on the set $\mathcal{P}$. The first metric is the average number of views of the questions that have the Git command appear in them or their accepted answers. Views by registered users and visitors of Stack Overflow are both considered, since the number of visitors is much more than the number of registered on Stack Overflow [32]. Multiple commands can appear in a question or its accepted answer. Since they are all closely related to the question, these commands should all take credit when calculating the number of views. This rule also applies to the following metrics. The second metric is the average number of favorites marked by users of questions with the command that appears in the questions or accepted answers [3, 6, 35, 45, 50]. The third metric is the average score of questions of the command that appears in the questions or accepted answers [3, 6, 35, 45, 50]. The fourth metric is the number of questions that contain the Git command. This metric is used because some commands may have higher values of average views, favorites, or scores by merely appearing a few times in several popular posts, and using this metric can help filter out such cases. To obtain the values for these metrics, we first need to check which Git commands are included in the question or its accepted answer. Then, for each Git command, we use the above information to select all the questions that contain the command in the question or its accepted answer. With all the questions related to a command, we extract the number of views, favorites, and score from the "ViewCount," "FavoriteCount," and "Score" fields of each question and then calculate the average number of views, favorites, and score for the command. Intuitively, a Git command with a higher number of views, favorites, a higher score, and more posts is more popular. Since not all the Git commands are available in every Git version, our analysis for RQ3 and RQ4 is limited to those commands that are available in all the Git versions. We have checked and determined that 136 commands are available in all recent versions of Git, and the complete list of these 136 commands and their corresponding experimental results are accessible in our supplemental materials.[3] Step 5 answers the third research question RQ3.

Step ❻: Determine command difficulty. We measure the difficulty of Git commands using five metrics adopted from previous work [3, 5, 50]. The first one is the percentage of questions with no accepted answer. The second metric is the percentage of questions with no answer, that is, these questions did not get any answers at all. The third metric is the median response time needed for questions to receive an accepted answer. The fourth metric is the number of questions that

contain the Git command. To investigate whether the Git command-related questions considered difficult are more likely to be raised by developers with more development experience, we count the average length of time the questioner had been registered on Stack Overflow at the time the question was asked, which is the fifth metric. In this step, we also use the questions in the set $\mathcal{P}$ for analysis and consider commands that have been available in all Git versions. Similarly to RQ3, each question is mapped to every Git command that appears in the question or its accepted answer. That is, when we determine the difficulty of each Git command, we will consider every question that the Git command appears in the question itself or its accepted answer. For a question post related to a Git command, there are two fields "AcceptedAnswerId" and "AnswerCount" in the post indicating the ID of the accepted answer if it exists and the number of answers, respectively. For an answer post, we obtain its creation time from its "CreationDate" field, which will be used to calculate the response time needed for the question to receive an accepted answer by comparing it with the "CreationDate" of the question post. Using the above information, we can obtain the values of the first four metrics, while the value of the fifth metric can be similarly calculated according to the method introduced in step 4. Intuitively, a Git command with less accepted answers received in a longer amount of time for more posts is more difficult. The average length of time that the questioners of the Git command had been registered on Stack Overflow can further help determine if the command causes difficulties for developers with years of development experience as well. Step 6 answers the fourth research question RQ4.

2.2 Survey

The above six steps are about the quantitative study of Stack Overflow posts. Those research questions mainly focus on the questions that developers asked about Git commands. The reason why developers asked questions is that they have doubts about the use of Git commands, i.e., they are encountering difficulties in using Git commands. In general, there should be a causal relationship between the difficulties and the developer's approach to learning Git commands. For example, developers may not be learning Git commands properly or may not be putting enough effort into it, so in this step, we designed an online survey to learn from real-world developers about their approach to learning Git commands and to elicit their suggestions for using Git commands. The online survey is detailed in the following step. Step 7 answers the fifth research question RQ5.

Step ❼: Survey developers' learning approaches. The questions in the survey were basically related to the developers' approaches for learning Git commands. In the meantime, to have a more comprehensive understanding of the developers who participated in the survey, we also asked some other related questions. The experience of developers using Git is an important type of information, so we surveyed the developers for the number of years they had been using Git and their own ratings of their Git usage ability level. We have divided the ability to use Git into five levels, which are novice, advanced beginner, competent, proficient, and expert. Through these questions, we can understand developers' self-evaluation of their ability to use Git, and the collected results can serve as the basis for further analysis. Afterward, we further surveyed the main approaches developers adopted to learn Git commands and explored the possible relationship between developers' learning approaches and their ability to use Git commands. We listed some common learning approaches (i.e., learning during the class, learning in online courses, learning from peers or seniors, self-learning from the documentation, and self-learning from the internet) but also allowed participants to specify other learning approaches that were not listed. Then, we surveyed the developers' demographics, including developers' professional area and education levels, and explored the possible relationship between developers' demographics and learning approaches. To further elicit the contributors' opinions, in some questions with predefined answers, we also included an optional comment box to encourage the respondents to provide other options that we

did not offer or reasons for their choices. Therefore, the questions in the survey are classified into four categories: (1) the developers' professional area (e.g., academia and industry) and education level, (2) their experience and expertise level of using Git commands, (3) their learning approaches for Git commands, and (4) their suggestions for potential new developers on learning Git commands. Except for the last one, which is an optional open-ended question, all the other questions are multiple-choice questions.

Target participants. The survey is released on SurveyMonkey,[5] which is one of the most famous web services for online surveys. We sent the link[6] of the survey to the participants. The target participants consist of both practitioners and researchers. The practitioners were those who had recently asked Git-related questions and listed GitHub accounts on Stack Overflow. We used the GitHub accounts to find their email addresses and send invitations.[7] We also sent invitations to some researchers who had their Git repositories listed on their websites. We have pilot-tested the survey. We asked five graduate students to independently complete the survey in one go, in the same way as we did on SurveyMonkey, and to raise questions if they had any doubts. We then improved the survey based on their feedback. We informed participants that we would reward their answers by donating a total of 150 dollars, proportionally divided to the three different institutions that participants could choose (GLOBAL IMPACT, UNICEF, and the WWF).

Respondents. In total, we have successfully sent invitations to 508 participants. Within a period of 45 days, we have received 92 responses (74 from the questionnaire in industry and 18 from the questionnaire in academia). Respondents vary regarding how many years of Git using experiences they have, with a range between 1 and 15 years of Git using experience (median = 8 years). Since we did not require the participants to fill in the comment field for all the questions, the number of received comments for different questions varied. Finally, 65 comments were received in total. We further categorized the collected comments and provided detailed results in Section 3.5. Related materials including the survey form and the analysis results of the survey are available online.[3]

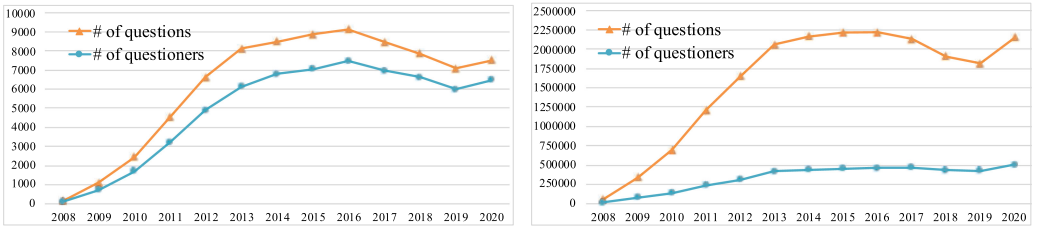
3 RESULTS

In this section, we present and discuss the results of our study for the five research questions RQ1–RQ5.

3.1 RQ1: Question Popularity Trend

The number of questions and the number of questioners are the most direct indicator of whether it is common for developers to encounter problems in using Git commands. Figure 2(a) shows the popularity trend of Git command-related questions in terms of the number of questions and the number of developers who asked the questions on Stack Overflow during the past 13 years. Since a questioner may ask more than one question each year, the number of questioners is less than the number of questions. As we can see from Figure 2(a), Git command is gaining increasing attention, demonstrating the timeliness and necessity of this study. In particular, the number of related questions and questioners has been increasing sharply until 2016. Although the total number has dropped a bit from 2017, it has still remained at a high level (more than 7,000 questions and 6,000 questioners each year). Moreover, we again observe a significant increase of the number in 2020. In fact, we find that the number of all questions on Stack Overflow also decreased from 2016 to 2019, as shown in Figure 2(b). After years of development of Git, the number of Git command-related

[5] https://www.surveymonkey.com.
[6] https://www.surveymonkey.com/r/3TS3CSS.
[7] We were unaware that it was discouraged to send emails to users by GitHub's policy for information usage restrictions. We do not recommend subsequent researchers to follow this practice.



(a) Git commands (b) Stack Overflow

Fig. 2. The trend of the numbers of questions and questioners over the years.

Table 1. Ratio of Git Command-related Questions and Questioners among Stack Overflow Questions and Questioners over Time

Year	*2008*	*2009*	*2010*	*2011*	*2012*	*2013*	*2014*	*2015*	*2016*	*2017*	*2018*	*2019*	*2020*
% of Qns.	0.2%	0.3%	0.4%	0.4%	0.4%	0.4%	0.4%	0.4%	0.4%	0.4%	0.4%	0.4%	0.3%
% of Qnrs.	0.6%	0.9%	1.2%	1.4%	1.6%	1.5%	1.5%	1.5%	1.6%	1.5%	1.5%	1.4%	1.3%

questions and questioners is still large, showing that a considerable amount of developers have been unclear about the usage of Git commands.

