Extracting Archival-Quality Information from Software-Related Chats

Preetha Chatterjee
Department of Computer and Information Science, University of Delaware, USA
Advisor: Lori Pollock
preethac@udel.edu

ABSTRACT

Software developers are increasingly having conversations about software development via online chat services. Many of those chat communications contain valuable information, such as code descriptions, good programming practices, and causes of common errors/exceptions. However, the nature of chat community content is transient, as opposed to the archival nature of other developer communications such as email, bug reports and Q&A forums. As a result, important information and advice are lost over time.

The focus of this dissertation is *Extracting Archival Information from Software-Related Chats*, specifically to (1) automatically identify conversations which contain archival-quality information, (2) accurately reduce the granularity of the information reported as archival information, and (3) conduct a case study to investigate how archival quality information extracted from chats compare to related posts in Q&A forums. Archiving knowledge from developer chats that could be used potentially in several applications such as: creating a new archival mechanism available to a given chat community, augmenting Q&A forums, or facilitating the mining of specific information and improving software maintenance tools.

CCS CONCEPTS

• **Software and its engineering** → **Maintaining software**; • **Information systems** → *Social networking sites*;

KEYWORDS

online software developer chats, archival quality social content

ACM Reference Format:
Preetha Chatterjee. 2020. Extracting Archival-Quality Information from Software-Related Chats. In *42nd International Conference on Software Engineering Companion (ICSE '20 Companion), May 23–29, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3377812.3381391

1 INTRODUCTION

More than ever, software developers are having conversations about software development via online chat services. In particular, developers are turning to public chat communities hosted on services such as Slack, IRC, Gitter, Microsoft Teams, and Freenode to discuss specific programming languages or technologies. Developers use these communities to ask and answer specific development questions, with the aim of improving their own skills and helping others.

Our preliminary study [6, 7] shows that chat communications contain valuable information, such as descriptions of code snippets and specific APIs, good programming practices, and causes of common errors/exceptions. Researchers have demonstrated that various software engineering tasks can be supported by mining similar information from emails and bug reports [5], tutorials [16], and Q&A forums [3, 17, 19]. Thus, availability of all these types of information in software related chats shows promise for mining those information in building and improving software maintenance tools.

The nature of chat community content is transient, as opposed to the archival nature of other developer communications such as email, bug reports and Q&A forums. Developers participate in informal conversations, where information is shared in short messages and in an unstructured manner. Multiple questions are discussed and answered in parallel by different participants. Due to the informal and unstructured nature of the medium, chat conversations also often contain both noise and useful information. As a result, it becomes difficult to find relevant information in a large chat history, and important advice is lost over time. Hence, identifying and preserving useful information from chats in the form of an archive, would serve as a source of knowledge for both software developers and researchers.

Assessing the quality of information is important, so that we can extract useful information when archiving. There has not been much analyses of developer chat communities to assess or improve the quality of the content, however, researchers have focused on assessing the quality of information in Q&A forums beyond built-in mechanisms of the websites [4, 9, 18, 22]. We observed that relative to other developer communications such as Stack Overflow, where quality feedback is explicitly signaled (in forms of accepted answers, vote counts, or duplicate questions), in chats quality feedback is signaled in the flow of the conversation. In developer chat communities, conversations contain mostly textual and emoji clues from other participants to reward good answers. Our **hypotheses** are: (1) Implicit quality indicators (e.g. emojis, textual clues) in chat communities do not insure archival worthiness. The quality indicators in Q&A forums could be adapted to assess the archival-worthiness in chat communications. However, since the structure and format of chat communications vary significantly from Q&A forums, identifying additional quality indicators specific to developer chat communications might be necessary. (2) Text and program

ICSE '20 Companion, May 23–29, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7122-3/20/05…$15.00
https://doi.org/10.1145/3377812.3381391

analysis techniques can be leveraged to automatically identify and curate archival-quality information shared in written developer chat communications.

To evaluate our hypothesis, this dissertation will focus on answering the following research questions (RQs):

- **RQ0:** How much archival-quality information exist in developer chat communications?
- **RQ1:** How accurately can we automatically identify conversations containing archival-quality information?
- **RQ2:** How much can we accurately reduce the granularity of archival-quality information?
- **RQ3:** Case Study: How can extracted archival-quality information from Slack chats compare to related Stack Overflow posts?

