DISCO: A Dataset of Discord Chat Conversations for Software Engineering Research

Keerthana Muthu Subash
School of Computer Science
Carleton University
Ottawa, Canada
keerthana.muthusubash@carleton.ca

Lakshmi Prasanna Kumar
School of Computer Science
Carleton University
Ottawa, Canada
lakshmi.prasannakumar@carleton.ca

Sri Lakshmi Vadlamani
School of Computer Science
Carleton University
Ottawa, Canada
sri.vadlamani@carleton.ca

Preetha Chatterjee
Department of Computer Science
Drexel University
Philadelphia, PA, United States
preetha.chatterjee@drexel.edu

Olga Baysal
School of Computer Science
Carleton University
Ottawa, Canada
olga.baysal@carleton.ca

ABSTRACT

Today, software developers work on complex and fast-moving projects that often require instant assistance from other domain and subject matter experts. Chat servers such as Discord facilitate live communication and collaboration among developers all over the world. With numerous topics discussed in parallel, mining and analyzing the chat data of these platforms would offer researchers and tool makers opportunities to develop software tools and services such as automated virtual assistants, chat bots, chat summarization techniques, Q&A thesaurus, and more.

In this paper, we propose a dataset called DISCO consisting of the one-year public **DIS**cord chat **CO**nversations of four software development communities. We have collected the chat data of the channels containing general programming Q&A discussions from the four Discord servers, applied a disentanglement technique [13] to extract conversations from the chat transcripts, and performed a manual validation of conversations on a random sample (500 conversations). Our dataset consists of 28, 712 conversations, 1, 508, 093 messages posted by 323, 562 users. As a case study on the dataset, we applied a topic modelling technique for extracting the top five general topics that are most discussed in each Discord channel.

KEYWORDS

Chat conversations, software developers, conversation disentanglement, online communities, Discord.

ACM Reference Format:
Keerthana Muthu Subash, Lakshmi Prasanna Kumar, Sri Lakshmi Vadlamani, Preetha Chatterjee, and Olga Baysal. 2022. DISCO: A Dataset of Discord Chat Conversations for Software Engineering Research. In *Proceedings of 19th International Conference on Mining Software Repositories (MSR'22).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3524842.3528018

MSR'22, May 23–24, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-9303-4/22/05…$15.00
https://doi.org/10.1145/3524842.3528018

1 INTRODUCTION

In recent years, more and more software development communities adopt online chat platforms such as Discord, Slack, IRC, Gitter and Microsoft Teams for more effective collaboration and communication on their projects. These chat platforms serve as a vital resource for getting technical help, sharing knowledge with fellow developers, as well as facilitating real-time conversations among community members. In spite of the wide adoption and benefits of chat platforms for software development communities, research offers only a few studies on mining these chat conversations compared to the studies on mining emails and bug reports [5], tutorials [24], and Q&A forums [1, 10, 26, 29]. Chatterjee et al. [7, 9] have mined and studied Slack chat conversations; their results show that these conversations contain valuable information such as code snippets' description and APIs, bug debugging techniques, best programming practices, and causes of common errors/exceptions. Esteban et al. [23] have studied Gitter data to help new developers get familiar with software products.

To the best of our knowledge, there have not been any studies on Discord server data. As a public chat platform with thousands of users all over the world, we believe that mining Discord conversation would provide numerous research opportunities and, in turn, help software communities. The chat conversations in Discord follows an informal, unstructured and asynchronous format. The conversation length might range from 2 messages to 100s spanning with numerous participating users. The conversations are not always continuous and are entwined with each other. For the researchers to mine and use Discord chat data, the conversations need to be subjected to a technique to separate or *disentangle* them.

In this paper, we present a disentangled dataset of chat conversations obtained from Discord servers of four programming language communities such as *Python*, *Go*, *Clojure*, and *Racket*. We have selected the following general technical help channels: `python#python-general`, `gophers#golang`, `racket$general`, and `clojurians#clojure`. The source chat transcripts from these channels are exported in JSON format and then converted into XML for the date range of November 2019 to October 2020 (12 months). The exception is the `golang` channel with the chat data being exported from November 2019 to September 2020 (11 months) due to the fair