For comparison, we also counted the number of all questions and the number of users who raised questions on Stack Overflow in the past 13 years based on Stack Overflow's official dataset $\mathcal{S}$. The results are shown in Figure 2(b). Based on the analyzed data, we calculated the growth rate of the number of all questions and questioners on Stack Overflow and Git command-related questions and questioners, respectively. Prior to 2013, the growth rate of the number of Git command-related questions in each year (666%, 121%, 87%, and 46%) was larger than the one of all questions on Stack Overflow (491%, 102%, 74%, and 37%). Afterward, they grew at about the same rate, alternately leading. In terms of the growth rate of the number of questioners, Git commands' rate was larger than the one of Stack Overflow until 2016, and after that, they also alternated the lead. This indicates that Git commands are always being lively discussed by developers. We also calculated the ratio of Git command-related questions and questioners among all the Stack Overflow questions and questioners over time. As shown in Table 1, the ratio of Git command-related questions ("% of Qns.") and questioners ("% of Qnrs.") on Stack Overflow has been growing rapidly for the first few years and then has been relatively stable. The slightly lower ratio for 2020 is due to the fact that the data in the dataset is only available until early December 2020 (i.e., one month of data is still missing for 2020). Of particular note is the difference between the number of questions and the number of questioners. For Stack Overflow, the difference is consistently large in each year, which suggests that many developers have raised more than one question. For Git commands, however, the difference is relatively much smaller, suggesting that many questions are raised by different developers. We can also learn this from Table 1, where the ratio of questioners is larger than the ratio of questions. This confirms from another perspective that many developers have faced difficulties when using Git commands.

To measure the popularity of Git command-related questions from a wider perspective, we calculated the average views, favorites, and scores of Git command-related questions. The results are shown in Table 2. Rows 2, 4, and 6 present the average views, average favorites, and average scores for Git command-related questions asked in each year from 2008 to 2020, respectively. Since these metrics are calculated from the time the questions were asked to December 2020, we see that the average views, favorites, and scores for questions asked in earlier years are generally larger than

Table 2. Average Views, Favorites, and Scores of Questions Asked in Each Year

Year of Qns. asked	*2008*	*2009*	*2010*	*2011*	*2012*	*2013*	*2014*	*2015*	*2016*	*2017*	*2018*	*2019*	*2020*
Git Avg. View	272587.3	104297.9	47228.8	25537.5	15300.4	8590.9	4541.1	2922.7	3069.3	1927.8	1067.5	449.4	112.5
SO Avg. View	32425.2	13689.5	8060.3	5937.0	4498.8	3608.1	2187.8	1685.7	1444.1	1192.0	815.0	394.7	102.2
Git Avg. Favorite	245.59	67.66	26.3	12.62	5.87	3.31	1.70	1.07	1.04	0.64	0.48	0.32	0.21
SO Avg. Favorite	14.96	4.42	2.19	1.38	0.95	0.70	0.48	0.39	0.34	0.29	0.22	0.15	0.08
Git Avg. Score	673.64	200.76	81.48	42.66	20.64	10.89	5.71	4.06	3.89	2.25	1.78	1.05	0.46
SO Avg. Score	39.53	13.13	7.14	4.77	3.35	2.44	1.68	1.60	1.38	1.11	0.91	0.67	0.29

those for questions asked in later years. This is because the earlier questions have fewer numbers but have been viewed for a longer period of time. For comparison, we also counted the average views (Row 3), average favorites (Row 5), and average scores (Row 7) of all Stack Overflow questions asked in each year. It shows that Git command-related questions have larger values on all three metrics than all questions on Stack Overflow for each year, especially for the questions raised in previous years. Besides the metrics presented in Table 2 that are calculated by questions asked in each year, we also have the results for the total Git command-related questions (80,370) and the total Stack Overflow questions (over 20 million) from 2008 to 2020: The average views, average favorites, and average scores are 8,440, 4.3, and 13.7 for all the Git command-related questions, and 2,444, 0.6, and 2.1 for all the Stack Overflow questions. From these values of the above five metrics, we can conclude that Git command-related questions are popular on Stack Overflow.

Answer: There are in total over 80,000 Git command-related questions on Stack Overflow from 2008 to 2020, many of which were asked by different developers. Over the years, the percentage of Git command-related questions and questioners has remained relatively stable as the number of questions on Stack Overflow has been growing. Meanwhile, Git command-related questions have higher average numbers of views, favorites, and scores than all questions on Stack Overflow, indicating the popularity of Git command-related questions on Stack Overflow.

3.2 RQ2: Questioner Distribution

As we mentioned above, there are many developers who have asked questions related to the usage of Git commands. Based on this result, a natural question to ask is whether the developers who raised Git command-related questions were those with years of software development experience or not. To answer this question, we analyze how many years the questioners had been registered on Stack Overflow when they asked those Git command-related questions. For comparison, we also analyze the questioner distribution for all questioners on Stack Overflow. For the Git command-related questions in $\mathcal{P}$, we have 64,193 questioners, of which 63,168 are users with registration information on Stack Overflow. The remaining 1,025 are deleted or anonymous users, since registration is not required to participate on Stack Overflow. For all the questions in $\mathcal{S}$, we have analyzed 6,884,984 questioners' registration information on Stack Overflow. Table 3 shows the distribution of the number of years these questioners had been registered at the time of raising questions in each year from 2008 to 2020. We divide the number of years of registration into six intervals, e.g., less than 1 year and 1–2 years, as shown in the first row of Table 3. Starting from the second row, the table shows the percentage of questioners in each time interval for Git command-related questioners and all questioners on Stack Overflow, respectively. Specifically, for each year's questions, we recorded the time that the questioners had been registered in that year. Then, based on the time difference, we counted the percentage of questioners in each of the six time intervals. Take the second row for example. In 2008, the gap between all questioners' time of registration and time of asking questions was all less than one year since Stack Overflow started operating in 2008, making 100% of the questioners fall into the shortest time interval. Due to the

Table 3. Distribution of Questioners' Registration Time in Each Year for Git Command-related Questions and All Stack Overflow Questions

Year	<1 year		1-2 yrs		2-3 yrs		3-4 yrs		4-5 yrs		>5 yrs	
	Git	SO	Git	SO	Git	SO	Git	SO	Git	SO	Git	SO
2008	100%	100%	0	0%	0	0%	0	0%	0	0%	0	0%
2009	91.6%	98.8%	8.4%	1.2%	0	0%	0	0%	0	0%	0	0%
2010	64.9%	89.9%	32.2%	9.6%	2.9%	0.4%	0	0%	0	0%	0	0%
2011	50.3%	83.7%	29.6%	11.5%	18.2%	4.6%	1.9%	0.2%	0	0%	0	0%
2012	42.3%	77.6%	26.8%	12.9%	19.7%	6.7%	10.4%	2.8%	0.8%	0.2%	0	0%
2013	34.7%	71.6%	25.2%	15.1%	19.5%	7.5%	13.3%	4.0%	6.6%	1.7%	0.7%	0.1%
2014	29.0%	63.7%	21.3%	16.2%	20.7%	10.5%	14.2%	5.4%	9.7%	2.9%	5.0%	1.3%
2015	25.5%	57.1%	17.2%	15.3%	17.8%	12.0%	16.5%	8.0%	11.3%	4.2%	11.7%	3.3%
2016	25.0%	53.3%	13.5%	13.3%	14.7%	11.6%	15.5%	9.5%	13.3%	6.3%	18.0%	6.1%
2017	23.5%	50.1%	12.9%	12.9%	12.1%	10.0%	12.5%	9.2%	12.2%	7.7%	26.9%	10.2%
2018	23.2%	46.9%	10.9%	12.2%	10.4%	9.8%	10.4%	8.1%	10.5%	7.7%	34.6%	15.4%
2019	22.3%	46.1%	9.9%	11.1%	9.5%	9.2%	8.4%	7.7%	8.6%	6.5%	41.3%	19.4%
2020	27.1%	47.5%	9.4%	9.9%	7.7%	8.3%	7.9%	7.1%	8.0%	6.2%	40.0%	21.2%

same reason, the majority of the questioners from 2008 to 2012 fall into the short time intervals (e.g., <1 year and 1–2 years). Since then, the percentage of long-registered questioners has been growing steadily both for Git command-related questions and Stack Overflow questions. In general, the percentage of questioners of Git command-related questions in the longer time intervals (e.g., 2–3 years, 3–4 years, and 4–5 years) is significantly larger than the percentage of questioners of all questions on Stack Overflow, while the percentage of questioners of all questions on Stack Overflow is consistently higher than the percentage of questioners of Git command-related questions in the first time interval (i.e., <1 year). In particular, the gap between the percentages of questioners of Git command-related questions and all Stack Overflow questions in the longest time interval (i.e., >5 years) increases every year since 2013. From 2017 onward, the largest percentage of Git command-related questioners in each year is the users registered for more than five years, while the largest percentage (about 50%) of questioners on Stack Overflow for all questions is still in the shortest time interval (i.e., <1 year). By 2020, 40.0% of the Git command-related questioners are users who had been registered for more than 5 years, and the percentage of all questioners on Stack Overflow in this time interval (>5 years) is only 21.2%. It is clear that many of the questioners who had been registered for a long time still have doubts about the use of Git commands. Note that this is a conservative statement: The questioners should have already started developing software or even been developing for some time when they make the registration on Stack Overflow, since Stack Overflow is a forum specifically for developers to seek advice for technical problems.

