On the maintenance of crowdsourced knowledge on Stack Overflow

by

Haoxiang Zhang

A thesis submitted to the Graduate Program in Computing

in conformity with the requirements for the

Degree of Doctor of Philosophy

Queen's University

Kingston, Ontario, Canada

September 2020

Abstract

THE enormous quantity of software artifacts (e.g., code, documentations, and online discussions) created by developers are in great need for maintenance. Among software artifacts built by developers, Stack Overflow, a question answering (Q&A) website for sharing programming knowledge, has tripled the number of its hosted answers from 9.3 million to 27.2 million over the past 6 years (as of 2019). Such a large-scale knowledge base (including code snippets along with the embedded knowledge in the question answering activities), is inevitably changing over time. In an effort to better understand the current knowledge maintenance practices and to improve the maintenance of such valuable knowledge on Stack Overflow, this PhD thesis empirically studies the Q&A activities on Stack Overflow over a decade (i.e., from 2008 to 2018). Our goal is to provide developers with lessons about the knowledge maintenance practices on Stack Overflow. Specifically, this thesis mines the question answering activities on Stack Overflow along three dimensions: 1) the obsolescence of answers, 2) the informativeness of comments that are associated with answers, and 3) the retrieval of information in hidden comments. First, we wish to understand the knowledge maintenance practices by studying obsolete answers. For example, obsolete answers can contain invalid links, or obsolete Application Programming Interface (API) usages. Secondly, we investigate the informativeness of comments that are associated with Stack Overflow answers. An informative comment can provide additional explanations, thus updating its associated answer. Furthermore, we examine comments that can be added long after an answer is posted. In particular, we study the comment organization mechanism and analyze whether hidden comments are informative as well. By empirically studying such crowdsourced knowledge, we wish to highlight the importance of maintaining such valuable crowdsourced knowledge and to understand developer practices.

Acknowledgments

Thanks a million to Prof. Ahmed E. Hassan for his continuous guidance and insightful advice throughout these three years. All this work would not have been possible without his vision and support. It is my pleasure to work with Prof. Ahmed Hassan, who encouraged me to pursue the research topic of mining software repositories and taught me the art of research in empirical software engineering. I truly enjoy working in a research environment enabling critical thinking and rigid analytics, which he created.

Very special thanks to Prof. Shaowei Wang and Prof. Tse-Hsun (Peter) Chen for the great collaborations over the past years. Thanks a lot for your fruitful discussions and inspiring comments on my work over countless times. I am looking forward to our next collaborations.

A sincere appreciation to my supervisory and examination committee members, Prof. Yuan Tian and Prof. Mohammad Zulkernine for their continued critique, support and guidance. Very special thanks to Prof. Ting Hu and Prof. Massimiliano Di Penta for being examiners of this thesis with their insightful feedback and valuable advice on my work.

Many thanks to everyone who co-authored a paper with me over the years (in alphabetical order): Tse-Hsun (Peter) Chen, Ahmed Hassan, Shaowei Wang, Ying Zou.

Thanks to all members of the Software Analysis and Intelligence Lab (SAIL) at Queen's University, including all the postdoc and students with whom I worked. It was a great time in Kingston to work with everyone at SAIL. A special thanks to Debby Robertson for all the help with my paper work.

Last but not least, I would like to express my deepest gratitude to my parents, Mei Rong and Chi Zhang, for their loving support throughout my studies.

Dedication

This thesis is dedicated to the memory of my grandfather Fuduan Zhang

For his encouragement and inspiration

Co-authorship

The work presented in the thesis is published as listed below:

- Haoxiang Zhang, Shaowei Wang, Tse-Hsun (Peter) Chen, Ying Zou, and Ahmed E. Hassan. 2019. An empirical study of obsolete answers on Stack Overflow. IEEE Transactions on Software Engineering (TSE). In press. This work is described in Chapter 3.
- Haoxiang Zhang, Shaowei Wang, Tse-Hsun (Peter) Chen, and Ahmed E. Hassan. 2019. Reading answers on Stack Overflow: Not enough! IEEE Transactions on Software Engineering (TSE). In press. This work is described in Chapter 4.
- Haoxiang Zhang, Shaowei Wang, Tse-Hsun (Peter) Chen, and Ahmed E. Hassan. ACM Transactions on Software Engineering and Methodology (TOSEM). Under review. This work is described in Chapter 5.

For each of the work, my contributions include proposing the initial research idea, investigating background knowledge and related work, proposing research methods, conducting experiments and collecting the data, analyzing the data and verifying the hypotheses, and writing and polishing the manuscript. My co-contributors supported me in refining the initial ideas, providing suggestions for potential research methods, providing feedback on experimental results and earlier manuscript drafts, and providing advice for polishing the writing.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

SOFTWARE engineering is a rapidly evolving field due to the emergence of new technologies and the advancing of existing ones. Such rapid evolution makes software artifacts that are generated by developers both large in quantity and complex in terms of content. For example, an estimated number of 180 – 200 billion lines of legacy code is still in use (Fanelli et al., 2016). In reality, developers are working in an environment where both legacy and up-to-date technologies coexist. "*You cannot step into the same river twice for other waters are continually flowing in*" (Heraclitus quoted in Plato's Cratylus, Section 402a). This poetical statement about "*all is change*" is reflected in software development. Software maintenance is defined as "the modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment (Bennett and

Rajlich, 2000).” For example, McDonnell et al. (2013) observes that Android is evolving at an average rate of 115 API updates per month. Thus, the enormous quantities of software artifacts (e.g., code and documentations) created by developers are constantly in need of maintenance. Such knowledge maintenance activities are equally as important and as challenging as code maintenance activities.

Stack Overflow[1], a question answering (Q&A) website for sharing programming knowledge, is widely used among developers. By Feb. 2020, Stack Overflow was ranked by Alexa as the 39th website[2] in global internet engagement, compared with GitHub which is ranked as the 66th[3]. The popularity of Stack Overflow can also be quantified from the number of its hosted answers to programming-related questions. More specifically, Stack Overflow has hosted 9.3 million answers within five years (i.e., as of July 2013) since its launch in 2008, while over the following 6 years (i.e., as of July 2019) its number of hosted answers has tripled (i.e., 27.2 million)[4].

The large-scale knowledge base hosted on Stack Overflow is inevitably changing over time. In an effort to better understand the current knowledge maintenance practices and to improve the maintenance of such knowledge on Stack Overflow, this PhD thesis empirically mines the Q&A activities on Stack Overflow over a decade (i.e., from 2008 to 2018). Our goal is to provide developers with a better understanding of knowledge maintenance practices.

On Stack Overflow, question answering activities can be illustrated as follows. First, a user (i.e., an asker) posts a question to describe a programming-related issue. Another user (i.e., an answerer) can then post an answer for solving the issue. A question

[1] `https://stackoverflow.com/`
[2] `https://www.alexa.com/siteinfo/stackoverflow.com`
[3] `https://www.alexa.com/siteinfo/github.com`
[4] `https://data.stackexchange.com/stackoverflow/queries`

can receive multiple answers, while only one answer can be selected by the asker as the accepted answer (i.e., a "selected" answer) (Anderson et al., 2012). During this question answering process, users (i.e., commenters) can further discuss with each other by posting a comment under either a question or an answer. The aforementioned activities of posting questions/answers/comments are all recorded in an individual webpage (i.e., a question thread). Question threads under the same technical domain (i.e., a tag) are manually tagged by their askers (e.g., [Java] and [Python]). As of Feb. 2020, there are a total number of 19 million questions, 29 million answers, and 73 million comments. These posts (i.e., questions, answers, and comments) are posted by 11.8 million users across 57 thousand tags[5]. The Stack Overflow community continues to play a vital role in software development practices around the globe, mainly through its crowdsourced knowledge sharing, enabling developers to either solve their programming issues or share their programming knowledge.

Prior studies investigated Stack Overflow Q&A activities through both empirical studies (Abdalkareem et al., 2017; Anderson et al., 2012; Zhang et al., 2018) and statistical models (Chen et al., 2017; Duijn et al., 2015; Treude and Robillard, 2016; Wong et al., 2013). These studies have two main limitations: 1) Much of the prior work analyzed Q&A activities (e.g., posting questions, answers, accepted answers, or comments) without considering the obsolescence of such knowledge. In fact, we observe that as of Feb. 2020 some question threads are over 12 years old. Over time an answer can become outdated, turning into an obsolete answer especially in a very rapidly moving technical domain. For example, chronologically a later answer can address a question that was previously answered and whose asker had selected an earlier answer as the accepted one. The later answer might provide up-to-date knowledge — making the

[5]https://data.stackexchange.com/stackoverflow/queries

new "most suitable" answer. As a result, a question thread contains different flavors of problem solving activities, such as invalid, legacy, and cutting edge solutions/discussions among its answers/comments, all due to the fact that different activities can take place at different times throughout the lifetime of a question thread. 2) Much of the prior work did not take into consideration the completeness of question answering activities. More specifically, prior studies considered that a question is solved through a single answer, signaling the closing of the question answering process. However, an answer, even an accepted answer, may not be the "most suitable" solution. Other answers, or alternatively, comments, may become relevant or even more informative while the knowledge evolves or additional content is added over time (or even due to the varying perspective of different participants in an answer thread). A comment can update the knowledge in its associated answer, even though Stack Overflow does not award reputation points to commenting activities. Even worse, a comment can be hidden thus is difficult for users to retrieve the information that it provides.

1.1 Research Hypothesis

Research hypothesis: The valuable crowdsourced knowledge on Stack Overflow continues to grow and our dependence on such knowledge continues to increase. Such knowledge must be maintained to ensure that it is not obsolete. Prior research has primarily focused on the acquisition of such knowledge without exploring the maintenance practices of such knowledge.

This thesis aims to study the knowledge maintenance practices of the crowdsourced knowledge of Stack Overflow. Developers rely on Stack Overflow for their daily problem solving tasks. A deeper understanding of how up-to-date the crowdsourced knowledge in answers is, and how these answers become outdated has not been addressed from prior work. More importantly, how the accumulated knowledge on Stack Overflow can be maintained by following best practices within its community, is an essential task for Stack Overflow to keep evolving without too much obsolete code, invalid documentation, or legacy instructions. This thesis aims to answer these inquiries. Through the analysis of both answers and comments on Stack Overflow, we wish to empirically identify real-world practices about the maintenance of such crowdsourced knowledge within these answers/comments, types of unmaintained knowledge, and propose approaches for enhancing the maintenenace of such knowledge.

A great amount of crowdsourced knowledge resides within the question threads of Stack Overflow, including questions, answers (including both accepted and non-accepted answers), and comments. Knowledge changes over time, especially in software engineering. Stack Overflow activities (i.e., posting questions, answers, and comments) create and revise knowledge. For example, an answer can contain a dead link with a later revision replacing the dead link with a working one, or with a later comment pointing out the dead link. The crowdsourced knowledge on Stack Overflow is not only actively leveraged by developers, but it is also exploited by researchers (e.g., to create recommendation systems). The update of knowledge can also affect the accuracy and effectiveness of such recommenders. Thus, to understand the evolving crowdsourced knowledge can not only be useful to improve Stack Overflow, but can also be useful to other researchers to improve their recommenders. **We are the first to study the evolution of crowdsourced knowledge on Stack Overflow. Specifically, this thesis studies**

the question answering activities on Stack Overflow along three dimensions: (1) the obsolescence of answers (Chapter 3), (2) the informativeness of the comments that are associated with answers (Chapter 4), and (3) the retrieval of information in hidden comments (Chapter 5), in order to gain a better understanding of the maintenance process of crowdsourced knowledge on Stack Overflow.

1.2 Thesis Overview

The goal of this PhD thesis research is to better understand the maintenance practices of the crowdsourced Stack Overflow knowledge through the mining of the question answering activities. First, we survey the state-of-the-art research on understanding and supporting the maintenance of the crowdsouced knowledge of Stack Overflow. We then highlight the problems that are not addressed by the prior work, and propose approaches to tackle these problems.

We now present a brief overview of the work in this thesis.

1.2.1 Chapter 2: Literature Review

For our literature review, we focus on prior studies that examined the crowdsourced knowledge on Stack Overflow. We characterize and compare the surveyed literature along two dimensions:

- **Understanding and improving the quality of Stack Overflow knowledge:** Prior studies empirically explore how developers share knowledge on Stack Overflow and investigate the quality of the crowdsourced knowledge.

- **Leveraging Stack Overflow knowledge:** Prior studies mine the rich source of Stack Overflow data to support developers in their question answering activities.

From the literature review, we observe that prior studies did not examine the obsolescence of Stack Overflow knowledge or the informativeness/organization of comments in enhancing the knowledge.

Our motivation of this thesis is to understand the maintenance practices for such crowdsourced knowledge. The knowledge in Stack Overflow answers can become obsolete or augmented by the comments that are associated with answers. Therefore, further research should examine the obsolescence of such crowdsourced knowledge.

1.2.2 Chapter 3: The obsolescence of answers on Stack Overflow

Stack Overflow answers provide developers with a rich repository of knowledge to solve many programming related issues. For over a decade (i.e., from 2008 to 2020), these answers have assisted developers to solve millions of questions. The crowdsourced knowledge on Stack Overflow is inevitably changing due to the evolution of technology. As a result, certain knowledge embedded in Stack Overflow answers may become obsolete.

In order to better cope with the obsolescence of knowledge, we investigate how the knowledge in answers becomes obsolete and identify the characteristics of such obsolete answers through both quantitative and qualitative analyses. Our findings suggest that Stack Overflow should develop mechanisms to encourage the maintenance of answers (to avoid obsolete answers) and answer seekers are encouraged to carefully go through all information (e.g. comments) in answer threads.

1.2.3 Chapter 4: The informativeness of comments under answers

Answers posted on Stack Overflow help developers solve issues during software development. In addition to posting answers, users can also post comments to further discuss their associated answers. The collection of 32.3 million comments (as of Aug 2017) forms another repository of crowdsourced knowledge on top of the commonly-studied Stack Overflow answers.

To understand how the commenting activities contribute to the crowdsourced knowledge sharing, we investigate what users discuss in comments and analyze the timing and users roles of commenting activities. Our study suggests that the Stack Overflow commenting system can be leveraged for improving the maintenance and organization of the crowdsourced knowledge.

1.2.4 Chapter 5: The retrieval of information in hidden comments

Comments on Stack Overflow can provide additional information to enhance their associated answers, such as the obsolescence of answers. It is nontrivial to show more informative comments and hide less informative ones. Proper organization of comments can help developers more effectively retrieve information from the large collection of comments that are associated with Stack Overflow answers. Currently, Stack Overflow prioritizes the display of comments and as a result, 4.4 million comments (possibly including informative comments) are hidden by default from developers.

To understand whether the comment organization mechanism can effectively organize informative comments, we conduct a study to characterize the shown and hidden comments that are associated with Stack Overflow answers. We observe that the current comment organization mechanism does not work well for the purpose of retrieving informative comments. We proposed a classifier that can effectively distinguish informative comments from uninformative comments. Furthermore, we evaluate two alternative comment organization mechanisms to help developers better retrieve informative comments on Stack Overflow.

1.3 Thesis Contributions

This thesis empirically studies the maintenance practices of the crowdsourced knowledge on Stack Overflow and proposes actionable suggestions to help developers better manage their knowledge. The findings of this thesis highlight the importance of maintaining the technical knowledge in software engineering. The thesis makes the following key contributions:

- The lack of maintenance to answer obsolescence on Stack Overflow (in Chapter 3) suggests future research opportunities for improving the management of crowdsourced technical knowledge.
- Our study of comments that are associated with answers (in Chapter 4) shows that comments are informative although they are rarely integrated back into answers. Comments can be leveraged as an additional knowledge sharing channel.
- We observe that the current comment organization mechanism leads to the hiding of informative comments (in Chapter 5). We provide alternative organization mechanisms in order to help developers better retrieve information from comments.

CHAPTER 2

Literature Review

MAINTENANCE practices of crowdsourced knowledge on Stack Overflow are studied in this thesis. Understanding the real-world question answering activities is a first step towards helping improve such maintenance practices. In this chapter, we survey prior work along two dimensions, i.e., understanding the quality of crowdsourced knowledge, and leveraging such crowdsourced knowledge.

2.1 Literature Selection

Prior studies investigated various artifacts that are associated with Stack Overflow. We choose to survey papers that were written between 2008 and 2019 since the Stack Overflow website was established in 2008 (spanning over a decade of extensive research efforts). We search for such papers in major software engineering journals and conferences to capture as many studies as possible.

Table 2.1 shows the research venues from which we started our literature review process. Another resource for identifying relevant research is a Meta Stack Exchange discussion – "*academic papers using Stack Exchange data*"[1]. To improve the coverage of our literature review, we also follow the papers that each reviewed paper cited. we detail each category of papers below.

2.2 Understanding and Improving the Quality of Knowledge On Stack Overflow

One significant challenge that Q&A websites have is ensuring the quality of their knowledge (Agichtein et al., 2008; Allamanis and Sutton, 2013; Asaduzzaman et al., 2013; Chua and Banerjee, 2015; Yao et al., 2013; Ponzanelli et al., 2014b; Zhang et al., 2018). Therefore, numerous studies have been done to understand and improve the quality of knowledge on Q&A websites. The majority of prior studies defined the quality of content on Stack Overflow from the presentation perspective (e.g., code and text) based on the latest status of the activities. For example, Allamanis and Sutton (2013) investigated

[1] https://meta.stackexchange.com/q/134495/

Table 2.1: Names of starting venues (conferences and journals) for our literature review.

Venue Type	Venue Name	Abbreviation
Journal	IEEE Transactions on Software Engineering	TSE
Journal	ACM Transactions on Software Engineering and Methodology	TOSEM
Journal	Empirical Software Engineering	EMSE
Conference	ACM SIGSOFT Symposium on the Foundation of Software Engineering/ European Software Engineering Conference	FSE/ESEC
Conference	International Conference on Software Engineering	ICSE
Conference	International Conference on Automated Software Engineering	ASE
Conference	International Conference on Software Maintenance and Evolution	ICSME
Conference	International Conference on Software Analysis, Evolution, and Reengineering	SANER
Conference	International Conference on Mining Software Repositories	MSR

Stack Overflow questions by finding their concepts, code, topics and types. Asaduzzaman et al. (2013) studied the unanswered questions on Stack Overflow and observed that some questions did not receive any answer due to the question being too short, not clear, too hard, or unrelated (not related to the Stack Overflow community). Chua and Banerjee (2015) examined the increase in the number of unanswered questions by constructing a framework to explain why some questions remain unanswered. Yao et al. (2013) observed that the quality of an answer is strongly associated with the quality of its question. Yao et al. (2015) also proposed a family of algorithms to identify high quality posts on Q&A websites. Ponzanelli et al. (2014b) studied the relationship between a set of proposed factors and the quality of a post on Stack Overflow. Ponzanelli et al. (2014c) also built a classification model to identify high quality and low quality questions. Duijn et al. (2015) investigated the impact of various factors that are related to text and code on the quality of a question on Stack Overflow and observed that the code/text ratio is the most important factor. Calefato et al. (2016) performed a study to identify the best answers on Stack Overflow and observed that the classifier that was built only on text features (e.g., length of body) could achieve good performance. Zhang et al. (2018) conducted an empirical study on the prevalence and severity of API misuse on Stack Overflow. They observed that even the posts that are accepted as correct answers or upvoted by others are not necessarily more reliable than other posts in terms of API misuse. Wang et al. (2017) investigated various factors (e.g., factors related to questions, answers, and users) that potentially affect the speed of getting an accepted answer and observed that the most important factor is the activity level of the answerer community. Chen et al. (2017, 2018) proposed an approach to help users on Stack Overflow fix grammar issues based on prior editing records. These aforementioned studies leveraged the knowledge on Stack Overflow without considering that such knowledge can become obsolete over time.

Without users, online Q&A websites would not gain large amount of knowledge. Pal et al. (2011) observed that experts can be identified from their prior behavior. Their study suggested that some potential experts had become inactive, thus pointing out the value of early identification of experts. Wang et al. (2018a) analyzed three badges (i.e., Strunk & White, Copy Editor, and Archaeologist) related to post editing on Stack Overflow, and observed a spike of post revisions on the badge-awarding days. Treude et al. (2011) categorized types of questions asked on Stack Overflow, and analyzed which answers are answered effectively. To better understand how complex problems on Stack Overflow are handled, Hanrahan et al. (2012) characterized both problem difficulty and user expertise. To gain insights into the development community, Barua et al. (2014) analyzed the textual content of Stack Overflow, discovered the main topics by latent Dirichlet allocation (LDA), then studied the relationships and trends among different topics. Vasilescu et al. (2014) studied how users migrate questions from the R user support mailing list (r-help) to crowdsourced knowledge sharing platforms (i.e., Stack Overflow and Cross Validated) and observed that users can get faster answers on crowdsourced sites than on specialized mailing lists. Murgia et al. (2016) conducted experiments to simulate a user to answer questions (i.e., an answering bot) on Stack Overflow, and observed that even though the performance of the bot is comparable to human users, humans have a higher expectation from a machine than a human. Upadhyay et al. (2017) model the evolution of a user's expertise over time, and observed that prolific learners tend to share knowledge with high value. Mamykina et al. (2011) analyzed Stack Overflow's design features and observed that its highly responsive and iterative design approach and its incentive system contributed to its success.

Prior studies defined the quality of knowledge on Stack Overflow from a static point of view (e.g., the presence of code, the text to code ratio, and the length of the text). However, these studies did not consider the change in the quality of content over time; namely, the obsolescence of knowledge on Stack Overflow. Storey et al. (2014) pointed out that crowdsourced documentation may be out of date although many consider such documentation to be very valuable. Calefato et al. (2016) identified the best answers in legacy developer forums and noticed that once a newer, better answer is entered the answer previously identified as the best may become obsolete. Ragkhitwetsagul et al. (2017) observed that 19% of the Stack Overflow answerers in their study rarely or never fix their outdated code. Zhou and Walker (2016) also observed that code examples on the web could easily become outdated. This thesis analyzes all obsolete answers and all the comments that are associated with answers on Stack Overflow to gain an empirical understanding of the obsolescence of answers and the updating of answers through commenting activities.

Most prior studies of Stack Overflow crowdsourced knowledge do not examine the obsolescence of such knowledge. Our motivation of this thesis is to understand the maintenance practices for such crowdsourced knowledge.

2.3 Leveraging the Knowledge from Stack Overflow

Stack Overflow accumulates a large amount of knowledge and researchers have done a remarkable number of studies to leverage the knowledge on Stack Overflow to facilitate various types of development and maintenance activities. For example, Zagalsky et al. (2012) recommended high-quality code fragments on Stack Overflow by leveraging knowledge from Stack Overflow. Treude and Robillard (2016) developed a tool to enrich API documentation with "insight sentences" extracted from Stack Overflow. Vassallo et al. (2014) extracted discussions from Stack Overflow to generate JavaDoc automatically. Wong et al. (2013) leveraged questions and answers on Stack Overflow to automatically generate comments in system source code. Gao et al. (2015) proposed an automated approach to fix recurring crash bugs by leveraging information (e.g., questions with similar crash traces) on Q&A websites. In order to help users identify the most appropriate channel to ask questions, several approaches were developed to help users generate tags automatically when they post questions (Wang et al., 2012, 2018b; Xia et al., 2013). For example, Wang et al. (2012) leveraged the tag information on Stack Overflow to infer semantically related software terms. By leveraging Stack Overflow discussions, Chen et al. (2019) built a software engineering thesaurus with software-specific terms and morphological forms using word embedding and lexical rules, they also demonstrated several thesaurus based applications, such as tag synonym detection and software-specific spell checker. Glassman et al. (2018) proposed an interactive visualization tool to summarize online code examples at scale. They observed that their interactive visualization helps users answer more API usage questions correctly. Mujumdar et al. (2011) proposed an approach to collect and display crowd-sourced bug fix suggestions for the Ruby programming language. Liu and Zhong (2018) mined program repair templates on Stack Overflow to repair bugs. Ma et al. (2019) proposed a multi-layer neural network model to extract API mentions from the informal text in Stack Overflow posts. Hoque and Carenini (2014) developed a tool to analyze textual content, and identify topics and opinions with visualization to support further exploration. Li et al. (2016) proposed a recommendation system to provide users with hyperlinks that are highly recognized by the Stack Overflow community. Luca et al. (2016) proposed a recommendation system to notify developers of relevant discussions on Stack Overflow within the Integrated Development Environment (IDE). They observed that the recommender can be affected by the change of information on Stack Overflow over time (Ponzanelli et al., 2014a; Luca et al., 2016).

Although prior studies of the Stack Overflow ecosystem mainly focus on questions and answers, some studies have taken comments into account. For example, Baltes et al. (2018) investigated the evolution of Stack Overflow posts and the temporal relationship between edits and comments. They observed that posts grow over time in terms of the amount of text and the number of code blocks, but the size of a single block is relatively stable. They also observed that some comments may trigger an edit on a post although the correlation is weak (i.e., 0.26). In predicting the long-term value of question threads, Anderson et al. (2012) observed that the number of comments in answers has a significant predictive power. Similarly, Tian et al. (2013) observed that answers with more comments are more likely to be accepted. Asaduzzaman et al. (2013) analyzed both questions and their comments to find out why questions were unanswered. They observed that users may post actual solutions in comments associated with these unanswered questions. Calefato et al. (2015) analyzed the sentiment of comments in their study of answer acceptance. They observed that the sentiment of comments significantly impacts the chance of answer acceptance. Dalip et al. (2013) observed that comment can provide additional information to improve the associated posts. In addition, they observed that commenting is useful for measuring the engagement of users in an answer, and this engagement improves the rating of answers. Chang and Pal (2013) proposed a question routing framework to recommend answerers and commenters to a question. They observed the importance of commenting in further clarifications and the improvement of the quality of an answer. Sengupta and Haythornthwaite (2020) examined comments that are associated with 50 posts on Stack Overflow and observed that the discussion is mainly about clarification and modification. This thesis leverages comments to identify obsolete answers on Stack Overflow, and to understand how comments enhance their associated answers.

Prior studies learned common developer practices from the crowdsourced activities, and built tools by leveraging Stack Overflow's crowdsourced knowledge. However, prior studies did not consider the obsolescence of such knowledge. The crowdsourced knowledge embedded in Stack Overflow answers can become obsolete or augmented by their associated comments. Therefore, follow-up research should examine the obsolescence of such crowdsourced knowledge.

CHAPTER 3

The obsolescence of answers on Stack Overflow

Stack Overflow accumulates an enormous amount of software engineering knowledge. However, as time passes, certain knowledge in answers may become obsolete. Such obsolete answers, if not identified or documented clearly, may mislead answer seekers and cause unexpected problems (e.g., using an out-dated security protocol). In this chapter, we investigate how the knowledge in answers becomes obsolete and identify the characteristics of such obsolete answers. We find that: 1) More than half of the obsolete answers (58.4%) were probably already obsolete when they were first posted. 2) When an obsolete answer is observed, only a small proportion (20.5%) of such answers are ever updated. 3) Answers to questions in certain tags (e.g., node.js, ajax, android, and objective-c) are more likely to become obsolete. Our findings suggest that Stack Overflow should develop mechanisms to encourage the whole community to maintain answers (to avoid obsolete answers) and answer seekers are encouraged to carefully go through all information (e.g., comments) in answer threads.

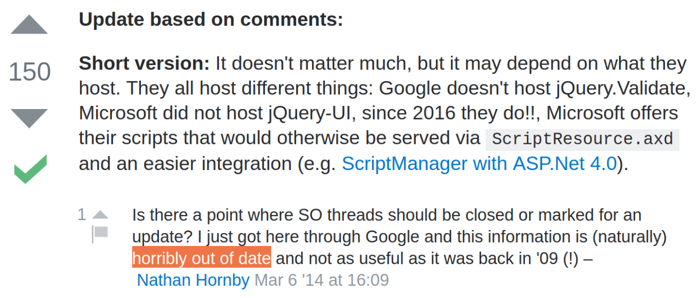
An earlier version of this chapter is published in the IEEE Transactions on Software Engineering Journal (TSE) (Zhang et al., 2019b).

3.1 Introduction

Technical Q&A websites have revolutionized how users seek knowledge on the Internet. When users face unsolvable problems, they often try to search for solutions via search engines (e.g., Google). A case study shows that Google developers perform an average of 12 code search queries each weekday (Sadowski et al., 2015). Search engines commonly direct users to technical Q&A websites in response to their queries. As a prominent example, Stack Overflow, one of the most popular technical Q&A websites, has collected an enormous amount of knowledge, which includes 15 million questions, 23 million answers, and 62 million comments as of September 2017[1].

Software systems evolve at a rapid pace nowadays. For instance, Android has released 16 major versions and 53 minor versions since September 2008 (as of August 2018) (SocialCompare, 2018). Android is evolving at a rate of 115 API updates per month on average according to a study by McDonnell et al. (2013). Such rapid evolution may make the knowledge in some Stack Overflow answers obsolete over time. Fig. 3.1 presents an example of such a case, where the user was directed from Google to a Stack Overflow answer. However, the user observed that the content of the answer thread (including the answer and the discussions in the comments) was obsolete and asked whether Stack Overflow has any mechanisms to handle such a situation[2]. Additionally, a survey of 453 Stack Overflow users reports that outdated code on Stack Overflow is one of the most important issues that users complain about (Wu et al., 2018).