My research will focus on Slack as the targeted chat platform due to its increasing popularity [23] and potential of a mining source for software engineering tools [6].

Expected Contributions:

- A technique to automatically disentangle conversations in software developer chats.
- Identification of properties of archival-quality information or quality metrics for developer chat communications.
- An approach based on text processing and machine learning to extract archival-quality information in chats.
- A knowledge archive of conversations in developer chat communications, suitable for research beyond this project.
- A case study to compare the archived information from developer chat conversations to another archival-based software artifact, specifically Q&A forums.

2 BACKGROUND AND RELATED WORK

2.1 Background

The most popular chat communities used by software developers include Slack, IRC, Microsoft Teams, and Flowdock. Slack, with over 8 million daily active users [23], is easily accessible to users as a mobile application as well as a web-based and OS-based application. Public chats in Slack are comprised of multiple communities focused on particular topics such as a technology (e.g., Python or Ruby-on-Rails), with specific channels within a given community assigned to general discussion or to particular subtopics [24]. Within each channel, users participate in chat conversations, or chats, by posting messages. Across all messaging options, users can send text, emojis, and/or multimedia (image and video) messages. Chats in some channels follow a Q&A format, with information seekers posting questions and others providing answers, possibly including code snippets or stack traces.

2.2 Related Work

Recent studies have focused on learning about how chat communities are used by development teams and the usefulness of the conversations for learning about developer behaviors. Shihab et al. [20, 21] analyzed developer Internet Relay Chat (IRC) meeting logs to analyze the content, participants, their contribution and styles of communications. Yu et al. [26] conducted an empirical study to investigate the use of synchronous (IRC) and asynchronous (mailing list) communication mechanisms in global software development projects. Elliott and Scacchi [10] showed that open source communities use IRC channels, email discussions and community digests to mitigate and resolve conflicts. Lin et al. [13] conducted an exploratory study to learn how Slack impacts development team dynamics. Lebeuf et al. [12] investigated how chatbots can help reduce the friction points that software developers face when working collaboratively. Paikari et al. [14] characterized and compared chatbots related to software development in six dimensions (type, direction, guidance, predictability, interaction style, and communication channel). Panichella et al. [15] investigated how collaboration links vary and complement each other by analyzing communication data from mailing lists, issue trackers, and IRC chat logs of seven OSS projects.

Chowdhury and Hindle [8] proposed an approach to automatically filter out off-topic IRC discussions by exploiting Stack Overflow programming discussions and YouTube video comments. Alkadhi et al. [1, 2] conducted exploratory studies to examine the frequency and completeness of available rationale in chat messages, contribution of rationale by developers, and the potential of automatic techniques for rationale extraction. The automation in these analyses was used to learn about behavior, frequency and the potential of machine learning techniques to extract specific types of information, but not to assess their quality to extract only archival worthy information from developer chat communications.

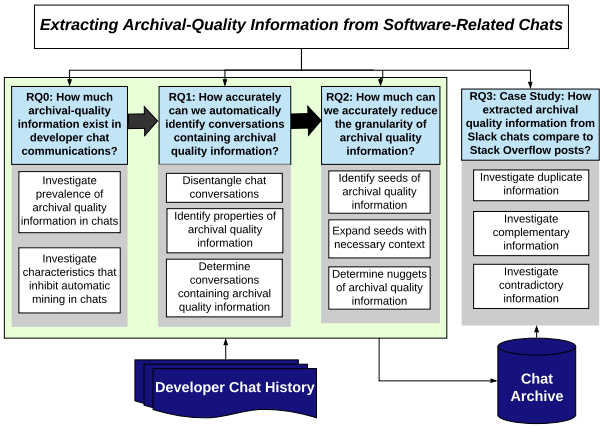

Figure 1: Overview of Dissertation Research Questions and Corresponding Research Problems

3 RESEARCH

This section describes my dissertation research towards archiving information from developer chat communications. Figure 1 illustrates how the individual research projects fit together to form the proposed research program.

3.1 RQ0: How much Archival-Quality Information Exist in Developer Chat Communications?

We conducted an exploratory study [6] to investigate if there is archival information in developer chat communications. We also

investigated the potential usefulness and challenges of mining developer Q&A chat conversations for supporting software maintenance and evolution tools.