Therefore, we can learn that among the many questioners, there are not only developers who have just started programming but also those who have been programming for years. In light of these results, we can say that Git commands are not easy to learn and master, even for developers with years of programming experience. Take a question [37] on Stack Overflow for example. In the question, the questioner said that "*I've been developing for several years now, and I've never had the time to learn about version control. Renaming directories with different version names always seemed enough. Now, I've finally decided to learn it, but some basic terminology and working principles still confuse me.*" In addition to developers with years of development experience, those with years of experience using Git also have doubts when learning and using Git commands, as illustrated by a questioner [41] on Stack Overflow: "*I'll preface this by mentioning that I've been working with*

Git for years, but my knowledge is limited to very basic workflows. Realizing this, I've been getting a lot better with "advanced" Git features, but here's a question I can't quite figure out: What exactly does git checkout [file] do?" Therefore, our findings reveal that it is common for developers to have doubts about using Git commands, which explains the importance and necessity of this study.

Answer: Many questions related to Git commands are asked by developers who have been registered on Stack Overflow for a long time. After 2017, about 40% of the questioners of Git command-related questions had been registered for more than four years at the time of asking the questions, compared to that over 50% of all questioners on Stack Overflow had been registered for less than 2 years. This suggests that even developers with years of development experience can have trouble using Git commands.

3.3 RQ3: Command Popularity

Since there are over a hundred Git commands, we investigate which Git commands are the most popular ones being asked on Stack Overflow. Answers to this research question could help understand with what Git commands developers are more likely to be confused. The popularity of a Git command is measured using the average number of views, the average number of favorites, the average scores, and the total number of related questions. Among the above four metrics, we use the average number of views as the main metric, since a popular question tends to attract more developers to view. Still, the other metrics also have reference values to estimate the popularity of commands. Table 4 shows the results of commands' popularity, sorted by the average number of views. Meanwhile, we filter out those commands with less than 200 questions in this table, because some commands only appeared a few times in some popular posts along with other commands, as discussed in Section 2.1. For example, command "`git verify-tag`" only appears in one popular post [38] with other commands and is not included in any other posts. Although it has a high average number of views, we do not consider it as popular. Due to space limitations, we list the Git commands with the top 30 views in this table, and the complete result for all commands can be found in the supplemental materials.[3]

According to Table 4, we notice that "`git revert`," "`git reflog`," "`git stash`," "`git clean`," and "`git reset`" are the top five most popular commands being asked. To recap, `git revert` is a forward-moving undo operation that offers a safe method of undoing changes, `git reflog` is an important command for recovering from local errors, `git stash` takes uncommitted changes (both staged and unstaged), saves them away for later use, and then reverts them from the working copy, `git clean` is a method for deleting untracked files in a repository's working directory, and `git reset` is a powerful command that is used to undo local changes to the state of a Git repository. As we can see, except for "`git clean`," all the commands are related to recovery. In addition, the average number of favorites and the average scores of the questions related to these commands are also ranked at the top, which further indicates that they are very popular. On average, each Git command-related question for the 30 listed commands receives 12,254 views, showing that developers indeed value these Git commands. Furthermore, we calculated the average number of views for the studied 136 commands, which is 7,326. The average number of views for these commands are higher than other well-studied topics, such as concurrency (*average views 1,641*) [3], big data (*average views 1,364*) [5], and security (*average views 1,696*) [50].

It is worth mentioning that since a question and its accepted answer may involve multiple Git commands and each command is relevant to the question, the question is taken into account when calculating the above metrics for each involved Git command. That is, each command takes credit for the question's number of views, favorites, and the score, as introduced in Section 2.1. It is common for multiple Git commands to be used together in Git. Figure 3 shows our statistics on the

Table 4. Popularity Measures of Git Commands

Command	*Views*	*Favorites*	*Score*	*Questions*
`git revert`	21,726	10.0	28.5	1,289
`git reflog`	20,330	14.0	42.4	1,282
`git stash`	19,202	10.5	33.9	2,113
`git clean`	18,592	10.0	31.5	827
`git reset`	17,544	8.9	27.9	6,987
`git help`	17,343	8.6	30.0	682
`gitk`	15,234	12.7	28.9	1,154
`git mergetool`	15,110	8.7	25.4	488
`git fetch`	12,843	6.4	19.6	6,366
`git branch`	12,089	6.5	20.4	9,048
`git config`	11,853	5.8	18.4	7,240
`git update-ref`	11,220	11.4	26.4	333
`git pull`	10,752	4.9	14.9	10,996
`git apply`	10,496	4.9	17.6	524
`git rm`	10,441	6.0	17.9	3,521
`git commit`	10,416	5.5	15.9	15,311
`git checkout`	10,356	5.4	16.4	16,505
`git show`	10,302	5.4	18.9	2,290
`git difftool`	10,038	5.6	18.6	395
`git status`	9,947	4.4	15.1	8,031
`git push`	9,803	5.0	13.9	17,977
`git diff-tree`	9,476	6.1	18.5	210
`git format-patch`	9,472	7.2	22.0	452
`git merge`	9,318	5.4	15.9	8,006
`git add`	9,296	4.6	14.5	12,659
`git log`	9,040	5.1	16.2	8,313
`git remote`	9,018	5.0	12.9	7,374
`git init`	8,912	4.3	11.2	4,535
`git diff`	8,741	4.9	17.4	5,613
`git tag`	8,703	5.4	18.1	1,708
Average	***12,254***	***7.0***	***21.0***	***5,408***

combinational use of Git commands based on the data in the set $\mathcal{P}$. As we can see from Figure 3, only 17% of the cases have only one command appearing alone, and the rest are all used in combination. Among them, the most common combination is five commands together, accounting for 22%, followed by four commands together, accounting for 15%.

The Git commands can be used in a variety of combinations, both in terms of the number of commands and the type of commands in a combination. To further explore the association between these commands, we analyzed the combinations of Git commands. Specifically, we leveraged the classical Apriori algorithm [2] to discover the possible association rules between these commands from the combinational use of Git commands. Apriori is an algorithm for frequent itemset mining and association rule learning. It proceeds by identifying the frequent individual items in the dataset and extending them to larger itemsets as long as those itemsets appear sufficiently often in the dataset. The frequent itemsets then can be used to determine association rules that highlight general trends in the dataset. Minimum support and confidence thresholds should be given to

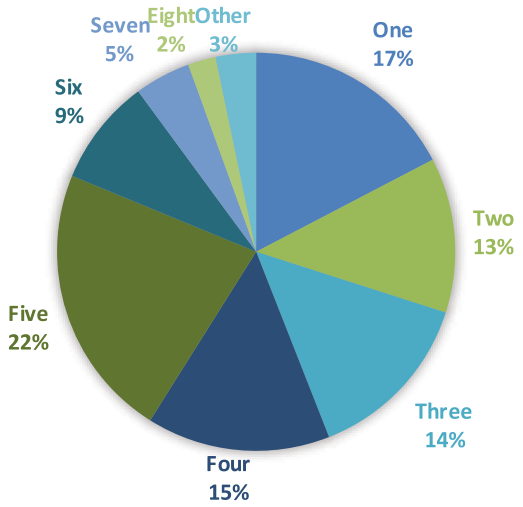


Fig. 3. The number of commands in combinational use.

Table 5. The Discovered Association Rules between Git Commands

Association rules	*Support*	*Confidence*
{git add} ⇒ {git push}	0.0556	0.3533
{git merge} ⇒ {git checkout}	0.0511	0.5127
{git add} ⇒ {git checkout}	0.0506	0.3214
{git pull} ⇒ {git push}	0.0507	0.3705
{git remote} ⇒ {git push}	0.0505	0.5500
{git add} ⇒ {git commit}	0.1038	0.6588
{git commit} ⇒ {git add}	0.1038	0.5447
{git commit} ⇒ {git checkout}	0.0601	0.3157
{git commit} ⇒ {git push}	0.0713	0.3741
{git push} ⇒ {git commit}	0.0713	0.3186
{git add} ⇒ {git commit, git push}	0.0508	0.3228

select interesting rules from the set of all possible rules. Support is an indication of how frequently the itemset appears in the dataset, and confidence is an indication of how often the rule has been found to be true. In our problem of finding association rules between Git commands, the minimum support threshold is set to 0.05 and the minimum confidence threshold is set to 0.3 [26]. Table 5 presents the 11 association rules we discovered. Take the rule {`git add`} ⇒ {`git push`} for example. It indicates that after a developer has used the command `git add`, he/she may also need to use the command `git push`. These association rules reveal some of the associations of Git command usages, many of which also match real-world usage scenarios for developers. For instance, the rule {`git add`} ⇒ {`git commit`} illustrates a common usage scenario where `git add` adds the file to the staging area, and then `git commit` adds contents in the staging area to the local repository. The usage scenarios of Git commands can be complex and varied, and what we have provided here are some of the association rules. We can modify the parameters (e.g., reducing the confidence threshold) to mine more association rules or use other techniques to mine Git command usage patterns. This goes to show that the use of Git commands is complex and needs further research efforts.

Answer: Git commands (e.g., git revert and git reflog) about recovery are among the most popular commands asked on Stack Overflow, followed by supporting Git commands (e.g., git clean and git help), and commands for branching and conflict resolution (e.g., git mergetool and git branch). Git commands are often used in combination to fulfill tasks.

