Software-related Slack Chats with Disentangled Conversations

Preetha Chatterjee*, Kostadin Damevski†, Nicholas A. Kraft‡, Lori Pollock*
* University of Delaware, Newark, DE, USA; *{preethac, pollock}@udel.edu*
† Virginia Commonwealth University, Richmond, VA, USA; *kdamevski@vcu.edu*
‡ Uservoice, Raleigh, NC, USA; *nkraft@gmail.com*

ABSTRACT

More than ever, developers are participating in public chat communities to ask and answer software development questions. With over ten million daily active users, Slack is one of the most popular chat platforms, hosting many active channels focused on software development technologies, e.g., python, react. Prior studies have shown that public Slack chat transcripts contain valuable information, which could provide support for improving automatic software maintenance tools or help researchers understand developer struggles or concerns.

In this paper, we present a dataset of software-related Q&A chat conversations, curated for two years from three open Slack communities (python, clojure, elm). Our dataset consists of 38,955 conversations, 437,893 utterances, contributed by 12,171 users. We also share the code for a customized machine-learning based algorithm that automatically extracts (or disentangles) conversations from the downloaded chat transcripts.

CCS CONCEPTS

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

KEYWORDS

online software developer chats, chat disentanglement

ACM Reference Format:
Preetha Chatterjee*, Kostadin Damevski†, Nicholas A. Kraft‡, Lori Pollock*. 2020. Software-related Slack Chats with Disentangled Conversations. In *17th International Conference on Mining Software Repositories (MSR '20), October 5–6, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3379597.3387493

1 INTRODUCTION

Increasingly, software developers are engaging in conversations via online chat services such as Slack, IRC, Gitter, Microsoft Teams, and Flowdock. Public, open-to-all Slack channels have been created around specific software technologies allowing participants to ask and answer a variety of questions. Our preliminary studies show that such chat communications on Slack contain valuable information, such as descriptions of code snippets and specific APIs, good programming practices, and causes of common errors/exceptions [9, 11]. Availability of these types of information in software-related chats suggests that mining chats could provide similar support for improving software maintenance tools as what researchers have already leveraged from emails and bug reports [7], tutorials [25], and Q&A forums [4, 10, 26, 28].

Different from many other sources of software development-related communication, the information on chat forums is shared in an unstructured, informal, and asynchronous manner. There is no predefined delineation of conversation in chat communications; each conversation could span from two messages to hundreds. Chat conversations are also often interleaved, where multiple questions are discussed and answered in parallel by different participants. Therefore, a technique is required to separate, or *disentangle*, the conversations for analysis by researchers or automatic mining tools.

In this paper, we describe a released dataset of software-related developer chat conversations. A subset of this dataset was analyzed as part of our research in understanding the content of developer chat conversations on publicly available Slack channels [9]. We publish our dataset in XML format, where each XML node represents a chat utterance, containing the anonymized name of the participant, a timestamp, the message text, and an attribute (conversation id) to associate the message with its corresponding conversation. The conversation id is created through a chat disentanglement technique, which is a modified version of Elsner and Charniak's well-known algorithm that better matches the constraints of Slack and the type of software-related Q&A conversations in our corpus [14].

The released conversations are from three programming communities on Slack (python, clojure, elm), gathered over two years (July 2017- June 2019). The overall dataset consists of 38,955 conversations, 437,893 utterances, contributed by 12,171 users. To enable others to process additional Slack transcripts and disentangle them into conversations, we also share the code we used to process daily chat logs, convert them to XML, and extract individual conversations from the collected chat transcripts. Both code and data[1] are openly available to be downloaded for further reuse by the community.