Methodology. We designed our study to answer the following study questions (SQs): *SQ1: How prevalent is the information that has been successfully mined from Stack Overflow Q&A forum to support software engineering tools in developer Q&A chats such as Slack? SQ2: Do Slack Q&A chats have characteristics that might inhibit automatic mining of information to support software engineering tools?* To answer SQ1, we focused on information that has been commonly mined in other software artifacts such as code snippets, links to code snippets, API mentions, and bad code snippets. To answer SQ2, we focused on measures that could provide some insights into the form of Slack Q&A conversations (participant count, questions with no answer, answer count) and measures that could indicate challenges in automation (how participants indicate accepted answers, questions with no accepted answer, natural language text describing code snippets, incomplete sentences, noise within a document, and knowledge construction process) that suggest a need to filter.

Findings. The findings of the study indicate that: (1) Much of the information mined from Stack Overflow is also available on Slack Q&A channels. (2) API mentions are available in larger quantities on Slack Q&A channels. (3) Links are rarely available on both Slack and Stack Overflow Q&A. (4) The largest proportion of Slack Q&A conversations discuss software design. (5) Accepted answers are available in chat conversations, but require more effort to discern. (6) Participatory conversations provide additional value but require deeper analysis of conversational context.

3.2 RQ1: How Accurately Can We Automatically Identify Conversations Containing Archival-Quality Information?

To answer this research question, I address two research problems: (1) Disentangle developer chat conversations, and (2) Identify properties of archival-quality information.

3.2.1 Disentangle Developer Chat Conversations. Messages in chats form a stream, with conversations often interleaving such that a single conversation thread is entangled with other conversations, thus requiring techniques to separate, or disentangle, the conversations for analysis.

Methodology. The disentanglement problem has been studied before by researchers in the context of IRC and similar chat platforms [11, 25]. In a recent study [6], we modified the technique proposed by Elsner and Charniak [11] to disentangle conversations on Slack. Specifically we 1) used a significantly larger window of messages, 2) computed the features on the last five messages regardless of elapsed time, and 3) introduced several features specific to Slack, for instance, the use of emoji or code blocks within a message.

Evaluation. The model with the enhancements produced a micro-averaged F-measure of 0.79; a strong improvement over Elsner and Charniak approach's micro-averaged F-measure of 0.57 on disentangling Slack conversations.

3.2.2 Identify Properties Of Archival-Quality Information. Conversations in public chat conversations may vary significantly in terms of archival-quality such as conciseness, readability, and correctness of information. We intuitively define *archival-quality information* as knowledge, which on archiving can serve as a good resource for software engineers and/or mining tools.

Research Strategy. As a first step to assess quality of conversations, we need to identify the properties of archival worthy knowledge in chat forums. Typically, there are no explicit built-in indicators of quality of information shared on chat forums. Hence, I plan to explore how well adaptations of the properties of valuable information in archival-based Q&A forums (specifically Stack Overflow), can identify the properties of archival-quality information in chat forums (specifically Slack). Additionally, I will also explore a data-driven approach of analyzing Slack conversations to understand the characteristics of conversations containing archival-quality information. The properties of archival-quality information thus identified could potentially be used as features to build a machine learning based approach to automatically determine archival worthiness of a conversation.

Evaluation. The first step in evaluation will be to create a gold set of conversations with each conversation assigned a quality score. I plan to recruit human judges with prior experience in programming and using Slack, to participate in a study and create the gold set. Next, the evaluation study will focus on addressing the following evaluation question: *"How effective is my approach in determining conversations that contain archival-quality information?"* Results will be evaluated using precision, recall, and F-measure. These evaluation measures are widely used for tasks in information retrieval and classification.

3.3 RQ2: How Much Can We Accurately Reduce The Granularity Of Archival-Quality Information?

Developer conversations can often contain extensive details, redundant information, and noise, along with archival-quality information. Reading, understanding and reusing the archival-quality information from those conversations therefore becomes arduous and time-consuming. Reducing the granularity of archival-quality information in chats could help in saving time and effort for mining specific information from the archive by both developers and mining tools.