[1]https://data.stackexchange.com/stackoverflow/
[2]https://stackoverflow.com/posts/comments/33754357/



Update based on comments:

Short version: It doesn't matter much, but it may depend on what they host. They all host different things: Google doesn't host jQuery.Validate, Microsoft did not host jQuery-UI, since 2016 they do!!, Microsoft offers their scripts that would otherwise be served via `ScriptResource.axd` and an easier integration (e.g. ScriptManager with ASP.Net 4.0).

1 Is there a point where SO threads should be closed or marked for an update? I just got here through Google and this information is (naturally) horribly out of date and not as useful as it was back in '09 (!) – Nathan Hornby Mar 6 '14 at 16:09

Figure 3.1: An example of a user complaining in a comment that the Stack Overflow answer thread (including the answer and the discussions in the comments) is obsolete.

Obsolete answers are detrimental to answer seekers. For example, a user found a piece of code that matches his/her needs and reused it in his/her own project. However, the user may not realize that the used APIs in the code are obsolete. Such obsolete APIs could potentially result in software quality problems (e.g., using an outdated security framework API), and may increase maintenance difficulties. Therefore, it is necessary to provide insights on how to track or alleviate this problem.

In this chapter, we study Stack Overflow answer threads (each answer thread includes all answers to a question (i.e., accepted & not-accepted answers) and all the comments that are associated with them) to understand how the knowledge that is embedded in answer threads becomes obsolete and the characteristics of such obsolete answers, and to provide actionable suggestions. We also perform a qualitative study to understand the evidence that users provide to support their obsolete observations and the activities that users perform after an answer is observed as obsolete. We structure our study by answering the following four research questions:

- **RQ1: What happens when an answer is observed as obsolete?**

  More than half of the studied obsolescence observations refer to answers that were probably already obsolete when they were first posted. Most users did not update obsolete answers or add new answers to address the observed obsolescence. On average, it took 118 days for users to react to an observed obsolete answer.

- **RQ2: Whether answers to questions that are associated with particular tags are more likely to become obsolete?**
  Answers to questions that are associated with certain tags (e.g., node.js, ajax, android, and objective-c) are more likely to become obsolete.

- **RQ3: What are the potential reasons for answers to become obsolete?**
  The majority of the answers become obsolete due to the evolution of their associated programming languages and/or third party libraries, APIs, and frameworks. Therefore, users need to pay more attention to such answers when looking for answers on Stack Overflow.

- **RQ4: Who observes obsolete answers and what evidence do they provide?**
  The majority of the obsolete answers were not observed by the original answerers. Also, most obsolescence observations are supported by evidence (e.g., updated information, a version information, or a reference).

Based on our observations, we provide actionable suggestions for Stack Overflow to alleviate the problem of obsolete answers. For example, an automated tool based on machine learning techniques or even simple keyword search could be built to identify existing obsolete answers on Stack Overflow, or help answerers identify obsolete answers in real-time as an answer is being typed. Moreover, Stack Overflow should develop mechanisms (e.g., rewarding badges or reputation scores) to encourage the whole community to maintain answers and flag obsolete answers. We also provide suggestions for users. For example, answerers are encouraged to include whenever possible information about the valid version or time of the knowledge when contributing answers. Answer seekers are encouraged to carefully go through the comments that are associated with answers in case the obsolescence of an answer is noted in the comments, especially for the answers in questions that are related to particular tags (e.g., node.js, ajax, android, and objective-c). We also shared our findings with Stack Overflow developers who concurred with our findings, and they were interested in investigating approaches to generate version tags to indicate the valid version for a platform or programming language used in obsolete answers.

Chapter Organization: The rest of the chapter is organized as follows. Section 3.2 presents the subject of study and introduces our data collection process. Section 3.3 presents the results of our research questions. Section 3.4 discusses the implications of our study. Section 3.5 presents the potential threats to the validity of our observations. Finally, Section 3.6 concludes the chapter.

3.2 Case Study Setup

This section describes the subject of study and the process that we follow to collect the data for our case study.

3.2.1 Subject of Study

We briefly introduce the mechanism of question answering on Stack Overflow and discuss how answers on Stack Overflow can become obsolete.

The question answering mechanism on Stack Overflow

Stack Overflow provides a platform for the asking and answering of questions. Askers post questions which include a textual description on Stack Overflow. Askers can include code snippets and other references (e.g., URLs or images) to enrich their posted question. Each question may receive multiple answers from different answerers. However, at most one answer could be accepted by the asker as the *accepted answer* to indicate that this particular answer is the most suitable/correct one.

In the rest of the thesis, we refer to a question, its corresponding answers (i.e., both accepted and not-accepted answers) and all the associated comments with these answers together as a *question thread*. We refer to an answer (could either be accepted or not-accepted answers) and its comments as an *answer thread*.

Users tag questions[3] into well-defined categories. Tags capture the topics with which a question is associated. Each question can have at most five tags and must have at least one tag. Askers need to specify the tags when they create a question. In the rest of the chapter, we say that an answer is associated with a particular tag if the answer belongs to a question that is associated with that tag. In RQ2, we study whether answers to questions that are associated with particular tags are more likely to become obsolete.

Obsolete answers on Stack Overflow

As we noted in Section 3.1, Stack Overflow users complain about the obsolescence of answers. There are various reasons that an answer could become obsolete on Stack Overflow. For instance, APIs could become deprecated later on when a new API version is released. For a better understanding of answer obsolescence on Stack Overflow,

[3]https://stackoverflow.com/help/tagging

we present the possible activities that could happen after an answer becomes obsolete in Fig. 3.2. An answer probably becomes obsolete after some time since its creation (alternatively an answer might be obsolete even before it is posted) (see Section 3.3.1). An obsolete answer probably would be observed by a user on Stack Overflow (i.e., obsolescence observation). Users may also discuss the obsolescence afterwards and update their answers accordingly.

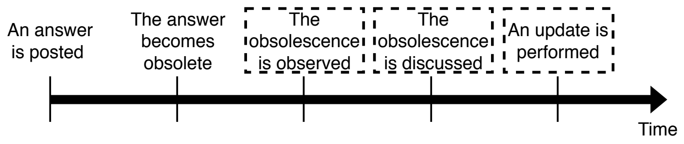

An answer is posted
The answer becomes obsolete
The obsolescence is observed
The obsolescence is discussed
An update is performed
Time

Figure 3.2: A possible flow of activities that could occur after an answer becomes obsolete. Activities in dotted box are optional and might not happen in all cases.

Obsolete answers are problematic on Stack Overflow. However, there exists no mechanisms in place today to alleviate the problem of obsolete answers. Thus, in this chapter, we wish to closely examine the obsolescence of answers in an effort to propose ways to help Stack Overflow deal with such answers in an effective & efficient manner. To do so, we investigate what happens once someone identifies that an answer has become obsolete and whether answers in questions that are associated with particular tags are more likely to become obsolete. We also investigate who observes obsolete answers and what evidence do they provide to support their observations.

Based on our study, obsolete answers could be categorized into two classes: *legacy* or *invalid*. We consider an obsolete answer as a *legacy* answer if it can still be used or applied, but it may not be recommended anymore since a newer answer might be better or more appropriate. For example, a comment[4] points out that an answer is "obsolete in Rails >= 3.0.0", which indicates that the accepted answer only applies to

[4]https://stackoverflow.com/posts/comments/30559321/

Rails version 3.0.0 or below. Nevertheless, users who use earlier versions may find this answer still useful. On the other hand, an *invalid* answer indicates that the obsolete answer is not valid or that it no longer works. Users who might have successfully applied the particular answer earlier would now run into errors or complete failures. One example of an invalid answer is related to an old http protocol (such as RFC 2616[5]), which is deprecated. For example, a comment[6] mentions that "RFC 2616 has been obsoleted".

Thus, we are interested in investigating obsolete answers on Stack Overflow, to understand obsolescence reasons that happen and to provide some insights into addressing the obsolescence of answers.

In this section, we describe how we collect the dataset that we used to answer our research questions.

3.2.2 Data Collection

To understand the obsolescence of answers on Stack Overflow, we need to identify answer threads (both accepted and not-accepted answers) with obsolete knowledge. We observe that users occasionally leave comments to indicate that an answer is obsolete (see Fig. 3.1). Based on this observation, we identify answer threads that have obsolete knowledge using the two following criteria:

1. A comment in an answer thread contains one of the keywords "deprecated", "outdated", "obsolete" or "out of date".

2. The same keyword from criteria 1 ("deprecated", "outdated", "obsolete" or "out of

[5] https://www.ietf.org/rfc/rfc2616.txt
[6] https://stackoverflow.com/posts/comments/61676900/

date") does not appear in the question (including the question title and body) of its thread or any of its answers. The reason behind this criteria is that if any of the keywords appear in the content of a question or an answer, it may indicate that the question or answer itself is related to an "obsolete" topic rather than being a sign that the answer is likely obsolete.

The purpose of our selection criteria is not to collect all possible answer threads with obsolete knowledge, but to collect sufficient data for a relatively comprehensive analysis, while minimizing the bias of our sample.

We downloaded the Stack Overflow data from archive.org[7]. The data was published on August 31, 2017 by the Stack Exchange community. The data contains information about badges, comments, post history, post links, posts, tags, users, and votes. Using our selection criteria, we ended up with 52,177 answer threads, which include 58,201 comments that mention obsolescence. These collected threads span 12,629 tags. We published our data set including the labeled data online[8].

The accuracy of our heuristic-based approach is 75% based on our manual verification of a statistically representative sample with a 99% confidence level and a 5% confidence interval. For each observed answer obsolescence, we examine the support evidence from the user who observed the obsolescence together with online information (e.g., documentation for API, programming language, and framework), to verify if the answer is really obsolete. If no obsolescence is identified, we label it as a false positive. 167 answers out of the 669 are false positives. The two main reasons for the false positive cases are: 1) Instead of indicating the obsolescence of an answer in the comment, the content that is discussed by users in the comments is related to certain

[7] www.archive.org/details/stackexchange/

[8] https://github.com/SAILResearch/replication-obsolete_answers_SO/

topics which use our keywords of interest (e.g., "obsolete" and "out of date"). For example, in a comment[9] the user mentions that "unless you have some kind of locking mechanism (which I'd argue against), the result of the call would be obsolete as soon as you got it". This comment did not indicate the obsolescence of the answer. 2) Users either ask whether the answer is obsolete or express that the answer probably will become obsolete soon. For example, in a comment[10] "because php is changing a lot and in upcoming versions this might be deprecated", the user did not observe any specific obsolete software artifact in the answer, but just simply expressed the user's general feeling that PHP is evolving very fast and that it is obsolete-prone.

3.3 Case Study Results

In this section, we present the results of our research questions. For each research question, we present the motivation of the research question, the approach to address the research question, and our experimental results for the research question.

3.3.1 RQ1: What happens when an answer is observed as obsolete?

Motivation: It is very important to keep answers up-to-date on Stack Overflow as we noted in Section 3.1. However, it is not known how the Stack Overflow community handles obsolete answers. In this RQ, we are interested in examining how the Stack Overflow community deals with obsolete answers after such obsolescences are observed. More specifically, we would like to investigate the activities that occur after someone observes the obsolescence of an answer. Through such analysis, we expect

[9] https://stackoverflow.com/posts/comments/2293838/
[10] https://stackoverflow.com/posts/comments/65380866/

to provide an overview of how the community handles the obsolescence of answers once they are observed and a reasonable understanding of the severity of the answer obsolescence problem for Stack Overflow developers to pay attention to.

Approach: Based on our observation during the data collection process, there are two types of actions that might occur after an answer is observed to be obsolete: 1) updating the obsolete answer (*update*); 2) creating a new updated answer (*new*). As a result of the above two types of actions, another action might occur, that is the switching of the accepted answer (*switch*). For example, the original asker may cancel the currently accepted answer and mark an updated one or a newly created one as the accepted answer. To understand what occurs after an obsolescence is observed, we perform both quantitative and qualitative analyses. An overview of our analyses is presented in Fig. 3.3.

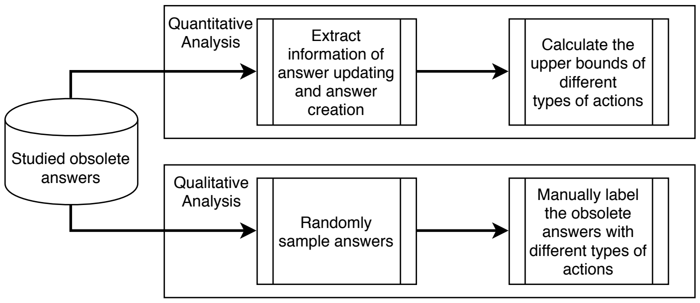

Studied obsolete answers
Quantitative Analysis
Extract information of answer updating and answer creation
Calculate the upper bounds of different types of actions
Qualitative Analysis
Randomly sample answers
Manually label the obsolete answers with different types of actions

Figure 3.3: An overview of our analysis in RQ1.

In the quantitative analysis, we captured an overall picture about when the obsolescence is observed and how users react to obsolescence observations in terms of the three types of actions (i.e., *update*, *new*, and *switch*). To compute the number of cases in which users update the obsolete answer (type *update*), we counted the number of obsolete answers that have been edited after an obsolescence observation. Such a number gives us an upper bound estimate since updating an obsolete answer is not the only reason for editing an existing answer. We computed the number of type *new*, using a similar way as type *update*, i.e., computing an upper bound estimate. Adding updated information is one possible reason for creating a new answer, but there could be other reasons, such as adding an alternative answer. Thus, by computing the number of question threads that have new answers after an obsolescence observation, we can get an upper bound on the number of instances of type *new*. We are able to compute the number of type *switch* instances based on the historical records of answers. *However, we did not find any case of type switch. Therefore we focus the rest of our analysis on type update and new.*

In the qualitative analysis, we performed a manual study to calculate the exact occurrences of type update and new actions. We randomly sampled a statistically representative sample of 669 obsolete answers (including all their associated comments) from our studied 52,177 obsolete answers using a 99% confidence level with a 5% confidence interval. Since there are 167 (25% out of these sampled 669 answers) false positive cases, to make sure we have enough number of actual obsolete answers (to achieve a 99% confidence level with a 5% confidence interval), we kept randomly sampling from the rest of the 52,177 obsolete answers until we reach a total number of 669 actual obsolete answers. We manually performed a lightweight open coding-like process (Seaman, 1999; Seaman et al., 2008) to check the sampled answers, their edit records, and the associated comments and other answers in the same question thread in order to label the types (update and new) of the performed actions. We also recorded the time for users to react. Note that the qualitative analysis of other RQs are also performed on these 669 actual obsolete answers.

This process involves 2 phases and is performed by two researchers (i.e., A1–A2, with me being one of them):

- Phase I: A1 and A2 independently categorize the types of performed actions for each of the studied 669 answers. A1 & A2 took notes regarding the deficiency or ambiguity of the labeling for these obsolete answers.
- Phase II: A1, A2 discussed the coding results that were obtained in Phase I to resolve any disagreements until a consensus was reached. The inter-rater agreement of this coding process has a Cohen's kappa of 0.96 (measured before starting Phase II), which indicates that the agreement level is high (Viera et al., 2005).

Quantitative Results: More than half of the studied obsolete answers were probably already obsolete as they were being posted. Fig. 3.4 presents the time gap between the answer creation time and the time at which the obsolescence observation was noted. An interesting observation is that 58.4% of the studied answers were noted as obsolete within 24 hours after their creation. This suggests that more than half of the answers were probably already obsolete when they were first posted. One possible explanation is that even the answerer himself/herself did not realize that their answer is obsolete. For example, Fig. 3.5 shows an answerer[11] who was using an obsolete API in his original answer. A commenter pointed out within 2 minutes that the answer is obsolete, then the answerer updated his answer.

More than half of the users do not update their answers or add new answers after their answers are noted as obsolete. In terms of an upper bound estimation, 49.8% of the studied obsolete answers were either updated (type *update*) or added with new

[11] https://stackoverflow.com/questions/4650483/

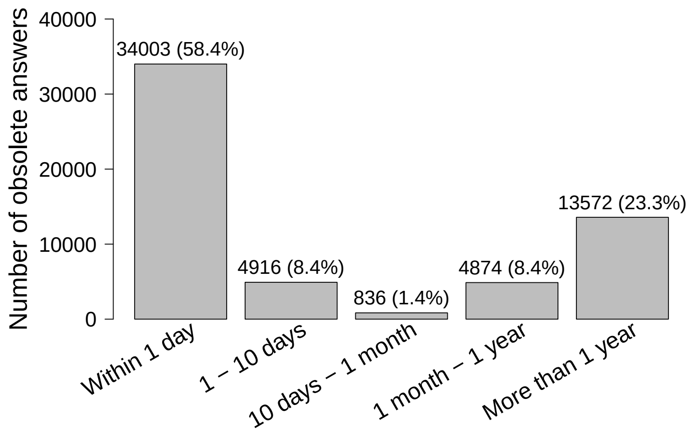
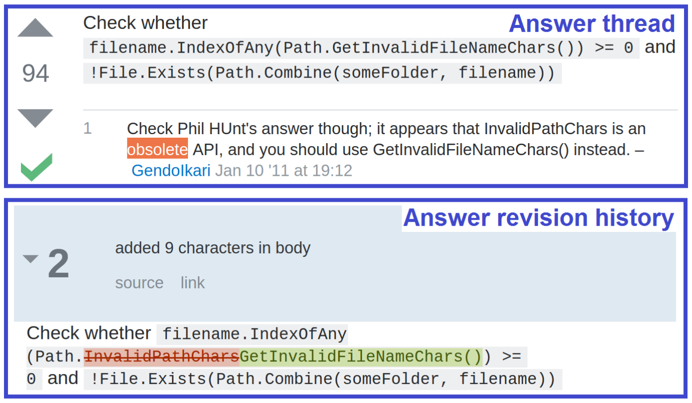


Figure 3.4: Number of obsolete answers vs number of days to point out the obsolescence of the answers.



Check whether `filename.IndexOfAny(Path.GetInvalidFileNameChars()) >= 0` and `!File.Exists(Path.Combine(someFolder, filename))`

1 Check Phil HUnt's answer though; it appears that InvalidPathChars is an obsolete API, and you should use GetInvalidFileNameChars() instead. – Gendolkari Jan 10 '11 at 19:12

added 9 characters in body

source link

Check whether `filename.IndexOfAny(Path.~~InvalidPathChars~~GetInvalidFileNameChars()) >= 0` and `!File.Exists(Path.Combine(someFolder, filename))`

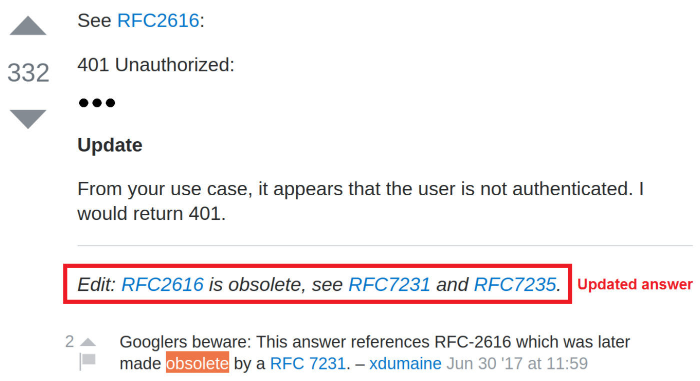
Figure 3.5: An example of an answer whose poster didn't realize his answer was obsolete when he created the answer.

answers (type *new*). More specifically, less than 27.4% (upper bound) of the obsolete answers got updated after being noted as obsolete, and in 33.3% of the cases users added new answers. Note that there are answer threads that have both updated and new answers after answer obsolescence is observed in comments. We also check the editing records of the accepted answers. We observe that 44.1% of the studied obsolete answers are the accepted answers. We find that 30.7% of the obsolete accepted answers got updated (type *update*) after being noted as obsolete, while only 24.8% of not-accepted answers got updated. These findings suggest that accepted answers are more likely to be updated after an obsolescence was noted compared with not-accepted answers. Nevertheless, it is interesting to note that users still do read not-accepted answers and do note their obsolescence (indicating the importance of all answers not just the accepted ones). Future studies of Stack overflow should also explore not-accepted answers instead of mostly focusing on accepted answers.

It takes 227 days on average for users to provide the first update for an obsolete answer after the obsolescence is observed in a comment, while it takes 198 days on average to add the first new answer after the obsolescence is observed.

Qualitative Results: Users updated their obsolete answers in 20.5% of the cases and added new answers in 6.3% of the cases in our qualitative study. On average, it took 118 days for users to react to an answer obsolescence observation. For example, we present a case[12] in Fig. 3.6. The answer was edited on August 11, 2017 to update the obsolete answer (i.e., information about a protocol). We also notice that it took 119 days on average for users to update obsolete answers, and it took 128 days on average to add new answers after an answer obsolescence was observed.

[12] https://stackoverflow.com/questions/3297081/



332

See RFC2616:

401 Unauthorized:

•••

Update

From your use case, it appears that the user is not authenticated. I would return 401.

Edit: RFC2616 is obsolete, see RFC7231 and RFC7235.

2 Googlers beware: This answer references RFC-2616 which was later made obsolete by a RFC 7231. – xdumaine Jun 30 '17 at 11:59

Figure 3.6: An example of an obsolete answer that was updated.

More than half of the studied obsolete answers were probably already obsolete as they were being first posted. Most users did not update obsolete answers nor add new answers to address the obsolescence of an answer. Even for users who performed actions to deal with the obsolete answers, on average it took them 118 days after the obsolescence of the answer was noted.

3.3.2 RQ2: Whether answers to questions that are associated with particular tags are more likely to become obsolete?

Motivation: Some particular topics (i.e., associated Stack Overflow tags) evolve more rapidly than others. For example, Android is evolving at a rather rapid pace (McDonnell et al., 2013). Such rapid evolution may lead to a higher likelihood for the answers of such tags to become obsolete. Therefore, in this RQ, we examine which topics (i.e., tags in our study) of answers are more prone to have obsolete answers. By understanding this, we could provide some suggestions for the answer seekers when they search for answers on Stack Overflow (e.g., which answers relative to their associated tags require more caution since they are more likely to become obsolete). We could also provide insights into the severity of answer obsolescence across different tags, so that Stack Overflow developers could implement mechanism to solve or alleviate the specific issue.

Approach: We conduct a quantitative analysis to examine which tags are more likely to have obsolete answers. To understand which tags are prone to have obsolete answers, we compute the number of obsolete answers to questions that are associated with a particular tag and *normalize this number by dividing it with the total number of answers for a particular tag on Stack Overflow.*

Results: Answers that are related to certain tags (e.g., node.js, ajax, android, and objective-c) are more likely to become obsolete. Fig. 3.7 ranks the tags according to the ratio of obsolete answers to the total number of answers in each tag in our studied questions. The most obsolete-prone tag is node.js, where 0.36% of the answers with this tag have been pointed out to be obsolete. 0.34%, 0.32%, and 0.32% of the answers with tags ajax, android and objective-c are obsolete, respectively.

Due to the popularity of mobile apps, many developers are involved in mobile app development, thus leading to an increase in the number of mobile app related questions and answers on Stack Overflow. Answers related to mobile app technologies are more likely to become obsolete because of the fast progress of this field. For instance, Android has released 16 major versions and 28 levels of API from September 2008 to Aug 2018 (SocialCompare, 2018) and there are, on average, 115 API updates per month (McDonnell et al., 2013). Another example is iOS where Apple has released 12 major versions and 103 minor versions of iOS from June 2007 to Sept 2018[13]. Such

[13]https://en.wikipedia.org/wiki/IOS_version_history

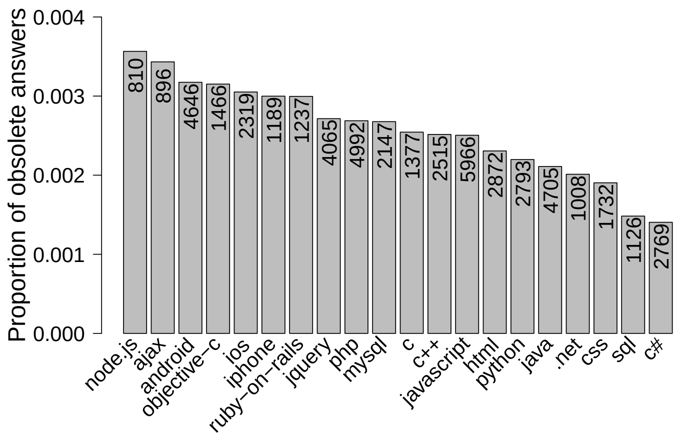


Figure 3.7: The top 20 tags ranked by the ratio of obsolete answers to the total number of answers in each tag.

rapid updating (in both mobile operating systems and their associated tooling) makes the answers related to mobile development more likely to become obsolete. This phenomenon has also been observed by users on Meta Stack Overflow[14]. For example, a user mentions that "... Android, which as a platform is only 7 years old. It has changed drastically over that time, and answers to questions that were posed 3 or 5 years ago are out of date. In some cases the answers are inappropriate or just wrong for current developers ..."[15]. A similar situation arises to answers related to web development, such as node.js, ajax, ruby-on-rails, and jquery.

There is no statistically significant difference in the obsolescence ratio (i.e., number of obsolete answers divided by total number of answers in a particular tag), between tags with large and small number of answers. We analyze all the tags with at least 1,000 answers. The Spearman correlation between the obsolescence ratio and the number of answers in a tag is -0.049. We divide Stack Overflow communities into 7 groups based on the number of answers that are associated with a tag (i.e., 1K - 5K,

[14]https://meta.stackoverflow.com/
[15]https://meta.stackoverflow.com/questions/309152/

5K - 10K, 10K - 50K, 50K - 100K, 100K - 500K, 500K - 1M, and >1M), then we run the Mann-Whitney test between each pair of different groups. We also perform the Benjamini Yekutieli procedure (Benjamini and Yekutieli, 2001) to adjust the p-values to handle the impact of multiple comparisons. We find that the adjusted p-values are greater than 0.05 for all the tests (i.e., no statistically significant difference), indicating that no matter how large the communities are, there are no differences in the obsolescence ratio of different communities. Answer obsolescence is a phenomenon across all communities on Stack Overflow.

Answers to questions that are associated with tags such as node.js, ajax, android, and objective-c are the most likely to become obsolete. There is no statistically significant difference in the obsolescence ratio between tags with a large versus a small number of answers.

3.3.3 RQ3: What are the potential reasons for answers to become obsolete?

Motivation: Various reasons could lead to obsolescence (e.g., a release of a new version of a framework). We are interested in investigating why answers on Stack Overflow become obsolete. Knowing this will help Stack Overflow plan better ways to avoid answer obsolescence. We can also provide insights for users to be more careful with such answers.

Approach: We perform a qualitative analysis to study the reasons of answer obsolescence. In this experiment, we use the same data, i.e., the randomly selected 669 answers (including all their associated comments) out of the 52,177 answers from RQ1, in order to achieve a confidence level of 99% with a confidence interval of 5%. We manually derived and categorized the obsolescence reasons (as shown in Table 3.1) from the randomly sampled answers threads. Note that an answer can have multiple reasons for becoming obsolete. We performed a lightweight open coding-like process (Seaman, 1999; Seaman et al., 2008) similar to RQ1 to identify the reasons of obsolescence. This process involves 3 phases and is performed by two researchers (i.e., A1–A2, with me being one of them):

- Phase I: A1 derived a draft list of obsolescence reasons based on 50 random answers. Then, A1 and A2 use the draft list to categorize the answers collaboratively. During this phase the reasons were revised and refined.

- Phase II: A1 and A2 independently applied the resulting reasons from Phase I to categorize all 669 answers. A1 & A2 took notes regarding the deficiency or ambiguity of the labeling for obsolete answers. During this phase no new labels (i.e., reasons) were introduced.

- Phase III: A1, A2 discussed the coding results that were obtained in Phase II to resolve any disagreements until a consensus was reached. The inter-rater agreement of this coding process has a Cohen's kappa of 0.76 (measured before starting Phase III), which indicates that the agreement level is substantial (Viera et al., 2005).

During our manual study process, we also labeled whether the obsolescence is a legacy or an invalid obsolescence.

Table 3.1: Reasons for obsolescence

Reason	Definition	Example
Third Party Library	An answer becomes obsolete due to third party libraries, Application Programming Interfaces (APIs), or frameworks becoming obsolete.	A comment points out that the way to delete a project in Google APIs Console has become obsolete[16].
Programming Language	Answer obsolescence is caused by obsolete features of the programming language and/or its standard APIs.	A comment points out that the -client option is ignored by a 64-bit capable JDK since Java 6[17].
Reference	References in an answer are obsolete.	A comment points out that the link to a whitepaper with detailed benchmarking for the Oracle TimesTen in-memory database is dead[18].
Tool	Tool information is obsolete, such as an old version.	A comment points out that a solution is out of date for Microsoft Kinect SDK version 1.0[19].
Mobile OS	An answer becomes obsolete due to an obsolete mobile platform.	A comment points out the event handling syntax for Mono for Android 4.2 is out of date[20].
Non-mobile OS	An answer becomes obsolete due to an obsolete non-mobile OS platform.	A comment points out that in order to work on macOS Sierra instead of macOS El Capitan, the new option is `-install-dir /usr/local/bin` [21].
Protocol	An answer is obsolete because a protocol is updated.	A comment points out that the internet text messages RFC 822 was replaced by RFC 2822[22].