2 BACKGROUND AND RELATED WORK

Background: The most popular chat communities used by software developers include Slack, IRC, Microsoft Teams, and Flowdock. Slack, with over 10 million daily active users [33], is easily accessible to users as a mobile application (Windows, iOS, and Android) as

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

ACM ISBN 978-1-4503-7517-7/20/05…$15.00
https://doi.org/10.1145/3379597.3387493

[1] https://zenodo.org/record/3627124

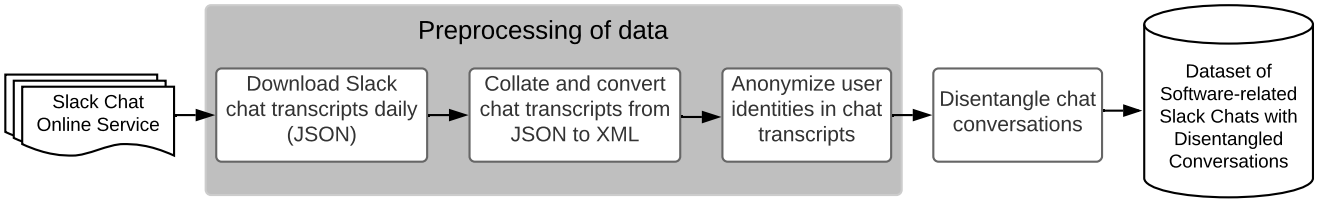


Figure 1: Overview of Data Collection, Preprocessing and Storage of Slack Chats

well as a web-based and OS-based (Windows, Linux, and Mac) 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 [34]. Within each channel, users participate in chat conversations by posting messages, emojis, and/or multimedia (image and video) messages. Conversations in some channels follow a Q&A format, with information seekers posting questions and others providing answers, possibly including code snippets or stack traces. Slack provides easy integration to frequently used developer tools (e.g., Github, Bitbucket, JIRA, and Jenkins) through a set of conversation-based bots and apps [37]. These bots and apps have been widely adopted by many developers for different software engineering tasks such as maintaining code quality, testing, conducting development operations, supporting customers, and creating documentation [18].

Chat Disentanglement Techniques: Most previous research on conversation disentanglement has focused on developing data and models based on chats extracted from IRC channels [14, 15]. Elsner and Charniak's dataset and disentanglement algorithm, extracted from the #Linux IRC channel, has been used for training and evaluation in subsequent disentanglement research [16, 22]. Riou et al. [30] adapted Elsner and Charniak's technique [15] to a French corpus extracted from the Ubuntu platform, while Adams and Martell [1] investigated methods of topic detection and topic thread extraction. Lowe et al. [20, 21] used a heuristic-based approach to extract conversations from the #Ubuntu channel. More recently, Kummerfeld et al. [17] released a manually annotated IRC conversation disentanglement dataset with reply-to relations between messages. To the best of our knowledge, our paper presents the first large-scale dataset of automatically disentangled software related conversations from the Slack platform.

Analysis of Chats: Researchers have studied chat communities to learn about how they are used by development teams and the usefulness of the conversations for understanding developer behaviors. Shihab et al. [32] analyzed developer Internet Relay Chat (IRC) meeting logs to analyze the content, participants, their contribution and styles of communications. Yu et al. [41] conducted an empirical study to investigate the use of synchronous (IRC) and asynchronous (mailing list) communication mechanisms in global software development projects. Lin et al. [19] conducted an exploratory study to learn how Slack impacts development team dynamics. Stray et al. [35] investigated how distributed global development teams use Slack. Panichella et al. [24] investigate collaboration links identified through data from three different kinds of communication channels: mailing lists, issue trackers, and IRC chat logs. Lebeuf et al. [18] investigated how chatbots can help reduce the friction points that software developers face when working collaboratively. Paikari et al. [23] characterized and compared chatbots related to software development in six dimensions (type, direction, guidance, predictability, interaction style, and communication channel). Alkadhi et al. [2, 3] conducted exploratory studies to examine the frequency and completeness of available rationale in chat messages, and the potential of automatic techniques for rationale extraction. In one of our earlier works, we assessed Slack public Q&A chat as a mining source for improving software tools [8, 9].