3.4 RQ4: Command Difficulty

Finding the most difficult Git commands can help developers value the difficult commands so that they can prioritize effort on challenging commands. In particular, if a command is both popular and difficult, then it should receive much more attention. To measure the difficulty of a command, we look at five measures, i.e., the percentage of questions with no accepted answers ("% w/o acc."), the percentage of questions with no answers ("% w/o ans."), the median time to get an accepted answer ("Hrs to acc."), the number of questions including the command ("# of Qns."), and the average number of years the questioners had been registered on Stack Overflow ("Avg. reg. yrs"). Table 6 presents the difficulty measurements using these metrics, sorted by the percentage of questions with no accepted answers. Intuitively, a command is more difficult if a high percentage of its related questions do not have accepted answers or take longer to receive accepted answers. In the meantime, the number of questions related to a command, the percentage of the questions with no answers, and the number of years of registration of the questioners on Stack Overflow can further help us understand the reasons for the difficulty of the command. Similarly, we show the top 30 commands in the table, and the full results of all commands can be found in the supplementary materials.[3]

As shown in the table, we find that different commands vary a lot in the percentage of questions with no accepted answers, especially for the first few commands. For example, there are no accepted answers to the question related to commands `git pack-redundant` and `git http-push`, and thus their percentage is 100% and "—" in the fourth column indicates no accepted answer yet. Commands `git citool` and `git upload-archive` have the third- and fourth-lowest percentages of questions with no accepted answers, which are 80.0% and 57.1%, respectively. Although they differ substantially in the percentage of questions with no accepted answers, a fact to be aware of is that there are only several questions related to these commands. We further analyzed the functions of these commands and found that these commands are actually used in very few scenarios. For example, `git pack-redundant` computes which packs in the repository are redundant, `git http-push` pushes objects over HTTP/DAV to another repository, `git citool` is an alternative to the less interactive git commit program, and `git upload-archive` is usually invoked by `git archive` to send a generated archive to the other end over the Git protocol. We believe this is one of the main reasons why there is a high percentage of questions related to these commands that do not have accepted answers. This is also the case for many of the other commands ranked in the top 10, e.g., `git check-attr`, `git credential-cache`, and `git credential-store`. They are seldom used by developers. However, though the number of questions related to them is small, their corresponding percentages of relevant questions with no accepted answers or answers are high and it takes a longer time for them to get an accepted answer. Also, we observed that in general, the questioners of these commands had been registered on Stack Overflow for relatively long years compared to other commands that also have fewer questions related to them but are ranked lower. These commands are less used, but they are advanced commands that can provide some special functions (e.g., `git credential-cache` caches credentials in memory for use by future Git programs), and there are few instructions in the documentation, so developers can easily have doubts about their use. Therefore, this kind of commands is the ones that are less used by developers but are also difficult for them to understand.

Table 6. Difficulty Measurements of Git Commands

Command	*% w/o acc.*	*% w/o ans.*	*Hrs to acc.*	*# of Qns.*	*Avg. reg. yrs*
`git pack-redundant`	100.0%	0.0%	—	1	4.01
`git http-push`	100.0%	0.0%	—	3	0.51
`git citool`	80.0%	0.0%	14.3	5	1.75
`git upload-archive`	57.1%	14.3%	2.6	7	3.15
`git p4`	57.0%	10.5%	9.5	86	2.91
`git check-attr`	52.2%	17.4%	7.8	23	4.67
`git credential-cache`	50.0%	8.3%	8.9	12	3.54
`git credential-store`	50.0%	50.0%	6.0	4	2.20
`git fast-import`	44.8%	10.3%	2.8	58	2.88
`git credential`	43.0%	16.2%	5.3	328	3.37
`git prune-packed`	42.9%	14.3%	3.0	7	4.30
`git cvsimport`	42.0%	8.0%	2.1	50	1.92
`git annotate`	38.7%	9.7%	2.1	31	3.66
`git shell`	38.0%	8.9%	1.3	179	2.21
`git submodule`	37.5%	11.5%	1.7	2,911	2.98
`git svn`	37.4%	7.5%	3.2	1,527	2.26
`git verify-pack`	37.3%	9.8%	2.8	51	4.00
`git clone`	37.2%	10.2%	1.1	9,923	2.59
`git blame`	37.0%	9.8%	0.8	449	3.55
`git difftool`	36.7%	11.4%	2.6	395	3.29
`git mergetool`	36.5%	11.1%	2.8	488	2.90
`git pull`	36.5%	8.8%	0.6	10,996	2.67
`git ls-remote`	36.0%	10.9%	1.8	817	3.03
`git send-email`	35.9%	15.1%	1.1	53	2.47
`git upload-pack`	35.4%	7.7%	4.2	65	2.77
`git instaweb`	35.0%	5.0%	1.6	20	1.79
`git daemon`	34.8%	11.6%	1.2	112	1.71
`git push`	34.8%	9.1%	0.7	17,977	2.55
`gitweb`	34.7%	6.2%	2.1	274	1.63
`git count-objects`	34.6%	9.1%	3.5	55	3.14
Average	***45.8%***	***10.8%***	***3.24***	***1,564***	***2.81***

From the perspective of many developers, they may be more interested in commands that are ranked high and have a relatively large number of questions related to them, because these commands are more likely to be used by developers. Among the top 20 commands, the number of questions related to commands `git credential`, `git submodule`, `git svn`, `git clone`, `git blame`, and `git difftool` is relatively high, which is 328, 2,911, 1,527, 9,923, and 449, respectively. This means that these commands are not only difficult but are also asked a lot. Command `git credential` exposes the interface for storing and retrieving credentials from system-specific helpers, as well as prompting the user for usernames and passwords, `git submodule` allows developers to keep a Git repository as a subdirectory of another git repository, `git svn` allows using Git to interact with Subversion (an open source version control system) repositories, `git clone` clones a repository into a newly created directory, `git blame` shows what revision and author last modified each line of a file, and `git difftool` allows developers to compare and edit files between revisions using common diff tools. These commands are more commonly used by

developers, but they have more complex and flexible usage scenarios and can be used in combination with many other commands to achieve more complex functionality. For example, developers often use a combination of `git submodule`, `git clone` and `git status` to add submodules to the main project. So these commands are more difficult to master, and we can see that the questions related to these commands not only have a large percentage of no accepted answers, but also a relatively high percentage of no answers. When considering the median time (3.24 hours) to receive accepted answers, it takes more time compared to the median time of all questions to receive accepted answers on Stack Overflow, which is 21 minutes [32]. Besides, the questioners asking questions related to these commands have also been registered on Stack Overflow for an average of nearly 3 years, which is also consistent with the results obtained in RQ2 that many Git command-related questions are raised by developers with years of development experience.

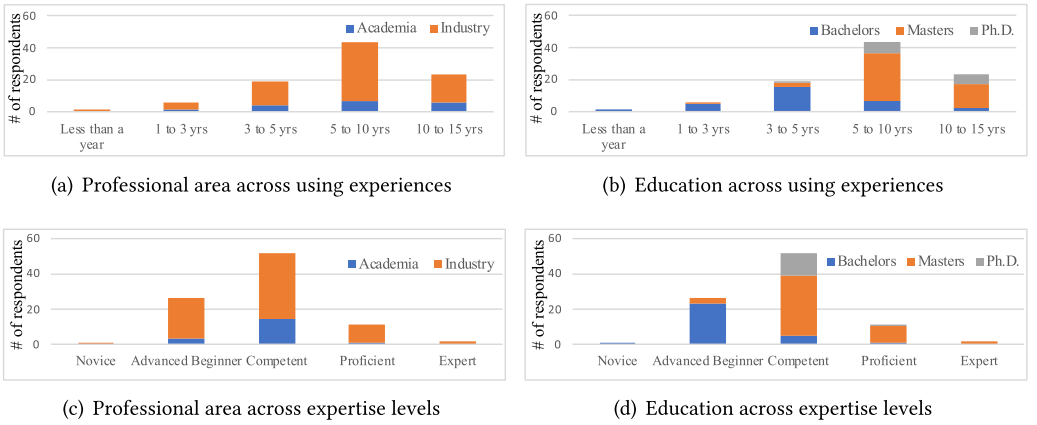
In fact, there is no uniform standard to judge whether a command is difficult or not. Some developers may consider that the difficulty of a Git command should mainly depend on the percentage of questions with no accepted answers related to the command, without referring to other metrics (e.g., the number of questions), while others may consider the time it takes to get accepted answers to be a better indicator of difficulty, or it is necessary to combine multiple metrics to judge the difficulty of a command. We mainly refer to the metric of existing work (i.e., the percentage of questions with no accepted answers) as the basis for sorting, but we also present four other metrics to help developers judge the difficulty of a command in a comprehensive way. The complete results of all the studied commands are also given[3] to facilitate developers to reorder these commands according to their understanding.

Answer: In terms of the percentage of questions with no accepted answers, some of the seldom used Git commands (e.g., git pack-redundant and git http-push) are ranked high. Commands that are asked in quite a number of questions with low percentages of accepted answers (e.g., git credential and git submodule) tend to be those can be used in complex or flexible scenarios.

3.5 RQ5: Learning Approaches

We present key survey findings here, focusing primarily on developers' learning approaches of Git commands. To help clarify the results, we also include some excerpts from the qualitative responses to the open-ended questions. Each of the excerpts is followed by a number representing a unique identifier for the respondent who expressed that opinion. For example, [#5] indicates a response from respondent number 5.