Results: 31.7% of the studied answers (after removing false positives) became obsolete due to the evolution of their associated third party libraries. The number of occurrence and percentage of each obsolescence reason is shown in Fig. 3.8, as well as the proportion of legacy or invalid obsolescence for each obsolescence reason. In our qualitative study, we find that most answers became obsolete due to the evolution of their associated third party libraries. In addition, **30.9% of the studied answers became obsolete due to the evolution of their programming languages**. Stack Overflow covers a broad range of questions and answers across various programming languages

[16]https://stackoverflow.com/posts/comments/56423259/
[17]https://stackoverflow.com/posts/comments/59707599/
[18]https://stackoverflow.com/posts/comments/803108/
[19]https://stackoverflow.com/posts/comments/12009382/
[20]https://stackoverflow.com/posts/comments/14581496/
[21]https://stackoverflow.com/posts/comments/75888652/
[22]https://stackoverflow.com/posts/comments/14278476/

and third party libraries, and it is very common for programming languages/third party libraries to release new versions, thereby making the older versions possibly obsolete. For example, in a question of how to serialize and restore an unknown class in c#, an answer[23] suggested to use SoapFormatter instead of XmlSerializer. Another user posted a comment 3 minutes later stating that "this class is obsolete. Use BinaryFormatter instead", including the .NET Framework version number and a reference link. Based on this observation, we recommend that users provide a version number for their answers, then Stack Overflow can note the active versions when an answer was posted and note in the UI how many versions come after it.

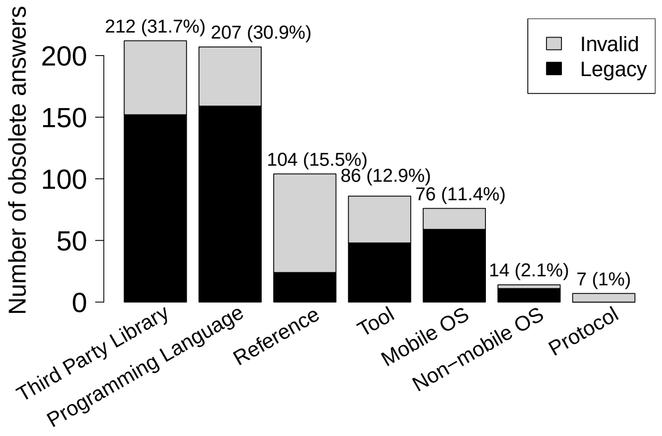


Figure 3.8: Number and percentage of each obsolescence reason based on our manual analysis. The figure also shows the proportion of legacy (black) and invalid (gray) obsolescence.

15.5% of the answers are obsolete due to obsolete references. 11.9% of the 5.5 million links (that are mentioned on Stack Overflow answers) are no longer available. Obsolete references include URL links, cited books, videos, and so on. Although it is convenient for a user to post an answer simply by referring to external URLs, it is common for references to become obsolete because the source of the reference may

[23]https://stackoverflow.com/questions/590722/

not be well maintained over time. This is especially a problem when users write an answer without providing too much concrete content, but instead simply offering URLs as the solution. In total, there are 5.5 million links in the 7.3 million answers on Stack Overflow. To better understand the obsolete URLs on Stack Overflow, we check all 5.5 million links to verify if they are still accessible (i.e., by returning 200 status code when requesting the URL). As of September 2018, we find that 11.9% of these links are no longer accessible.

12.9% of the studied obsolete answers are due to outdated tools, and 27.9% of these outdated tools are related to IDEs. To further understand what types of tools are more likely to be associated with obsolete answers, we manually study the related answer threads. Among these tools, 27.9% are related to IDEs, such as Visual Studio, Eclipse, Xcode, and Android Studio. For example, in an outdated answer for Xcode, the commenter not only pointed out the obsolescence, but also provided an updated answer[24]. One possible explanation is that IDEs are frequently updated in order to provide support for evolving programming languages and environments (e.g., mobile development).

Besides these obsolescence reasons, we also observe others, such as obsolete operating systems, and protocols. For example, a comment[25] in an answer pointed out that since Windows 7 cacls is deprecated for displaying and modifying access control lists (ACLs).

Obsolete answers should not simply be removed as a solution because they may still be applicable to users who are using legacy technologies/systems. We find that 63.8% of the studied obsolete answers belong to the legacy category. **However, we observe**

[24]`https://stackoverflow.com/posts/comments/16320934/`
[25]`https://stackoverflow.com/posts/comments/54010530/`

that the studied answers that are related to protocols are all invalid. This is reasonable since once a protocol becomes obsolete, it is most likely no longer used anymore. We get the complete list of RFCs[26] as of May 2018. This list contains the 8,286 RFCs, in which 1,188 RFCs are obsolete because of 1,112 newly added RFCs. We collected all answers (i.e., 21,591) containing "RFC" information from Stack Overflow, and we find that the RFCs in 10,793 answers became obsolete (i.e., were replaced by new RFCs). However, among such obsolete answers, only 611 answers were updated to reflect the new RFC versions. In other word, **only 5.7% of answers mentioning obsolete RFCs were updated to reflect the new RFC version.**

> The majority of answers are obsolete due to the evolution of their associated third party libraries, programming languages, and references. Therefore, users need to pay more attention to such answers when looking for answers on Stack Overflow.

3.3.4 RQ4: Who observes obsolete answers and what evidence do these observers provide?

Motivation: Uncovering obsolete knowledge on Stack Overflow is not trivial, especially if the user is not an expert in the specific knowledge domain. Therefore, it is essential to identify experts who might observe answer obsolescence and support their observations. In this RQ, we examine who identifies obsolete answers. Furthermore, we are interested in investigating how they support their obsolescence observation. By analyzing these aspects, we expect to get insights into how to assist users on Stack Overflow to identify obsolete answers.

[26]https://www.ietf.org/download/rfc-index.txt

Approach: To understand who observes the obsolescence of an answer, we first perform a quantitative study on all the studied answer threads. Based on the role of the user who observes the obsolescence of an answer thread, we categorize observers into one of the following 5 groups:

1. **Asker**: the user who posted the question;
2. **Answerer**: the user who posted the obsolete answer;
3. **Other answerer**: the user who posted another answer other than the obsolete one;
4. **Commenter**: the user who posted comments in the question thread;
5. **Outsider**: the user who never had any prior activities (including posting question, answer or comment) in that question thread.

We refer to an asker, answerer, other answerer(s), or commenter who are involved in the question thread (groups 1 – 4) as an *insider* (since they were involved earlier in the question thread).

To understand the type of evidence that users provide when observing the obsolescence of an answer, we performed a qualitative study. We used the studied answers from Section 3.3.1. We manually extracted and categorized the evidence of obsolescence from the sampled answers. We performed a lightweight open coding-like process (Seaman, 1999; Seaman et al., 2008) as mentioned in Section 3.3.3. We categorized the support evidence for obsolete answers into 8 types, as shown in Table 3.2. The inter-rater agreement of this coding process has a Cohen's kappa of 0.95, which indicates that the agreement level is high (Viera et al., 2005).

Table 3.2: Types of support evidence for an obsolescence observation.

Type	Definition
Provide updated info	The user provides updated information as an explanation why an answer is obsolete.
Provide version info	The user mentions the version number of either the obsolete answer (e.g., framework) or the updated information.
No support	No supportive material is given to prove the answer is obsolete. The user simply claims that something is obsolete.
Provide links	The user posts a link as a further reference to her/his obsolescence observation.
Highlight time	The user mentions the time when the answer worked.
Provide running errors	The user shows the running errors due to the obsolescence.
Refer to other answers	The user points to another answer on Stack Overflow to support why the current answer is obsolete.
Refer to this answer	The user points to this answer because it updated the obsolete content.

Results: The obsolescences of answers are more frequently observed by outsiders (38.2%), compared to askers (20.5%) and answerers (24.3%) The number and proportion of obsolete answers that were observed by each group of users (i.e., asker, answerer, other answerer, commenter, and outsider) are shown in Fig. 3.9. Only 24.3% of the obsolete answers were observed by answerers. 10.1% of the obsolete answers were observed by commenters. 6.9% of the obsolete answers were observed by other answerers in the same question thread. 20.5% of the obsolete answers were observed by askers. The lowest proportion among the insiders are other answerers. The rest of the obsolete answers (38.2%) are observed by users who have never participated in the discussion before observing that the answer is obsolete.

In summary, only 24.3% of the obsolete answers were observed by answerers. One possible reason is that some answerers are no longer active on Stack Overflow. Another possible reason is that even if the answerers are still active on Stack Overflow, they may not really want to maintain their answers after a long period of time. Even worse, they may not even be active in that domain anymore. For example, one user asked how to handle obsolete answers, and one commenter mentioned that "*Two years down the line I don't want to have to regularly rework my answers. I might not even be active in*

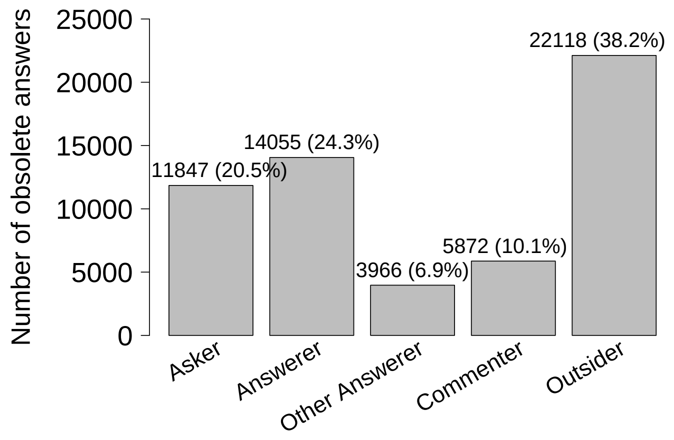


Figure 3.9: The number (as well as the percentage) of the obsolete answers that are observed by each type of user. A role of user is assigned using the following priority: asker > answerer > other answerer > commenter > other user. For example, if a user has multiple roles, such as an answerer and a commenter, we consider the user as an answerer.

that field anymore"[27]. Therefore, it's very important for Stack Overflow to encourage the whole community, not just the answerers to maintain answers by taking care of obsolete answers.

The majority (78.6%) of the obsolete observations are supported with evidence (e.g., updated information, a version information, or a reference). Fig. 3.10 shows the proportion of each type of supporting evidence for obsolescence observations. An obsolescence observation could have multiple types of support evidence. For example, a user can provide both version information and a link to the new version. We observe that in the majority of cases, users provide supporting evidence (e.g., updated information and a version information). In 42% of the cases, users provide updated information about the obsolete answers. For example, in a comment[28], the user not only pointed out that numpy is out of date, but also provided the code to check the

[27]https://meta.stackexchange.com/posts/comments/21537/
[28]https://stackoverflow.com/posts/comments/56525745/

numpy version in the code to install the latest version. Such cases are not rare; we observe that 44.8% of cases a solution (an updated answer) is provided in the comments. Furthermore, version numbers are also used by some users to support obsolescence observation. Once a version number is given, it is convenient for users to identify the obsolete knowledge. We find that 27.4% of obsolescence observations mentioned version numbers. For example, in an answer that uses AutoMapper (a convention-based object-to-object mapper and transformer for .NET), one comment[29] started with "*as of AutoMapper 4.2 Mapper.CreateMap() is now obsolete ...*". However, we find that 21.4% of obsolescence observations do not provide any supporting evidence. During our qualitative study, we find other types of support for obsolescence observations. For example, 7.6% of obsolescence observations are supported by highlighting time information (e.g., the validity period for the answer) related to the obsolescence.

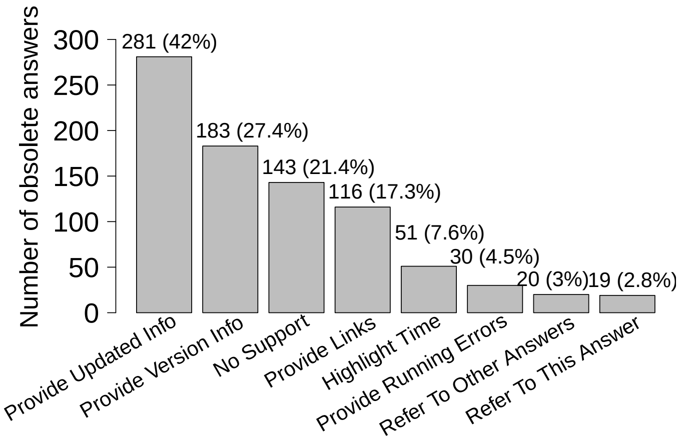


Figure 3.10: The proportion of each type of evidence that users provide when pointing out obsolescence.

Obsolescence observers tend to provide different evidence to support their observations. As shown in Fig. 3.11, askers are more likely to report runtime errors. One

[29] https://stackoverflow.com/posts/comments/58514542/

possible explanation is that askers are more likely to have a chance to run the code that is proposed in the answer and find out that it does not work due to runtime errors. Then, they report the error in the comment. In general, outsiders are the main evidence providers for pointing out obsolete answers.

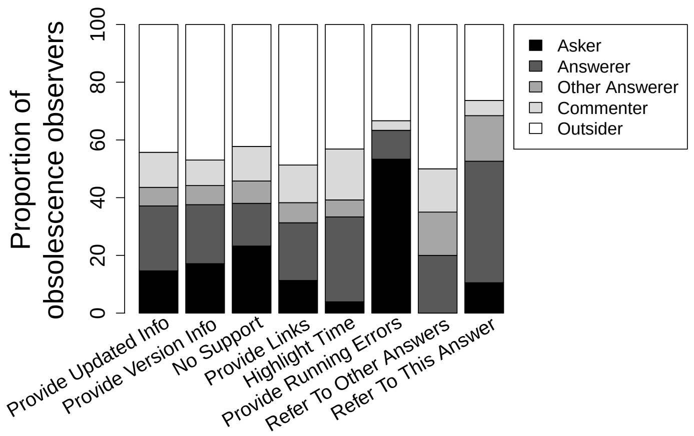


Figure 3.11: The proportion of each type of evidence that different observers provide when pointing out the obsolescence of an answer.

The majority of the obsolete answers were not observed by the original answerers. To help resolve obsolete answers, Stack Overflow should develop mechanisms to encourage the various members of the Stack Overflow community to maintain and flag obsolete answers. We also find that most (78.6%) obsolescence observations are supported by evidence.

3.4 Discussion

3.4.1 Actionable Suggestions for Stack Overflow

An automated tool could be built to identify existing obsolete answers on Stack Overflow, or help answerers identify obsolete answers in real-time during answer creation. We find that more than half of the obsolete answers were identified as obsolete within 24 hours of their initial posting, which indicates that users may not even realize that their posted answers are already obsolete. An automated tool could be developed to identify the possible obsolescence of an answer as it is being typed in. More specifically, we observed that there are many obsolescence reasons and the two major ones are related to third party libraries and programming languages. Future research could possibly leverage the evolution information of third party libraries and programming languages to detect the obsolescence of related answers. For example, a tool could analyze third party libraries to check their latest version, or the time of their latest update, and determine the valid API version for an API related answer so that version information is highlighted in appropriate answers. As an example, Tran and Cao (2013) automatically detected outdated information on Wikipedia by using pattern-based fact extraction from both Wikipedia and the web. A similar tool may be developed to scan existing answers and classify those that are obsolete with valid version information of a programming language, library, or framework where applicable.

An automated mechanism to detect obsolete references is needed. We scanned all links (i.e., 5.5 million) in Stack Overflow answers and observed that as of September 2018, 11.9% of the links are inaccessible. Hence, Stack Overflow could scan links to identify the availability of links. Similar to the dead link template and other inline cleanup tags (such as obsolete source) on Wikipedia[30], Stack Overflow could also include a "dead link" tag as well as the last retrieved time once an obsolete link is detected. As a result, users are made aware of obsolete links when reading the answer, and users who posted obsolete links could also be notified when their links are detected as obsolete. Additional actions are therefore encouraged, such as updating obsolete links or archiving snapshots of links as soon as they are created.

Our heuristic-based approach for identifying obsolete answers using comments has an accuracy of 75%. Future work could improve the accuracy of our approach using machine learning techniques (e.g., classification). Machine learning techniques could be applied to identify whether a comment indicates that an answer is obsolete based on the content of the comment and other features, such as the associated tags of the answer, and answer/comment score. Note that we characterize the false positives in the data collected by our heuristic-based approach (in Section 3.2), so future work could pay special attention to these corner cases in order to improve the accuracy of any automated approach. For example, a comment mentioning "function ABC was replaced by XYZ in year N" would be a strong indication of an obsolete answer. As a result, such comments could be highlighted to assist users in identifying obsolete answers.

Stack Overflow should develop mechanisms to encourage users (especially question thread insiders) to pay more attention to the obsolescence of answers (their own or others') and make efforts to maintain any obsolete answers. In RQ1, we find that only around 1 out of 4 users updated their answers when their answers were noted as obsolete. Moreover, it took users about 4 months on average (i.e., 118 days) to update their answers or add new updated answers. In other words, users do not pay much

[30]`https://en.wikipedia.org/wiki/Template:Dead_link`

attention to the obsolescence of their answers and do not frequently maintain their answers. For example, a comment of an obsolete answer mentioned that the answer was obsolete and asked the answerer to update it. The answerer replied in comment "*Feel free to update the answer yourself, if you like. I honestly would, but I don't have the time*"[31]. The gamification system (e.g., badges and/or reputation scores) should be adjusted to encourage users to identify and update obsolete answers. For example, Stack Overflow could reward badges or reputation scores to users who identify or maintain obsolete answers.

3.4.2 Actionable Suggestions for Users

Answerers are encouraged to include relevant information about the valid version or the time of their knowledge when creating answers. In RQ4, we observe that 78.6% of the obsolescence observations included supportive evidence, such as when the answer became obsolete (e.g., time and version information). Such information is very helpful for answer seekers to verify whether the knowledge in the answers is still valid or not (especially for their context).

Answer seekers are encouraged to carefully go through the comments that are associated with answers in case these answers become obsolete, especially for answers that are related to web and mobile development, such as node.js, ajax, android, and objective-c. In RQ2, we observe that answers related to some specific tags are more likely to become obsolete, such as tags that are related to mobile development (e.g., Android and iOS) and web development (e.g., node.js and ajax). Therefore, answer seekers are encouraged to pay more attention when reading through answers

[31]https://stackoverflow.com/posts/comments/61093395/

that are related to such tags. One actionable way is to go through the comments under accepted answers or not-accepted (yet highly voted) answers, which may have useful information to indicate whether the answer has became obsolete or not. Even more, in 44.8% of the observed obsolete answers, a comment provided a solution to update the answer. In addition, we strongly advise users to carefully read all highly ranked comments when reading an answer, since we observe that 73.5% of the comments that indicate the obsolescence of an answer are the top 1 ranked comment for the obsolete answers.

3.4.3 Feedback from Stack Overflow

To understand whether our research uncovered a relevant problem on Stack Overflow and whether our findings are useful for Stack Overflow, we shared our findings with members of the Stack Overflow team. They concurred with our findings and mentioned that it is interesting to see a breakdown of this problem (*"obsolete info is an ongoing issue on the site, so it's interesting to see this breakdown of how that issue manifests itself"*). They asked us to examine whether the answer obsolescence issue would vary based on different community sizes. We observed that answer obsolescence is a widespread issue that is not influenced by the size of the tag (the details of this analysis is included in RQ2). Moreover, they were specifically interested in our analysis about the version information of platforms and programming languages. Based on our findings, the Stack Overflow team was also interested in investigating approaches to generate tags that indicate the valid version for a framework, an API, or a programming language for an answer. Future research efforts should continue working with the Stack Overflow team to solve/alleviate the obsolete problem.

3.5 Threats to Validity

3.5.1 External Validity

Threats to external validity are related to the generalizability of our findings. In this study, we focus on Stack Overflow, which is one of the most popular and largest Q&A websites for developers; hence, our results may not generalize to other Q&A websites. To alleviate this threat, more Q&A websites should be studied in the future. We needed to conduct several qualitative analysis in our RQs; however, it is impossible to manually study all answers. To minimize the bias when conducting our qualitative analysis, we took statistically representative random samples of all relevant revisions, in order to ensure a 99% confidence level and 5% confidence interval for our observations (Boslaugh and Watters, 2008).

3.5.2 Internal Validity

Threats to internal validity are related to experimenter bias and errors. Our study involved qualitative analysis in RQs. To reduce the bias, each answer was labeled by two researchers (with me being one of them) and discrepancies were discussed until a consensus was reached. We also showed that the level of inter-rater agreement of the qualitative studies is high (i.e., the values of Cohen's kappa ranged from 0.76 to 0.96). Another threat to our study is related to our data collection process. Due to the large number of answers and lack of mechanism on Stack Overflow to identify obsolete answers, we used a heuristic-based approach to uncover obsolete answers. The accuracy of our heuristic-based approach is 75% based on our manual verification, which implies that there may be noise in our quantitative study. Hence we followed all presented quantitative studies with qualitative studies of randomly representative samples. Future study should develop a more accurate method to identify the obsolescence of an answer on Stack Overflow. In RQ1, a quantitative analysis shows an upper bound for both the proportion of obsolete answers that were updated and the proportion of new answers that were created after such obsolete answers. The values do not show how many answers are actually updated or created due to answer obsolescence, but only indicate an upper bound of such cases. Other reasons (e.g., provide alternative solutions) could cause users to update and/or add answers. This represents a possible threat to the internal validity of this particular analysis. To tackle this concern, we performed a qualitative study in RQ1 to manually analyze how many answers are updated or added due to answer obsolescence. An additional threat lies in the evaluation of our heuristic approach to find obsolete answers. Two researchers (with me being one of them) evaluated this heuristic approach. We calculated Cohen's kappa to measure the inter-rater agreement between both researchers and the agreement is high (i.e., 0.76). Last but not least, the status of obsolete answers on Stack Overflow can change, as Luca et al. (2016) observed that information on Stack Overflow can update over time. Although the associated findings of answer obsolescence may change, our study highlights the obsolete problem that can deteriorate, as more and more answers can become obsolete over time.

3.6 Chapter Summary

In this chapter, we present an empirical study of the obsolete knowledge on Stack Overflow, as an inevitable step towards understanding the evolution of knowledge on Stack Overflow. We find that: 1) Answers in certain tags (e.g., node.js, ajax, android, and objective-c) are more likely to become obsolete mainly due to the evolution of their associated third party libraries and programming languages. 2) Most of the studied obsolete answers are pointed out by non-answerers and are supported by evidence. 3) When an obsolete answer is identified, only a small proportion of such answers are updated afterwards. More importantly, more than half of the obsolete answers were probably already obsolete when they were posted. Based on our findings, we offer the following suggestions: 1) Stack Overflow should develop mechanisms (i.e., incentive systems) to encourage the whole community to identify and/or maintain obsolete answers. 2) Answerers are encouraged to include information of the valid version or time of the knowledge when creating answers. 3) Answer seekers are encouraged to go through all the information in an answer thread carefully in case someone had pointed out the obsolescence of an answer, especially for the answers that are related to web and mobile development.

There are two possible directions for future work. First, we encourage future studies to develop advanced approaches to detect obsolete knowledge on Stack Overflow. For example, machine learning techniques can be employed to detect the comments that indicate obsolescence based on the semantic meaning of the text instead of keywords matching. Second, we encourage future studies to develop approaches to extract useful information from the comments so that answer seekers could easily find the useful information from the list of long and unorganized comments.

CHAPTER 4

The informativeness of comments under answers

Stack Overflow is one of the most active communities for developers to share their programming knowledge. Answers posted on Stack Overflow help developers solve issues during software development. In addition to posting answers, users can also post comments to further discuss their associated answers. As of Aug 2017, there are 32.3 million comments that are associated with answers, forming a large collection of crowdsourced repository of knowledge on top of the commonly-studied Stack Overflow answers. In this chapter, we wish to understand how the commenting activities contribute to the crowdsourced knowledge. We investigate what users discuss in comments, and analyze the characteristics of the commenting dynamics, (i.e., the timing of commenting activities and the roles of commenters). We find that: 1) the majority of comments are informative and thus can enhance their associated answers from a diverse range of perspectives. However, some comments contain content that is discouraged by Stack Overflow. 2) The majority of commenting activities occur after the acceptance of an answer. More than half of the comments are fast responses occurring within one day of the creation of an answer, while later comments tend to be more informative. Most comments are rarely integrated back into their associated answers, even though such comments are informative. 3) Insiders (i.e., users who posted questions/answers before posting a comment in a question thread) post the majority of comments within one month, and outsiders (i.e., users who never posted any question/answer before posting a comment) post the majority of comments after one month. Inexperienced users tend to raise limitations and concerns while

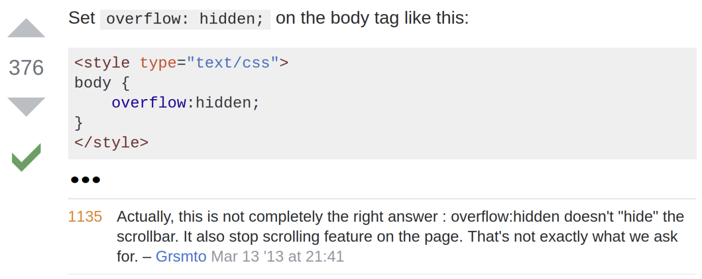
experienced users tend to enhance the answer through commenting. Our study provides insights into the commenting activities in terms of their content, timing, and the individuals who perform the commenting. For the purpose of long-term knowledge maintenance and effective information retrieval for developers, we also provide actionable suggestions to encourage Stack Overflow users/engineers/moderators to leverage our insights for enhancing the current Stack Overflow commenting system for improving the maintenance and organization of the crowd-sourced knowledge.

An earlier version of this chapter is published in the IEEE Transactions on Software Engineering Journal (TSE) (Zhang et al., 2019a).

4.1 Introduction

STACK Overflow provides a knowledge sharing platform to help developers share knowledge and seek answers to their problems. It has accumulated a large amount of programming knowledge in the form of questions and answers. As of Feb. 2020, there are 19 million questions on Stack Overflow. The question answering activities cover various software development domains, and have generated 29 million answers.

Such crowdsourced knowledge is not only generated by the question answering process, but it is also produced by commenting activities. Comments are appended under their associated questions and answers to facilitate further discussion[1]. For example, in Fig. 4.1, a comment is posted to provide additional information (i.e., the limitation of *overflow:hidden*) to its associated answer.


Set `overflow: hidden;` on the body tag like this:

376

```
<style type="text/css">
body {
    overflow:hidden;
}
</style>
```

•••

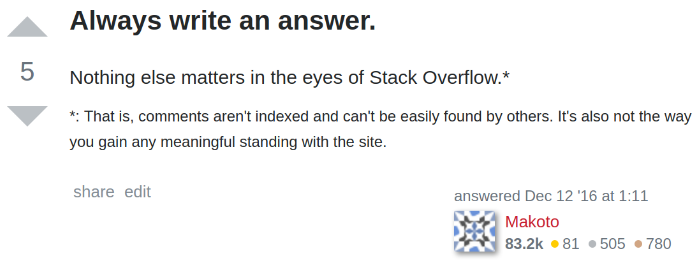
1135 Actually, this is not completely the right answer : overflow:hidden doesn't "hide" the scrollbar. It also stop scrolling feature on the page. That's not exactly what we ask for. – Grsmto Mar 13 '13 at 21:41

Figure 4.1: An example of a comment that is associated with an answer. This comment points out a flaw in the accepted answer and has gained a higher score (i.e., 1,135) than its associated answer (i.e., 376).

Although commenting is a popular online communication channel, commenting activities on Stack Overflow have never been studied in depth before. Note that posting of comments on Stack Overflow does not generate any reputation point for a

[1]Note that we refer to comments that are associated with answers on Stack Overflow as comments, if not specified otherwise in the rest of the thesis. Comments that are associated with questions are not studied in this thesis.

user. From a user's perspective, comments can be easily missed in contrast to answers. Hence, Stack Overflow has suggestions about what should (e.g., request clarification and leave constructive criticism) and should not (e.g., answer a question and suggest corrections) be posted in comments. On the other hand, users may not know Stack Overflow's commenting guideline. For example, on Stack Overflow Meta[2], some users hold an opinion[3] that knowledge sharing should only be conducted in the form of answers and not comments (in Fig. 4.2). Others consider comments as *temporary "Post-It" notes* to improve their associated answers[4], as shown in Fig. 4.3. A prior study shows that certain comments contain informative content. For instance, comments can point out the obsolescence of their associated answers (as we examined in Chapter 3). However, no systematic study has ever been done on the comments of Stack Overflow to better understand how comments are used. For instance, whether users use comments by following Stack Overflow commenting guidelines?


Always write an answer.

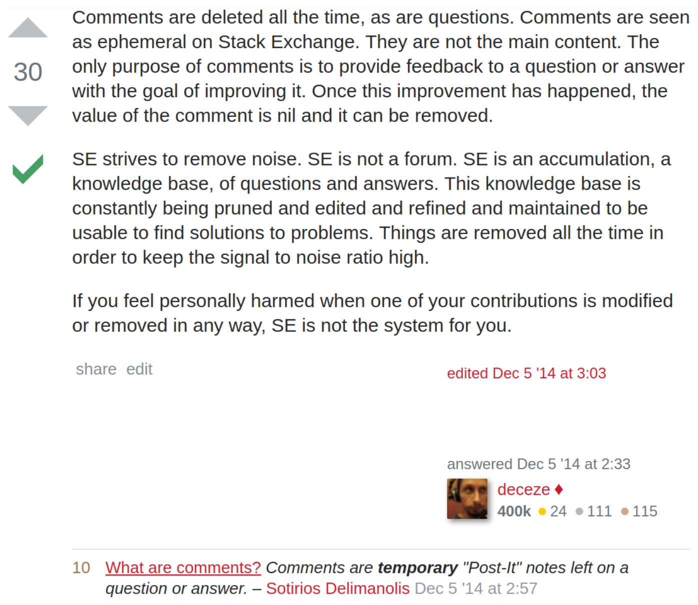
Nothing else matters in the eyes of Stack Overflow.*

*: That is, comments aren't indexed and can't be easily found by others. It's also not the way you gain any meaningful standing with the site.