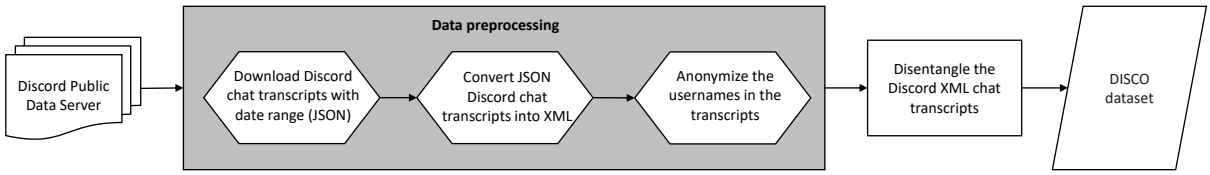

Figure 1: Overview of the data collection and disentanglement process.

dealing policy [36]. Under this policy, researchers are allowed to use upto 10% of the copyrighted data.

This dataset was then subjected to the disentanglement process by applying the modified Elsner and Charniak's algorithm used by Chatterjee et al. [7] for Slack chat mining. The resulting disentangled dataset has each XML node representing a chat message utterance. The message utterance has three tags including *anonymized user name*, a *timestamp*, the *message text*, and a computed *conversation id* to identify the conversation. The DISCO (**DIS**cord **CO**nversations) dataset consists of 28, 712 conversations, 1, 508, 093 utterances, and 323, 562 participants.

As an exploratory case study, we have applied an LDA topic modelling to the original dataset before disentanglement to identify the general chat topics discussed in all the four programming language channels. To facilitate further research on chat data and use of the DISCO dataset, we make the source XML data, JSON to XML conversion script, dataset with disentangled conversations, the modified Elsner and Charniak's algorithm code, and the LDA models publicly available[1].

2 BACKGROUND AND RELATED WORK

Background: Discord, an online communication platform which was originally developed for communication among online gamers has become very popular like Slack, IRC, and Microsoft Teams. In 2021, Discord has over 150 million active monthly users [40]. Users can easily access Discord on their computers, mobile phones, and web browsers for communicating with others using audio/video calls, text messages, files, media, etc. The collection of channels which form communities in Discord are called *servers*, and users can access these channels to post different questions related to a particular topic, and to participate in discussions related to these topics. Chatterjee et al. [7] have extracted 38, 955 conversations from three open Slack communities and performed a disentanglement on these downloaded chat transcripts. Compared to [7] we have extracted 28, 712 conversations from four programming server channels. GitterCom [23] is a dataset which is also based on online communication among developers. It has 10,000 messages collected from ten Gitter communities which are manually annotated to state the purpose of the communication.

Chat Disentanglement Techniques: The task of deciding to which conversation an utterance can be linked to is called *chat disentanglement*. According to Liu et al. [18], chat disentanglement can be divided into two-step methods and end-to-end methods. In two-step methods, the relation between a message pair is identified, and the messages are clustered to obtain the conversation threads according to this relations. In end-to-end methods, a global conversation flow is captured. One of the earliest research in this field by Elsner and Charniak [13] is based on two-step methods. They have also presented a corpus extracted from the IRC (Internet Relay Chat) channel at `freenode.net`. Mehri and Carenini [21] have explored the idea suggested by Elsner and Charniak [14] of using a classifier to predict the upcoming reply messages. Additionally, a RNN is also used to identify if a message falls in a particular thread. Jiang et al. [15] have also leveraged a deep learning based model called Siamese Heirarchical Convolutional Neural Network for finding the similarities between messages to identify the messages which are under the same conversation. Riou et al. [30] have tailored the disentanglement technique by Elsner and Charniak [14] for French language corpus extracted from the IRC channel of French language Ubuntu platform. Lowe et al. [19, 20] have proposed the Ubuntu Dialogue Corpus and performed heuristics-based disentanglement technique on the dataset. Kummerfeld at al. [16] have released a manually annotated disentanglement dataset consisting of more than 70,000 messages of IRC. The messages are annotated with reply-to relations. One of the recent works in disentangling is by Yu and Joty [42] which proposes a pointer module for modelling the interactions between utterances and a joint training framework to capture contextual information. Liu et al. [18] have suggested a deep co-training algorithm for disentanglement with two classifiers such as a message pair classifier and a session classifier. Our paper presents the first disentangled Discord conversations dataset related to software development.