3 METHODOLOGY

Figure 1 presents an overview of our process for automatic data collection, preprocessing, disentanglement and storage of Slack developer chats. First, we download daily chat transcripts from each Slack channel in JSON format. Second, we collate the daily chat transcripts and convert them into XML format. Next, we anonymize the user identities of the chat participants to preserve privacy, as, otherwise, the Slack user ids can be used to retrieve the participant's e-mail via the channel of origin. Finally, we run a disentanglement algorithm to produce XML attributes that associate identified utterances (i.e., messages) with their corresponding conversations.

3.1 Data Selection

For the purpose of creating a dataset reusable for software developers and maintenance tools, we identified groups that primarily discuss software development topics and have a substantial collection of participants. We selected three programming communities who have active presence on Slack, and were willing to provide us API tokens for download. Within those selected communities, we focused on four channels that follow a Q&A format: pythondev#help, clojurians#clojure, elmlang#beginners, and elmlang#general. The channels are advertised on the Web and allow anyone to join, with a joining process only requiring the participant to create a username (any unique string) and a password. Once joined, on these channels, participants can ask or answer any question, as long as it pertains to the main topic (e.g., programming in Python).

3.2 Data Collection and Preprocessing

Because programmatic access to the data in Slack communities is controlled by the administrators of the Slack team, we contacted several public Slack teams and asked for an API token that would allow us to read and store their data. Public Slack teams typically use Slack's free tier, which only stores the most recent 10,000 messages. Thus, for each Slack community, we downloaded all of the discussion data from each channel incrementally, every day for two years (July 2017- Jun 2019).

```
<message conversation_id = T3610>
  <ts>2018-06-11T11:56:24.000781</ts>
  <user>Harrison</user>
  <text>Hi guys. How can we delete all line breaks from .docx file?
  I'm using python-docx library. In docx - I store some Jinja2 template,
  which later I'm rendering with some data.</text>
</message>
<message conversation_id = T3611>
  <ts>2018-06-11T12:24:04.000597</ts>
  <user>Minna</user>
  <text>Not sure if I should ask here or job_board, I wanted to expand
  my github and use it as a portfolio of sorts, are there certain types
  of projects that are good to have in there to show my comptency?</text>
</message>
...
<message conversation_id = T3611>
  <ts>2018-06-11T12:58:11.000201</ts>
  <user>Raul</user>
  <text><Minna; no one has real time to browse through your repo i would
  think. So if you want a position that uses django/react then do a project
  that does so. If you're trying to get into scraping, do a scraping project
  etc</text>
</message>
<message conversation_id = T3610>
  <ts>2018-06-11T12:58:58.000549</ts>
  <user>Raul</user>
  <text>Harrison: yes just open the file and remove all the line breaks.
  They are essentially the special character `"\n"`</text>
</message>
```

Figure 2: Data Format

The downloaded chats from Slack were in JSON format. We collated all the downloaded chat transcripts and converted them to XML format, in which each message contains a timestamp, the id of the participant, and the message text. During the JSON to XML file conversion, we only use Slack events that correspond to messages, ignoring all other recorded events (e.g., channel joins). In the next step, we obfuscated the participant's ids for privacy, by replacing the original usernames with randomly generated human names.

3.3 Conversation Disentanglement

Since messages in chats form a stream, with conversations often interleaving such that a single conversation thread is entangled with other conversations, a technique is required to separate, or *disentangle*, the conversations for analysis. Figure 2 shows an example of an interwoven conversation in pythondev#help channel on Slack. In this example, a question follows another question, while the answers do not follow a chronological order; the third and the fourth utterances are answers to the second and first questions, respectively. This free form nature of chat communications makes the task of tracing and understanding chat transcripts difficult for automated tools.