Figure 4.2: A discussion on Stack Overflow Meta shows an opinion that knowledge sharing should exclusively occur in answers.

Therefore, in this chapter, we investigate the comments (i.e., 32.3 million) that are associated with Stack Overflow answers (i.e., 22.7 million) to gain a better understanding of how the commenting activities contribute to the generation and maintenance

[2]Stack Overflow META is the part of the site where users discuss the workings and policies of Stack Overflow.
[3]`https://meta.stackoverflow.com/a/339395/`
[4]`https://meta.stackoverflow.com/a/278517/`


30

Comments are deleted all the time, as are questions. Comments are seen as ephemeral on Stack Exchange. They are not the main content. The only purpose of comments is to provide feedback to a question or answer with the goal of improving it. Once this improvement has happened, the value of the comment is nil and it can be removed.

SE strives to remove noise. SE is not a forum. SE is an accumulation, a knowledge base, of questions and answers. This knowledge base is constantly being pruned and edited and refined and maintained to be usable to find solutions to problems. Things are removed all the time in order to keep the signal to noise ratio high.

If you feel personally harmed when one of your contributions is modified or removed in any way, SE is not the system for you.

share edit

edited Dec 5 '14 at 3:03

answered Dec 5 '14 at 2:33
deceze ♦
400k ● 24 ● 111 ● 115

10 What are comments? *Comments are **temporary** "Post-It" notes left on a question or answer.* – Sotirios Delimanolis Dec 5 '14 at 2:57

Figure 4.3: A discussion on Stack Overflow Meta shows an opinion that comments are temporary.

of crowdsourced knowledge. We wish to provide insights to Stack Overflow users so that they can more effectively identify relevant information from comments. We also wish to provide actionable suggestions to Stack Overflow designers and engineers so that the Stack Overflow commenting system can be used more effectively to enhance the knowledge sharing process.

More specifically, we first conduct a preliminary study on how active users are in posting comments. We find that a large collection of comments exist, and that the number of posted comments continues to exceed the number of posted answers every year since 2009. 23% (i.e., 2.6 million) of the answers with comments have a commenting-thread (i.e., all the comments that are associated with an answer) that is longer than the actual answer. Then, we answer the following three research questions:

- **RQ1: What do users discuss in comments?**

  Most comments (i.e., 75.6%) provide useful information, such as pointing out weaknesses and providing references to their associated answer. However, some of the informative comments (e.g., suggesting a correction and answering a question) do not follow the commenting guidelines that are outlined by Stack Overflow.

- **RQ2: When do users post comments?**

  The acceptance of an answer is not the end of commenting activities; instead, the majority of commenting activities occur after the acceptance of an answer. Generally, more than half of the comments are fast responses occurring within one day of the answer creation. Comments that point out the advantage and weakness of answers tend to be posted later than other informative comments, and later comments tend to be more informative. However, the knowledge in comments is rarely integrated back into answers.

- **RQ3: What types of users participate in the commenting activities?**

  Users are highly involved in commenting. Askers mainly comment to express praise, inquiry, and point out weakness, while answerers mainly comment to highlight the advantage of an answer, provide improvement and additional information. Askers and answerers are more likely to post comments within one month, while other users are more likely to post comments after one month. Among informative comments, inexperienced users tend to raise limitations and concerns while experienced users tend to enhance answers by commenting. Among uninformative comments, inexperienced users tend to praise answers while experienced users tend to post irrelevant information.

Based on our findings, we encourage users to read comments carefully since the majority of comments provide a diverse variety of information that enhances their associated answers. Thus, comments are an important resource of knowledge. For example, a comment can provide helpful clarification to its associated answer, or point out flaws. Especially, we highlight that later comments tend to be more informative than comments that are posted soon after the posting of their associated answer. However, the informative content in comments is rarely integrated back into their associated answers. Thus, Stack Overflow should consider adopting a mechanism to reward reputation points or certain badges to encourage the maintenance and integration of commenting knowledge. We also suggest that Stack Overflow designers should improve the current commenting system because users post comments in unrecommended manners (i.e., not following Stack Overflow's commenting guideline), such as suggesting corrections, answering questions, and praising answers.

Chapter Organization: The rest of the chapter is organized as follows. Section 4.2 introduces the background of the commenting system on Stack Overflow and describes our studied dataset. Section 4.3 explores the commenting activities on Stack Overflow as a preliminary study. Section 4.4 details the results from our case study. Section 4.5 discusses our findings and their implications. Section 4.6 discusses the potential threats to the validity of our findings. Finally, Section 4.7 concludes our study.

4.2 Case Study Setup

This section describes the subject of study and the process that we follow to collect the data for our case study.

4.2.1 Subject of Study

Stack Overflow is an online platform for question answering in the domain of software programming. Users can post their questions with descriptive text about their problems. Once a question is posted by a user (i.e., an asker), others (i.e., answerers) can post answers to this question. When an asker is satisfied with any solution, she/he can select the answer as the accepted answer among all the posted answers.

A reputation system is implemented on Stack Overflow to measure and encourage the contributions of users to the community. There are various ways for users to gain reputation points. For instance, an answer can be upvoted by other users; as a result, the answerer gains 10 reputation points as reward. On the other hand, commenting activities do not lead to any gain of reputation points, since they "*are all secondary activities*" according to Stack Overflow[5].

Users (i.e., commenters) can post additional text/code under either questions or answers. As stated by Stack Overflow (2019), "*comments are temporary 'Post-It' notes left on a question or answer ... but do not generate reputation*". We consider a question with all its answers and their associated comments as a *question thread*. Similarly, we consider an answer with its associated comments as an *answer thread*, and all the comments that are associated with an answer as a *commenting-thread*.

Comments can be posted by three types of users (Stack Overflow, 2019):

- The owner of an answer can post comments under the answer;
- The owner of a question can post comments under the question and any of its answers;

[5]`https://stackoverflow.com/help/whats-reputation`

- Users with at least 50 reputation points can post comments everywhere.

Similar to answers, comments can be also upvoted; however, upvoting a comment does not generate any reputation points. Officially, Stack Overflow recommends users to post comments under the following circumstances: ***request clarification***, ***leave constructive criticism***, and ***add relevant information***, and recommends users *not to* post comments under circumstances such as: ***suggest corrections***, ***answer a question***, and ***compliment*** (Stack Overflow, 2019). In the following RQs, we investigate the content of comments to examine if users follow Stack Overflow commenting guidelines.

4.2.2 Data Collection

We downloaded all the comments from the Stack Overflow data dump on archive.org[6] that was released in Aug 2017. In this dataset, there are 60.1 million comments from either questions or answers. Since we focus on the comments that are associated with answers, we end up with 32.2 million comments in this study, which are associated with 22.7 million answers and 1.9 million users. In general, comments are short. The median length of a comment is 115 characters.

4.3 Preliminary Study

Stack Overflow is the largest Q&A website tailored for software developers, with 7.6 million registered users. As of Aug 2017, 14.5 million questions have been asked across more than 5,000 tags (i.e., user-provided topics of a question). Developers leverage

[6]https://archive.org/details/stackexchange

answers posted on Stack Overflow to tackle their coding issues or learn programming knowledge. Similar to other online platforms (e.g., Reddit, Hacker News, and Quora), users can also post comments to answers. In order to get an understanding of the commenting activities on Stack Overflow, we first conduct a preliminary study on such comments. More specifically, we wish to find the popularity of commenting activities, considering that posting comments does not result in extra reputation points.

We compare the number of comments and answers posted on a yearly basis. We also characterize the amount of information in comments that are associated with answers. Since each comment is associated with an answer, we measure the number of characters in all the comments that are associated with an answer (i.e., a commenting-thread) and compare it with that of its associated answer.

Stack Overflow has a large collection of comments. More comments were posted than answers yearly since 2009. As of August 2017, among 11.4 million answers, 32.3 million comments are posted. The number of comments is greater than the number of answers on a yearly basis since 2009, as shown in Fig. 4.4. The ratio between the number of comments and answers keeps increasing until 2013, and remains stable afterwards[7]. The ratio between the number of comments and answers is around 1.5:1 since 2013. In other words, users are actively commenting on answers.

23% (i.e., 2.6 million) of the answers with comments have a commenting-thread that is longer than the actual answer (in terms of characters). In such answer threads, users may require more time and effort to read the comments than the answer, not only due to the longer text, but also due to the free style of comments and the way comments

[7]Note that we used the data dump that is published in Aug 2017; therefore, the numbers of both answers and comments in 2017 do not cover the entire year.

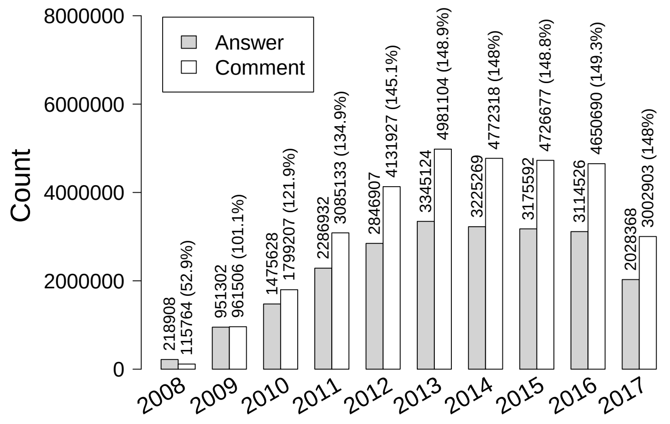


Figure 4.4: The number of answers and comments created on a yearly basis. The percentage value shown in the bracket is the ratio of the number of comments to the number of answers on a yearly basis.

are organized and presented. A commenting-thread can be massive and lack organization, thus leading to information overload and hindering information retrieval. In an example[8], the answer has 3,175 characters, while the answer has 28 comments which have 5,460 characters in total. It is difficult to understand the conversations in the commenting-thread from the default Stack Overflow view of comments. Additionally, 23 comments are hidden from this view. Thus, it is ineffective to share and retrieve information in a commenting-thread, even when a comment can enhance its associated answer. Only from the view that displays all the comments, then it is more clear what is the context of an individual comment, and whom a commenter mentions in his/her conversation. The unorganized nature of comments increases the difficulty to read and understand conversations. Due to the above-mentioned obstacles, comprehending all the information requires extra time and effort. It can be intimidating that some users may not bother to read comments at all. This polarizing view of comments has

[8]https://stackoverflow.com/a/47990/

been investigated by Reich (2011). In his study, one interviewee said that "*Most people don't want to comment. And actually, most people don't want to read other people's comments*". Therefore, we conduct an empirical study to gain a better understanding of what users discuss in comments, when users post comments, and who participate in such commenting activities.

Based on the above-mentioned preliminary results, we find that users actively participate in commenting activities, and sometimes comments that are associated with an answer can be longer than their associated answer – making it time-consuming to read comments. Therefore, in this chapter, we wish to explore this large collection of comments. In the following RQs, we study how comments provide useful information to their associated answers, and understand the characteristics of commenting. We hope to provide insights on how to leverage these comments as well as to suggest mechanisms to more effectively organize comments for easy information retrieval and knowledge management.

Stack Overflow has a large collection of comments, whose number is even larger than answers. The amount of information in comments cannot be neglected, with 23% of the answers having a commenting-thread that is longer than their actual answer.

4.4 Case Study Results

In this section, we present the results of our research questions. For each research question, we present the motivation of the research question, the approach to address the research question, and our experimental results for the research question.

4.4.1 RQ1: What do users discuss in comments?

Motivation: Commenting activities on Stack Overflow open up an alternative channel for users to participate in the knowledge crowdsourcing process. As shown in Section 4.3, in 23% of answers with comments, the commenting-thread is even longer than the answer itself. Some comments significantly add value to their associated answers. For example, in a comment[9] shown in Fig. 4.1, the commenter pointed out that the accepted answer is not completely correct. Although the answer has been accepted by the asker and upvoted by the community to reach a score of 376, this comment has been upvoted even more and has a higher score (i.e., 1,135) than the associated answer. On the other hand, users can post comments in a relatively free style, and a commenting-thread can even appear unorganized.

Such a large collection of commentary text that is associated with answers is important for users but not well understood nor studied carefully. It is unclear what do users actually discuss in comments and whether or not Stack Overflow commenting guidelines (see Section 4.2) are followed by users. Therefore, in this RQ, we investigate what users actually discuss in comments. Moreover, we categorize the types of discussions in comments (i.e., the comment types), and investigate the advantages and disadvantages of different comment types with regard to the official guidelines from Stack Overflow. A better understanding of the comment types can provide Stack Overflow engineers with firsthand insights into how commenting as a communication channel is used in practice. The identified comment types can also be leveraged to better organize comments and improve the maintenance of crowdsourced knowledge so that answer seekers can effectively extract relevant information.

[9] https://www.stackoverflow.com/posts/comments/21766075/

Approach: We study what do users discuss in comments through qualitative analysis. First, we randomly select a statistically representative sample of 3,000 comments from the 32.2 million comments, providing us with a confidence level of 99% and a confidence interval of 2.4%. We manually label the types of discussions in each comment at the sentence level with a lightweight open coding-like process. For example, in a comment[10], the user says that "*This one works and should be answer to this question. Although dot, coma and other values are displayed to user, the user can not insert them. So only input you receive from this is numeric.*". This comment is assigned the type *praise* and *clarification*. This lightweight open coding-like process involves 3 phases and is performed by two researchers (i.e., A1-A2, with me being one of them) as follows:

- **Phase I:** A1 derives a draft list of comment types based on 50 randomly sampled comments. Then, A1 and A2 use the draft list to label the sampled comments collaboratively. During this phase the comment types are revised and refined.

- **Phase II:** A1 and A2 independently apply the resulting types from Phase I to label the rest of the 3,000 comments. A1 and A2 take notes regarding the deficiency or ambiguity of the labeling for the comments. Note that new labels (i.e., the comment types) are introduced during this phase if A1 or A2 observes more comment types. At the end of this phase, we obtain 7 types of comments that are further divided into 17 subtypes (see Table 4.1).

- **Phase III:** A1 and A2 discuss the coding results that are obtained in Phase II to resolve any disagreements until a consensus is reached. No new types and subtypes are added during this discussion. The inter-rater agreement of this coding

[10]https://www.stackoverflow.com/posts/comments/12535822/

process at the subtype and type level has a Cohen's kappa of 0.86 and 0.90 (measured at the start of Phase III) respectively. These kappa levels indicate that the agreement level is substantial.

Table 4.1: The definition of types of comments with their subtypes and the proportion of each type.

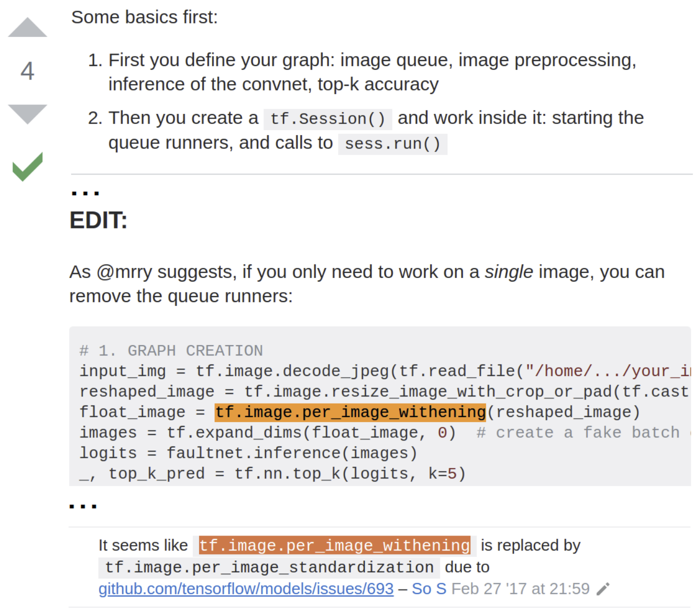
Type	Count	Subtype	Count	Description
Praise	521 (17.4%)	Praise	521 (17.4%)	Praises the answer
Advantage	134 (4.5%)	Support	43 (1.4%)	Gives reasons to support the answer
		Highlight	50 (1.7%)	Highlights the working circumstances or use case of the answer
		Performance	43 (1.4%)	Discusses the performance of the answer
Improvement	176 (5.9%)	Correction	152 (5.1%)	Provides correction to the answer
		Extension	24 (0.8%)	Extends the answer to other cases
Weakness	518 (17.3%)	Flaw	238 (7.9%)	Points out flaws or limitations
		Error	139 (4.6%)	Points out errors in the code
		Obsolete	31 (1.0%)	Points out obsoleteness
		Disagree	112 (3.7%)	Disagrees with the answer
Inquiry	591 (19.7%)	Question	446 (14.9%)	Asks clarification questions
		Request	147 (4.9%)	Requests information
Addition	1,130 (37.7%)	Solution	293 (9.8%)	Provides alternative solutions to the answer
		Example	66 (2.2%)	Adds a concrete example
		Clarification	682 (22.7%)	Adds a clarification
		Reference	163 (5.4%)	Adds a reference
Irrelevant	441 (14.7%)	Irrelevant	441 (14.7%)	Discusses irrelevant topics to the answer

We analyze each comment type and present concrete examples. Furthermore, to evaluate how the actual commenting activities are aligned with the official commenting guidelines from Stack Overflow, we compare the recommended commenting scenarios with the actual commenting scenarios, and examine the reasons for agreements and disagreements.

Results: 75.6% of the tagged comments are informative. Table 4.1 shows the identified comment types with their subtypes. Users often post informative comments, i.e., comments of type *advantage*, *improvement*, *weakness*, *inquiry*, and *addition*, to enhance their associated answers. Users also post uninformative comments, i.e., comments of type *praise* and *irrelevant*. **Furthermore, diverse subtypes of comments exist for each informative comment type.** More specifically, 37.7% of the studied comments belong to type *addition*. These comments add value to their associated answers by providing new content, i.e., an alternative solution, a concrete example, clarification, or a reference link. For example, in Fig. 4.5, the comment[11] points out an update in TensorFlow and provides a reference link. 19.7% of the comments belong to the type *inquiry*. These comments pose additional questions that are related to their associated answers, or request extra information for better understanding their associated answer, such as asking the answerer where "JsonConvert" is originated in the code snippet of the answer[12]. Comments of type *inquiry* aim to motivate the answerers to disclose more details, it helps answers to become clearer, and thus, are more likely to be used by other users. 17.3% of the comments are of type *weakness*. In these comments, a weakness in an answer (i.e., flaws, coding errors, obsolescence, and disagreements) is noted. For example, in a question about issues converting a javascript object to a query string, a comment[13] points out that an answer "... *only do convert plain js object to query string. If you need to resolve for nested objects go with some recursive strategy*". Last but not least, 5.9% and 4.5% of the comments improve existing answers, and comment on the advantage of an existing answer, respectively. The uninformative comments either praise an answer (i.e., 17.4%, such as "*Thank you. It worked for me :)*"[14]) or discuss irrelevant topics (i.e., 14.7%, such as "*If it correct please vote up*

[11] https://www.stackoverflow.com/posts/comments/72132287/
[12] https://www.stackoverflow.com/posts/comments/65851467/
[13] https://www.stackoverflow.com/posts/comments/72503806/
[14] https://www.stackoverflow.com/posts/comments/74500847/

Thanks :)"[15]). They are not informative because they do not directly add value to their associated answers.



4

Some basics first:

1. First you define your graph: image queue, image preprocessing, inference of the convnet, top-k accuracy
2. Then you create a `tf.Session()` and work inside it: starting the queue runners, and calls to `sess.run()`

...

EDIT:

As @mrry suggests, if you only need to work on a *single* image, you can remove the queue runners:

```
# 1. GRAPH CREATION
input_img = tf.image.decode_jpeg(tf.read_file("/home/.../your_i
reshaped_image = tf.image.resize_image_with_crop_or_pad(tf.cast
float_image = tf.image.per_image_withening(reshaped_image)
images = tf.expand_dims(float_image, 0)  # create a fake batch
logits = faultnet.inference(images)
_, top_k_pred = tf.nn.top_k(logits, k=5)
```

...

It seems like `tf.image.per_image_withening` is replaced by `tf.image.per_image_standardization` due to github.com/tensorflow/models/issues/693 – So S Feb 27 '17 at 21:59

Figure 4.5: An example of a comment that points out a TensorFlow update with a reference link.

The majority of informative comments (i.e., 67.4% of the tagged comments) follow the Stack Overflow commenting guidelines. Stack Overflow recommends users to post comments when they want to ***request clarifications*** from the author of posts. We observe that comments of type *inquiry* match this guideline. Comments of type *weakness* point out flaws, coding errors, obsolescence, or disagreements, thus also follow the guideline to ***leave constructive criticism***. Comments of type *addition* are encouraged by Stack Overflow as well because they ***add relevant information*** to an answer.

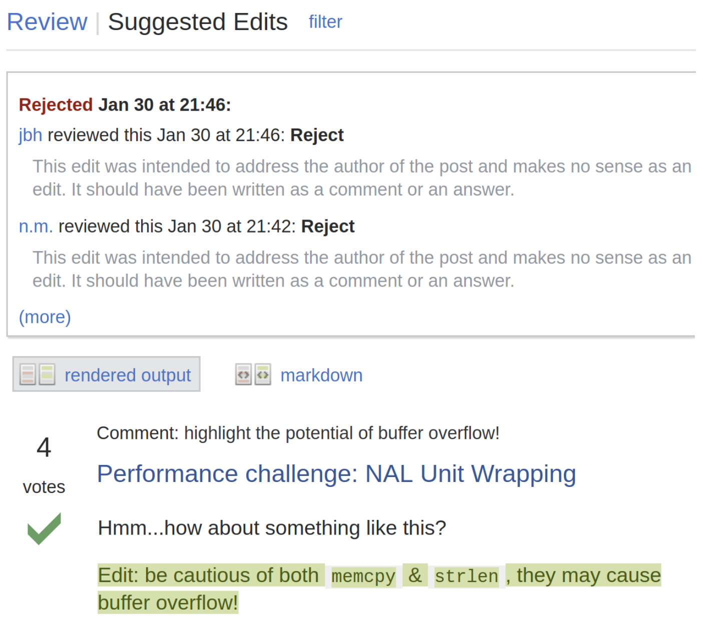
Comments that *suggest corrections*, *answer a question*, or are *compliment* exist

[15]https://www.stackoverflow.com/posts/comments/66524665/

(i.e., 31.3% of the tagged comments), although they are discouraged by Stack Overflow. For example, Stack Overflow does not recommend comments that ***suggest corrections***; however, comments of type *improvement* point out and take actions to the weakness in their associated answers, such as making corrections, or providing extensions or solutions, to fix an answer's weakness. Nevertheless, such comments are not recommended by Stack Overflow. Instead, users are recommended to suggest or make an edit to an existing answer. As shown in Fig. 4.12, the median reputation points for all comment types are below 2,000 (Stack Overflow only allows users with more than 2,000 reputation points to edit an answer directly). Note that when a user does not have enough reputation points (i.e., 2,000) to directly edit the answer, he/she can suggest an edit. Under this circumstance, the suggested edit is placed in a review queue[16]. However, it is unknown whether or not the suggested edit will be accepted, therefore the commenter faces uncertainty if he/she attempts to enhance an answer by editing directly. An example of a suggested edit[17] is shown in Fig. 4.6. The edit was rejected by two reviewers because: "*This edit was intended to address the author of the post and makes no sense as an edit. It should have been written as a comment or an answer*". The reviewers suggested the editor to write a comment or an answer to correct the answer instead of editing the answer. However, Stack Overflow does not recommend users to post comments to "*suggest corrections that don't fundamentally change the meaning of the post; instead, make or suggest an edit*". Additionally, it is unnecessary to create a new answer that simply corrects an existing answer. As a result, a contradictory situation occurs for such users with lower than 2,000 reputation points. Namely, ***edits that improve an answer can be rejected and suggested to become a comment,***

[16]https://stackoverflow.com/help/editing
[17]https://stackoverflow.com/review/suggested-edits/22075882

although, simultaneously, Stack Overflow does not recommend the posting of comments to suggest corrections – thus, users who wish to correct an answer are suggested not to edit the answer nor to post a comment, but to remain silent! As a matter of fact, posting a comment has a lower barrier, i.e., anyone with at least 50 reputation points can post comments, while any user with less than 2,000 reputation points has to wait for approval for editing an answer directly. Therefore, even while being discouraged to maintain an answer in the above scenario, users who still attempt to maintain the answer are likely to still post comments to suggest corrections. Even if a suggested edit is accepted, or the user has at least 2,000 reputation points, the answerer may not prefer others to change the post and might simply rollback the edit (Wang et al., 2018a). Overall, the above-mentioned obstacles create a cumbersome situation for users who wish to suggest corrections to an answer.


Review | Suggested Edits filter

Rejected Jan 30 at 21:46:
jbh reviewed this Jan 30 at 21:46: **Reject**
This edit was intended to address the author of the post and makes no sense as an edit. It should have been written as a comment or an answer.
n.m. reviewed this Jan 30 at 21:42: **Reject**
This edit was intended to address the author of the post and makes no sense as an edit. It should have been written as a comment or an answer.
(more)

rendered output markdown

4 votes
Comment: highlight the potential of buffer overflow!
Performance challenge: NAL Unit Wrapping
Hmm...how about something like this?
Edit: be cautious of both `memcpy` & `strlen`, they may cause buffer overflow!

Figure 4.6: A suggest edit with rejecting reasons.

Such conflict also applies to the Stack Overflow commenting guideline that users should not post comments to ***answer a question***. Even though it is not recommended to answer a question in a comment, we still observe cases of posting an answer in a comment instead of editing an answer or creating a new answer. Another type of comments that is not recommended by Stack Overflow is comments of type *praise*, although users still praise in comments. In conclusion, we suggest the implementation of new mechanisms to tackle these issues. For example, answerers can be notified when comments are posted to correct their answers or answer new questions. Once commenters post information to enhance answers, the answerers or the community can decide whether or not to accept their effort for knowledge maintenance. Comments of type *praise* can also be detected automatically, and Stack Overflow can suggest these commenters to upvote the answer instead. Meanwhile, users can be provided with an option to post a short comment when they upvote an answer.

The majority of comments enhance their associated answer from a diverse range of perspectives (e.g., pointing out weaknesses or providing additional references). Even though the majority of informative comments follow Stack Overflow commenting guidelines (e.g., requesting clarification and adding relevant information), users are still posting a considerable portion of comments that are discouraged by Stack Overflow (e.g., praising an answer, or suggesting a correction).

4.4.2 RQ2: When do users post comments?

Motivation: Once an answer is created, comments can be posted under that answer at any time. Meanwhile, the answer can be edited to reflect any update to its content. In Section 4.4.1, we observed that the majority of comments are informative, and thus, can be potentially leveraged to enhance their associated answers. Therefore, in this RQ, we analyze the temporal dynamics of comments to find out when the commenting activities occur and when their associated answers are edited. Our findings may be an indicator of the ability of the community to effectively integrate comments back into their associated answers, because "*comments are temporary 'Post-It' notes left on a question or answer*" (Stack Overflow, 2019). Ideally, the value of a comment can be best reflected in its associated answer. In addition, accepted answers are considered as the "best" solutions given to their associated questions (Anderson et al., 2012). Therefore, we wish to analyze how the timing of commenting activities is associated with the timing of answer acceptance. Understanding the temporal dynamics of comments provides us with insights into how to effectively manage the large collection of commenting-threads.

Approach: We first investigate when do users post comments. Namely, how long it takes for a user to post a comment since the creation of an answer, and whether a comment is posted before or after an answer is accepted. We also classify the commenting time into three categories: within one day, from one day to 30 days, and more than 30 days, to characterize whether or not a commenting activity occurs as a fast response to an answer. Among the accepted answers, we analyze whether the comment was posted before or after its associated answer was accepted. We analyze when do users post comments across different comment types that we identified from the 3,000 statistically representative samples of comments in RQ2 to understand the relationship between the type of comments and the timing of commenting. Lastly, from all the answers with comments, we extract the creation time and the last edit time of the answers. We also extract the creation time of the latest comments in these answers. By examining the proportion of answers that are edited after the discussion through comments, we wish to estimate an upper bound on the time it takes for the comments to be integrated back into their associated answers

Results: Most comments were posted within one day after the creation of their associated answers. Comments of type *advantage* and *weakness* are more likely to be posted later than other informative comment types. The proportion of comments posted during different ranges of commenting time is shown in Fig. 4.7 for each comment type. More than half of the tagged comments are posted within one day for each comment type. Comments of type *irrelevant* and *improvement* have a higher chance to be posted within one day. Within one day that the answer is posted, 70.8% of the posted comments are informative. From day one to day three, 75.5% of the posted comments are informative. After three days, 78.4% of the posted comments are informative. After one year, the proportion of comments that are informative increases to 81.0%. **Therefore, later comments are more likely to be informative.** As an example, in Jan 2016 an online tutorial was posted in an accepted answer about installing and switching PHP versions. More than one year later (i.e., in May 2017), a comment[18] pointed out that the tutorial was moved. In another example, a comment[19] mentioned more than one year later (i.e., April 2017) that "*this no longer works ... changing that class selector fixes it*" under an accepted answer that was originally posted in January 2016. Hence, we encourage users to carefully read late arriving comments because of

[18]https://www.stackoverflow.com/posts/comments/75190051/
[19]https://www.stackoverflow.com/posts/comments/73451366/

their likelihood to point out incorrect or updated information (e.g., an answer is obsolete, as we examined in Chapter 3).