Analysis of Chats: A lot of research has been done on analysis of chat data to understand the topics discussed in the chat, how developers interact, their style of communication, etc. Shi et al. [32] have explored the live chat of developers from eight Gitter communities and offered an understanding of developer communication profiles, community structures, discussion topics, and interaction patterns. Sahar et al. [31] have performed a study of issue reports and the resolution time of issues from 24 open source Gitter project chat rooms. Wang et al. [38] have analyzed the communication style in various Slack channel groups and studied the relation between communication style and team performance. Several researchers [6, 9, 17, 33] have studied Slack data to understand their use in software engineering. Our dataset and case study focus on Discord data and extraction of the general topics discussed on these channels.

3 METHODOLOGY

The overall process of the Discord data collection and conversation disentanglement is presented in Figure 1. The chat transcripts are first downloaded in JSON format from the selected channels using

[1]https://zenodo.org/record/5909202

a date range. They are then cleaned to retain only helpful information such as timestamp, user name, and message content and converted into XML format. This was followed by anonymizing the usernames in XML to ensure the privacy of the users that eliminates the possibility of identifying the original Discord users. The disentanglement algorithm [7] was then leveraged to extract disentangled Discord conversations (in XML format). The final dataset includes an additional computed attribute, `<conversation id>`, as part of the each message utterances.

3.1 Data Selection

We select Discord public server channels as the source in creating the dataset that can support interesting research opportunities and tool development. Our Discord chat data complements Slack data [7] to foster further research on studying distributed software development communities, communication among community and team members, informal documentation, etc. While Slack's free plan supports only 10,000 of the most recent messages to be searched and viewed, Discord does not impose such limitations and preserves all the historical chat data. Hence, many software development communities have started to migrate their communication from Slack to Discord; while some communities continue to maintain both communication mediums [12, 25]. Gitter is another instant messaging and chat platform designed for GitHub and GitLab users where the discussions are happened on specific projects. Since many open-source communities use Discord as their communication platform, it is important that the conversation data from 150 million active Discord users is collected and available to researchers.

For creating our dataset, we select Discord servers for four programming languages such as Python, Go (or GoLang), Racket and Clojure which demonstrate a good daily activity and a substantial number of members (e.g., Python Discord server has a total of 300, 919 members) compared to other available Discord programming servers. Anyone with a Discord user ID can join these servers as they are publicly visible and start asking general or technical help questions on these channels. We identified the following server channels that follow a Q&A format and offer general technical help including `python#python-general`, `gophers#golang`, `racket#general`, and `clojurians#clojure` for our data collection and conversation disentanglement process. To allow triangulation with previous datasets [7], we have selected similar channels.

3.2 Data Collection and Preprocessing

Data from the Discord channels is exported as JSON files using an open-source application, Discord Chat Exporter [35], with a specific date range. The date range for three channels (Python, Clojure, Racket) is from Nov-2019 to Oct-2020, while for `gophers#golang` the date range is Nov-2019 to Sep-2020 due to our University's Fair Dealing Policy in using public copyrighted data for research purposes [36].

The collected Discord chat transcripts in JSON format are then converted to XML files. Each message in the resulting XML has three tags such as a *timestamp*, the *ID of the user* and the *message text*. All other information in the JSON files such as the user-related details, reactions on the messages, etc. is removed during JSON to XML conversation. The user IDs are then anonymized using the randomly selected person names to preserve the privacy of the channel users. Metrics such as the number of conversations, utterances, users, and average conversation length (CL) for each channel are reported in Table 1.

Table 1: Dataset of disentangled Discord conversations.

Channel	#Conver.	#Utter.	#Users	Avg CL
python#python-general	19, 155	1, 254, 362	300, 919	57.49
gophers#golang	8, 860	247, 179	19, 983	27.47
racket#general	538	4, 975	917	8.95
clojurians#clojure	159	1, 577	1, 743	9.99
Total	**28,712**	**1,508,093**	**323,562**	–

3.3 Conversation Disentanglement

In chat servers, the message transcripts are formed by different conversations (both formal and informal) happening simultaneously. Figure 2 illustrates an example of a preprocessed Discord XML file. The presented XML snippet covers two separate conversations entangled with each other. The 2^{nd} question was asked when the 1st conversation was in progress. The 1^{st} conversation was then continued before a relevant reply to the 2^{nd} question was given. This interlinked conversation flow makes it difficult to mine the chat data.