The chat disentanglement problem has been studied before in the context of IRC and similar chat platforms [38]. We leveraged the effective technique proposed by Elsner and Charniak [14] that learns a supervised model based on a set of features between pairs of chat messages that occur within a window of time of each other. The features include the elapsed time between the message pair, whether the speaker in the two messages is the same, occurrence of similar words, use of cue words (e.g., hello, hi, yes, no), the use of technical jargon, among others. For the training set, we manually disentangled a set of 500 messages from each Slack channel and trained the model using the combined set.

Table 1: Dataset of Disentangled Slack Conversations

Community	#Conver.	#Utterances	#Partic.
pythondev#help	8,887	106,262	3,295
clojurians#clojure	7,918	72,973	2,422
elmlang#beginners	13,169	168,689	3,695
elmlang#general	8,981	899,69	2,759
Total	**38,955**	**437,893**	**12,171**

After we observed that some Slack channels can become dormant for a few hours at a time and that participants can respond to each other with considerable delay, we modified Elsner and Charniak's algorithm to expand the window of message pairs. Our modification computes features between the current utterance and every utterance that 1) occurred <= 1477 (1.5^{18}) seconds prior to it, or 2) is within the last 5 utterances observed in the channel. We also added to the set of features used by Elsner and Charniak, introducing several specific to Slack, for instance, the use of URLs, Slack channel references, or code blocks within a message. Leveraging the fact that our conversations are mostly Q&A, we added features corresponding to gratitude (e.g., thanks, this works, makes sense), which sometimes occurs at the end of a conversation, when the question is answered satisfactorily. We also followed the procedure prescribed by Elsner and Charniak to create a better set of technical words for the model by extracting all of the words occurring in Stack Overflow documents tagged with a particular tag that do not co-occur in the English Wikibooks corpus. To measure the accuracy of the disentangling process, we manually disentangled separate sets of messages from each of the channels. The model with our enhancements produced a micro-averaged F-measure of 0.80; a strong improvement over the vanilla Elsner and Charniak approach's micro-averaged F-measure of 0.66. Since during the process of creating the gold set of disentangled chat conversations annotators can disagree whether a new conversation branches off from the original or not, micro-averaged F-measure is considered more appropriate than the standard F-measure [14]. With the permision of the original authors, we provide our modified Elsner and Charniak disentanglement code along with the Slack dataset.

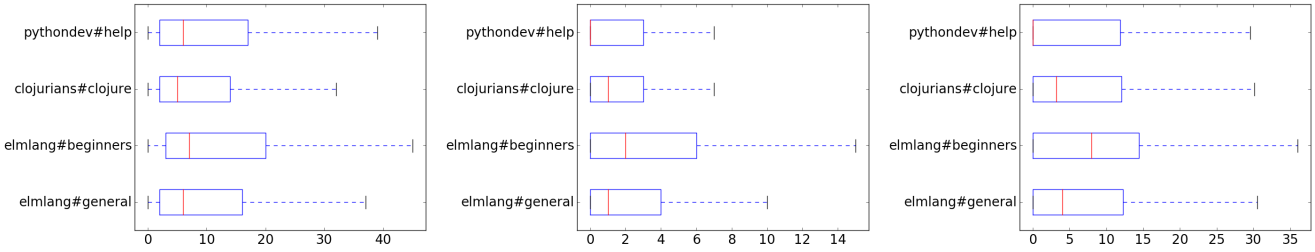
3.4 Data Format

We publish our dataset in an XML format as shown in Figure 2, produced as an output of conversation disentanglement. In the disentangled files, each message has *<message conversation_id>* which is a markup to associate each message with its corresponding conversation id, timestamp *<ts>* in Epoch format, anonymized participant names *<user>*, and the content *<text>* of the message.

4 DATA METRICS

In Table 1, we show the breakdown of number of conversations, utterances, and participants for each of the 4 channels in our dataset. We also computed and report a few additional measures on our dataset that describe its basic characteristics: conversation length, code snippets, and urls. The results are displayed as boxplots in Figure 3.