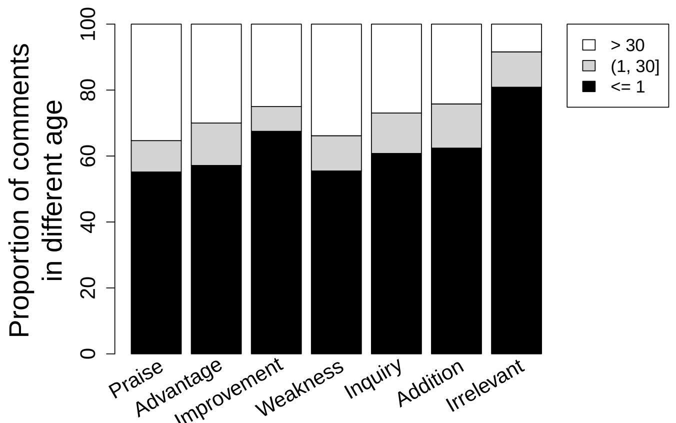


Figure 4.7: The proportion of comments within different ranges of commenting time (in days) for each comment type.

The majority of commenting activities occur after their associated answers is accepted. Among all the accepted answers with comments, 77.1% of the comments are posted after their associated answer is accepted. Fig. 4.8 shows the proportion of the tagged comments posted before and after their associated accepted answer. Even though askers have selected the accepted answers and the community tends to consider accepted answers as the "best" answers, **the community does not necessarily stop discussing these answers through commenting**. We suggest answer seekers to carefully read through the commenting-thread (i.e., the flattened list of all the comments that are associated with an answer after clicking "show N more comments" instead of only the top 5 comments that are displayed by default), even in the accepted answers. **Even though an answer is accepted, it does not necessarily mean that it is proven to be the "best", and any comment that is associated with this answer can potentially enhance the answer itself.** For example, in Fig. 4.1, the answer was accepted in July 2010, and the comment that made a correction to the answer was actually posted in March 2013, i.e., after more than two years since the answer was accepted. So far, this comment has gained a score that is 3 times higher than the score of its associated answer. Zhang et al. (2019b) also observed that "30.7% of the (studied) obsolete accepted answers got updated after being noted as obsolete (by a comment that is associated with the obsolete answer)".

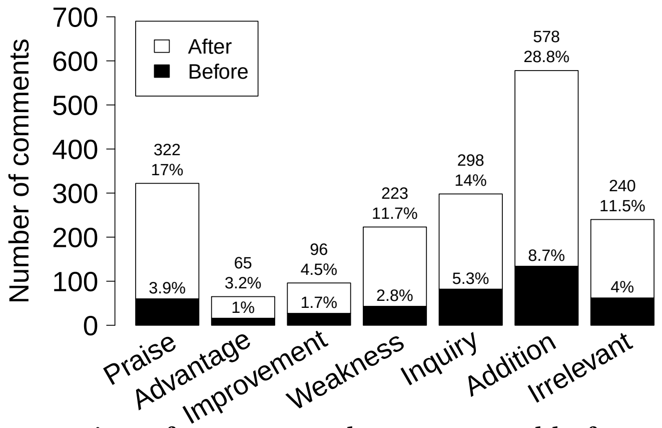


Figure 4.8: The proportion of comments that are posted before and after their associated answer is accepted for each comment type. Note that the total number of the comments shown in this figure is less than 2,000 since some manually studied comments were posted under non-accepted answers.

Answers are rarely updated after comments are posted, indicating that comments are rarely integrated back into answers – thus, users have to carefully read the comments. In the 11.4 million answers with comments, 61.9% (i.e., 7.1 million) of the answers have never been edited since their creation. Only 14.1% (i.e., 1.6 million) of the answers are edited after any comment. Note that 14.1% is an upper bound, since the edits in answers may not be related to the posted comments. Although the majority of comments are informative based on our findings in Section 4.4.1, comments are rarely integrated back into their associated answers. Therefore, we suggest the Stack Overflow team to encourage users to maintain answers with badges and reputation points, e.g., rewarding users who actively update answers by leveraging their associated comments.

The acceptance of an answer is not the end of commenting activities; instead, the majority of commenting activities occur after the acceptance of an answer. Generally, more than half of the comments are fast responses occurring within one day of the answer creation. Comments of type *advantage* and *weakness* are more likely to be posted later than other informative comments, and later comments tend to be more informative. Even though most comments provide useful information, they are rarely integrated back into answers.

4.4.3 RQ3: What types of users participate in the commenting activities?

Motivation: Stack Overflow sets restrictions on who can post comments. Namely, any user with at least 50 reputation points, the owner of the answer, and the owner of the question thread can post comments. To better organize commenting-threads, it is important to understand what types of users participate in these activities. Part of the reasons why organizing such a large collection of comments is challenging, is that comments are posted in a crowdsourced manner by different users. Therefore, in this RQ, we wish to understand how commenters with different roles (e.g., asker and answerers) are involved in the commenting-threads.

Approach: Based on the role of a commenter in the entire question thread, we categorize commenters into one of the following three groups:

1. **Asker**: the user who posted the question;

2. **Answerer**: the user who posted the answer;
3. **Outsider**: the user who belongs to neither of the two above-mentioned roles in that question thread.

We refer to an asker or answerer who is involved in the question thread (groups 1 & 2) as an *insider* (since they were involved earlier in the question answering process). The role of a commenter is assigned using the priority: asker > answerer > outsider. For example, if a user has multiple roles, such as an asker and an answerer, we consider the user as an asker.

Furthermore, we analyze how the roles of commenters are correlated with the comment types and the temporal dynamics of comments, in terms of the commenting time and whether commenting occurs before or after an answer acceptance.

To find out how experienced are the commenters when they post comments, we analyze a user's reputation within each comment type. Since a user's reputation changes over time, we crawl the daily activities of a user from their user profile webpages and calculate the reputation points of the user when he/she posted a comment. We analyze the relationship between the types of comments and the reputation points of users when they posted the comments.

Results: In general, users are actively posting comments. All the 32.3 million comments are posted by 1.9 million users, compared with 1.7 million users who post 22.7 million answers. On average, among all the answers and their associated comments in each question thread, the ratio of the number of commenters to answerers is 1.675:1, that is, there are 67.5% more commenters than answerers per question thread. The median of the ratio of the number of commenters to answerers is 1.75:1. As of April

16, 2019, the top commenter (i.e., Jon Skeet) has posted 43,876 comments that are associated with 24,568 answers. Stack Overflow uses badges to encourage users to leave comments under posts. As a result, 881,649 *Commentator* badges (i.e., to award users who leave 10 comments) and 10,232 *Pundit* badges (i.e., to award users who leave 10 comments with a score of 5 or more) are rewarded to users[20].

The majority of comments are posted by insiders within one day since the creation of answers, while outsiders are more likely to post comments after one month. Among all the comments that are associated with answers, 45.2% are posted by askers, and 31.1% are posted by answerers. 62.3% of the commenters are users who post under their own answers or questions (i.e., users with lower than 50 reputation points). As shown in Fig. 4.9, within one month, the majority of comments are posted by either askers or answerers (i.e., insiders). The dynamics of commenting activities are similar to the dynamics of answering activities on Stack Overflow, i.e., most questions get their accepted answers in half an hour (Wang et al., 2017). However, after one month, the majority of comments are posted by outsiders (i.e., the user did not post the question nor the answer, before posting the comment). In Section 4.4.2, we find that later comments are more informative; therefore, after one month, outsiders start to play an important role in maintaining the associated answers through commenting activities.

These results suggest that the maintenance of crowdsourced knowledge is a long-term task, and later activities should not be neglected. While outsiders do not contribute to the discussion in earlier stages, they are significantly involved later on. In addition, we observe that askers, answers, and outsiders are all involved in the commenting activities, both before and after the acceptance of an answer, as shown in Fig. 4.10.

[20]Data obtained on April 11, 2019 from `https://stackoverflow.com/help/badges.`

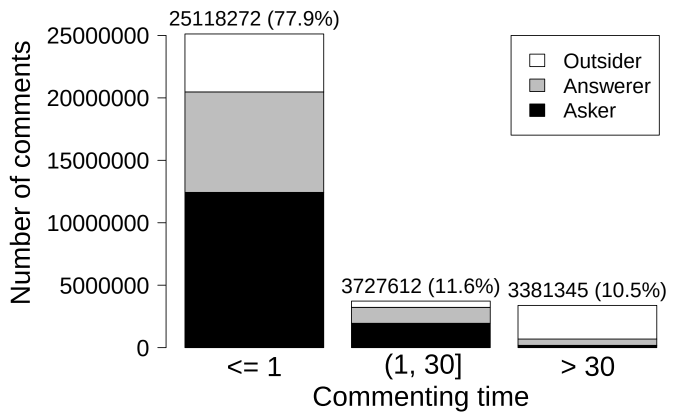


Figure 4.9: The number and proportion of comments that were posted by different user roles in different ranges of commenting time.

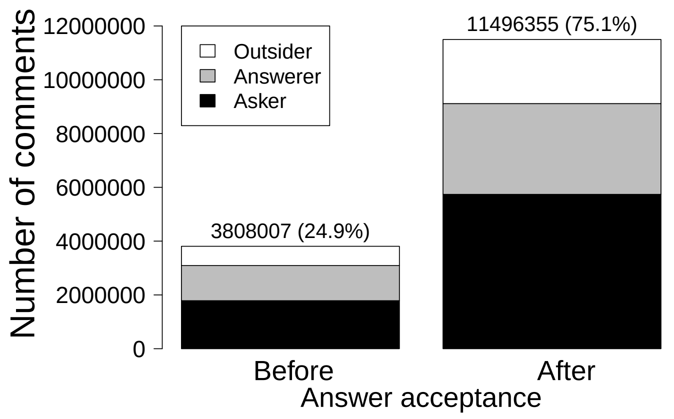


Figure 4.10: The number and proportion of comments that were posted by different user roles before and after the answer acceptance.

Askers mainly post comments that belong to type *praise*, *inquiry*, and *weakness*. Answerers mainly post comments in type *advantage*, *improvement*, *addition*, and *irrelevant*. The proportion of commenter roles in each comment type is shown in

Fig. 4.11. We notice that a significant proportion of comments of type *advantage, improvement,* and *weakness* are posted by outsiders, although these outsiders never participate in the entire question thread before posting the comment. Based on the above-mentioned observations, Stack Overflow can design a better channel for askers to appreciate answers. **A new praise channel instead of commenting can separate praising activities from commenting activities that can enhance the value of an answer.** The praise channel helps make a commenting-thread less crowded with irrelevant comments. As examples, both GitHub reactions[21] and Basecamp boosts[22] are such designs to channel praise comments. Stack Overflow can also provide the associated askers and answerers with alternative channels instead of posting comments of type *praise* and *inquiry*, such as praising by sending iconic expressions to answerers or other commenters instead of praising in comments. Only the praised users are concerned with such praising activities, while the commenting area can be a place for the community to discuss the answer with praising content hidden or removed.

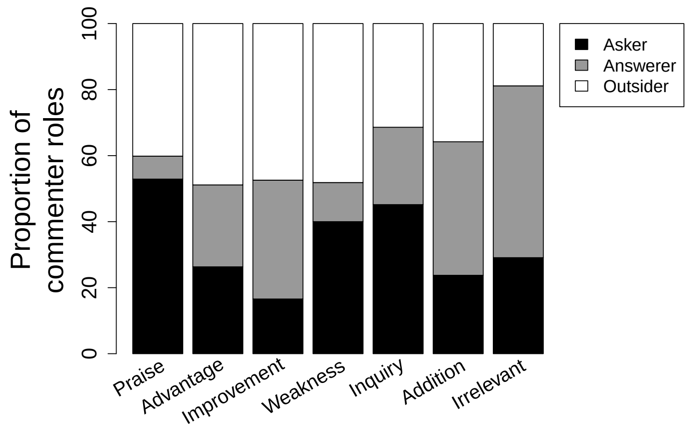


Figure 4.11: Proportion of commenter roles in each comment type.

[21]https://developer.github.com/v3/reactions/
[22]https://3.basecamp-help.com/article/391-boosts

Among users who post informative comments, inexperienced users (i.e., ones with lower reputation) tend to raise limitations and concerns by posting comments of type *weakness* and *inquiry*, while experienced users (i.e., ones with higher reputation) tend to enhance the answer with their comments by posting comments of type *advantage*, *improvement*, and *addition*. Fig. 4.12 shows the distribution of reputation points for users who post in each comment type. Even though comments of type *weakness* and *inquiry* are often posted by users with lower reputation points (with a median value of 465 and 423) than comments of type *advantage*, *improvement*, and *addition* (with a median value of 1,069.5, 1,341.5, and 956, respectively), all these users are actively contributing to enhance the associated answers. To further test if these differences are statistically significant, we ran the Mann-Whitney U test between the distribution of user reputation points for the comments of type *weakness* and each one of the three other types (i.e., *advantage*, *improvement*, and *addition*). We find that the difference is statistically significant with p-value $< 0.05/3$ (adjusted with a Bonferroni correction) in all three cases. The reputation points of users who post the comments of type *inquiry* is also statistically significantly lower than each one of the three other comment types that enhance their associated answers (i.e., *advantage*, *improvement*, and *addition*) with the p-values all below 0.05/3 (adjusted with a Bonferroni correction).

Among users who post uninformative comments, inexperienced users (i.e., with median reputation points of 292.5) tend to post comments of type *praise*. These commenters are probably not familiar with Stack Overflow, and simply express their appreciation through commenting instead of upvoting or accepting answers. **Experienced users (i.e., with median reputation points of 1,090.5) tend to post comments of**

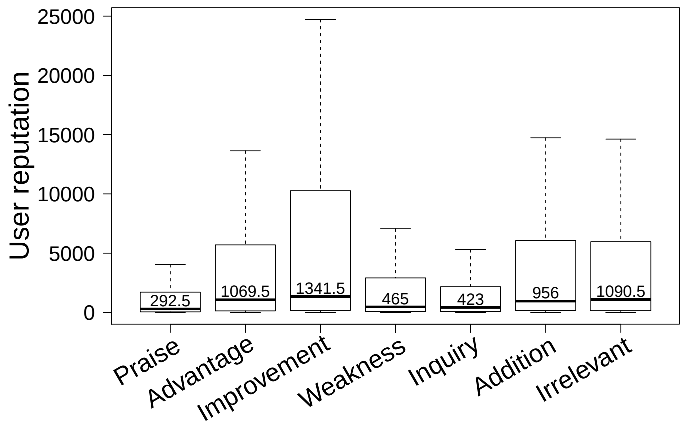


Figure 4.12: The distribution of user reputation points in each comment type (the median value is shown inside each box).

type *irrelevant.* Even though these users have reputation as high as users who post informative comments, they do not necessarily enhance the associated answers by commenting.

Furthermore, we group users by their reputation points. Fig. 4.13 shows the proportion of comments in different types that are posted by different user groups. The reputation thresholds among different user groups are defined by Stack Overflow[23]. We find that users with higher reputation points are more likely to post a lower proportion of comments of type *praise, weakness,* and *inquiry,* and are more likely to post a higher proportion of comments of type *advantage, improvement, addition,* and *irrelevant.* Such users with higher reputation points are probably more aware of the community rules; thus, posting fewer comments to praise an answer or make an additional inquiry. Users with higher reputation points also contribute to the crowdsourced knowledge sharing through the frequent posting of comments of type *advantage, improvement,* and *addition.* Surprisingly, users with higher reputation points post fewer

[23]https://stackoverflow.com/help/privileges?tab=milestone

comments to point out weaknesses and more irrelevant comments.

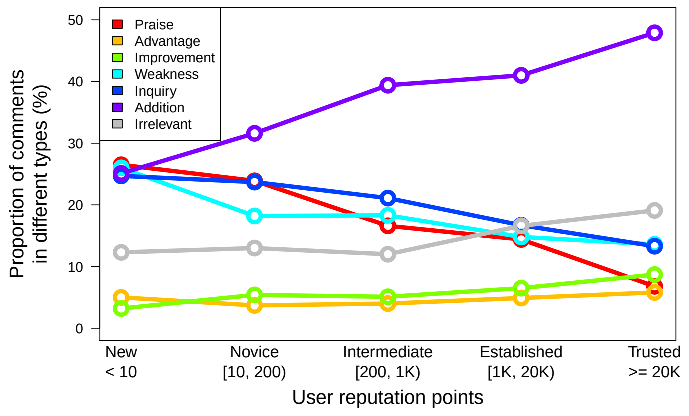


Figure 4.13: The proportion of comments in different types among different user groups.

Currently, commenting activities do not reward any reputation point on Stack Overflow. Even if a comment is extremely helpful and gets a large number of upvotes, the commenter will not gain any reputation points. In the example shown in Fig. 4.1, although the comment got 1,135 scores compared to a score of 376 for the answer, the commenter gained **no** reputation points while the answerer gained 3,760 reputation points (i.e., 10 reputation points for each one of the 376 upvotes). This commenter contributed to the maintenance of the crowdsourced knowledge and is recognized by the community (i.e., through the comment score), but he did not receive any reward. Although there exists 2 badges (i.e., *Commentator*, which is given to commenters who leave 10 comments, and *Pundit*, which is given to commenters who leave 10 comments with score of 5 or more) are related to commenting activities on Stack Overflow, only the users who reach these specific criteria can receive these badges, regardless of the usefulness and importance of any of their comments. These two badges are also designed for comments posted under both questions and answers; therefore, they are not directly designed for encouraging users to maintain Stack Overflow answers (note that 2,000 reputation points are required to maintain an answer by directly editing). On the other hand, the upvoting of an answer by other users directly adds reputation points to the answerer.

Users are highly involved in commenting. Askers mainly post comments that belong to type *praise*, *inquiry*, and *weakness*, while answerers mainly post comments of type *advantage*, *improvement*, and *addition*. Insiders post the majority of comments within one month, while outsiders are more likely to post comments after one month. Among informative comments, inexperienced users tend to raise limitations and concerns while experienced users tend to enhance the answer by commenting. Among uninformative comments, inexperienced users tend to praise the answer while experienced users tend to post irrelevant information.

4.5 Discussion

4.5.1 Implications for Stack Overflow and Users

Although existing answers can be revised and new answers can be created in their associated question threads for updating existing knowledge, it is unclear how effectively do users maintain answers. In addition, the evolution of the underlying programming languages, APIs, and other software artifacts makes it challenging to keep the 22.7 million Stack Overflow answers up to date, i.e., it is challenging to evaluate the answer quality in the long term. On the other hand, comments provide additional observations to their associated answers, such as answer obsolescence (as we examined in

Chapter 3) and security flaws in answers[24]. Under these scenarios, users who post these informative comments play an important role in maintaining the existing crowdsourced knowledge by observing and even addressing issues in answers. Therefore, these commenting activities can improve the long term value of their associated answers.

Based on our findings, we encourage Stack Overflow designers & engineers to focus on how to more effectively maintain the crowdsourced knowledge on Stack Overflow by leveraging the large collection of comments. We note that the proportion of answers that currently have been updated based on the rich content in comments is low. We provide below some implications for Stack Overflow and users based on our findings:

1. Since informative comments can significantly enhance their associated answers, we propose that these commenters are rewarded with reputation points, thus motivating the maintenance of crowdsourced knowledge. 4.4 million (i.e., 38.9%) of the answers with comments have a comment with an equal or higher score than the answer itself. However, under the current reputation system, a commenter does not gain any reputation points while an answerer gains 10 reputation points from each upvote.

2. Stack Overflow should encourage users to maintain answers, e.g., by rewarding users who leverage comments to update answers with badges and reputation points. As Stack Overflow and the knowledge within it age (Stack Overflow is over 10 years old today), many answers on Stack Overflow are likely to become outdated relative to the latest technologies. We already observed many answers that are not updated to reflect informative comments on these answers. Therefore,

[24]https://www.attackflow.com/Blog/StackOverflow

knowledge maintenance should be actively encouraged. For example, a checkbox of “answer maintenance” can be provided to users who post comments to indicate that the posted comments can be potentially used to maintain the answer, and a review queue can be added for these types of comments. When a user posts a comment which could be used to maintain the answer, the user can check the “answer maintenance” checkbox then this comment will be added into a queue for the community to review. If the community agrees with the comment, the comment could be labeled as “answer maintenance” to indicate its value. If these comments that serve the purpose of maintaining answers get approval after the review process, they can be highlighted and their corresponding users can be rewarded through the gamification mechanism (e.g., through badges as done for answer editing badges (Wang et al., 2018a)).

3. Users can tag their comments based on our existing comment types. With tagged comments, a better organization scheme can be implemented to display comments, thus leveraging the massive collection of informative comments for the purpose of both answer maintenance and information retrieval. In addition, an automated classifier can be developed to identify informative comments and comments of different types. The observed characteristics from our study of the temporal dynamics of commenting activities can provide insights for future work to build such an automated classifier.

4. Comments of type *praise* exist while they do not improve the quality of an answer. A classifier can be implemented to detect comments of type *praise*. Users can be suggested to upvote an answer instead of posting a comment. By removing these comments of type *praise*, users can retrieve informative comments more

effectively, which eventually assists them in solving their issues.

5. Unrecommended uses of comments can be flagged to help users follow Stack Overflow's guidelines. Comments that suggest corrections, answer a question, or relay a compliment can be automatically detected, and proper actions can be suggested to these commenters. A classifier to automatically identify such unrecommended uses of comments can be built, or individual classifiers can be built to tackle each unrecommended case. As a result, unnecessary comments can be deleted and users can retrieve informative comments more effectively. Similarly, a classifier to identify informative comments can be built to effectively assist users in retrieving relevant information from comments. Our findings can be leveraged by future work for comment classification. For example, we find in Fig. 4.13 that users with higher reputation points post a higher proportion of certain types of informative comments (e.g., *addition* and *improvement*). The reputation of a commenter may be used as a potential metric to identify informative comments. Furthermore, Stack Overflow can provide notifications to users about posting potentially uninformative comments, thus the overall informativeness of comments throughout Stack Overflow can be further enhanced.

6. Without any active organizing effort, the best suggestion so far for users is to read every single comment carefully, regardless of whether it is displayed or not. In particular, a reader is suggested to read later comments since they are more likely to be informative. Finally, to gain a closer look at users' opinions on Stack Overflow comments, we conduct a preliminary user survey to ask 22 participants the following question: "*Do you read comments when you use Stack Overflow?*". Out

of the 22 responses, 9 participants read comments occasionally and 1 participant never read comments. Our study shows that 45.4% of the participants do not actively read comments. Hence, based on our findings, we encourage users to read comments carefully since the majority (i.e., 75.6%) of comments provide a diverse variety of information that enhances their associated answers.

Note that any gamification mechanism on Stack Overflow may have adverse side effects as noted by Wang et al. (2018a) in recent work on the use of badges in Stack Overflow. For example, awarding reputation points for commenting activities could lead to an increase in the number of uninformative comments by users who attempt to fish for reputation points by posting comments. Future studies are needed to study the impact of gamification mechanism on the user participation and its side effect to have a better balance.

4.5.2 Implications for Researchers

Another implication of our study is for researchers. Since 2009, many research efforts continue to leverage the Stack Overflow dataset. The majority of the studies only leveraged the information related to questions and answers. There exists a limited number of prior studies that leveraged the information from comments. For example, Zou et al. (2015) analyzed both posts and comments to investigate non-functional requirements on Stack Overflow. Castelle (2018) evaluated the classification models of abusive language from Stack Overflow comments. We encourage future research to leverage the

32.3 million comments that are associated with answers to actively support maintenance efforts of such crowdsourced knowledge. We observe that answers can be updated through the leaving of informative comments on these answers. Therefore, reviewing comments is recommended when analyzing Stack Overflow answers. Furthermore, researchers can leverage such rich and informative comments to enhance various software engineering tasks, e.g., API documentation enhancement (Treude and Robillard, 2016) and question answering bot (Tian et al., 2017).

In addition, our study is the first work to empirically study the types of information in comments. In comparison, Poché et al. (2017) observed that 30% of the comments on YouTube coding tutorials are informative, while Chen et al. (2014) observed that 35% of app reviews from Google Play are informative. We observe that the majority (∼76%) of these comments are informative and enhance answers from a diverse range of perspectives. Future studies may propose data-driven solutions for retrieving informative comments to either identify or summarize such comments in an automated manner. Future research can leverage approaches from the machine learning and natural language processing communities to automatically identify the comment types/-subtypes that we identified. The identified comments may assist developers with the reading of Stack Overflow posts or assist researchers to better leverage the information in comments.

4.6 Threats to Validity

4.6.1 External Validity

Threats to external validity relate to the generalizability of our findings. The number of comments is large and it is impossible to study all of the comments in our qualitative study. In order to minimize the bias, we randomly sampled 3,000 statistically representative comments, giving us a confidence level of 99% and a confidence interval of 2.4%. In this study, we focus on Stack Overflow, which is one of the most popular Q&A websites for developers, hence, our results may not generalize to other Q&A websites. To alleviate this threat, more Q&A websites should be studied in the future. Furthermore, in this study, we analyzed comments that are associated with answers. The comments that are associated with questions can also be informative, and thus, contribute to the crowdsourced knowledge sharing on Stack Overflow. Future research should investigate questions' comments and explore how such comments enhance the question answering activities on Stack Overflow.

4.6.2 Internal Validity

Threats to internal validity are related to experimenter errors and bias. Our study involved qualitative studies which were performed by humans. Bias may be introduced. To reduce the bias of our analysis, each comment is labeled by two researchers (with me being one of them) individually and discrepancies are discussed until a consensus is reached. We measured the level of the inter-rate agreement in our qualitative study, and the agreement value is substantial (i.e., 0.86 and 0.90 at the subtype and type level, respectively) even before the consensus is reached.

4.7 Chapter Summary

In this chapter, we investigate 32.3 million comments that are associated with answers on Stack Overflow. Since 2009, users create more comments than answers on a yearly basis. 23% (i.e., 2.6 million) of the answers with comments even have a commenting-thread longer than the actual answer, indicating the richness of information in comments.

Our empirical study provides an in-depth understanding of the commenting activities on Stack Overflow. We identify various types of comments and find that the majority of comments are informative as they enhance answers from a diverse range of perspectives. We also characterize the commenting activities in terms of time and user roles. We find that comments are rarely integrated back into their associated answers. Insiders (i.e., askers and answerers) post the majority of comments within one day, while outsiders (i.e., users with no earlier activity within a question thread) post the majority of comments after one month. These outsiders also post informative comments.

Our analysis can be leveraged to create alternative channels for askers and answerers to request detailed information and receive compliments, respectively. The informative comments can also be further utilized to actively maintain their associated answers and improve their presentation. Our findings can be leveraged for crowdsourced knowledge maintenance and organization.

CHAPTER 5

The retrieval of information in hidden comments

Many Stack Overflow answers have associated informative comments that can strengthen them and assist developers. As examined in Chapter 4, we observed that comments can provide additional information to point out issues in their associated answer, such as the obsolescence of an answer. By showing more informative comments (e.g., the ones with higher scores) and hiding less informative ones, developers can more effectively retrieve information from the comments that are associated with an answer. Currently, Stack Overflow prioritizes the display of comments and as a result, 4.4 million comments (possibly including informative comments) are hidden by default from developers.

In this study, we investigate whether this mechanism effectively organizes informative comments. We find that: 1) The current comment organization mechanism does not work well due to the large amount of tie-scored comments (e.g., 87% of the comments have 0-score). 2) In 97.3% of answers with hidden comments, at least one comment that is possibly informative is hidden while another comment with the same score is shown (i.e., unfairly hidden comments). The longest unfairly hidden comment is more likely to be informative than the shortest one. Our findings highlight that Stack Overflow should consider adjusting the comment organization mechanism to help developers effectively retrieve informative comments. Furthermore, we build a classifier that can effectively distinguish informative

comments from uninformative comments. We also evaluate two alternative comment organization mechanisms (i.e., the Length mechanism and the Random mechanism) based on text similarity and the prediction of our classifier.

An earlier version of this chapter is under review at the ACM Transactions on Software Engineering and Methodology Journal (TOSEM).

5.1 Introduction

STACK Overflow answers offer developers solutions to address their questions. This collection of answers enables developers to learn and share valuable programming knowledge. An answer creates a starting point for a discussion related to a question. Users can post alternative answers, edit existing answers, or post comments under the answer. We consider an answer thread as consisting of an answer and all the comments associated with the answer. Comments can provide additional information to support their associated answer (as we examined in Chapter 4), or even point out issues in the answer, such as the obsolescence of answers (as we examined in Chapter 3). Such information can be very helpful to developers.

However, the more comments that are posted under an answer, especially those answers that attract large user traffic, the more needed effort and time that developers need to retrieve information on Stack Overflow. In order to keep each answer thread compact, Stack Overflow implements a comment organization mechanism to only show the top 5 comments. Aiming at showing the most informative comments and hiding less informative ones, the mechanism first ranks these comments based on their scores. When multiple comments have the same score, they are then ranked by their creation time. In addition, informative comments might be hidden by the comment organization mechanism in turn reducing the chances of someone voting on them. To read the hidden comments, users need to click a link under the last shown comments. In addition, hidden comments are not indexed by Google[1], which also hinders the accessibility to the information in comments for answer seekers.