To enable mining of the chat transcripts for researchers and tool makers, we need to disentangle these conversations. The disentanglement techniques have been previously proposed for IRC [37], Gitter [23], and Slack [7]. One of the recent research on Slack data leveraged the well-known Elsner and Charniak disentanglement technique [13] with some modifications. The original Elsner and Charniak disentanglement technique used a supervised model that considers the time frame and features between the message pairs. It also considers the user similarity between the message pairs, cue words, similar word usage, and technical expressions while disentangling the chats. For Slack data, the Elsner and Charniak technique was modified on the feature computation between the message utterances [7].

The features were calculated 1) when the time frame of <= 1477 (1.5^{18}) seconds was observed between the message utterances, or 2) when the utterance was within the last 5 messages from one another. New features that are specific to Slack including gratitude words (e.g., "thanks", "this works", "makes sense") were also added in the modified algorithm. The modified classifier was then trained on 500 manually disentangled Slack conversations.

We have adopted Chatterjee et al.'s disentanglement technique [7] on Slack data for our Discord data since both Slack and Discord channels follow the same type of conversations in Q&A format. To check the accuracy of the disentanglement process, two first authors have selected a random block of 500 Discord messages extracted from the Python channel, manually disentangled these messages into conversations, and calculated a micro-averaged F-score. Our average F-score was 0.79, similar to the one reported by Chatterjee et al. [7] for Slack disentanglement (i.e., F-score of 0.80) which is higher than the F-score of 0.66 reported by Elsner and Charniak. As the annotators can have disagreement over the disentanglement process, micro-averaged F-score can be used as an appropriate metric to calculate the quality of disentaglement [13]. This result further supports our observation that Slack and Discord

Figure 2: Format of conversation data.

follow similar chat conversation patterns where multiple questions are asked and answered simultaneously. This observation also suggests that the same disentanglement algorithm used on Slack could be applied for Discord.

4 CASE STUDY: TOPIC MODELLING

In order to discover the key topics discussed in a software community, researchers have previously used LDA modelling [2, 4, 32]. Similarly, we preformed a case study to identify the general topics that are discussed in the four Discord servers of programming language communities, using a Latent Dirichlet allocation (LDA), an NLP topic modelling technique [3]. We preprocessed the data by removing the stop words and punctuation and then applying lemmatization. We then determined an optimal number of topics for each dataset based on the coherence score as a metric. This metric evaluates the quality of the learned topics by calculating the relative distances between the words appearing in a topic [22]. A high coherence score indicates a high probability that a words pertain to a specific topic. When evaluating the LDA model for each of these four datasets, we have observed that the highest coherence score varied between 0.37 to 0.42 for the topic number, N = 15. This means that the optimal number of topics is 15 for each of these four datasets. Although, the optimal number of topics was 15, we noticed that the distribution of words tend to be repetitive after five (5) topics, thus not providing any meaningful insights. Two authors have then performed a manual labelling of the topics identified by LDA resolving any disagreement. These topics for each of the four datasets are presented in Table 2. While this case study is not offering an extensive analysis of chat topics, it demonstrates a potential for leveraging topic modelling for analyzing Discord conversations and studying developers' communications. Moreover, the initial results from the LDA modelling identify the five topics that are most discussed in each of the four Discord channels.

5 LIMITATIONS AND EXTENSIONS

The Discord data we used is public and consists of conversations in the form of Q&A that offer developers technical help and support. Our dataset does not provide information on team dynamics and interpersonal relationships as the participants are not part of any particular organization. Discord as a real-time communication channel has gained recent popularity among developers in collaborating on projects, exchanging ideas, and getting technical help. The channels we selected cover general technical Q&As for each programming language. If researchers are interested in mining information on specific topics, the dataset can be extended by collecting data for specific channels of interest. Our goal was to share a larger dataset; but since the Discord chat data is copyrighted, we can only collect 10% of the data for research purposes [36]. While the currently shared Discord dataset serves as a starting point, we will be investigating other ways to expand this dataset in future.

Table 2: General discussion topics in each Discord channel.

Dataset	Topics
python#python-general	Basic Python help; Installing and configuring Python packages; Game development; Django and Flask framework tutorials; Python resources and tools.
gophers#golang	General error handling; Golang learning resources; Channels in Go; Installing packages; Dockerizing.
racket#general	Graph plotting, GUI for Racket; Lists recursions and iterations; Syntax or datum libraries; Reference to Racket documentation.
clojurians#clojure	Cron job handling and configuration; Data structures; Channel management; Memory management; Prime number generation.