Research Strategy. I plan to answer this research question through a research strategy which involves solving three steps: (1) identify seeds of archival-quality information, (2) expand seeds with necessary context, and (3) construct nuggets of archival-quality information. The first step is to identify archival-quality information seeds, which would be utterances in a conversation that are directly related to an archival property through the structure, content or metadata of the utterance. The second step is to identify utterances related to the archival-quality information seeds that would be highly likely to also be containing archival-quality information, but not directly identifiable without the seed utterance. Finally, the third step is to design an approach to combine the seeds and context to identify archival-quality nuggets of information.

Evaluation. The first step in evaluation will be to create a gold set of developer chats tagged with nuggets of archival-quality information. The next step of the evaluation study will focus on addressing the evaluation question: *"How accurate is my approach in reducing granularity of archival-quality information in developer chats?"*. Precision, recall and F-measure will be used as evaluation measures.

3.4 RQ3: Case Study: How Does Extracted Archival-Quality Information from Slack Chats Compare to Related Stack Overflow Posts?

Borrowing from data triangulation used by qualitative researchers, it is possible to envision a system where software developers' social communication channels serve as the multiple sources of evidence to establish quality of information in each channel. Information in one social communication channel that complements or contradicts information in another channel is identified and used to provide feedback within the social communities. Thus, I plan to conduct a case study to explore this potential by comparing Slack archival quality information with another archival-based software artifact, specifically Q&A forums (e.g. Stack Overflow).

Research Strategy. I plan to answer the following case study questions (CSQ) through a qualitative study: *CSQ1: How much information in Slack and Stack Overflow is duplicate? What kinds of information is duplicate? CSQ2: How much information in Slack and Stack Overflow is complementary? What kinds of information is complementary? CSQ3: Can we determine if information in Slack and Stack Overflow contradict each other?* I plan to investigate an inductive approach to qualitatively analyze the information in the Slack-Stack Overflow comparison dataset. Understanding the different kinds of duplicate, complementary and contradictory information is important to understand the potential of improving the quality of shared information in developer communications.

4 PROPOSED TIMELINE

The author is a fifth year PhD student, who passed her PhD proposal in May 2019. RQ0 was presented at the International Conference on Mining Software Repositories (MSR'19). RQ1 is currently under review at a software engineering conference. She plans to submit RQ2 and RQ3 in prestigious conferences and journals in Software Engineering.

REFERENCES

[1] R. Alkadhi, T. Lata, E. Guzmany, and B. Bruegge. 2017. Rationale in Development Chat Messages: An Exploratory Study. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. 436–446.

[2] R. Alkadhi, M. Nonnenmacher, E. Guzman, and B. Bruegge. 2018. How do developers discuss rationale?. In *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, Vol. 00. 357–369.

[3] Alberto Bacchelli, Luca Ponzanelli, and Michele Lanza. 2012. Harnessing Stack Overflow for the IDE. In *Proc. 3rd Int'l Wksp. on Recommendation Systems for Software Engineering*. 26–30.

[4] Antoaneta Baltadzhieva and Grzegorz Chrupala. 2015. Question Quality in Community Question Answering Forums: A Survey. *SIGKDD Explor. Newsl.* 17, 1 (Sept. 2015), 8–13.

[5] Gerardo Canfora, Massimiliano Di Penta, Rocco Oliveto, and Sebastiano Panichella. 2012. Who is Going to Mentor Newcomers in Open Source Projects?. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering (FSE '12)*. Article 44, 11 pages.

[6] P. Chatterjee, K. Damevski, L. Pollock, V. Augustine, and N.A. Kraft. 2019. Exploratory Study of Slack Q&A Chats as a Mining Source for Software Engineering Tools. In *Proceedings of the 16th International Conference on Mining Software Repositories (MSR'19)*. https://doi.org/10.1109/MSR.2019.00075

[7] P. Chatterjee, M. A. Nishi, K. Damevski, V. Augustine, L. Pollock, and N. A. Kraft. 2017. What information about code snippets is available in different software-related documents? An exploratory study. In *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 382–386. https://doi.org/10.1109/SANER.2017.7884638

[8] S. A. Chowdhury and A. Hindle. 2015. Mining StackOverflow to Filter Out Off-Topic IRC Discussion. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. 422–425.