In the survey, we collected some information about respondents' demographics, including their professional area, education, Git command using experiences, and their self-evaluation of the expertise level of Git command usage. The results show that among our respondents, 80.4% (74 of 92) are from industry compared to 19.6% (18 of 92) from academia. In terms of the highest level of education, our respondents are 32.6% (30 of 92) with a bachelor's degree, 52.2% (48 of 92) with a master's degree, and 15.2% (14 of 92) with a Ph.D. The higher educational qualifications of our respondents may not be surprising as most of them come from large companies and universities or research institutions. Regarding the experience of using Git commands, 71.7% of respondents (66 of 92) have more than five years of experience in using Git commands. To gain a clearer picture of the distribution of respondents' using experience with Git across different professional areas and education levels, we provide detailed results in Figure 4(a) and (b). We can see that the highest percentage of Git using experience among the respondents in the industry is between 5 to 10 years, followed by 10 to 15 years, which is the same as for the respondents in academia. Regarding the using experience of respondents with different education levels, respondents with a bachelor's degree have the most using experience of 3 to 5 years, then 5 to 10 years, while those with a master's



(a) Professional area across using experiences

(b) Education across using experiences

(c) Professional area across expertise levels

(d) Education across expertise levels

Fig. 4. The proportion of respondents' professional areas and education among different expertise levels and usage experiences.

degree have the most using experience of 5 to 10 years, then 10 to 15 years, and most of the using experience of Ph.D. is between 5 to 10 years and 10 to 15 years.

When asked about the self-evaluation of the expertise level of Git command usage, only a small number of respondents (14.1%) thought that their level was proficient or above, and the vast majority of them (12 of 13) were from industry, as shown in Figure 4(c). In general, respondents from both academia and industry, when assessing their own expertise level in Git usage, mostly consider themselves to be competent. In terms of respondents' self-evaluation of their Git usage abilities with different education levels, the majority of those with a bachelor's degree considered themselves advanced beginners, and the majority of those with a master's degree considered themselves competent, similar to respondents with a Ph.D. In particular, the vast majority of respondents who considered themselves to be proficient or above were master's degree holders from the industry. We found that most respondents considered their expertise level is just advanced beginner or competent, and many of them are even developers with more than five years of experience in using Git commands. This result, although surprising, is consistent with the conclusion we obtained in RQ2, indicating that even experienced developers still have doubts about Git usage. As our respondent #20 (8 years experiences of using Git) stated, "*I often face problems in using Git commands, and almost every time I solve them by checking the documentation or searching on Internet.*" Similarly, there is a question asked on Stack Overflow [39], and the questioner mentioned that "*I [have] been using git for years but never used diff command and I started to use it today but I really don't understand the output....*"

To better understand developers' major learning approaches of Git commands, we made inquiries about how developers learned the skills of using Git commands in the survey. We offered five choices that can be multi-selected for respondents but also allowed them to specify additional learning approaches that were not provided in the survey. The listed five learning approaches can be divided into two categories: instructional teaching and self-learning. The first category includes traditional classroom learning, learning in online courses, and learning from peers or seniors. For the second category, we have self-learning from the documentation and self-learning from the internet (e.g., Q&A sites, blogs, and video).

Table 7 shows the results of developers' learning approaches of Git commands according to the respondents of our survey. The second column shows the percentage of respondents' choice of each learning approach and the specific number of times each approach was selected by the

Table 7. Percentage of Respondents' Choice of Learning Approaches and Details of the Choice of Respondents in Different Professional Areas, Education Levels, and Expertise Levels

Learning Approach		*Area*		*Education*			*Expertise*				
Category	*%/#*	*Acad.*	*Ind.*	*BA*	*MS*	*Ph.D.*	*Novice*	*Begin.*	*Comp.*	*Pro.*	*Exp.*
During class	4.1% / 8	1	7	5	3	0	0	5	3	0	0
Online courses	5.1% / 10	1	9	3	7	0	0	3	6	1	0
Peers or seniors	7.6% / 15	4	11	5	8	2	0	6	7	2	0
Documentation	38.6% / 76	14	62	22	41	13	1	17	46	10	2
Internet	43.1% / 85	18	67	24	47	14	0	21	51	11	2
Other	1.5% / 3	0	3	2	1	0	0	2	1	0	0

respondents. A key finding from the survey was that 81.7%, a vast majority, of all the learning approaches used by our respondents were in the category of self-learning, but only 16.8% were in the category of instructional teaching. Among them, the most used learning approach is self-learning from the internet (43.1%), followed by self-learning from the documentation (38.6%). Since this question was multiple choice in the survey, many respondents selected more than one option, and the above percentages are calculated by the number of times each learning approach was selected from the total number of all options selected. In fact, 85 of 92 respondents chose self-learning from the internet and 76 of 92 respondents chose self-learning from the documentation. This shows that self-learning is the primary way for developers to learn to use Git commands. We further provide the distribution of respondents who chose different learning approaches in terms of the professional area ("Acad." for the academia and "Ind." for the industry), the education level ("BA" for bachelors, "MS" for masters, and Ph.D.), and the expertise level (novice, "Begin." for advanced beginner, "Comp." for competent, "Pro." for proficient, and "Exp." for expert), which are shown in columns 3 to 12 of Table 7.

As we can see, respondents in different professional areas do not differ much in their choice of learning approaches. Bachelors and masters are more diverse in their choice by covering all possible learning approaches, while the vast majority of Ph.Ds. learn only through documentation and the internet. In terms of the choice of learning approaches by respondents with different levels of expertise, beginners and competent respondents are relatively similar in their choice of learning approaches, and respondents who consider themselves proficient or above tend to learn Git usage from the documentation and the internet. We can also see that the number of respondents who chose to learn from the internet was higher than the number of respondents who chose to learn from the documentation, which may reflect the fact that the documentation for Git is not yet complete. Thus, researchers could further investigate whether more samples of Git command usage could be filtered from the internet (e.g., Q&A sites) to enrich the documentation for Git.

Comments from respondents. We also asked the respondent of our survey to provide suggestions for potential new developers on learning Git commands if they were willing to do so. We received a total of 65 comments, which covered a number of different aspects. We have therefore further categorized these comments for better delivery purposes. As shown in Table 8, the first column is the identified category, the second column is the number of comments in each category, and the fourth column provides some concrete examples of respondents' comments. Some of these comments are specific suggestions from respondents to potential new developers on learning Git commands (the first three categories in Table 8), and some are their own thoughts on using Git (the last three categories in Table 8).

The three categories regarding the suggestions received from the respondents are as follows: (1) Getting started with collaboration by learning simple commands and improving skills in

Table 8. Classification of Respondents' Comments and Some Concrete Examples

Category	Num.	Concrete example
Learning and improving from practice	18	**#77:** *Git is my VCS of choice, as I've learned how it works and have use for its power. But there is a quite high barrier for getting started with Git. I recommend learning to use the basic commands first, so you can keep improving your skills in practice.* **#32:** *There are so many commands, I think it is better to learn them in practice instead of learning to master them at the beginning.*
Making the best use of the internet	17	**#42:** *Years on, I still constantly have to search the internet for help because it's not intuitive. We must learn to build on the work of others. I prefer to find the answers I need online (like Stack Overflow) than its poor documentation.*
Understanding basic concepts and principles first	14	**#26:** *The difficulty with git is that when problems arise, you have to dive into the solutions in the tutorials, which is hard to do without understanding the basic concepts. At this point, if you don't understand them you have to learn them all over again.*
Emphasizing the importance of Git	9	**#56:** *Git is an extremely important tool to our routine as software developers. I often use Git with GitHub, which makes the software development process a lot more comforting. I got started with Git by learning some simple commands when I was in college.*
Mastering only the basic commands	5	**#89:** *Git is not a tool you have to go pretty deep to learn. Since it is a fairly trivial tool, and its man page is not clear, I would recommend not spending too much time learning Git. I've been using Git for at least five years now, if not more. The only commands I've ever had to use are probably git pull, git push, git commit, and git rebase (in rare cases).*
Having the basic needs satisfied by Git GUI tools	2	**#21:** *All I want is to concentrate on my development, and dump the code into a repository. I do not want to waste time learning all kinds of commands. I am quite satisfied with the existing graphical tools because they meet my needs.*

practice; (2) Making the best use of the internet (e.g., Stack Overflow) to search for the solutions of problems on Git commands; (3) Understanding the basic principles of Git and using them in practice is more effective than memorizing a bunch of commands. Each of these categories has been mentioned by a number of developers as shown in the second column of Table 8. Also, we have given some concrete examples. For example, our respondent #77 recommended that potential new developers can start with basic commands and continue to improve their skills in practice, respondent #42 suggested making full use of the internet, while respondent #26 touched on the importance of understanding the basic concepts in Git. In addition to suggestions, our respondents also mentioned some of their own views on Git, as shown by the last three categories in Table 8. Many respondents emphasized the importance of Git in software development today, as mentioned by respondent #56. Some respondents (e.g., respondent #89) believed that developers actually only need to master some basic Git commands. However, according to the previous research questions, we have seen that there are still many developers asking questions related to some advanced Git commands. A small fraction of respondents mentioned that they felt that the current Git graphical tools were sufficient for their needs. Nevertheless, we believe that while graphical tools can help developers achieve some simple tasks, they are also inadequate in many complex tasks, and it is still up to the developer when they occur.