Conversation length is defined as the number of sentences in a conversation. We computed this measure on the natural language text in each document using the sentence tokenizer from NLTK [6]. *Code snippet count* is computed as the number of code snippets



(a) Conversation length (b) Code Snippet count (c) Code Snippet length

Figure 3: Box Plots of Measures by Community

per conversation, counting both inline and multiline code snippets in a conversation. In Slack, inline code snippets are enclosed in single quotes, whereas multiline code snippets are enclosed in triple quotes. *Code snippet length* is the number of non-whitespace characters in each code snippet.

As shown in Figure 3a, the median conversation lengths for each of the communities are similar, ranging from 5-7 sentences. Figure 3b indicates that *elmlang#beginners* can have larger number of code snippets than the other communities. The median code snippet count in *elmlang#beginners* is 2, whereas the median code snippet count in *elmlang#general* and *clojurians#clojure* is 1. The median code snippet count for *pythondev#help* is zero, probably because sufficient resources about coding in python are already available online. From Figure 3c, we observe that both the median and the variation of code snippet length for *elmlang#beginners* are larger than the rest of the communities. Intuitively, this is because *elmlang#beginners* is for novice programmers who frequently ask and answer more programming-related questions, such as errors and exceptions related to specific code snippets.

5 LIMITATIONS AND EXTENSIONS

Our dataset originates from public Slack channels, focusing on conversations that start with a question followed by a discussion with answers. Thus, the content of our dataset, to some extent, resembles Q&A based forums such as Stack Overflow. If others are interested in datasets that represent team dynamics inside an organization, they would need to augment with private conversations.

We selected the chat transcripts from Slack, which is one of the most popular software developer chat communities. We chose three active programming language communities (4 Slack channels) for our dataset. There is a broad set of topics related to a particular programming language in each channel; however, if others want broader topics represented in their datasets, they will need to broaden the set.

We modified Elsner and Charniak's disentanglement algorithm to account for several features specific to Slack. The code of our modified disentanglement algorithm may need to be adapated to work well on other chat platforms or developer communications. Any changes in disentangled conversations could be handled manually by post processing or by further automation adaptation.

6 RESEARCH OPPORTUNITIES

In our previous study [9], we found that Q&A chats in Slack provide the same information as can be found in Q&A posts on Stack Overflow. Over the years, researchers have mined the knowledge embedded in Q&A forums, such as Stack Overflow, for supporting IDE recommendation [4, 26, 28], learning and recommendation of APIs [12, 27, 39], automatic generation of comments for source code [29, 40], and in building thesauri and knowledge graphs of software-specific terms and commonly-used terms in software engineering [13, 36]. Presence of similar information in Slack Q&A chats suggests that it can serve as a resource for several mining-based software engineering tools.

Developers use Slack to share opinions on best practices, APIs, or tools (e.g., API X has better design or usability than API Y). Q&A forums such as Stack Overflow explicitly forbid posting of questions that ask for opinions or recommendations. However, it is clear that receiving opinions is valuable to software developers. The availability of opinions or recommendations in chats may lead to new mining opportunities for software tools.

We noticed that, along with few links to Stack Overflow and GitHub Gists, there were sporadic links to other sites in our dataset. We believe that embedded links on Slack are used in many different contexts, and as such can be mined to provide more context to other data sources (tutorials, Q&A forums), and thus improve or augment developer learning resources.

Due to its increased popularity, Slack is becoming a popular media to disseminate information between software engineers across the globe. Lin et al. [19] have shown that developers use Slack to discover news/information on technological trends. Our dataset could be studied to identify 'hot' topics of discussion in a programming community [31], and understand common challenges and misconceptions among developers [5]. The results of these studies would provide guidance to future research in developing software support and maintenance tools.

The widespread use of chat communication platforms such as Slack provides a thriving opportunity to build new conversation-based tools and integrations, such as chatbots. Bots have become increasingly prominent due to the ease of their integration with communication tools and accessibility to various APIs and data sources [18]. Sharing chat datasets such as ours could potentially facilitate further research on training and designing chatbots for software development activities [23].