We observe that 4.4 million comments are hidden as of September 2017 and as a

[1]https://meta.stackexchange.com/a/304906/

result a large amount of useful information is probably buried in such hidden comments. As once commented by David Fullerton (the president of Stack Overflow) in a discussion of "hide trivial comments" on Stack Exchange META, "***if I have to click that link every time just in case there's something useful in the comments, haven't we failed?***"[2]. In other words, it is essential for Stack Overflow to show the most useful comments and hide less useful ones to ease the information retrieval for developers. Therefore, the decision of how to organize Stack Overflow comments is essentially an information retrieval (IR) task within a Software Engineering context – how to select the most informative comment to display to developers in a non ad hoc manner. Selecting informative comments from answers is similar to prior IR & SE research efforts, such as bug localization (Tantithamthavorn et al., 2018; Wang and Lo, 2014; Gu et al., 2018) and source code recommendation (Lv et al., 2015), which aim to assist developers by effectively retrieving information from large and unorganized corpuses. The organization of comments can negatively impact the effective retrieval of useful comments since the ordering of comments leads to some comments to be hidden from developers and search engines (which is a major problem since many developers reach Stack Overflow answers through search engine results (An et al., 2017; Abdalkareem et al., 2017; Yang et al., 2016; Wu et al., 2018)).

Therefore, it is important to understand how well the comment organization mechanism works. Does this mechanism, in fact, show the more informative comments and hide less informative comments as designed? What types of information is actually discussed in hidden and shown comments? By answering these questions, we explore and raise the importance of a proper comment organization mechanism, and wish to provide insights to improve the current mechanism to make it easier for developers to

[2]https://meta.stackexchange.com/posts/comments/653443/

retrieve information on Stack Overflow.

In this chapter, we study 22.7 million answers and all of their 32.2 million associated comments. We first qualitatively study whether the shown comments are more informative than the hidden comments. In other words, is the comment organization mechanism actually hiding less informative comments? We investigate:

- **RQ1: What are the characteristics of both hidden and shown comments?**
  We observe that hidden comments have a similar amount of text as shown comments. Hidden comments have an additional vocabulary and add a greater variety of textual content to their associated answers than that of shown comments to the same answer. Based on our qualitative study, we observe that more than 70% of the comments (both hidden and shown) are informative, as they provide alternative answers, or point out flaws in answers.

From the previous qualitative study, we observe that many informative comments can be hidden due to the current comment organization mechanism. To further understand the reason for such cases, we perform an empirical analysis to evaluate the efficacy of the comment organization mechanism by answering the following two research questions:

- **RQ2: How effective is the current comment organization mechanism?**
  The comment organization mechanism does not work effectively. Instead of giving priority to highly scored comments, it gives priority to early comments since the comment organization mechanism fails to consider a very common case: multiple comments may have the same score (i.e., tie-scored comments). More specifically, in 97.3% of the answers that have hidden comments, at least one

comment is hidden (i.e., unfairly hidden comment) while other comments with the same score are shown (i.e., unfairly shown comments).

- **RQ3: What are the characteristics of unfairly hidden comments?**
  Based on our qualitative study, we obverse that the longest unfairly hidden comments are more likely to be informative than the shortest unfairly shown comments (when the shortest unfairly shown comments are less than 50 characters).

Based on above findings, we suggest that Stack Overflow enhances its comment organization mechanism to better handle tie-scored comments which represent 92.9% of the hidden comments. Useful information can be embedded in these hidden comments. Developers could retrieve such information more effectively if hidden comments are more properly organized. Instead of simply ranking comments by their score then their creation time, the comment organization mechanism needs to introduce a higher priority for more informative comments, which can ease the retrieval of information for developers. For example, Stack Overflow can replace the shortest unfairly shown comments with the longest unfairly hidden comments (i.e., with tied score). Such a new mechanism will ensure that such informative comments are indexed by search engines. To evaluate our findings, we build a classifier (with an AUC of 0.8) to distinguish informative comments from uninformative comments. We evaluate two alternative comment organization mechanisms (i.e., the *Length* mechanism and the *Random* mechanism) based on text similarity and the prediction of our classifier. Our analysis shows that the *Random* mechanism is an option to diversify the comments, and the *Length* mechanism enables an improvement to show informative comments. In addition, we encourage developers to read through all comments (including hidden comments) in case any further correction/improvement is made by such comments, such as observations of answer obsolescence, security vulnerabilities, and errors.

Chapter Organization: The rest of the chapter is organized as follows. Section 5.2 introduces background about Stack Overflow's comment organization mechanism and details the dataset that we used in this study. Section 5.3 presents the case study results of our research questions. Section 5.4 discusses our findings and provides actionable suggestions. Section 5.5 discusses potential threats from our case study. Finally, Section 5.6 concludes our study.

5.2 Case Study Setup

This section describes the subject of study and the process that we follow to collect the data for our case study.

5.2.1 Subject of Study

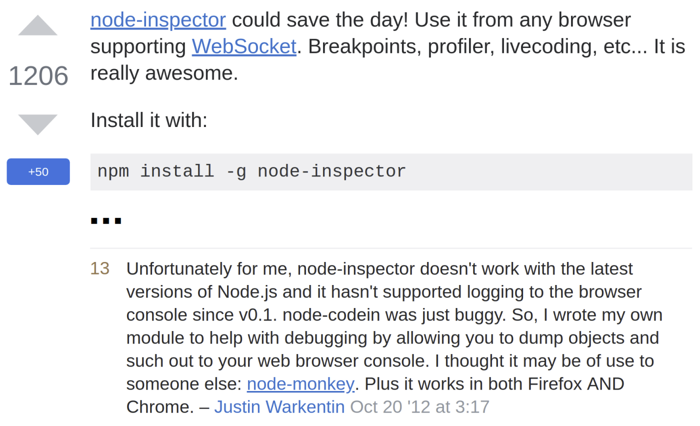
We briefly introduce the informativeness of comments under answers and the organization of comments on Stack Overflow.

The informativeness of comments on Stack Overflow

Commenting on Q&A websites can lead to more comprehensive discussions which improves the knowledge sharing process (Gazan, 2010; Poché et al., 2017; Zhang et al., 2019b). For example, as shown in Fig. 5.1, a user on Stack Overflow posted a comment[3] that pointed out a problem with an existing answer and provided an alternative solution. On Stack Overflow, comments are "*temporary 'Post-It' notes left on a question or*

[3] https://stackoverflow.com/posts/comments/17612489

answer"[4]. **As of September 2017, users on Stack Overflow have posted 60.2 million comments** under questions or answers, which represents more than the total number of questions and answers combined according to the data from Stack Exchange Data Explorer[5]. Note that in this study we refer to comments as *the comments that are associated with answers if not specified otherwise* since we focus on studying how comments contribute to providing additional value to answers. Even though new answers and revisions that contribute to existing answers can also provide additional values to existing answers, our study focuses on investigating whether comments are organized properly.



node-inspector could save the day! Use it from any browser supporting WebSocket. Breakpoints, profiler, livecoding, etc... It is really awesome.

Install it with:

```
npm install -g node-inspector
```

...

13 Unfortunately for me, node-inspector doesn't work with the latest versions of Node.js and it hasn't supported logging to the browser console since v0.1. node-codein was just buggy. So, I wrote my own module to help with debugging by allowing you to dump objects and such out to your web browser console. I thought it may be of use to someone else: node-monkey. Plus it works in both Firefox AND Chrome. – Justin Warkentin Oct 20 '12 at 3:17

Figure 5.1: An answer with one of its associated comments. The comment notes that the node-inspector no longer works with the recent versions of Node.js and proposes the use of an alternative module called node-monkey.

Comments open up a channel for developers to add informative discussions to their associated answers. Highly informative comments can expedite the problem solving process, or add knowledge that is worthwhile to share for a particular answer, such as the observation or update of an obsolete answer (as we examined in Chapter 3).

[4] https://stackoverflow.com/help/privileges/comment
[5] https://data.stackexchange.com/stackoverflow/query/945995

As explained by Jeff Atwood (the co-founder of Stack Overflow), "*there are often* ***important clarifications and addendums left as comments*** *that* ***substantially improve*** *the original post*" (Stack Overflow, 2009). In addition, Chang and Pal (2013) note that comments are a *first class citizen* of a Q&A platform. They can critically improve an answer by providing additional clarifications and refinements to the answer; thus, increasing the overall value of an answer. In Section 5.3.1, we also observe that comments provide value for various aspects of an answer, such as pointing out errors, making corrections, and providing alternative solutions. Therefore, commenting is an indispensable Stack Overflow feature that leads to improving and ensuring the long-lasting value of its crowdsourced knowledge. In other words, such informative comments can provide additional knowledge to developers when they seek answers.

However, uninformative comments could potentially threaten the Stack Overflow community by decreasing the information density of an answer thread (i.e., the compactness of an interface in terms of the amount of information) (Kandogan, 1998). If an increasing number of answer threads are filled up with discussions, such as through uninformative comments, developers may face difficulties in identifying relevant information.

Therefore, to promote informative comments and avoid uninformative ones, several sorting rules are applied to comments on Stack Overflow. First, comments can only be posted by the following three types of users: the asker, the answerer, and any user with at least 50 reputation points[6]. 11.7% of the users have at least 50 reputation points on Stack Overflow. On Stack Overflow, reputation points can be earned by various approaches, for example, 10 points if a question is upvoted, 10 points if an answer is upvoted, 15 points if an answer is marked accepted, and 2 points if a suggested edit

[6]https://stackoverflow.com/help/privileges/comment

is accepted[7]. Stack Overflow also incorporates a voting system to regulate the quality of comments[8]. **Comments can be up-voted, but cannot be down-voted. Thus, the lowest score a comment can have is 0.**

In short, commenting is widely used to structure discussions around their associated answers. In addition, comments can be as informative as their associated answers to some extent. Thus, in this chapter, we study what developers actually discuss in comments and characterize the informativeness of comments.

The organization of comments on Stack Overflow

Initially, Stack Overflow used to hide all comments under answers; however, this rule was abandoned because it hid too much information. As explained by Jeff Atwood that "*comments were all locked behind ... information was being lost*" (Stack Overflow, 2009). On the other hand, developers may find it difficult to locate useful information if all comments were shown under each answer. The single webpage that contains question, answers, and comments would have an increasing amount of content over time. Hence, developers will need to spend more effort and time to read and locate the relevant information.

To have a better balance between showing and hiding all comments, Stack Overflow implemented its current comment organization mechanism to enhance the readability of answer threads. Since 2009 (Stack Overflow was launched in 2008), this comment organization mechanism only shows the "top 5 comments" (Stack Overflow, 2009) for each answer (i.e., ***shown comments***). Note that if two comments have the same score

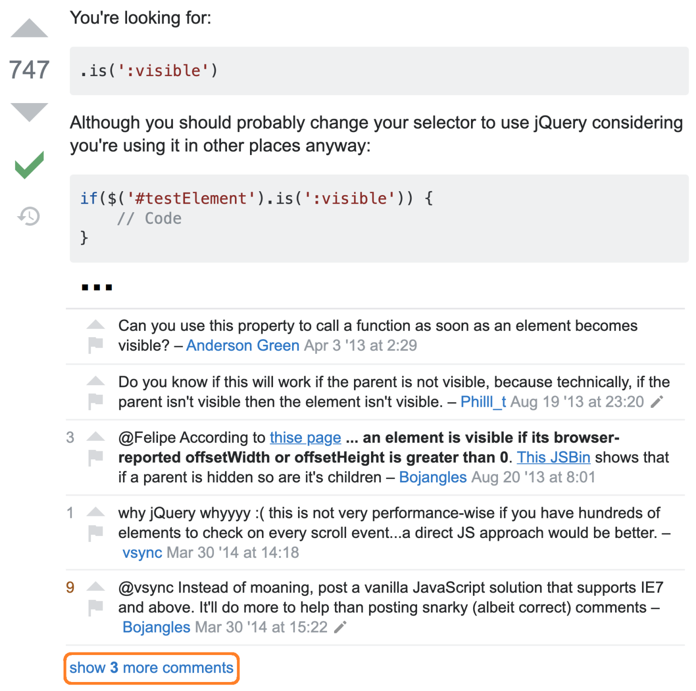
[7]https://stackoverflow.com/help/whats-reputation
[8]https://meta.stackexchange.com/a/17365

(i.e., tied score), they will be ranked by their creation time and the earlier created comment will be ranked higher. Additionally, if more than 30 answers are posted for a question, only the comments with a score greater than 0 will be shown. For example, if a question has more than 30 answers and one associated answer only has three comments that have a score greater than 0, then only these three comments will be shown under the answer. Other comments under an answer are hidden (i.e., ***hidden comments***). Even some comments with a score greater than 0 can be hidden if there are more than 5 comments with a score greater than 0 under their associated answer.

Users can click a link saying "show n more comments" to read all comments. An example[9] of an answer thread is shown in Fig. 5.2 with the shown comments for the answer and the clickable link at the bottom of the comments. By using the comment organization mechanism, ideally the 3 hidden comments are less likely to add additional information to their associated answers, while all the informative comments are shown. In this study, we investigate this comment organization mechanism to understand how it enables effective information retrieval on Stack Overflow.

The comment organization mechanism has been in place for nearly 10 years (as of October 2019) while the Stack Overflow community has expanded significantly since the launch of Stack Overflow. The number of users and answers have increased considerably. The number of comments also grows over the years as we examined in Chapter 4. However, the effectiveness of Stack Overflow's comment organization mechanism in showing informative comments and hiding uninformative comments remains unknown. Therefore, we wish to study whether comments (including both hidden and shown comments) are informative, and we investigate the efficacy of the comment organization mechanism. Moreover, it is common that developers use search engines

[9]https://stackoverflow.com/a/8774101/


747

You're looking for:

```
.is(':visible')
```

Although you should probably change your selector to use jQuery considering you're using it in other places anyway:

```
if($('#testElement').is(':visible')) {
    // Code
}
```

...

Can you use this property to call a function as soon as an element becomes visible? – Anderson Green Apr 3 '13 at 2:29

Do you know if this will work if the parent is not visible, because technically, if the parent isn't visible then the element isn't visible. – Philll_t Aug 19 '13 at 23:20

3 @Felipe According to thise page **... an element is visible if its browser-reported offsetWidth or offsetHeight is greater than 0**. This JSBin shows that if a parent is hidden so are it's children – Bojangles Aug 20 '13 at 8:01

1 why jQuery whyyyy :(this is not very performance-wise if you have hundreds of elements to check on every scroll event...a direct JS approach would be better. – vsync Mar 30 '14 at 14:18

9 @vsync Instead of moaning, post a vanilla JavaScript solution that supports IE7 and above. It'll do more to help than posting snarky (albeit correct) comments – Bojangles Mar 30 '14 at 15:22

Figure 5.2: An example of an answer with its shown comments and a link at bottom saying “show 3 more comments”. Once a user clicks the link, all comments (including the hidden comments) under the answer will be shown.

to look for solutions to their questions, and many top search results are from Stack Overflow answers (An et al., 2017; Abdalkareem et al., 2017; Yang et al., 2016; Wu et al., 2018). However, hidden comments are not indexed by search engines[10] (Stack Overflow, 2009). Additionally, a new informative comment might be already hidden by the user interface thus reducing the chances of users voting on it. As a result, hidden comments may get less public attention. It is unknown how the comment organization mechanism negatively affects developers in locating relevant information. For example, if a developer is seeking certain information that is actually from a hidden comment, the comment organization mechanism would then lead to information loss. In

[10]https://meta.stackexchange.com/a/23772/

this sense, it is necessary to examine the actual efficacy of the current comment organization mechanism.

5.2.2 Subject of Study

We download the data dump[11] that was published by Stack Exchange in September 2017. Table 5.1 lists the statistics of our studied data. We focus our study on the answers that have comments (i.e., $Answer_{comment}$). There are 32.2 million comments associated with these $Answer_{comment}$. 1.3 million (i.e., 11.4%) of $Answer_{comment}$ are answers with hidden comments (i.e., $Answer_{hidden}$). We identify such hidden comments by applying the comment organization mechanism (Stack Overflow, 2009) to the collected data in the data dump. We also randomly checked 100 answers on Stack Overflow and all of their associated comments are shown/hidden using the documented mechanism. Under such $Answer_{hidden}$, 4.4 million (i.e., 40.5%) of the comments are hidden. Note that $Answer_{comment}$ includes both $Answer_{hidden}$ and answers that have comments but none of these comments are hidden (i.e., $Answer_{noHidden}$). In general, $Answer_{hidden}$ are more popular than $Answer_{noHidden}$ on Stack Overflow. More specifically, the mean score of $Answer_{hidden}$ is 3.7 times higher than $Answer_{noHidden}$.

Table 5.1: Statistics of our studied data.

	Number	**Proportion**	**Mean/Median Score**
All Answers	22,668,556	100%	2.6 / 1
$Answer_{comment}$	11,396,766	50.3%	3.9 / 1
$Answer_{noHidden}$	10,093,265	44.5%	3.0 / 1
$Answer_{hidden}$	1,303,501	5.8%	11.1 / 1

[11] https://archive.org/details/stackexchange

5.3 Case Study Results

In this section, we present the results of our research questions. For each research question, we present the motivation of the research question, the approach to address the research question, and our experimental results for the research question.

5.3.1 RQ1: What are the characteristics of both hidden and shown comments?

Motivation: As shown in Section 5.2, comments can be informative, thus augmenting their associated answers with valuable knowledge. Identifying what developers discuss in comments helps us better understand how comments actually augment their associated answers, thus providing insights for improving the current commenting system and helping developers retrieve information on Stack Overflow. Stack Overflow uses a comment organization mechanism to split comments under answers into two groups: hidden and shown comments. However, it is not clear whether the shown comments are really more informative than hidden comments as designed. To have a better understanding of hidden and shown comments, we first conduct a quantitative study to investigate the characteristics of the textual content in both hidden and shown comments. Then we conduct a qualitative study to understand whether comments (including both hidden and shown comments) are informative. By knowing this, we can gain insight into the efficacy of the current comment organization mechanism.

Approach: To understand what are the characteristics of both hidden and shown comments, we perform both quantitative and qualitative analysis.

We study the characteristics of both hidden and shown comments using a quantitative approach that is described in the two following steps. We first compare the length of hidden comments with that of shown comments. The length of a comment is measured by the character number of the comment. We use the length of a comment as a baseline metric to reflect how informative a comment is. In each $Answer_{hidden}$, we calculate the median length (L_{hidden}) of all hidden comments within the answer, and the median length (L_{shown}) of all shown comments within the same answer. The length of a comment is widely used to characterize the quality of the comment in other sites (e.g., MetaFilter and YouTube[12]). As observed in prior research in the natural language processing community, sentences with more words often contain more variant information (i.e., larger entropy) (Tang et al., 2002). Prior research also observed that a minimum text length is required to reach a stable text coverage (Chujo and Utiyama, 2005). To compare L_{hidden} and L_{shown}, we define the ***median length ratio of comments*** in a pairwise manner as $Ratio_L = L_{hidden}/L_{shown}$. A value of $Ratio_L = 1$ means that the median length of all hidden comments under an answer is equal to the median length of all shown comments under the same answer. We also compare L_{hidden} with L_{shown} using the Wilcoxon signed-rank test and the Cliff's delta test (Cliff, 1993) to determine if there is any statistically significant difference between L_{hidden} and L_{shown}.

Second, to understand whether hidden comments add diverse information to the associated answers in contrast to their shown comments, we use the vector space model (VSM) to calculate the textual similarity between the hidden comments and the $Answer_{hidden}$ versus the similarity between the shown comments and

[12] http://ignorethecode.net/blog/2009/09/29/comments_size_does_matter/

the $Answer_{hidden}$. In VSM, each document is represented by a vector where the dimension of the vector is equal to the number of unique tokens (i.e., words) in all the documents (i.e., corpus), and the value of each element in a vector is represented by the term frequency – inverse document frequency weight (i.e. TF-IDF). VSM is commonly used for measuring the textual similarity between software engineering artifacts. Readers may refer to the prior studies (Wang et al., 2011, 2014; Thung et al., 2013; Chen et al., 2016; Oliveto et al., 2010; Gethers et al., 2011) for more details on VSM.

To apply VSM, we treat each answer as one document, all of its associated hidden comments together as one document, and all of its associated shown comments together as one document. For each document, we first perform the following common pre-processing steps (Wang et al., 2014; Chen et al., 2016): remove HTML tags/URL, split words by punctuation marks, split words using camel cases, convert upper case letters to lower case letters, and remove stop words. We use the stop words from the NLTK English stop words collection[13]. Note that we remove URLs for the purpose of building the VSM to compare the textual similarity between comments and their associated answer. In our qualitative analysis, we consider these URLs and find that comments pointing to relevant resources through URLs can be informative (see Section 5.3.1 for details). We then convert each pre-processed document to a vector, in which the weight of each element of the vector is calculated based on term frequency (i.e., the frequency of the term in the document) and inverse document frequency (i.e., the reciprocal of the number of documents containing the term). Finally, we compute the cosine similarity between answers and their associated comments (hidden and shown, respectively).

[13]https://www.nltk.org/book/ch02.html

In order to compare the textual similarity between hidden and shown comments under the same answer, we calculate the cosine similarity in a pairwise manner. In each $Answer_{hidden}$, we calculate the cosine similarity($S_{Answer\ vs.\ Hidden}$) between the answer and all of its associated hidden comments. In the same $Answer_{hidden}$, we calculate the cosine similarity($S_{Answer\ vs.\ Shown}$) between the answer and all of its associated shown comments. We define the ***pairwise cosine similarity ratio*** as $Ratio_S = S_{Answer\ vs.\ Shown}\ /\ S_{Answer\ vs.\ Hidden}$. Note that a value of $Ratio_S > 1$ means that, the shown comments are more similar to their associated answer as compared to that of the hidden comments.

We study the characteristics of both hidden and shown comments using a qualitative approach. To capture the information that developers could obtain from comments, we investigate what is discussed in hidden and shown comments. To do so, we randomly select 384 hidden comments from $Answer_{hidden}$, and randomly select 384 shown comments from $Answer_{hidden}$, in order to obtain a statistically representative sample with a 95% confidence level and a 5% confidence interval (Boslaugh, 2012).

We manually label the category of each comment. If a comment has multiple sentences, we assign a label to each sentence individually. Therefore, one comment can be assigned with multiple categories because a developer may discuss more than one category in a comment. We perform a lightweight open coding-like process that is similar to prior studies (Seaman, 1999; Seaman et al., 2008; Zhang et al., 2019b) to identify the category of topics that are discussed in a comment (i.e., *comment category*). This process involves 3 phases and is performed by two researchers (i.e., A1–A2, with me being one of them):

- Phase I: A1 identifies a draft list of comment categories based on a sample of 50 comments from hidden comments and another 50 comments from shown comments. Then, A1 and A2 use the draft list to label the comments collaboratively, during which the comment categories are revised and refined. For instance, A1 and A2 discussed their individual list of comment categories and created new labels to resolve any disagreement in the names of labels.

- Phase II: A1 and A2 independently apply the resulting categories from Phase I to label all 768 comments (i.e., 384 hidden comments and 384 shown comments). A1 and A2 take notes regarding the deficiency or ambiguity of the already-identified categories when labeling certain comments. Note that new categories are added during this phase if A1 and A2 observe the need for more categories when the existing labels are unable to represent a new comment. Saturation was constantly monitored during the labeling process and was reached after 200 comments. At the end of this phase, we end up with 7 categories of comments (see Table 5.2). Cohen's kappa (Gwet, 2002) is used to measure the inter-rater agreement, and the kappa value is 0.72 (measured at the end of Phase II), implying a high level of agreement.

- Phase III: A1 and A2 discuss the coding results obtained in Phase II to resolve any disagreement until a consensus is reached. No new categories are added during this phase with both A1 and A2 agreeing on all categories. We published our labeled data online[14].

We also compare if there is a statistically significant difference in the categories of

[14] `https://bit.ly/3bZ5GUQ`; for the final version of the chapter, the content will be made available on GitHub.

Table 5.2: Comment categories

Category	**Explanation**	**Example**
Praise	Praise an answer	Thank you. It worked for me :)[1]
Advantage	Discuss the advantage of an answer	This is ES6 syntax, it works fine in a sufficiently modern browser.[2]
Improvement	Make improvement to an answer	This should be an OR condition. It can't be both Saturday AND Sunday.[3]
Weakness	Point out the weakness of an answer	@ChrisPatterson I tried that but it would give me errors when I did anything but use the default settings[4]
Inquiry	Make inquiry based on an answer	What's the difference between the two?[5]
Addition	Provide additional information to an answer	an additional point for anyone looking at this. Make sure you haven't set both leading & trailing constraints as those override 'greater than' constraints[6]
Irrelevant	Discuss irrelevant topics to an answer	I'll leave a rebuttal here if that's ok with you. :)[7]

[1] https://www.stackoverflow.com/posts/comments/74500847/
[2] https://www.stackoverflow.com/posts/comments/63870245/
[3] https://www.stackoverflow.com/posts/comments/69209752/
[4] https://www.stackoverflow.com/posts/comments/59923929/
[5] https://www.stackoverflow.com/posts/comments/66174272/
[6] https://www.stackoverflow.com/posts/comments/74556909/
[7] https://www.stackoverflow.com/posts/comments/2771708/

hidden and shown comments using Wilcoxon signed-rank test and Cliff's delta test.

Quantitative Results: There is no statistically significant difference between hidden and shown comments in terms of median length. We plot the distribution of $Ratio_L$ in $Answer_{hidden}$ (as shown in Fig. 5.3). $Ratio_L$ shows a normal distribution with a mean value of 1. Our statistical test results show that there is no statistically significant difference between hidden and shown comments in terms of length (p-value $>$ 0.05).

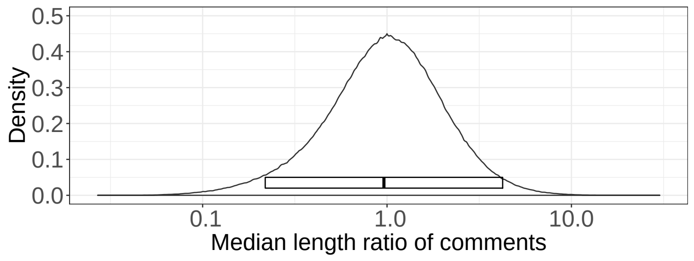


Figure 5.3: The distribution of the median length ratio of comments in a pairwise manner ($Ratio_L$). $Ratio_L = 1$ means that the median length of all hidden comments is equal to the median length of all shown comments under the same answer.

In the majority of $Answer_{hidden}$, hidden comments share less semantic similarity with the associated answers than that of shown comments. The finding may suggest that even though the hidden comments have a similar amount of text compared with the shown comments, the hidden ones have an additional vocabulary and add a greater variety of content than the shown comments to the associated answers. In 73.8% of $Answer_{hidden}$, the cosine similarity between the shown comments and the associated answers are no less than the cosine similarity between the hidden comments and the associated answers. The Wilcoxon signed-rank test shows a statistically significant difference (p-value < 0.05) for the cosine similarity between shown comments to their associated answers and hidden comments to their associated answers. The Cliff's delta test also shows that the difference is medium (i.e., -0.39).

Hidden comments have a similar amount of text as shown comments. Hidden comments have an additional vocabulary and add a greater variety of textual content to their associated answers than the shown comments for the same answer.

Qualitative Results: More than half of the comments are informative in both hidden and shown comments. Except for category *praise* and *irrelevant*, we consider the comments of other categories (i.e., *advantage*, *improvement*, *weakness*, *inquiry*, and *addition*) as ***informative***. Note that we consider the category *praise* as redundant for up-voting answers, thus we do not consider such comments as informative. The category *irrelevant* also does not add any direct value to the question answering process.

The distribution of comment categories *advantage*, *weakness*, *inquiry* and *addition* are very similar between hidden and shown comments. The top category is *addition* in both hidden and shown comments. In this category, developers provide additional information to the associated answers of the comment. Namely, by providing an alternative answer to a question, adding an example, adding an explanation, or adding a reference. An example of an informative comment is shown in Fig. 5.1, in which a user pointed out that the node-inspector did not work any more in the latest Node.js version (category *weakness*). He also provides an alternative in the same comment (category *addition*).

There is no statistically significant difference between hidden and shown comments in terms of the distribution of comment categories. Fig. 5.4 shows the distribution of comment categories for both hidden and shown comments. The result of the Wilcoxon signed-rank test shows the differences between hidden and shown comments are insignificant (i.e., p-value > 0.05). The result of Cliff's delta test is negligible (i.e., 0.14). The comparison of the proportion of informative comments in both hidden and shown comments is shown in Table 5.3. The studied hidden comments share similar information (in terms of the comment categories) compared to shown comments. That being said, **hidden comments are as informative as the shown comments for**

their associated answer.

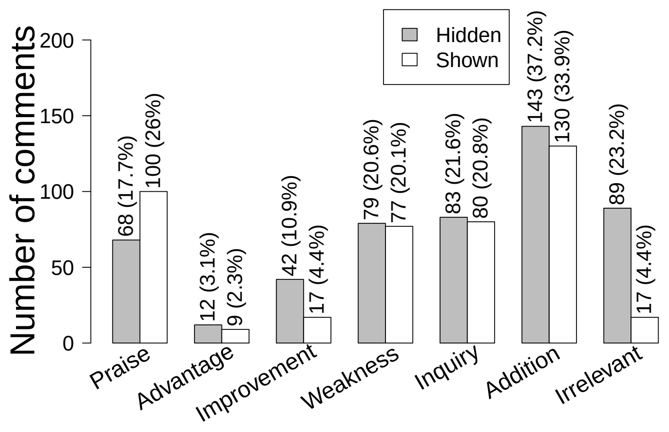


Figure 5.4: The distribution of comment categories.