Answer: Most respondents (from both the academia and the industry) considered their expertise level of using Git commands to be only advanced beginner or competent. A vast majority (81.7%) of all the learning approaches used by our respondents were self-learning (i.e., from the internet and the documentation). Many respondents stressed the importance of the Git command and gave some suggestions for learning it (e.g., understanding basic concepts and principles first).

4 IMPLICATIONS

The results of our study can help not only developers but also educators and researchers to better decide where to focus their efforts. In this section, we discuss our insights and some practical implications based on the preceding derived findings.

4.1 Researchers

Understanding what developers ask about Git commands on Q&A sites, such as Stack Overflow, can help the research community understand the challenges they are facing. As demonstrated in our study, questions related to Git commands have been extensively asked on Stack Overflow since Git was proposed. The percentage of Git command-related questions and questioners has remained relatively steady, while the number of questions on Stack Overflow continues to grow, showing that developers are continuously encountering a spectrum of difficulties in using Git commands. Our findings encourage researchers to develop technologies and tools to help developers overcome these difficulties. Here, we briefly discuss some potential opportunities for Git commands to the research community. *(1) Git command recommendations.* In Sections 3.1 and 3.2. We found that there are over 80,000 Git command-related questions on Stack Overflow in total and that the largest percentage of all questioners each year after 2017 are those who had been registered for more than five years. In 2020, 40.0% of questioners who asked Git command-related questions had been registered on Stack Overflow for more than five years. This finding indicates the difficulty in using Git commands and highlights the need for researchers to propose assistance for developers when they have trouble choosing Git commands. Recommendation techniques can be proposed to recommend appropriate Git commands for developers based on their queries, or recommend related Stack Overflow posts to help developers resolve problems in using Git commands, similarly to Reference [31]. *(2) Mining Git command usage patterns.* As demonstrated in Section 3.3, in practice, it is often necessary to use a combination of several Git commands to solve a problem. During software development and collaboration, a developer often needs to discover specific usage patterns of Git commands. However, these usage patterns are often not well documented. This observation motivates researchers to propose mining techniques to help developers to get such usage patterns. Besides, the Git command documentation can be augmented with mined usage patterns of Git commands and examples from Stack Overflow, considering that some respondents in Section 3.5 mentioned that the Git documentation was not clear and complete. Developers of Git GUI clients can also use this information to further improve their tools and add more features. We hope that understanding the Q&A characteristics related to Git commands will help guide future research in this area.

4.2 Educators

Modern software development is mostly done by teams or groups, which requires cooperation between developers and version management. Git has become the most popular version control tool in use for software development. Git is on the top of the list for inexperienced developers wanting to build up valuable skills in version control of software development. As illustrated in Section 3.4, among all the learning approaches used by our respondents, only 16.8% are instructional teaching, while 81.7% are self-learning. Thus, the need to provide training in Git should be seriously considered today. Our work studies how developers learn and develop their skills of using Git commands, and by being aware of this, educators can better understand students' educational needs and find ways to best support them through the learning process in practice. *(1) Prioritizing work on challenging commands.* In Section 3.4, we found that different Git commands have different levels of difficulty, some of which are relatively more difficult with a high percentage of related questions having no accepted answers. Among them, some are seldom used Git commands (e.g., *git pack-redundant* and *git http-push*), and the other are used relatively more frequently (e.g., *git credential* and *git submodule*). By knowing this, educational efforts can be prioritized on these challenging commands, such as devoting more material and teaching time to the more difficult commands and scheduling to teach the more popular and less difficult commands before the difficult ones. *(2) Adopting a practice-oriented and problem-oriented teaching*

method. In Section 3.5, we solicited our respondents' suggestions for potential new developers on learning Git commands, and they emphasized the importance of learning Git commands through practice. Some respondents expressed that it was easier to deepen their understanding of Git commands by solving real-world problems and thus mastering the use of Git commands. Many respondents also mentioned that they often learn to use Git commands only after they encounter problems. Hence, it is important for educators to provide practical training programs and adopt a problem-oriented approach to teaching Git commands.

4.3 Developers

Git is considered one of the fundamental skills that developers need to acquire, since developers need version control and Git is an industry standard. Developers who have worked with Git are well represented in the pool of available software development talent. The study outcome can serve as a checklist for developers to improve their Git skills in a targeted manner. *(1) Targeted learning of required commands.* Based on our findings in Sections 3.3 and 3.4, a developer who starts to learn Git may decide to focus their learning on commands with higher popularity and less difficulty compared to difficult but unpopular commands. In this case, they should focus more on the Git commands related to recovery, e.g., *git revert* and *git reflog*, some supporting Git commands (e.g., *git clean* and *git help*) and commands for branching and conflict resolution (e.g., *git mergetool* and *git branch*), since they are among the most popular commands. In contrast, a more knowledgeable developer who likes to learn about advanced commands of more than average difficulty may decide to learn about the difficult commands in Table 6, especially those commands having complex and flexible usage scenarios (e.g., *git credential*, *git submodule*, *git blame*, and *git ls-remote*). *(2) Making full use of Git to manage projects.* Considering the widespread use of Git, we recommend that developers use Git to manage their projects whenever possible. Currently, there are many excellent code hosting platforms on the market that offer convenient Git services. By using the Git service as early as possible, developers can master the rules of using Git and gain valuable experience for future work. However, developers can take full advantage of the features Git provides and unleash the capabilities of Git as much as possible. In our survey (Section 3.5), some respondents said that they initially thought they could use Git well by learning a few simple commands (e.g., `git pull` and `git push`) and there was no need to learn advanced commands. But then they realized that Git provides a lot of rich features that can greatly help teams collaborate better, e.g., branching and patching. Our respondent #12 stated that "*Git is very useful to developers. Whether you accidentally delete or modify the code, Git can help you revert to any previous versions if needed. Git also lets you share and exchange code with other developers easily.*"

5 THREATS TO VALIDITY

This section discusses some threats to the validity of our study.

Selection of posts. When determining whether a question post is related to Git commands, we use tags and keyword matching mechanisms and thus may result in potential research bias. This is because using tags and keywords may not be able to identify a precise and complete set of Git command-related posts. To make the collection of posts related to Git commands as complete as possible, we only use the tag "git." However, this could cause posts with the tag "git" that are not related to Git commands to be included. Therefore, we further use Git commands as keywords to filter the posts by matching keywords with the title, the body, and the accepted answer of the post. An alternative to checking for Git commands in accepted answers is to check all the answers in the same question thread. As pointed out in articles [12, 18], some Q&A sites (e.g., Ask Ubuntu, Super User, and Stack Overflow) have a considerable amount of questions without accepted answers, which, in some cases, is due to the fact that questioners may forget to mark accepted answers. By

checking all answers (not just accepted answers), the dataset could be further expanded. However, this can pose a threat to our study. Since we do not know the reason why the questioners have not marked the accepted answer, it is possible that the Git commands in the answers are not relevant to the posted question. In addition, since there can be different Git commands in different answers to a post, it is impossible to correctly associate the post with the specific Git commands if there is no accepted answer. Considering that over 80,000 posts related to Git commands have been identified in our study (a quite large number compared to those of some recent well-studied topics), we believe that such a dataset can be used to derive meaningful results while being representative.

Data analysis. One threat is concerned with potential errors in our implementation of data analysis. To reduce errors, we have double-checked and fully tested our implementation. Also, we have made our dataset and results public to facilitate other researchers to replicate and extend our work. However, the metrics we employed may not be comprehensive enough when measuring the question popularity trend, the popularity, and the difficulty of Git commands. Nevertheless, we do try to refer to the metrics used in the related work and use as many metrics as possible. In the meantime, many of our metrics are not only calculated for Git command-related questions, but also evaluated on all questions across Stack Overflow, allowing readers to interpret their values in a broader context. Another threat is that we use developers' registration date on Stack Overflow as a construct for development experience level in RQ2. This treatment could lead to inaccurate judgment of the developers' development experience. However, it is infeasible to know the exact number of years of experience of the Stack Overflow questioners. Considering that users on Stack Overflow are mostly developers, using the registration time as a proxy is a compromise.

Selection of data source. Similarly to previous studies [3, 5, 6, 14, 45, 50], we use Stack Overflow as the only dataset for study. This is a potential threat, since Stack Overflow posts may not be representative of developer interests and difficulties, and we may overlook valuable insights from other sources. However, considering that Stack Overflow has a large number of participant developers and posts and is popular among developers, this threat could be reduced. In addition, we use not only the title and body of the questions, but also their accepted answers to alleviate this risk. Meanwhile, we conduct a survey with researchers and practitioners to further validate our results.

Design and skewness of survey. The design of the survey in this study presents threats to validity. For instance, respondents may misunderstand our questions or the questions may be inappropriate. Therefore, to reduce potential threats, we carefully word the questions in an unbiased manner and provide clear instructions for respondents. The actual results of our questions indicate that the respondents understand the intent of the survey questions. Therefore, we think this threat is minimal. Another threat is that the skewness of the survey responses is inevitable. Although our survey's response rate (18.5%) is similar to prior SE surveys [11, 27], the results might suffer from a potential "non-response bias," which means that the opinions of the respondents who chose to participate may be different from those who did not. To encourage responses, the amount and type of the questions in the survey were carefully designed according to Reference [44].