6 RESEARCH OPPORTUNITIES

Over the years, researchers have mined Q&A forums such as Stack Overflow and chat servers including IRC, Slack, and Gitter to offer insights and recommendations on APIs [28, 39], IDEs [1, 10, 29], automatic comment generation for source code [27, 41], opinion mining [8], and developing software engineering related thesauri and knowledge graphs [11, 34]. Our Discord dataset can be leveraged for similar research lines and topics. In particular, opinion mining is an interesting research topic having potential for developing new tools and applications given the abundance of opinions in chats than any other developer communications. Topic modelling and identifying topics that are prevalent in programming communities is another interesting research opportunity. We have conducted an exploratory case study using the LDA topic modelling technique. Potential extension would be to study the evolution of the topics discussed on these channels. Analyzing the most discussed topics and the problems developers face can help to make efficient support documents such as user manuals and maintenance guides.Another direction would be to use our dataset for designing and evaluating chat bots or new disentanglement techniques. Machine learning models can be trained on the DISCO dataset for summarizing the conversations to help communities document the key topics by extracting summaries from the numerous conversations. This dataset can be also used as an additional source of data to complement existing chat datasets.

Acknowledgements

Muthu Subash, Prasanna Kumar and Baysal acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), RGPIN-2021-03809.

REFERENCES

[1] Alberto Bacchelli, Luca Ponzanelli, and Michele Lanza. 2012. Harnessing stack overflow for the ide. In *2012 Third International Workshop on Recommendation Systems for Software Engineering (RSSE)*. IEEE, 26–30.

[2] Anton Barua, Stephen W. Thomas, and Ahmed E. Hassan. 2014. What are developers talking about? An analysis of topics and trends in Stack Overflow. *Empir. Softw. Eng.* 19, 3 (2014), 619–654. https://doi.org/10.1007/s10664-012-9231-y

[3] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* 3, null (mar 2003), 993–1022.

[4] Joshua Charles Campbell, Abram Hindle, and Eleni Stroulia. 2015. Chapter 6 - Latent Dirichlet Allocation: Extracting Topics from Software Engineering Data. In *The Art and Science of Analyzing Software Data*, Christian Bird, Tim Menzies, and Thomas Zimmermann (Eds.). Morgan Kaufmann, Boston, 139–159. https://doi.org/10.1016/B978-0-12-411519-4.00006-9

[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*. 1–11.

[6] Preetha Chatterjee. 2020. Extracting archival-quality information from software-related chats. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings*. 234–237.

[7] Preetha Chatterjee, Kostadin Damevski, Nicholas A Kraft, and Lori Pollock. 2020. Software-related slack chats with disentangled conversations. In *Proceedings of the 17th International Conference on Mining Software Repositories*. 588–592.

[8] Preetha Chatterjee, Kostadin Damevski, and Lori Pollock. 2021. Automatic extraction of opinion-based Q&A from online developer chats. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 1260–1272.

[9] Preetha Chatterjee, Kostadin Damevski, Lori Pollock, Vinay Augustine, and Nicholas A Kraft. 2019. Exploratory study of slack Q&A chats as a mining source for software engineering tools. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 490–501.

[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.

[11] Chunyang Chen, Zhenchang Xing, and Ximing Wang. 2017. Unsupervised software-specific morphological forms inference from informal discussions. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, 450–461.

[12] Daniel Crowe. 2020. Moving from Slack to Discord. https://medium.com/vaticle/moving-from-slack-to-discord-82a5817cec43.

[13] Micha Elsner and Eugene Charniak. 2008. You talking to me? a corpus and algorithm for conversation disentanglement. In *Proceedings of ACL-08: HLT*. 834–842.

[14] Micha Elsner and Eugene Charniak. 2010. Disentangling chat. *Computational Linguistics* 36, 3 (2010), 389–409.

[15] 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)*. 1812–1822.

[16] Jonathan K Kummerfeld, Sai R Gouravajhala, Joseph Peper, Vignesh Athreya, Chulaka Gunasekara, Jatin Ganhotra, Siva Sankalp Patel, Lazaros Polymenakos, and Walter S Lasecki. 2018. A large-scale corpus for conversation disentanglement. *arXiv preprint arXiv:1810.11118* (2018).

[17] 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*. 333–336.

[18] Hui Liu, Zhan Shi, and Xiaodan Zhu. 2021. Unsupervised Conversation Disentanglement through Co-Training. *arXiv preprint arXiv:2109.03199* (2021).

[19] 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. *arXiv preprint arXiv:1506.08909* (2015).