Table 5.3: Comparison of hidden and shown comments in providing additional value to the associated answers

	Informative	**Not informative**	**Total**
Hidden comment	280 (72.9%)	104 (27.1%)	384
Shown comment	289 (75.3%)	95 (24.7%)	384

We also note that commenters with higher reputation points are more likely to post more informative comments. In the studied hidden comments, the median reputation points of these commenters are 595.5 and 457.5 for the informative and uninformative comments, respectively. In the studied shown comments, the median reputation points of these commenters are 677 and 364 for the informative and uninformative comments, respectively.

Hidden comments are as informative as shown comments. More than half of the comments are informative in both hidden and shown comments.

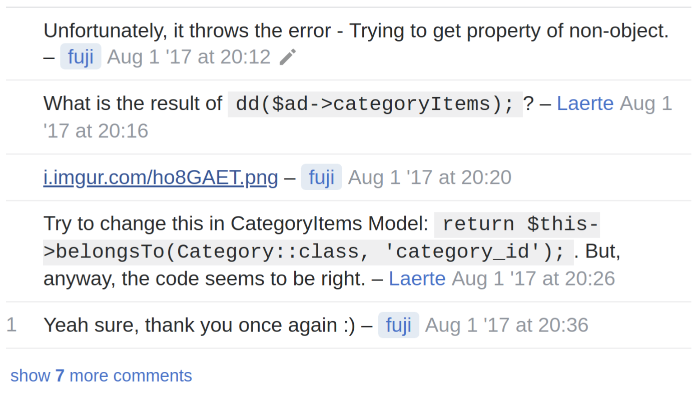
5.3.2 RQ2: How effective is the current comment organization mechanism?

Motivation: In Section 5.3.1, we discover that the majority of the hidden comments are at least as informative as the shown comments. Thus, although the comment organization mechanism has been designed to hide uninformative comments, many informative comments are also hidden.

In addition, developers commonly make use of web search engines, such as Google, to locate online resources to improve their productivity (Xia et al., 2017). However, Google does not index such a large number of hidden comments instead it only indexes the shown comments, which prevents developers from accessing the information in these hidden comments from search engines. Therefore, we focus on studying the current comment organization mechanism, that is, the principle that determines whether a comment should be hidden or shown. By investigating the efficacy of the current comment organization mechanism, we wish to offer deeper insights into enhancing the Stack Overflow commenting system, so that developers can more conveniently and effectively perceive informative discussions through comments.

Stack Overflow's current comment organization mechanism aims at showing comments with higher scores while hiding ones with lower scores. The assumption is that comments with higher scores are more informative than the ones with lower scores. However, in Section 5.3.1 we find that hidden comments are as informative as shown comments, which suggests that the current comment organization mechanism is not working as expected. Therefore, it is important to investigate the reason behind this.

During our manual study, we also notice that this comment organization mechanism may not work well if comments do not have a hierarchy of different scores. For example, if many comments do not get any up-vote (i.e., their scores are 0), apparently, the current comment organization mechanism would not work in such a scenario. More generally, as long as comments have the same score, the comments would not be ranked nor shown based on their scores (i.e., tie-scored comments). As an example, in an answer[15], there are 12 comments with only 1 comment with a non-zero score (i.e., 1) as shown in Fig. 5.5. In such cases, the shown comments may not be more informative than the hidden comments.


Unfortunately, it throws the error - Trying to get property of non-object. – fuji Aug 1 '17 at 20:12

What is the result of `dd($ad->categoryItems);` ? – Laerte Aug 1 '17 at 20:16

i.imgur.com/ho8GAET.png – fuji Aug 1 '17 at 20:20

Try to change this in CategoryItems Model: `return $this->belongsTo(Category::class, 'category_id');` . But, anyway, the code seems to be right. – Laerte Aug 1 '17 at 20:26

1 Yeah sure, thank you once again :) – fuji Aug 1 '17 at 20:36

show 7 more comments

Figure 5.5: An example of an answer where a large proportion (i.e., 11) of the comments under an answer have 0 score and only 1 comment has a score of 1.

In order to study the efficacy of the comment organization mechanism, in this RQ, we investigate how comments are actually ranked and therefore shown.

Approach: Intuitively, an uninformative comment (i.e., category *Praise* and *Irrelevant*) should be hidden by the comment organization mechanism so that another informative comment (such as category *Improvement* and *Weakness*) could be shown under the same answer. The comment organization mechanism is designed for this purpose, i.e., re-arrangement of comments based on their scores. To evaluate the efficacy of the

[15]https://stackoverflow.com/a/45446651/

comment organization mechanism in action, we first characterize the comment scores and analyze how they affect the comment organization mechanism, since the current comment organization mechanism is designed based on the comment score.

More specially, we investigate how the tie-scored comments impact the comment organization mechanism. For this purpose, we make the following definitions. We define that a comment is ***unfairly hidden*** when it is hidden not because it has a lower score than another comment, but because it is posted later than another shown comment with the same score (i.e., *unfairly hidden comments*). In other words, an unfairly hidden comment is hidden because of its later creation time instead of its lower score (note that all such unfairly hidden cases only happen in tie-scored comments based on our definition). We show an example of unfairly hidden comments and unfair comments set in Fig. 5.6. Currently, an unfairly hidden comment occurs in the following situation: for all comments of an answer sorted by score, the score of the sixth comment (i.e., a hidden comment) is equal to the score of the fifth comment (i.e., a shown comment). In this situation, the fifth comment does not need to compete with the sixth comment to be shown by the user interface, it gains its position (as a shown comment) because it is created earlier.

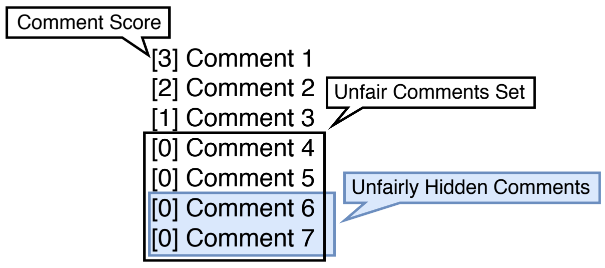


Figure 5.6: An example of an unfair comments set and its unfairly hidden comments. Comment 6 and 7 are unfairly hidden comments since they have the same score as Comment 4 and 5; but were posted at a later date.

Furthermore, we define a set of comments under an answer as an ***unfair comments set***, if there are some hidden and shown comments that have the same score (e.g., a comment with a score of 0 is hidden but another comment with a score of 0 is shown, see Fig. 5.6). We conduct a quantitative study to find out how many answers have unfairly hidden comments. If there were a large proportion of such unfair comments set, it may indicate that the current comment organization mechanism is not working as expected.

Besides the above-mentioned aspects related to comment score, we calculate the proportion of $Answer_{hidden}$ in which their comments are actually ordered and shown by their creation time (i.e., the comment organization mechanism has no effect). By investigating these characteristics, we wish to understand the impact of comment scores and creation times on the current comment organization mechanism.

Results: Due to the widespread existence of tie-scored comments, unfairly hidden comments exist in 97.3% (i.e., 1,268,416 out of 1,303,501) of the $Answer_{hidden}$. Currently, the comment organization mechanism fails to consider tie-scored comments, leading to new comments being hidden while old comments with the same score being shown in almost all $Answer_{hidden}$ (i.e., resulting in the stagnation of showing new comments). Even more, unfairly hidden comments sets have 4,105,956 hidden comments. In other words, 92.9% of all the 4,418,563 hidden comments are actually unfairly hidden (i.e., they are hidden not because of the score or content but the time they are posted). To illustrate the issue, in the same example shown in Fig. 5.5, only 1 comment has a score of 1 while the other 11 comments have a score of 0; therefore, 4 of the 0-score comments are shown simply because they were created earlier than the other

7 comments with 0-score. Therefore, any new comment, even if they are informative, will be automatically hidden. The lack of visibility makes unfairly hidden comments less likely to get any up-voting, and thus are even more likely to remain hidden.

944,950 (i.e., 72.5%) of $Answer_{hidden}$ have unfairly hidden comments with a score of 0. More than half (i.e., 56.5%) of $Answer_{hidden}$ have all of their comments with the same score of 0. In other words, as an upper bound estimation, the comment organization mechanism surprisingly only works as expected at most for 43.5% of $Answer_{hidden}$. Moreover, even in such 43.5% cases, it is not guaranteed that every single comment is ranked based on the score since perhaps only a portion of comments have a score greater than 0. For example, in Fig. 5.5, the answer has 12 comments, and only one of them has a score greater than 0 while the remaining of comments have a score of 0. In this example, the remaining 11 comments are not ranked based on their score anymore. We notice that 87.7% of all the comments under all answers have a score of 0. One possible reason for such a large number of comments that do not have any up-voting is as Calefato et al. (2015) mentioned in their previous study, comments are considered as a "free zone" for users since comments do not generate any reputation point. Thus, users may not be motivated to up-vote comments.

Comments are ranked based on their creation time if their scores are the same. Given the fact that most of $Answer_{hidden}$ have all of their associated comments with the same score, we wish to determine how many $Answer_{hidden}$ have their associated comments actually ordered by the comment creation time.

In 79.4% of $Answer_{hidden}$, comments are ranked and shown by the order of their creation time. In these answers, the result of the comment organization mechanism is equivalent to a queue of comments that are sorted by the creation time of their comments. Namely, only the first 5 oldest comments are shown, and any newer comment will be hidden. In other words, **the current comment organization mechanism gives priority to older comments — promoting stagnation of comments**. As we explained before, one possible reason that a large proportion of $Answer_{hidden}$ are actually shown based on their creation time is the widespread existence of 0-score comments.

Another possible explanation is that older comments tend to get higher scores. Note that the comment age is defined as the time interval between the creation of the comment and its associated answer. Among the 303,035 $Answer_{hidden}$ that have at least 2 comments whose scores are ≥ 1, 187,714 (i.e., 61.9%) have a negative correlation, and 65,205 (21.5%) have at least a moderate negative correlation (correlation < -0.5) (Mukaka, 2012) between comment age and score. Among the 46,935 $Answer_{hidden}$ that have at least 5 comments whose scores are ≥ 1, 36,655 (i.e., 78.1%) have a negative correlation, and 15,525 (i.e., 33.1%) have at least a moderate negative correlation (correlation < -0.5) between comment age and score. Therefore, older comments are more likely to get higher scores.

The current comment organization mechanism does not work effectively. If an answer has hidden comments, it is highly likely (97.3%) that it has unfairly hidden comments. The current mechanism fails to consider the widespread of comments with tie-score, especially 0-score, and gives a higher priority to show older comments.

5.3.3 RQ3: What are the characteristics of unfairly hidden comments?

Motivation: In Section 5.3.2, by examining the age of comments, their score, and their correlation, we find that, in most cases, the current comment organization mechanism actually fails to rank and show comments based on their scores. The current mechanism does not consider tie-scored comments (i.e., comments that have the same score). For example, during our manual study in Section 5.3.1, we observe that some unfairly shown comments are very short and uninformative (e.g., expressing praise) while some unfairly hidden comments are informative. In order to improve the current comment organization mechanism, we investigate the characteristics of shown and hidden comments in the unfair comments set. By understanding this, we can provide insightful suggestions for improving the current comment organization mechanism for Stack Overflow.

Approach: We first investigate the length of unfairly shown comments as a baseline metric to measure how informative they are. If some unfairly shown comments are very short, it is highly likely that they are uninformative. Furthermore, we investigate the informativeness of the shortest unfairly shown comment compared with the longest unfairly hidden comment in the same unfair comments set. The reason that we conduct such comparison is because we probably could provide insights into improving the comment organization mechanism. For example, one simple solution is to replace the shortest unfairly shown comment with the longest unfairly hidden comment if we can show that the longest unfairly hidden comment is more likely to be informative than the shortest unfairly shown comment in the same unfair comments set.

To do so, we randomly select 384 sets of comments in the unfair comments sets, with at least 1 unfairly shown comment with length < 50 and 1 unfairly hidden comment with length >= 50 to achieve a significance level of 95% and a significance interval of 5%. We manually label the comment category using the same qualitative approach in Section 5.3.1, for both the shortest unfairly shown comment and the longest unfairly hidden comment in each sampled set of comments. The Cohen's Kappa value is 0.81 before discussion.

We then perform a qualitative analysis to investigate the comment categories (see Table 5.2) of each unfairly hidden and unfairly shown comment pair to see if an unfairly hidden comment would be more informative than the corresponding unfairly shown comment under the same answer.

Results: In around half (i.e., 46.6%) of the answers that have unfairly hidden comments, the shortest unfairly shown comments have a length that is less than 50 characters. Fig. 5.7 shows the distribution of answers that have unfairly shown comments against different ranges of the length of the shortest comment under the same answers. Through our observation, we find that short comments are usually not informative. For example, a short comment saying "of course ... that's obvious"[16] does not add any information to the associated answer.

More specifically, in answers that have unfairly hidden comments with the shortest unfairly shown comments being less than 50 characters ($L_{ShownMin}$), we pick the longest unfairly hidden comment ($L_{HiddenMax}$) in the same unfair comments set, and calculate the length ratio as $Ratio_{unfair} = L_{HiddenMax}/L_{ShownMin}$. The distribution of $Ratio_{unfair}$ is shown in Fig. 5.8. **In 63.4% of such cases, the length of the longest**

[16] https://stackoverflow.com/posts/comments/56897172/

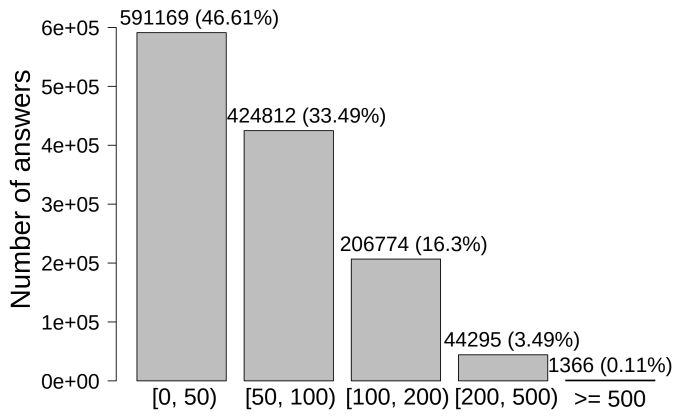


Figure 5.7: The distribution of the shortest unfairly shown comments.

unfairly hidden comment is at least 5 times as long as the length of the shortest unfairly shown comment. Such a high ratio between the longest unfairly hidden comment and the shortest unfairly shown comment of the same answers may indicate that the longest unfairly hidden comment is more informative than the shortest one.

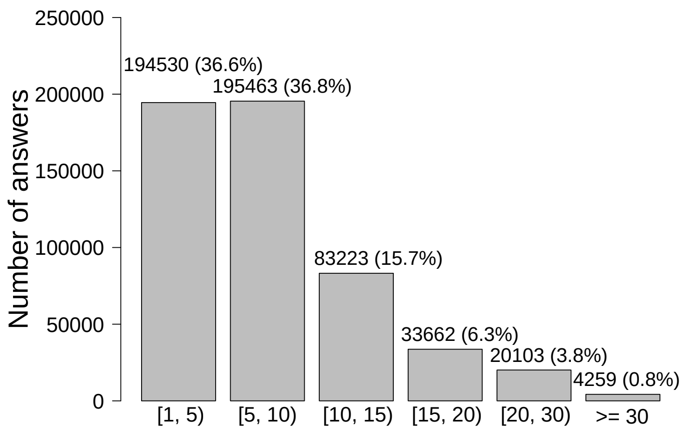


Figure 5.8: The distribution of the length ratio.

In cases where the shortest unfairly shown comment has fewer than 50 characters in the unfairly hidden comments set, the longest unfairly hidden comment is more likely to be informative than the shortest unfairly shown comment. As shown in Fig. 5.9, in the unfairly hidden comments, only 15.9% of comments are related to irrelevant information and praise, while in unfairly shown comments 51.3% are related to irrelevant information and praise. As a result, Stack Overflow could replace such short shown comments with another long hidden comment from the unfair comments set.

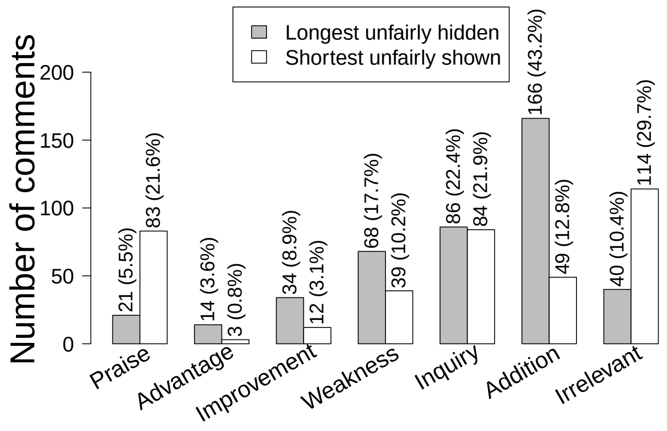


Figure 5.9: The distribution of the categories of comments in both the shortest unfairly shown and the longest unfairly hidden comments.

Discussion: As an exploratory experiment, we inspect how the comment organization mechanism impacts certain informative observations in comments, such as answer obsolescence as we examined in Chapter 3 (Zhang et al., 2019b), security vulnerability, and error message. An example of such a comment[17] says "*works awesome, the only*

[17]https://stackoverflow.com/posts/comments/78752711

thing that i had to change was the onAttach of the fragment, since it has been deprecated". This comment pointed out that the onAttach() function of the Android Fragment class is deprecated; however, this comment is hidden by the comment organization mechanism while none of the currently shown comments bring up this deprecation issue. This motivates us to search among all $Answer_{hidden}$ for comments that mention the word "obsolete" or "outdate", and we find that in such 6,523 comments of answer obsolescence observations, 42.5% are actually hidden. **Furthermore, 85.5% (i.e., 2,370) of the 2,771 hidden comments of answer obsolescence observations are actually unfairly hidden by the comment organization mechanism.** Since software obsolescence are more likely to happen over time, the observation of answer obsolescence tends to happen in newer comments instead of older ones. In Section 5.3.2, we find that in the majority of $Answer_{hidden}$, comments are ranked and shown by their creation time. Therefore, the current comment organization mechanism is much more likely to hide comments that observe answer obsolescence. If these unfairly hidden observations of answer obsolescence could have been replaced by other unfairly shown comments, developers would be more aware of the answer obsolescence issue on Stack Overflow.

In addition, we find similar trends from other informative observations in comments. For example, among comments mentioning the word "vulnerable" (i.e., 1,603), 38.9% are hidden. Among comments mentioning the word "error" (i.e., 566,756), 36.7% are also hidden.

Among all of the above-mentioned observations that are related to answer obsolescence, security vulnerability, or error message, a significant proportion of such observations are buried in hidden comments. Our finding suggests that the comment organization mechanism does not facilitate developers in finding these obvious flaws in answers. Note that in Section 5.3.2, we find that the comment organization mechanism only applies to less than half of $Answer_{hidden}$ in the best-case scenario while in the remaining of $Answer_{hidden}$ all comments (due to all of them having a 0-score) are simply ranked and shown by time. Therefore, the comment organization mechanism may bury other informative observations as well.

In the unfair comments set, the longest unfairly hidden comment is more likely to be informative than the shortest unfairly shown comment, especially if the shortest unfairly shown comment has fewer than 50 characters. As a solution to improve the comment organization mechanism, Stack Overflow can swap these pairs of comments.

5.4 Discussion

5.4.1 Implications for Stack Overflow

Crowdsourced knowledge sharing for developers can be undermined due to tie-scored comments in the current comment organization mechanism on Stack Overflow. The comment organization mechanism is unable to prioritize any comment among tie-scored comments. Therefore, these comments could simply be ranked by their creation time, and any new comment other than the oldest 5 can be hidden. Older comments are more likely to be shown and get attention (e.g., have a higher score), while newer comments are more likely to be hidden. The observation exhibits ***the Matthew effect: the rich get richer and the poor get poorer*** (Merton, 1968). We observe that 87.7% of all the comments under answers are not upvoted at all (i.e., with a score of 0). One possible explanation is that not everyone can upvote a comment since at least 15 reputation points are required to upvote a comment. Note that 20.2% of the users have at least 15 reputation points on Stack Overflow. We manually studied the hidden and shown comments and observed that hidden comments are as informative as shown comments, even though such shown comments are ranked higher than hidden comments in terms of their scores. Namely, the score of a comment may not reflect the usefulness of the comment (especially since many informative comments might be already hidden by the user interface in turn reducing the chances of someone voting on them). Hence ranking comments based on their score is likely to lead to biased observations as they are confounded by the fact that a hidden comment is less likely to get voted on.

In the next subsections, we discuss our actionable suggestions in details and the evaluation of them.

How well can we classify informative comments?

One possible solution is to develop an automated classifier to identify informative comments from uninformative comments in the unfair comments set. Although Stack Overflow allows users to up-vote comments, and comments with a higher score are more likely to be among the top 5 thus shown, the current comment organization mechanism does not effectively reinforce its goal. By using this automated approach, the mechanism could be optimized without massive effort of manual labeling by developers. It could also assist developers in flagging comments as we examined in Chapter 3 (i.e., as a way of bringing inappropriate content to the attention of the community[18], such as labeling unfriendly or unkind comments). Moderators process approximately 1500 flags per day[19]. The classifier could indicate whether a comment is informative from the flagged comments, so he/she can efficiently determine noisy comments for removal and informative comments to keep.

We built a classifier to automatically identify informative comments based on the possible indicators (e.g., the length of a comment) as we observed in RQs. We evaluate our classifier on a dataset of 3,000 manually labeled comments from Chapter 4. Note that since some comments may have multiple sentences, and some sentences are informative while others are not, we further split a comment into different sentences using periods as separators then label each sentence with the comment category defined in Table 5.2. In total, we collect 3,654 sentences. As mentioned above, we consider sentences of category *advantage*, *improvement*, *weakness*, *inquiry*, and *addition* as ***informative*** and sentences of other categories as ***uninformative***. We apply the same text preprocessing approach as in RQ1, with a modification to convert any URL address string into a <URL> identifier and to convert any @user_name string into a <CALL> identifier.

Table 5.4 shows the 13 features that are used in our classifier. Note that we express the textual information of a sentence (i.e., comment text) as an individual feature since we wish further to examine which feature is important for the built classifier. To capture the textual information of each sentence and express it as an individual feature, we first build a random forest classifier using the unigram TF-IDF score (Shihab et al., 2013) for each word in a sentence as a feature and the label of informative or not for

[18]`https://stackoverflow.com/help/privileges/flag-posts`
[19]`https://meta.stackexchange.com/a/166628/`

the sentence. Then we collect the prediction score of each sentence that is outputted by the built random forest classifier as the comment text feature to express the textual information for each sentence (for the detail of this approach, see Shihab et al. (2013)). We use the nltk.sentiment.vader package (NLTK, 2020) to calculate the sentiment score for each sentence. Finally, we combine the score of the comment text with the other 12 features to build a second random forest classifier at the sentence level.

We use out-of-sample validation with a bootstrap of 100 iterations to evaluate the effectiveness of our proposed classifier (Zhou et al., 2020; Wang et al., 2018a). We measure the effectiveness of our classifier using accuracy, precision, recall, and AUC, which are widely used in the machine learning area for evaluating classifiers (Zhou et al., 2020; Wang et al., 2018a). Our classifier achieves 0.86, 0.89, 0.92, and 0.80 in terms of accuracy, precision, recall, and AUC, which demonstrates the effectiveness of our proposed classifier. Note that we consider a comment to be informative if at least one sentence is predicted as informative.

To further understand the important features for our built classifiers, we use the default feature importance calculation technique of random forest classifiers to compute the feature importance. We use the variable importance computation method in the RandomForestClassifier of the scikit-learn package (scikit-learn, 2020). After computing the variable importance value for each feature across 100 bootstrap iterations, the Scott-Knott clustering (Jelihovschi et al., 2014) is applied to group both the feature importance and the word importance. Comment text has a significant power in identifying the informativeness of a comment. Table 5.5 shows the feature importance of all our features. We also show the top words that contribute to the text score of the comment text in Table 5.6.

Table 5.4: Features used in our classifier

Dimension	**Feature**	**Type**	**Explanation**
Text	Text score	Numeric	A score to capture textual information
	Length	Numeric	The number of characters in the text
	Sentiment score	Numeric	The sentiment score of the text
Activity	Comment age	Numeric	The time it took to post the comment since the creation of the answer, measured in days
	Comment score	Numeric	The score of the comment
	In an accepted answer	Boolean	Whether the comment is associated with an accepted answer
User	Commenter reputation	Numeric	The reputation points of the commenter before posting the comment
	By asker	Boolean	Whether the comment is posted by the asker of the associated question thread
	By answerer	Boolean	Whether the comment is posted by the answerer of the associated answer
	By another answerer	Boolean	Whether the comment is posted by another answerer of the associated question thread
	By commenter	Boolean	Whether the comment is posted by a commenter who posted an earlier comment in the associated question thread
	By insider	Boolean	Whether the comment is posted by a user who posted the question, any answer, or any comment in the associated question thread before posting the current comment
	By outsider	Boolean	Whether the comment is posted by a user who never posted the question, any answer, or any comment in the associated question thread before posting the current comment

Table 5.5: Feature importance of the classifier

SK cluster (5%)	**Feature**	**Means**
a	Text score	0.7828
b	Length	0.1037
c	Sentiment score	0.0618
d	Comment age	0.0177
	Commenter reputation	0.0168
e	Comment score	0.0046
	In an accepted answer	0.0038
	By asker	0.0033
	By answerer	0.0015
	By outsider	0.0011
	By insider	0.0011
	By commenter	0.0009
	By another answerer	0.0008

How effective are the comment reorganization mechanisms?

One possible approach to alleviate the issue from the comment organization mechanism is to replace shorter shown comments with longer hidden ones. Another possibility is to randomly show comments in the unfair comments set. Other algorithms focusing on sorting tie-scored unfair comments set can be exploited to effectively hide noisy comments while still retaining informative comments[20]. Stack Overflow can also allow for the down-voting of comments to break the tie-scores. This effort can continue to improve the overall quality of knowledge sharing on Stack Overflow.

In order to explore the effectiveness of these two suggested reorganization mechanisms, we run additional experiments using the cosine similarity to all the 1.3 million $Answer_{hidden}$ with the two reorganization mechanisms:

- the *Random* mechanism: we replace the unfairly shown comments with randomly selected comments from the unfair comments set (i.e., hidden and shown

[20]https://meta.stackexchange.com/q/204402/

Table 5.6: Word importance of the classifier

SK cluster (5%)	**Feature**	**Means**
a	thank	0.0957
b	not	0.0288
	answer	0.0283
	?	0.0271
c	<CALL>	0.0235
	<CODE>	0.0228
d	work	0.0171
e	<URL>	0.0149
f	help	0.0139
	update	0.0131
	good	0.0121
g	use	0.0110
	much	0.0104
	accept	0.0096
h	right	0.0080
	let	0.0077
	but	0.0075
i	edit	0.0071
	great	0.0068
	very	0.0065

comments that have the same score).

- the *Length* mechanism: we replace the unfairly shown comments with the longest comments from the unfair comments set.

Fig. 5.10 shows the distribution of cosine similarity between answer and shown comments in the different settings: the original comment organization mechanism, the *Random* mechanism, and the *Length* mechanism. The Wilcoxon signed-rank test shows a statistically significant difference (p-value < 0.05) both for the original comment organization mechanism and the *Random* mechanism, and for the original comment organization mechanism and the *Length* mechanism. For the *Random* mechanism, the similarity between shown comments and answers is decreased, suggesting that users can retrieve more different information from the shown comments compared with the original comment organization mechanism. However, for the *Length* mechanism, the similarity between shown comments and answers increases. Note that shown comments with a longer text length introduce more vocabulary thus it is more likely to overlap with the vocabulary of answers. Therefore, for the purpose of textual diversity, we recommend the *Random* mechanism instead of the *Length* mechanism.

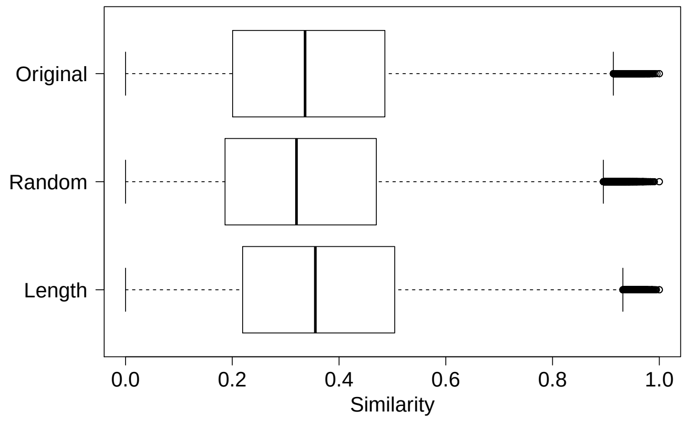


Figure 5.10: The distribution of the cosine similarity between answer and shown comments in the different settings.