ACKNOWLEDGMENTS

We acknowledge the support of the National Science Foundation under grants 1812968 and 1813253.

REFERENCES

[1] P. H. Adams and C. H. Martell. 2008. Topic Detection and Extraction in Chat. In *2008 IEEE International Conference on Semantic Computing*. 581–588. https://doi.org/10.1109/ICSC.2008.61

[2] 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. https://doi.org/10.1109/MSR.2017.43

[3] 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. https://doi.org/10.1109/SANER.2018.8330223

[4] 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.

[5] Kartik Bajaj, Karthik Pattabiraman, and Ali Mesbah. 2014. Mining Questions Asked by Web Developers. In *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR 2014)*. Association for Computing Machinery, New York, NY, USA, 112?121. https://doi.org/10.1145/2597073.2597083

[6] Steven Bird. 2002. Nltk: The natural language toolkit. In *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*.

[7] 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)*. ACM, New York, NY, USA, Article 44, 11 pages. https://doi.org/10.1145/2393596.2393647

[8] Preetha Chatterjee. 2020. Extracting Archival-Quality Information from Software-Related Chats. In *Proceedings of the 42nd International Conference on Software Engineering*.

[9] 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

[10] Preetha Chatterjee, Minji Kong, and Lori Pollock. 2020. Finding Help with Programming Errors: An Exploratory Study of Novice Software Engineers' Focus in Stack Overflow Posts. *Journal of Systems and Software* 159 (2020), 110454. https://doi.org/10.1016/j.jss.2019.110454

[11] 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

[12] C. Chen, S. Gao, and Z. Xing. 2016. Mining Analogical Libraries in Q&A Discussions — Incorporating Relational and Categorical Knowledge into Word Embedding. In *Proc. IEEE 23rd Int'l Conf. on Software Analysis, Evolution, and Reengineering*. 338–348. https://doi.org/10.1109/SANER.2016.21

[13] Chunyang Chen, Zhenchang Xing, and Ximing Wang. 2017. Unsupervised Software-specific Morphological Forms Inference from Informal Discussions. In *Proc. 39th Int'l Conf. on Software Engineering*. 450–461. https://doi.org/10.1109/ICSE.2017.48

[14] 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.

[15] M. Elsner and E. Charniak. 2010. Disentangling chat. *Computational Linguistics* 36, 3 (2010), 389–409.

[16] Jyun-Yu Jiang, Francine Chen, Yan-Ying Chen, and Wei Wang. 2018. Learning to Disentangle Interleaved Conversational Threads with a Siamese Hierarchical Network and Similarity Ranking. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, 1812–1822. https://doi.org/10.18653/v1/N18-1164

[17] Jonathan K. Kummerfeld, Sai R. Gouravajhala, Joseph J. Peper, Vignesh Athreya, Chulaka Gunasekara, Jatin Ganhotra, Siva Sankalp Patel, Lazaros C Polymenakos, and Walter Lasecki. 2019. A Large-Scale Corpus for Conversation Disentanglement. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 3846–3856. https://doi.org/10.18653/v1/P19-1374

[18] Carlene Lebeuf, Margaret-Anne Storey, and Alexey Zagalsky. 2017. How Software Developers Mitigate Collaboration Friction with Chatbots. In *Proc. 20th ACM Conf. on Computer-Supported Cooperative Work and Social Computing*.

[19] Bin Lin, Alexey Zagalsky, Margaret-Anne Storey, and Alexander Serebrenik. 2016. Why Developers Are Slacking Off: Understanding How Software Teams Use Slack. In *Proc. 19th ACM Conf. on Computer Supported Cooperative Work and Social Computing Companion*. https://doi.org/10.1145/2818052.2869117

[20] Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, Prague, Czech Republic, 285–294. https://doi.org/10.18653/v1/W15-4640

[21] Ryan Thomas Lowe, Nissan Pow, Iulian Serban, Laurent Charlin, Chia-Wei Liu, and Joelle Pineau. 2017. Training End-to-End Dialogue Systems with the Ubuntu Dialogue Corpus. *D&D* 8 (2017), 31–65.