[20] Ryan Lowe, Nissan Pow, Iulian Vlad Serban, Laurent Charlin, Chia-Wei Liu, and Joelle Pineau. 2017. Training end-to-end dialogue systems with the ubuntu dialogue corpus. *Dialogue & Discourse* 8, 1 (2017), 31–65.

[21] 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)*. 615–623.

[22] David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic Evaluation of Topic Coherence. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (Los Angeles, California) *(HLT '10)*. Association for Computational Linguistics, USA, 100–108.

[23] Esteban Parra, Ashley Ellis, and Sonia Haiduc. 2020. GitterCom: A Dataset of Open Source Developer Communications in Gitter. In *Proceedings of the 17th International Conference on Mining Software Repositories*. 563–567.

[24] Gayane Petrosyan, Martin P Robillard, and Renato De Mori. 2015. Discovering information explaining API types using text classification. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 1. IEEE, 869–879.

[25] Romaric Philogène. 2020. Why we moved from Slack to Discord? https://www.qovery.com/blog/why-we-moved-from-slack-to-discord.

[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 *Proceedings of the 11th Working Conference on Mining Software Repositories*. 102–111.

[27] Mohammad Masudur Rahman, Chanchal K Roy, and Iman Keivanloo. 2015. Recommending insightful comments for source code using crowdsourced knowledge. In *2015 IEEE 15th International Working Conference on Source Code Analysis and Manipulation (SCAM)*. IEEE, 81–90.

[28] Mohammad Masudur Rahman, Chanchal K Roy, and David Lo. 2016. Rack: Automatic api recommendation using crowdsourced knowledge. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Vol. 1. IEEE, 349–359.

[29] Mohammad Masudur Rahman, Shamima Yeasmin, and Chanchal K Roy. 2014. Towards a context-aware IDE-based meta search engine for recommendation about programming errors and exceptions. In *2014 Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*. IEEE, 194–203.

[30] Matthieu Riou, Soufian Salim, and Nicolas Hernandez. 2015. Using discursive information to disentangle French language chat. In *2nd Workshop on Natural Language Processing for Computer-Mediated Communication (NLP4CMC 2015)/Social Media at GSCL Conference 2015*. 23–27.

[31] Hareem Sahar, Abram Hindle, and Cor-Paul Bezemer. 2021. How are issue reports discussed in Gitter chat rooms? *Journal of Systems and Software* 172 (2021), 110852.

[32] Lin Shi, Xiao Chen, Ye Yang, Hanzhi Jiang, Ziyou Jiang, Nan Niu, and Qing Wang. 2021. A first look at developers' live chat on gitter. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 391–403.

[33] Viktoria Stray, Nils Brede Moe, and Mehdi Noroozi. 2019. Slack me if you can! using enterprise social networking tools in virtual agile teams. In *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)*. IEEE, 111–121.

[34] Yuan Tian, David Lo, and Julia Lawall. 2014. Automated construction of a software-specific word similarity database. In *2014 Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*. IEEE, 44–53.

[35] Oleksii Holub (Tyrrrz). 2021. DIscord Chat Exporter - Open Source application. https://github.com/Tyrrrz/DiscordChatExporter

[36] Carleton University. 2022. Carleton University Fair Dealing Policy. https://carleton.ca/secretariat/wp-content/uploads/Fair-Dealing-Policy.pdf.

[37] David C Uthus and David W Aha. 2013. Multiparticipant chat analysis: A survey. *Artificial Intelligence* 199 (2013), 106–121.

[38] Dakuo Wang, Haoyu Wang, Mo Yu, Zahra Ashktorab, and Ming Tan. 2021. Group Chat Ecology in Enterprise Instant Messaging: How Employees Collaborate Through Multi-User Chat Channels on Slack. arXiv:1906.01756 [cs.HC]

[39] Wei Wang and Michael W Godfrey. 2013. Detecting api usage obstacles: A study of ios and android developer questions. In *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, 61–64.

[40] Wikipedia contributors. 2022. Discord (software) — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Discord_(software)&oldid=1066743572 [Online; accessed 21-January-2022].

[41] Edmund Wong, Jinqiu Yang, and Lin Tan. 2013. Autocomment: Mining question and answer sites for automatic comment generation. In *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 562–567.

[42] Tao Yu and Shafiq Joty. 2020. Online conversation disentanglement with pointer networks. *arXiv preprint arXiv:2010.11080* (2020).