Furthermore, to empirically demonstrate the effectiveness of the two suggested comment reorganization mechanisms, we randomly select 10,000 answers that have unfair comments set, and use the built classifier (in Section 5.4.1) to identify the informative comments from all the associated comments with an answer. Then we calculate the proportion of the shown informative comments over all shown comments of an answer in the different settings of organization mechanisms. If the proportion of the shown informative comments improved in the setting of the suggested mechanisms compared to the current mechanism, it suggests the effectiveness of our suggested mechanisms. In Fig. 5.11, we observe that in the *Random* mechanism, the proportion of shown informative comments drops compared with the original comment organization mechanism, while in the *Length* mechanism the proportion of shown informative comments increases compared with the original comment organization mechanism. In the *Length* mechanism, 2,994 (i.e., 30.0%) of the answer threads have an increased proportion of shown informative comments, while in the *Random* mechanism, 1,357 (i.e., 13.6%) of the answer threads have an increased proportion of shown informative comments compared with the original comment organization mechanism. Compared with the original comment organization mechanism, 6,162 (i.e., 61.6%) of the answer threads have a proportion of shown informative comments that remains the same in the *Length* mechanism, while 6,120 (i.e., 61.2%) of the answer threads have a proportion of shown informative comments that remain the same in the *Random* mechanism.

In summary, our explorative analysis shows that the *Random* mechanism is an option to diversify the comments, and the *Length* mechanism enables an improvement to show informative comments. Based on ***the Matthew effect*** (Merton, 1968), every comment can receive an equal chance to be shown in the *Random* mechanism that gives every comment a chance. Namely, each user may be shown with a list of randomized comments (i.e., not generating a randomized list once for all users). Thus, different

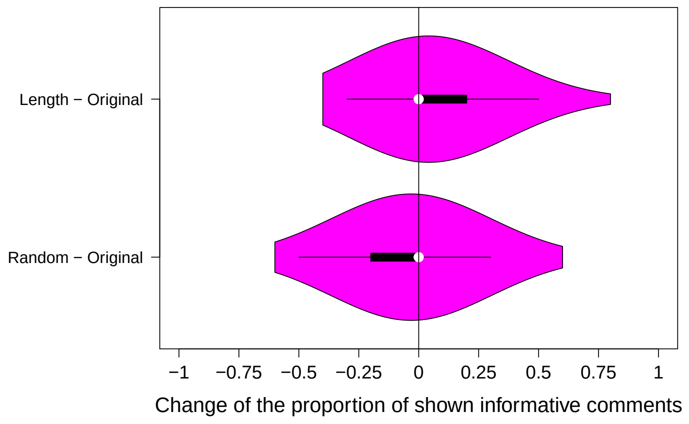


Figure 5.11: The distribution of the cosine similarity between answer and shown comments in the different settings.

users would have different shown comments and we conjecture that the informative comments would be eventually voted up.

5.4.2 Implications for developers

Developers are encouraged to read through all comments (including hidden comments) in case any further corrections are made in such comments, such as observations of answer obsolescence, security vulnerability, and error messages. The value of Stack Overflow answers can change over time even when answerers have not yet noticed the change. Therefore, comments provide another channel to notify a wider audience on Stack Overflow about changes to existing answers. Especially in highly attractive answers, many comments are hidden without taking into account whether they are informative observations to answers. To prevent using an obsolete solution, an insecure code snippet, or a running error, developers are encouraged to read through all comments under an answer before attempting to solve their issues based on the answer, especially more recent ones because they are more likely to be hidden.

5.5 Threats to Validity

5.5.1 External Validity

Threats to external validity are related to the generalization from our study. In this study, we focus on the comment organization mechanism of Stack Overflow, which is the most popular technical Q&A sites in the world. However, our findings and suggestions may not generalize well to other Q&A sites (especially, other sites under Stack Exchange that have the same question, answer, comment layout). Future studies could focus on other Q&A sites since some of these sites (such as Super User, Server Fault, and Ask Ubuntu) contribute significantly to knowledge sharing in their specific domains.

We conduct two qualitative studies in our case study, the first of which investigates what developers discuss in both hidden and shown comments, and the second one explores whether longer unfairly hidden comments are more informative than shorter unfairly shown comments. Since it is impossible for us to manually study all comments in this study, we attempt to minimize the bias by selecting a statistically representative samples of comments with a 95% confidence level and a 5% confidence interval.

Another threat of our study is that only 5.8% of all answers have hidden comments. Even though the proportion of the $Answer_{hidden}$ is low, we observe that there are *1.3 million* of such answers with such answers having a statistically significantly higher score compared with those without hidden comments, suggesting that such answers usually attract more attention from the community. Therefore, we believe that by studying these answers, our findings can still benefit many developers who are using Stack Overflow on a daily basis.

5.5.2 Internal Validity

Threats to internal validity are related to experimental errors. Comment categories are determined by two researchers (with me being one of them) of this study, and later on the informative comment categories are evaluated by the same researchers. To reduce the bias of this process, each comment is labeled by two researchers and discrepancies are discussed until a consensus. We provide the level of the inter-rater agreement in our qualitative analysis, and the values of the agreement are high (i.e., 0.72 and 0.81) in both qualitative studies.

5.5.3 Construct Validity

One threat to construct validity is related to the informativeness of a comment due to revisions of its associated answer. An informative comment can become uninformative after revision, e.g., the comment is integrated back into the associated answer. However, as we examined in Chapter 4, answers are not updated frequently after the posting of a comment – 85.9% of the answers never get updated after the posting of a comment.

Other threats to construct validity are related to our approaches to characterize comments. To compare the degree of information between hidden and shown comments, we first conduct a quantitative analysis. More specifically, we calculate the character length and compute similarity metrics to understand the differences between hidden and shown comments. However, such quantitative measures would not capture all aspects of the differences between hidden and shown comments. Therefore, we also conduct a qualitative study to investigate the difference between hidden and shown comments on top of the quantitative analysis.

5.6 Chapter Summary

Stack Overflow uses a comment organization mechanism where at most 5 comments are shown under each answer. The goal of this mechanism is to improve the compactness of answer threads while retaining the informative comments to facilitate the knowledge sharing process. Among answers with hidden comments, 40.5% of the comments are hidden by the in-use comment organization mechanism.

In this study, we analyzed 1.3 million answers that have hidden comments to understand the impact of the comment organization mechanism on comments. We observed that more than half of hidden and shown comments are informative. In addition, hidden comments are as informative as shown comments, and these hidden comments even add a greater variety of informative content than shown comments to their associated answers.

Furthermore, we evaluated the efficacy of the comment organization mechanism and observed that it fails to show informative comments. Comments are unfairly hidden due to the existence of tie-scored comments (especially 0-score comments). Finally, we discuss possible solutions to improve the comment organization mechanism, such as replacing longer unfairly hidden comments with shorter unfairly shown comments. We build a classifier to distinguish informative comments from uninformative comments with an AUC of 0.8. Our analysis shows that the *Random* mechanism can diversify the information in comments, and the *Length* mechanism enables an improvement to show informative comments.

CHAPTER 6

Conclusion and future work

Question answering activities on Stack Overflow provide valuable information for developers to solve their programming issues and share their knowledge. However, the posted answers along with their additional discussions through commenting evolve as the software technologies evolve. Most prior work considers the acquisition of such knowledge as the end but our work considers it as the actual beginning of crowdsourced knowledge acquisition/sharing. We are the first to consider the evolution of crowdsourced knowledge on Stack Overflow, and to explore the maintenance practices of such knowledge.

This PhD thesis aims to understand the maintenance practices through the mining of Stack Overflow crowdsourced knowledge, e.g., obsolete answers, comments associated with answers, and hidden comments associated with answers. More specifically, we observe that the obsolescence of answers is not actively addressed on Stack Overflow. The majority of comments that are associated with answers enhance their associated answers, although these comments are rarely integrated back into answers. Last but not least, comments that are hidden by the comment organization mechanism can be as informative as shown comments for the purpose of enhancing Stack Overflow answers. Therefore, we propose alternative mechanisms to support the practices of crowdsourced knowledge maintenance. We believe that understanding the crowdsouced knowledge maintenance activities can help practitioners improve the quality of their software engineering knowledge since more and more communities are moving towards a crowdsourced platform for knowledge accumulation and sharing.

6.1 Thesis Contributions

Below, we highlight the main contributions of this thesis.

- **The obsolescence of answers on Stack Overflow**

  We study the obsolete answers on Stack Overflow to understand the maintenance practices of the crowdsourced knowldedge. We perform a qualitative study to understand what happens when an answer is observed as obsolete. We observe that answers to questions that are associated with certain tags are more likely to become obsolete. We also identify the potential reasons for answers to become obsolete. In addition, we identify who observes obsolete answers and what evidence do they provide.

- **The informativeness of comments under answers**

  We mine all the comments that are associated with Stack Overflow answers. We identify the types of discussion in comments and observe that the majority of comments provide useful information to their associated answers. However, we observe that some of the informative comments do not follow the commenting guideline by Stack Overflow. We observe that the majority of commenting activities occur after the acceptance of an answer. Comments that point out the advantage and weakness of answers tend to be posted later. However, the knowledge within comments is rarely integrated back into answers.

- **The retrieval of information in hidden comments**
  Comments can be hidden by the comment organization mechanism on Stack Overflow. We study the characteristics of both hidden and shown comments and observe that hidden comments can be as informative as shown comments. We further evaluate the efficacy of the comment organization mechanism and observe that the mechanism does not work effectively. We build a classifier to distinguish informative comments from the uninformative comments and evaluate the two alternative comment organization mechanisms.

6.2 Future Research

Although the crowdsourced technical knowledge on Stack Overflow is widely leveraged by developers on a daily basis, this thesis highlights that better maintenance of the Stack Overflow knowledge is needed. Below, we propose some potential research opportunities that may benefit the technical knowledge sharing practices.

- **Automated identification of obsolete answers on Stack Overflow**

Prior studies proposed classifiers to identify different aspects of knowledge sharing activities. We observe in Chapter 3 that more than half of the obsolete answers were identified as obsolete within 24 hours of the posting of answers. An automated tool could be developed to identify the likelihood of an answer being obsolete.

- **Flagging of unrecommended uses of comments on Stack Overflow**

  In Chapter 4 we observe that some categories of comments do not follow Stack Overflow's guidelines. These comments can be automatically detected by a classifier and further actions can be suggested to these commenters.

- **Improvement to the current comment organization mechanism**

  We propose alternative comment organization mechanisms and we observe in Chapter 5 that the *Random* mechanism is an option to diversify the comments and the *Length* mechanism enables an improvement to show informative comments. Future studies can extend the effort for enabling better informative retrieval, e.g., through user studies to evaluate how effective are the alternative comment organization mechanisms. In addition, automated approaches to summarize informative comments can be proposed to assist developers in retrieving information in comments.

Bibliography

Abdalkareem, R., Shihab, E., and Rilling, J. (2017). On code reuse from StackOverflow: An exploratory study on android apps. *Information and Software Technology*, 88(C):148–158.

Agichtein, E., Castillo, C., Donato, D., Gionis, A., and Mishne, G. (2008). Finding high-quality content in social media. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, pages 183–194.

Allamanis, M. and Sutton, C. (2013). Why, when, and what: Analyzing Stack Overflow questions by topic, type, and code. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR '13, pages 53–56.

An, L., Mlouki, O., Khomh, F., and Antoniol, G. (2017). Stack overflow: A code laundering platform? In *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering*, SANER '17, pages 283–293. IEEE.

Anderson, A., Huttenlocher, D., Kleinberg, J., and Leskovec, J. (2012). Discovering value from community activity on focused question answering sites: A case study of Stack Overflow. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 850–858, New York, NY, USA. Association for Computing Machinery.

Asaduzzaman, M., Mashiyat, A. S., Roy, C. K., and Schneider, K. A. (2013). Answering questions about unanswered questions of Stack Overflow. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR '13, pages 97–100.

Baltes, S., Dumani, L., Treude, C., and Diehl, S. (2018). Sotorrent: Reconstructing and analyzing the evolution of stack overflow posts. In *Proceedings of the 15th International Conference on Mining Software Repositories*, MSR '18, pages 319–330.

Barua, A., Thomas, S. W., and Hassan, A. E. (2014). What are developers talking about? an analysis of topics and trends in Stack Overflow. *Empirical Software Engineering*, 19(3):619–654.

Benjamini, Y. and Yekutieli, D. (2001). The control of the false discovery rate in multiple testing under dependency. *The Annals of Statistics*, 29(4):1165–1188.

Bennett, K. H. and Rajlich, V. T. (2000). Software maintenance and evolution: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 73–87.

Boslaugh, S. (2012). *Statistics in a nutshell: a desktop quick reference*. O'Reilly Media.

Boslaugh, S. and Watters, P. (2008). *Statistics in a Nutshell: A Desktop Quick Reference*. O'Reilly Media.

Calefato, F., Lanubile, F., Marasciulo, M. C., and Novielli, N. (2015). Mining successful answers in Stack Overflow. In *Proceedings of the 12th Working Conference on Mining Software Repositories*, MSR '15, pages 430–433.

Calefato, F., Lanubile, F., and Novielli, N. (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, pages 13:1–13:10.

Castelle, M. (2018). The linguistic ideologies of deep abusive language classification. In *Proceedings of the 2nd Workshop on Abusive Language Online*, ALW2 '18, pages 160–170, Brussels, Belgium. Association for Computational Linguistics.

Chang, S. and Pal, A. (2013). Routing questions for collaborative answering in community question answering. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '13, pages 494–501.

Chen, C., Xing, Z., and Liu, Y. (2017). By the community & for the community: A deep learning approach to assist collaborative editing in Q&A sites. In *Proceedings of the ACM on Human-Computer Interaction*, volume 1, pages 32:1–32:21.

Chen, C., Xing, Z., and Liu, Y. (2018). By the community & for the community: A deep learning approach to assist collaborative editing in Q&A sites. In *Proceedings of the 21st ACM Conference on Computer-Supported Cooperative Work and Social Computing*, CSCW '18, pages 32:1–32:21. ACM.

Chen, N., Lin, J., Hoi, S. C. H., Xiao, X., and Zhang, B. (2014). AR-miner: Mining informative reviews for developers from mobile app marketplace. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE '14, pages 767–778, New York, NY, USA. ACM.

Chen, T.-H., Thomas, S. W., and Hassan, A. E. (2016). A survey on the use of topic models when mining software repositories. *Empirical Software Engineering*, pages 1843–1919.

Chen, X., Chen, C., Zhang, D., and Xing, Z. (2019). Sethesaurus: Wordnet in software engineering. *IEEE Transactions on Software Engineering*.

Chua, A. Y. and Banerjee, S. (2015). Answers or no answers: Studying question answerability in Stack Overflow. *Journal of Information Science*, 41(5):720–731.

Chujo, K. and Utiyama, M. (2005). Understanding the role of text length, sample size and vocabulary size in determining text coverage. *Reading in a Foreign Language*, 17(1):1–22.

Cliff, N. (1993). Dominance statistics: Ordinal analyses to answer ordinal questions. *Psychological bulletin*, 114(3):494–509.

Dalip, D. H., Gonçalves, M. A., Cristo, M., and Calado, P. (2013). Exploiting user feedback to learn to rank answers in q&a forums: A case study with Stack Overflow. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 543–552.

Duijn, M., Kučera, A., and Bacchelli, A. (2015). Quality questions need quality code:

Classifying code fragments on Stack Overflow. In *Proceedings of the 12th Working Conference on Mining Software Repositories*, MSR '15, pages 410–413.

Fanelli, T. C., Simons, S. C., and Banerjee, S. (2016). A systematic framework for modernizing legacy application systems. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering*, volume 1 of *SANER '13*, pages 678–682.

Gao, Q., Zhang, H., Wang, J., Xiong, Y., Zhang, L., and Mei, H. (2015). Fixing recurring crash bugs via analyzing Q&A sites. In *Proceedings of the 2015 30th International Conference on Automated Software Engineering*, ASE '15, pages 307–318.

Gazan, R. (2010). Microcollaborations in a social Q&A community. *Information Processing & Management*, 46(6):693–702.

Gethers, M., Oliveto, R., Poshyvanyk, D., and Lucia, A. D. (2011). On integrating orthogonal information retrieval methods to improve traceability recovery. In *2011 27th IEEE International Conference on Software Maintenance*, ICSE '11, pages 133–142.

Glassman, E. L., Zhang, T., Hartmann, B., and Kim, M. (2018). Visualizing api usage examples at scale. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 580:1–580:12, New York, NY, USA. ACM.

Gu, X., Zhang, H., and Kim, S. (2018). Deep code search. In *2018 IEEE/ACM 40th International Conference on Software Engineering*, ICSE '18, pages 933–944. IEEE.

Gwet, K. (2002). Inter-rater reliability: dependency on trait prevalence and marginal homogeneity. *Statistical Methods for Inter-Rater Reliability Assessment Series*, 2(1):9.

Hanrahan, B. V., Convertino, G., and Nelson, L. (2012). Modeling problem difficulty and expertise in StackOverflow. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work Companion*, CSCW '12, pages 91–94.

Hoque, E. and Carenini, G. (2014). Convis: A visual text analytic system for exploring blog conversations. In *Proceedings of the 16th Eurographics Conference on Visualization*, EuroVis '14, pages 221–230, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Jelihovschi, E., Faria, J., and Allaman, I. (2014). ScottKnott: a package for performing the Scott-Knott clustering algorithm in R. *TEMA (São Carlos)*, 15:3 – 17.

Kandogan, E. (1998). *Hierarchical multi-window management with elastic layout dynamics*. PhD thesis, University of Michigan.

Li, J., Xing, Z., Ye, D., and Zhao, X. (2016). From discussion to wisdom: web resource recommendation for hyperlinks in stack overflow. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pages 1127–1133.

Liu, X. and Zhong, H. (2018). Mining stackoverflow for program repair. In *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering*, SANER '18, pages 118–129. IEEE.

Luca, P., Gabriele, B., Di Penta, M., Rocco, O., and Michele, L. (2016). Prompter-turning the IDE into a self-confident programming assistant. *Empirical Software Engineering*.

Lv, F., Zhang, H., Lou, J.-g., Wang, S., Zhang, D., and Zhao, J. (2015). CodeHow: effective code search based on API understanding and extended boolean model. In *2015 30th*

IEEE/ACM International Conference on Automated Software Engineering, ASE '15, pages 260–270. IEEE.

Ma, S., Xing, Z., Chen, C., Chen, C., Qu, L., and Li, G. (2019). Easy-to-deploy api extraction by multi-level feature embedding and transfer learning. *IEEE Transactions on Software Engineering*.

Mamykina, L., Manoim, B., Mittal, M., Hripcsak, G., and Hartmann, B. (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, pages 2857–2866, New York, NY, USA. ACM.

McDonnell, T., Ray, B., and Kim, M. (2013). An empirical study of API stability and adoption in the android ecosystem. In *Proceedings of the 2013 IEEE International Conference on Software Maintenance*, ICSM '13, pages 70–79.

Merton, R. K. (1968). The matthew effect in science. *Science*, 159(3810):56–63.

Mujumdar, D., Kallenbach, M., Liu, B., and Hartmann, B. (2011). Crowdsourcing suggestions to programming problems for dynamic web development languages. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '11, pages 1525–1530, New York, NY, USA. ACM.

Mukaka, M. M. (2012). A guide to appropriate use of correlation coefficient in medical research. *Malawi Medical Journal*, 24(3):69–71.

Murgia, A., Janssens, D., Demeyer, S., and Vasilescu, B. (2016). Among the machines:

Human-bot interaction on social Q&A websites. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '16, pages 1272–1279. ACM.

NLTK (2020). Sentiment analysis. https://www.nltk.org/howto/sentiment.html. [Accessed: 02-May-2020].

Oliveto, R., Gethers, M., Poshyvanyk, D., and Lucia, A. D. (2010). On the equivalence of information retrieval methods for automated traceability link recovery. In *2010 IEEE 18th International Conference on Program Comprehension*, ICPC '10, pages 68–71.

Pal, A., Farzan, R., Konstan, J. A., and Kraut, R. E. (2011). *Early Detection of Potential Experts in Question Answering Communities.*

Poché, E., Jha, N., Williams, G., Staten, J., Vesper, M., and Mahmoud, A. (2017). Analyzing user comments on YouTube coding tutorial videos. In *Proceedings of the 25th International Conference on Program Comprehension*, ICPC '17, pages 196–206.

Ponzanelli, L., Bavota, G., Di Penta, M., Oliveto, R., and Lanza, M. (2014a). Mining StackOverflow to turn the IDE into a self-confident programming prompter. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR '14, pages 102–111.

Ponzanelli, L., Mocci, A., Bacchelli, A., and Lanza, M. (2014b). Understanding and classifying the quality of technical forum questions. In *Proceedings of the 2014 14th International Conference on Quality Software*, QSIC '14, pages 343–352.

Ponzanelli, L., Mocci, A., Bacchelli, A., Lanza, M., and Fullerton, D. (2014c). Improving low quality Stack Overflow post detection. In *Proceedings of the 2014 IEEE International Conference on Software Maintenance and Evolution*, ICSME '14, pages 541–544.

Ragkhitwetsagul, C., Krinke, J., and Oliveto, R. (2017). Awareness and experience of developers to outdated and license-violating code on Stack Overflow: An online survey. *RN*, 17(10):10.

Reich, Z. (2011). User comments. In *Participatory Journalism*, chapter 6, pages 96–117. John Wiley & Sons, Ltd.

Sadowski, C., Stolee, K. T., and Elbaum, S. (2015). How developers search for code: A case study. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, FSE '15, pages 191–201.

scikit-learn (2020). Feature importance. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. [Accessed: 02-May-2020].

Seaman, C. B. (1999). Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, 25(4):557–572.

Seaman, C. B., Shull, F., Regardie, M., Elbert, D., Feldmann, R. L., Guo, Y., and Godfrey, S. (2008). Defect categorization: Making use of a decade of widely varying historical data. In *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '08, pages 149–157.

Sengupta, S. and Haythornthwaite, C. (2020). Learning with comments: An analysis of comments and community on stack overflow. In *Proceedings of the 53rd Hawaii International Conference on System Sciences*.

Shihab, E., Ihara, A., Kamei, Y., Ibrahim, W. M., Ohira, M., Adams, B., Hassan, A. E., and Matsumoto, K.-i. (2013). Studying re-opened bugs in open source software. *Empirical Software Engineering*, 18(5):1005–1042.

SocialCompare (2018). Android versions comparison. [Accessed: 09-Oct-2018].

Stack Overflow (2009). Comments: Top n Shown. https://stackoverflow.blog/2009/04/23/comments-top-n-shown/. [Accessed: 02-May-2020].

Stack Overflow (2019). Privileges - comment everywhere. https://stackoverflow.com/help/privileges/comment. Online; accessed May 16 2019.

Storey, M.-A., Singer, L., Cleary, B., Figueira Filho, F., and Zagalsky, A. (2014). The (R) Evolution of social media in software engineering. In *Proceedings of the on Future of Software Engineering*, pages 100–116. ACM.

Tang, M., Luo, X., and Roukos, S. (2002). Active learning for statistical natural language parsing. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 120–127, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Tantithamthavorn, C., Abebe, S. L., Hassan, A. E., Ihara, A., and Matsumoto, K. (2018). The impact of IR-based classifier configuration on the performance and the effort of method-level bug localization. *Information and Software Technology*, 102:160 – 174.

Thung, F., Wang, S., Lo, D., and Lawall, J. L. (2013). Automatic recommendation of API methods from feature requests. In *2013 28th IEEE/ACM International Conference on Automated Software Engineering*, ASE '13, pages 290–300.

Tian, Q., Zhang, P., and Li, B. (2013). Towards predicting the best answers in community-based question-answering services. In *Proceedings of the 7th International Conference on Weblogs and Social Media*, ICWSM '13.

Tian, Y., Thung, F., Sharma, A., and Lo, D. (2017). APIBot: Question answering bot for API documentation. In *Proceedings of the 32Nd IEEE/ACM International Conference on Automated Software Engineering*, ASE '17, pages 153–158, Piscataway, NJ, USA. IEEE Press.

Tran, T. and Cao, T. H. (2013). Automatic detection of outdated information in Wikipedia Infoboxes. *Research in Computing Science*, 70:183–194.

Treude, C., Barzilay, O., and Storey, M. A. (2011). How do programmers ask and answer questions on the web?: NIER track. In *2011 33rd International Conference on Software Engineering*, ICSE '11, pages 804–807.

Treude, C. and Robillard, M. P. (2016). Augmenting API documentation with insights from Stack Overflow. In *Proceedings of the 38th International Conference on Software Engineering*, ICSE '16, pages 392–403.

Upadhyay, U., Valera, I., and Gomez-Rodriguez, M. (2017). Uncovering the dynamics of crowdlearning and the value of knowledge. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 61–70.

Vasilescu, B., Serebrenik, A., Devanbu, P., and Filkov, V. (2014). How social Q&A sites are changing knowledge sharing in open source software communities. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '14, pages 342–354.

Vassallo, C., Panichella, S., Di Penta, M., and Canfora, G. (2014). CODES: Mining source code descriptions from developers discussions. In *Proceedings of the 22Nd International Conference on Program Comprehension*, ICPC '14, pages 106–109.

Viera, A. J., Garrett, J. M., et al. (2005). Understanding interobserver agreement: the kappa statistic. *Fam Med*, 37(5):360–363.

Wang, S., Chen, T.-H., and Hassan, A. E. (2017). Understanding the factors for fast answers in technical Q&A websites. *Empirical Software Engineering*, pages 1–42.

Wang, S., Chen, T. P., and Hassan, A. E. (2018a). How do users revise answers on technical Q&A websites? A case study on Stack Overflow. *IEEE Transactions on Software Engineering*, pages 1–15.

Wang, S. and Lo, D. (2014). Version history, similar report, and structure: Putting them together for improved bug localization. In *Proceedings of the 22nd International Conference on Program Comprehension*, pages 53–63. ACM.

Wang, S., Lo, D., and Jiang, L. (2012). Inferring semantically related software terms and their taxonomy by leveraging collaborative tagging. In *28th IEEE International Conference on Software Maintenance*, ICSM '12, pages 604–607.

Wang, S., Lo, D., and Lawall, J. (2014). Compositional vector space models for improved bug localization. In *30th IEEE International Conference on Software Maintenance and Evolution, Victoria, BC, Canada, September 29 - October 3, 2014*, ICSME '14, pages 171–180.

Wang, S., Lo, D., Vasilescu, B., and Serebrenik, A. (2018b). Entagrec ++: An enhanced tag recommendation system for software information sites. *Empirical Software Engineering*, 23(2):800–832.

Wang, S., Lo, D., Xing, Z., and Jiang, L. (2011). Concern localization using information retrieval: An empirical study on Linux kernel. In *18th Working Conference on Reverse Engineering*, WCRE '11, pages 92–96.

Wong, E., Yang, J., and Tan, L. (2013). AutoComment: Mining question and answer sites for automatic comment generation. In *Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering*, ASE'13, pages 562–567.

Wu, Y., Wang, S., Bezemer, C.-P., and Inoue, K. (2018). How do developers utilize source code from Stack Overflow? *Empirical Software Engineering*.

Xia, X., Bao, L., Lo, D., Kochhar, P. S., Hassan, A. E., and Xing, Z. (2017). What do developers search for on the web? *Empirical Software Engineering*, pages 3149–3185.

Xia, X., Lo, D., Wang, X., and Zhou, B. (2013). Tag recommendation in software information sites. In *Proceedings of the 10th Working Conference on Mining Software Repositories, MSR '13, San Francisco, CA, USA, May 18-19, 2013*, pages 287–296.

Yang, D., Hussain, A., and Lopes, C. V. (2016). From query to usable code: an analysis of stack overflow code snippets. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories*, MSR '16, pages 391–401. IEEE.

Yao, Y., Tong, H., Xie, T., Akoglu, L., Xu, F., and Lu, J. (2013). Want a good answer? Ask a good question first! *arXiv preprint arXiv:1311.6876*.

Yao, Y., Tong, H., Xie, T., Akoglu, L., Xu, F., and Lu, J. (2015). Detecting high-quality posts in community question answering sites. *Information Sciences*, 302:70–82.

Zagalsky, A., Barzilay, O., and Yehudai, A. (2012). Example overflow: Using social media for code recommendation. In *Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering*, RSSE '12, pages 38–42.

Zhang, H., Wang, S., Chen, T.-H., and Hassan, A. E. (2019a). Reading answers on Stack Overflow: Not enough! *IEEE Transactions on Software Engineering*.

Zhang, H., Wang, S., Chen, T.-H. P., Zou, Y., and Hassan, A. E. (2019b). An empirical study of obsolete answers on Stack Overflow. *IEEE Transactions on Software Engineering*.

Zhang, T., Upadhyaya, G., Reinhardt, A., Rajan, H., and Kim, M. (2018). Are code examples on an online Q&A forum reliable? a study of API misuse on Stack Overflow. In *Proceedings of the 40th International Conference on Software Engineering*, ICSE '18, pages 886–896.

Zhou, J. and Walker, R. J. (2016). API deprecation: A retrospective analysis and detection method for code examples on the web. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE '16, pages 266–277, New York, NY, USA. ACM.

Zhou, J., Wang, S., Bezemer, C.-P., Zou, Y., and Hassan, A. E. (2020). Studying the association between bountysource bounties and the issue-addressing likelihood of github issue reports. *IEEE Transactions on Software Engineering*.

Zou, J., Xu, L., Guo, W., Yan, M., Yang, D., and Zhang, X. (2015). Which non-functional requirements do developers focus on? An empirical study on Stack Overflow using topic analysis. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, MSR '15, pages 446–449.