6 RELATED WORK

The research most closely related to our work comes from studies that use data from Stack Overflow to understand the interests and difficulties of developers and literature related to the usage of Git commands.

Studies using Stack Overflow data. As the most popular Q&A site among developers, Stack Overflow is widely used to study the software engineering practice from the developer's perspective. The impact of Stack Overflow on software engineering research is growing as shown in Reference [34]. There is a number of studies using Stack Overflow data to categorize its questions [10, 47], identify its design features [32], and analyze the topics discussed by developers [8].

However, existing work has not comprehensively studied the usage of Git commands using Stack Overflow. More specifically, researchers have analyzed the posts on Stack Overflow to understand developers' concerns in developing various types of software and facing different programming tasks, including but not limited to web applications [6], mobile applications [45], concurrency [3], security [50], and privacy [46]. Recently, due to the rapid development of big data and machine learning, researchers have also started to study the problems and challenges that developers encounter in the development practices of these two types of applications based on the data on Stack Overflow [5, 7, 51]. For example, studies of deep learning bug characteristics [23], TensorFlow program bugs [52], deep learning deployment [14, 19], and cloud computer vision [16] have emerged. These studies are about how to identify or understand the challenges and characteristics related to software development based on the developers' discussions on Stack Overflow. In addition, there are studies exploring how information on Stack Overflow can help programmers develop software. Vasilescu et al. [48] investigated the interplay between Stack Overflow activities and the development process. Abdalkareem et al. [1] analyzed 1,414 Stack Overflow-related code commits and found that developers use the crowd knowledge on Stack Overflow to support development tasks and collect user feedback. To the best of our knowledge, our work is the first attempt to investigate the Q&A characteristics related to Git commands and practices for using and learning Git commands from the developer's perspective through an empirical study with a large number of posts on Stack Overflow and 92 developers.

Git command usage. There are currently more than 140 Git commands. Nevertheless, existing research on Git commands has focused on certain commands, e.g., `git commit` and `git pull`. The commit command is used to save changes to a local repository. Before saving changes in Git, developers are required to provide a commit message describing the changes. To automatically generate short and high-level commit messages, Jiang et al. [25] adapt neural machine translation to translate diffs into commit messages. Liu et al. [29] further perform an in-depth analysis of the experimental results in Reference [25] and propose a simpler and faster approach to generate concise commit messages using the nearest neighbor algorithm based on their findings. Xu et al. [49] propose to combine both code structure and code semantics to enrich the representations of code changes for a better generation. There have been ongoing studies that continue to investigate how to generate better commit messages [28]. As regards the use of `git commit`, the command itself is a basic and common command in Git, and developers do not have many questions about its use, which matches the experimental results we have obtained, namely that it ranks high in popularity and relatively low in difficulty.

Pull requests are a mechanism for developers to notify team members that they have completed a feature and ask their upstream developers to pull the changes. This mechanism is widely used in existing code hosting platforms and involves commands like `git pull` and `git merge`. Similar to commit messages, when submitting a pull request, developers are required to add a description to describe what changes are made in the pull request and/or why. Liu et al. [30] propose an approach to automatically generating pull request descriptions based on the commit messages and the added source code comments in the pull requests. They treat this problem as a text summarization problem and solve it using a sequence-to-sequence model. Since there can be many developers in a project, besides using descriptions, another common way to facilitate the organization of pull requests in projects is to use tags for pull requests. Thus, Jiang et al. [24] conduct a survey to understand the usage of tags in GitHub and propose a method that uses a feed-forward neural network to analyze titles, descriptions, file paths, and contributors for recommending tags of pull requests. Commands `git pull` and `git merge` are also common and popular, while their usage scenarios are richer and can be combined with many other commands, so it is more difficult to master, and developers have asked many questions related to them.

The above works provide tremendous assistance for developers to use those Git commands. However, they are only concerned with the usage of some Git commands' secondary features, without focusing on how the commands themselves are used, which happens to be the subject of our study. Ross et al. [17, 42] argue that there are deficiencies in the design of Git based on their understanding of Git and propose alternative designs. To back up their argument, they analyzed about 2,400 Stack Overflow posts related to Git and found that 41 posts were matching their proposed Git design problems. Therefore, the goal of the Stack Overflow post study in that work was to collect evidence about whether the proposed Git design problems were practical ones. Their work echoes our findings from another angle, which is that developers are not using the Git command in a very favorable way. Developers have asked lots of questions about the use of Git commands, and the reason for this could be due to the overly complex design of Git. Rather than changing the design of Git, our work studies the collective trends and Q&A characteristics about developers using Git commands through a large-scale study and provides implications that can help researchers, educators, and developers to better decide when and where to focus their efforts on learning or building support for Git commands.

7 CONCLUSION

This work is motivated by the need to empirically understand the problems and difficulties that developers have encountered in using Git commands. To that end, we retrieved a large dataset from Stack Overflow and conducted an empirical study based on the 80,370 questions related to Git commands to answer four research questions. We present key findings to show that the number of questions related to Git commands has been growing steadily and among the many questioners, there are not only novices but also experienced developers. Our findings also reveal that the most popular Git commands being asked on Stack Overflow are related to recovery and it takes a long time for Git command questions to receive accepted answers. We answer one additional research question regarding the Git learning approaches through a survey of 92 developers both from academia and industry, and find that a vast majority of developers learned how to use Git commands on the internet or through the documentation by themselves. The results of our study offer actionable implications for researchers, educators, and developers, with the goal of highlighting good practices and valuable research avenues in Git command usage.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable and constructive comments that are vital to the improvement of this article. We also thank the participants of the survey for providing feedback about their experience of using Git.

REFERENCES

[1] R. Abdalkareem, E. Shihab, and J. Rilling. 2017. What do developers use the crowd for? A study using stack overflow. *IEEE Softw.* 34, 2 (2017), 53–60. https://doi.org/10.1109/MS.2017.31

[2] Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 487–499.

[3] Syed Ahmed and Mehdi Bagherzadeh. 2018. What do concurrency developers ask about? A large-scale study using stack overflow. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'18)*. Association for Computing Machinery, New York, NY, USA, Article 30, 10 pages. https://doi.org/10.1145/3239235.3239524

[4] M. Alshangiti, H. Sapkota, P. K. Murukannaiah, X. Liu, and Q. Yu. 2019. Why is developing machine learning applications challenging? A study on stack overflow posts. In *Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'19)*. 1–11. https://doi.org/10.1109/ESEM.2019.8870187

[5] Mehdi Bagherzadeh and Raffi Khatchadourian. 2019. Going big: A large-scale study on what big data developers ask. In *Proceedings of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'19)*. Association for Computing Machinery, New York, NY, 432–442. https://doi.org/10.1145/3338906.3338939

[6] Kartik Bajaj, Karthik Pattabiraman, and Ali Mesbah. 2014. Mining questions asked by web developers. In *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR'14)*. Association for Computing Machinery, New York, NY, 112–121. https://doi.org/10.1145/2597073.2597083

[7] Abdul Ali Bangash, Hareem Sahar, Shaiful Chowdhury, Alexander William Wong, Abram Hindle, and Karim Ali. 2019. What do developers know about machine learning: A study of ML discussions on StackOverflow. In *Proceedings of the 16th International Conference on Mining Software Repositories (MSR'19)*. IEEE Press, 260–264. https://doi.org/10.1109/MSR.2019.00052

[8] Anton Barua, Stephen W. Thomas, and Ahmed E. Hassan. 2014. What are developers talking about? An analysis of topics and trends in stack overflow. *Emp. Softw. Eng.* 19, 3 (2014), 619–654.

[9] S. Bennett. 2012. 10 Things I Hate About Git. Retrieved August 15, 2021 from http://stevebennett.me/2012/02/24/10-things-ihate-about-git/.

[10] S. Beyer, C. Macho, M. Di Penta, and M. Pinzger. 2018. Automatically classifying posts into question categories on stack overflow. In *Proceedings of the IEEE/ACM 26th International Conference on Program Comprehension (ICPC'18)*. 211–21110.

[11] A. Bosu, J. C. Carver, C. Bird, J. Orbeck, and C. Chockley. 2017. Process aspects and social dynamics of contemporary code review: Insights from open source development and industrial practice at microsoft. *IEEE Trans. Softw. Eng.* 43, 1 (2017), 56–75. https://doi.org/10.1109/TSE.2016.2576451

[12] Fabio Calefato, Filippo Lanubile, and Nicole Novielli. 2016. Moving to stack overflow: Best-answer prediction in legacy developer forums. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'16)*. Association for Computing Machinery, New York, NY, Article 13, 10 pages. https://doi.org/10.1145/2961111.2962585

[13] Scott Chacon and Ben Straub. 2014. *Pro Git*. Springer Nature.

[14] Zhenpeng Chen, Yanbin Cao, Yuanqiang Liu, Haoyu Wang, Tao Xie, and Xuanzhe Liu. 2020. *A Comprehensive Study on Challenges in Deploying Deep Learning Based Software*. Association for Computing Machinery, New York, NY, 750–762. https://doi.org/10.1145/3368089.3409759

[15] Luke Church, Emma Söderberg, and Elayabharath Elango. 2014. A case of computational thinking: The subtle effect of hidden dependencies on the user experience of version control. In *Proceedings of the Psychology of Programming Interest Group Annual Conference*. 123–128.