[9] Denzil Correa and Ashish Sureka. 2013. Fit or Unfit: Analysis and Prediction of 'Closed Questions' on Stack Overflow. In *Proceedings of the First ACM Conference on Online Social Networks (COSN '13)*. 201–212.

[10] Margaret S. Elliott and Walt Scacchi. 2003. Free Software Developers As an Occupational Community: Resolving Conflicts and Fostering Collaboration. In *Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work (GROUP '03)*. 21–30.

[11] Micha Elsner and Eugene Charniak. 2008. You talking to me? A Corpus and Algorithm for Conversation Disentanglement. In *Proc. Association of Computational Linguistics: Human Language Technology*. 834–842.

[12] Carlene Lebeuf, Margaret-Anne D. Storey, and Alexey Zagalsky. 2017. How Software Developers Mitigate Collaboration Friction with Chatbots. *CoRR* abs/1702.07011 (2017).

[13] Bin Lin, Alexey Zagalsky, Margaret-Anne Storey, and Alexander Serebrenik. 2016. Why Developers Are Slacking Off: Understanding How Software Teams Use Slack. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion (CSCW '16 Companion)*. 333–336.

[14] Elahe Paikari and André van der Hoek. 2018. A Framework for Understanding Chatbots and Their Future. In *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE '18)*. 13–16.

[15] S. Panichella, G. Bavota, M. D. Penta, G. Canfora, and G. Antoniol. 2014. How Developers' Collaborations Identified from Different Sources Tell Us about Code Changes. In *2014 IEEE International Conference on Software Maintenance and Evolution*. 251–260.

[16] Gayane Petrosyan, Martin P. Robillard, and Renato De Mori. 2015. Discovering Information Explaining API Types Using Text Classification. In *Proceedings of the 37th International Conference on Software Engineering - Volume 1 (ICSE '15)*. 869–879.

[17] Luca Ponzanelli, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, and Michele Lanza. 2014. Mining StackOverflow to Turn the IDE into a Self-confident Programming Prompter. In *Proc. 11th Working Conf. on Mining Software Repositories*. 102–111.

[18] L. Ponzanelli, A. Mocci, A. Bacchelli, M. Lanza, and D. Fullerton. 2014. Improving Low Quality Stack Overflow Post Detection. In *2014 IEEE International Conference on Software Maintenance and Evolution*. 541–544.

[19] M.M. Rahman, S. Yeasmin, and C.K. Roy. 2014. Towards a context-aware IDE-based meta search engine for recommendation about programming errors and exceptions. In *Proc. IEEE Conf. on Software Maintenance, Reengineering, and Reverse Engineering*. 194–203.

[20] Emad Shihab, Zhen Ming Jiang, and Ahmed E. Hassan. 2009. On the Use of Internet Relay Chat (IRC) Meetings by Developers of the GNOME GTK+ Project. In *Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories (MSR '09)*. 107–110.

[21] E. Shihab, Z. M. Jiang, and A. E. Hassan. 2009. Studying the Use of Developer IRC Meetings in Open Source Projects. In *2009 IEEE International Conference on Software Maintenance*. 147–156.

[22] Jonathan Sillito, Frank Maurer, Seyed Mehdi Nasehi, and Chris Burns. 2012. What Makes a Good Code Example?: A Study of Programming Q&A in StackOverflow. In *Proceedings of the 2012 IEEE International Conference on Software Maintenance (ICSM) (ICSM '12)*. 25–34.

[23] The Statistics Portal Statista. 2018. https://www.statista.com/statistics/652779/worldwide-slack-users-total-vs-paid.

[24] Margaret-Anne Storey, Alexey Zagalsky, Fernando Figueira Filho, Leif Singer, and Daniel M. German. 2017. How Social and Communication Channels Shape and Challenge a Participatory Culture in Software Development. *IEEE Transactions on Software Engineering* 43, 2 (2017).

[25] David C Uthus and David W Aha. 2013. Multiparticipant Chat Analysis: A Survey. *Artificial Intelligence* 199 (2013), 106–121.

[26] Liguo Yu, Srini Ramaswamy, Alok Mishra, and Deepti Mishra. 2011. *Communications in Global Software Development: An Empirical Study Using GTK+ OSS Repository*. Springer Berlin Heidelberg, Berlin, Heidelberg, 218–227.