[22] Shikib Mehri and Giuseppe Carenini. 2017. Chat Disentanglement: Identifying Semantic Reply Relationships with Random Forests and Recurrent Neural Networks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Asian Federation of Natural Language Processing, Taipei, Taiwan, 615–623. https://www.aclweb.org/anthology/I17-1062

[23] 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)*. ACM, New York, NY, USA, 13–16. https://doi.org/10.1145/3195836.3195859

[24] 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. https://doi.org/10.1109/ICSME.2014.47

[25] 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)*. IEEE Press, Piscataway, NJ, USA, 869–879. http://dl.acm.org/citation.cfm?id=2818754.2818859

[26] 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. https://doi.org/10.1145/2597073.2597077

[27] M.M. Rahman, C.K. Roy, and D. Lo. 2016. RACK: Automatic API Recommendation Using Crowdsourced Knowledge. In *Proc. IEEE 23rd Int'l Conf. on Software Analysis, Evolution, and Reengineering*. 349–359. https://doi.org/10.1109/SANER.2016.80

[28] 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. https://doi.org/10.1109/CSMR-WCRE.2014.6747170

[29] M. M. Rahman, C. K. Roy, and I. Keivanloo. 2015. Recommending Insightful Comments for Source Code using Crowdsourced Knowledge. In *Proc. IEEE 15th Int'l Working Conf. on Source Code Analysis and Manipulation*. 81–90. https://doi.org/10.1109/SCAM.2015.7335404

[30] Matthieu Riou, Soufian Salim, and Nicolás Borrego Hernández. 2015. Using discursive information to disentangle French language chat.

[31] A. Sharma, Y. Tian, and D. Lo. 2015. What's hot in software engineering Twitter space?. In *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 541–545. https://doi.org/10.1109/ICSM.2015.7332510

[32] 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. https://doi.org/10.1109/ICSM.2009.5306333

[33] The Statistics Portal Statista. 2019. https://www.statista.com/statistics/652779/worldwide-slack-users-total-vs-paid/.

[34] 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). https://doi.org/10.1109/TSE.2016.2584053

[35] Viktoria Stray, Nils Brede Moe, and Mehdi Noroozi. 2019. Slack Me if You Can!: Using Enterprise Social Networking Tools in Virtual Agile Teams. In *Proceedings of the 14th International Conference on Global Software Engineering (ICGSE '19)*. IEEE Press, Piscataway, NJ, USA, 101–111. https://doi.org/10.1109/ICGSE.2019.00031

[36] Y. Tian, D. Lo, and J. Lawall. 2014. Automated construction of a software-specific word similarity database. In *Proc. IEEE Conf. on Software Maintenance, Reengineering, and Reverse Engineering*. 44–53. https://doi.org/10.1109/CSMR-WCRE.2014.6747213

[37] Slack Development Tools. 2018. https://slack.com/apps/category/At0EFRCDNY-developer-tools.

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

[39] W. Wang and M.W. Godfrey. 2013. Detecting API usage obstacles: A study of iOS and Android developer questions. In *Proc. 10th Working Conf. on Mining Software Repositories*. 61–64. https://doi.org/10.1109/MSR.2013.6624006

[40] Edmund Wong, Jinqiu Yang, and Lin Tan. 2013. AutoComment: Mining Question and Answer Sites for Automatic Comment Generation. In *Proc. 28th IEEE/ACM Int'l Conf. on Automated Software Engineering*. 562–567. https://doi.org/10.1109/ASE.2013.6693113

[41] 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. https://doi.org/10.1007/978-3-642-25126-9_32