[16] A. Cummaudo, R. Vasa, S. Barnett, J. Grundy, and M. Abdelrazek. 2020. Interpreting cloud computer vision pain-points: A mining study of stack overflow. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering (ICSE'20)*. 1584–1596. https://doi.org/10.1145/3377811.3380404

[17] Santiago Perez De Rosso and Daniel Jackson. 2016. Purposes, concepts, misfits, and a redesign of git. In *Proceedings of the ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'16)*. Association for Computing Machinery, New York, NY, 292–310. https://doi.org/10.1145/2983990.2984018

[18] Zhipeng Gao, Xin Xia, David Lo, and John Grundy. 2021. Technical Q8A site answer recommendation via question boosting. *ACM Trans. Softw. Eng. Methodol.* 30, 1, Article 11 (Dec. 2021), 34 pages. https://doi.org/10.1145/3412845

[19] Qianyu Guo, Sen Chen, Xiaofei Xie, Lei Ma, Qiang Hu, Hongtao Liu, Yang Liu, Jianjun Zhao, and Xiaohong Li. 2019. An empirical study towards characterizing deep learning development and deployment across different frameworks and platforms. In *Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering (ASE'19)*. IEEE Press, 810–822. https://doi.org/10.1109/ASE.2019.00080

[20] GitHub Inc. 2020. The 2020 State of the Octoverse. Retrieved May 20, 2021 from https://octoverse.github.com/.

[21] GitLab Inc. 2020. Is It Any Good? Retrieved May 20, 2021 from https://about.gitlab.com/is-it-any-good/.

[22] Stack Exchange Inc. 2020. Stack Exchange Dump. Retrieved May 20, 2021 from https://archive.org/details/stackexchange.

[23] Md Johirul Islam, Giang Nguyen, Rangeet Pan, and Hridesh Rajan. 2019. A comprehensive study on deep learning bug characteristics. In *Proceedings of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'19)*. Association for Computing Machinery, New York, NY, 510–520. https://doi.org/10.1145/3338906.3338955

[24] Jing Jiang, Qiudi Wu, Jin Cao, Xin Xia, and Li Zhang. 2021. Recommending tags for pull requests in GitHub. *Inf. Softw. Technol.* 129 (2021), 14 pages. https://doi.org/10.1016/j.infsof.2020.106394

[25] Siyuan Jiang, Ameer Armaly, and Collin McMillan. 2017. Automatically generating commit messages from diffs using neural machine translation. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE'17)*. IEEE Press, 135–146.

[26] Branko Kavšek, Nada Lavrač, and Viktor Jovanoski. 2003. APRIORI-SD: Adapting association rule learning to subgroup discovery. In *Advances in Intelligent Data Analysis V*, Michael R. Berthold, Hans-Joachim Lenz, Elizabeth Bradley, Rudolf Kruse, and Christian Borgelt (Eds.). Springer, Berlin, 230–241.

[27] Oleksii Kononenko, Olga Baysal, and Michael W. Godfrey. 2016. Code review quality: How developers see it. In *Proceedings of the 38th International Conference on Software Engineering (ICSE'16)*. Association for Computing Machinery, New York, NY, 1028–1038. https://doi.org/10.1145/2884781.2884840

[28] S. Liu, C. Gao, S. Chen, N. Lun Yiu, and Y. Liu. 2020. ATOM: Commit message generation based on abstract syntax tree and hybrid ranking. *IEEE Trans. Softw. Eng.* (2020), 1–1. https://doi.org/10.1109/TSE.2020.3038681

[29] Zhongxin Liu, Xin Xia, Ahmed E. Hassan, David Lo, Zhenchang Xing, and Xinyu Wang. 2018. Neural-machine-translation-based commit message generation: How far are we? In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE'18)*. Association for Computing Machinery, New York, NY, 373–384. https://doi.org/10.1145/3238147.3238190

[30] Z. Liu, X. Xia, C. Treude, D. Lo, and S. Li. 2019. Automatic generation of pull request descriptions. In *2019Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering (ASE'19)*. 176–188. https://doi.org/10.1109/ASE.2019.00026

[31] Sonal Mahajan, Negarsadat Abolhassani, and Mukul R. Prasad. 2020. *Recommending Stack Overflow Posts for Fixing Runtime Exceptions Using Failure Scenario Matching*. Association for Computing Machinery, New York, NY, 1052–1064. https://doi.org/10.1145/3368089.3409764

[32] Lena Mamykina, Bella Manoim, Manas Mittal, George Hripcsak, and Björn Hartmann. 2011. Design lessons from the fastest Q&a site in the west. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'11)*. Association for Computing Machinery, New York, NY, 2857–2866. https://doi.org/10.1145/1978942.1979366

[33] Slashdot Media. 2020. About Sourceforge. Retrieved May 20, 2021 from https://sourceforge.net/about.

[34] Sarah Meldrum, Sherlock A. Licorish, and Bastin Tony Roy Savarimuthu. 2017. Crowdsourced knowledge on stack overflow: A systematic mapping study. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering (EASE'17)*. Association for Computing Machinery, New York, NY, 180–185. https://doi.org/10.1145/3084226.3084267

[35] Sarah Nadi, Stefan Krüger, Mira Mezini, and Eric Bodden. 2016. Jumping through hoops: Why do Java developers struggle with cryptography APIs? In *Proceedngs of the International Conference on Software Engineering (ICSE'16)*. Association for Computing Machinery, New York, NY, 935–946. https://doi.org/10.1145/2884781.2884790

[36] [n.d.]. 2009. Retrieved May 20, 2021 from https://stackoverflow.com/questions/927358/how-do-i-undo-the-most-recent-local-commits-in-git.

[37] [n.d.]. 2009. Retrieved May 20, 2021 from https://stackoverflow.com/questions/1469623/a-few-basic-version-control-questions.

[38] [n.d.]. 2013. Retrieved May 20, 2021 from https://stackoverflow.com/questions/17371955/verifying-signed-git-commits.

[39] [n.d.]. 2014. Retrieved May 20, 2021 from https://stackoverflow.com/questions/27508982/interpreting-git-diff-output.

[40] [n.d.]. 2018. Retrieved May 20, 2021 from https://stackoverflow.com/questions/50827060/where-can-i-report-a-github-bug.

[41] [n.d.]. 2020. Retrieved May 20, 2021 from https://stackoverflow.com/questions/62701501/git-checkout-vs-restore-single-file.

[42] Santiago Perez De Rosso and Daniel Jackson. 2013. What's wrong with git? A conceptual design analysis. In *Proceedings of the ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software (Onward! 2013)*. Association for Computing Machinery, New York, NY, 37–52. https://doi.org/10.1145/2509578.2509584

[43] Atlassian Corp Plc. 2019. Celebrating 10 Million Bitbucket Cloud Registered Users. Retrieved May 20, 2021 from https://bitbucket.org/blog/celebrating-10-million-bitbucket-cloud-registered-users.

[44] T. Punter, M. Ciolkowski, B. Freimut, and I. John. 2003. Conducting on-line surveys in software engineering. In *Proceedings of the International Symposium on Empirical Software Engineering (ISESE 2003)*. 80–88. https://doi.org/10.1109/ISESE.2003.1237967

[45] Christoffer Rosen and Emad Shihab. 2016. What are Mobile developers asking about? A large scale study using stack overflow. *Emp. Softw. Eng.* 21, 3 (June 2016), 1192–1223. https://doi.org/10.1007/s10664-015-9379-3

[46] Mohammad Tahaei, Kami Vaniea, and Naomi Saphra. 2020. Understanding privacy-related questions on stack overflow. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–14. DOI: https://doi.org/10.1145/3313831.3376768

[47] C. Treude, O. Barzilay, and M. Storey. 2011. How do programmers ask and answer questions on the web?: NIER track. In *Proceedings of the 33rd International Conference on Software Engineering (ICSE'11)*. 804–807. https://doi.org/10.1145/1985793.1985907

[48] B. Vasilescu, V. Filkov, and A. Serebrenik. 2013. StackOverflow and GitHub: Associations between software development and crowdsourced knowledge. In *Proceedings of the International Conference on Social Computing*. 188–195. https://doi.org/10.1109/SocialCom.2013.35

[49] Shengbin Xu, Yuan Yao, Feng Xu, Tianxiao Gu, Hanghang Tong, and Jian Lu. 2019. Commit message generation for source code changes. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI'19)*, Sarit Kraus (Ed.). International Joint Conferences on Artificial Intelligence, 3975–3981. https://doi.org/10.24963/ijcai.2019/552

[50] Xin-Li Yang, David Lo, Xin Xia, Zhi-Yuan Wan, and Jian-Ling Sun. 2016. What security questions do developers ask? A large-scale study of stack overflow posts. *J. Comput. Sci. Technol.* 31, 5 (2016), 910–924.

[51] T. Zhang, C. Gao, L. Ma, M. Lyu, and M. Kim. 2019. An empirical study of common challenges in developing deep learning applications. In *Proceedings of the IEEE 30th International Symposium on Software Reliability Engineering (ISSRE'19)*. 104–115. https://doi.org/10.1109/ISSRE.2019.00020

[52] Yuhao Zhang, Yifan Chen, Shing-Chi Cheung, Yingfei Xiong, and Lu Zhang. 2018. An empirical study on TensorFlow program bugs. In *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA'18)*. Association for Computing Machinery, New York, NY, 129–140. https://doi.org/10.1145/3213846.3213866

Received May 2021; revised August 2021; accepted October 2021