Paper#01

Need for Tweet: How Open Source Developers Talk About Their GitHub Work on Twitter

Hongbo Fang, Daniel Klug, Hemank Lamba, James Herbsleb, BogdanVasilescu

Link: program-MSR 2020

 General [social media](https://en.wikipedia.org/wiki/social_media) platforms like Twitter, Facebook, are impacting the professional lives of many [software developer](https://en.wikipedia.org/wiki/software_developer)s, facilitating communication and coordination, learning and knowledge sharing, lack information and activities of the study subjects.  
Among these platforms, Twitter is especially popular with [software developer](https://en.wikipedia.org/wiki/software_developer)s and actively studied by [software engineering](https://en.wikipedia.org/wiki/software_engineering) researchers.

Still missing are studies testing, refining, extending, To enable and encourage such future research, in this paper we propose a computational approach to cross-link users on Twitter and GitHub, the dominant platform for hosting open-source development, revealing 70,427 users with accounts on both. . We find that different developer roles tend to have different tweeting behaviors, with repository owners being perhaps the most distinctive group compared to other project contribut or sand followers

Despite a widespread recognition that activities on open source platforms like GitHub are only a part of the overall software development ecosystem and that open source developers are often active on multipleplat forms simultaneously,there is apaucity of research studying the same populations of developers across platforms

Using this dataset, we report on a [case study](https://en.wikipedia.org/wiki/case_study) of 786 tweets by open-source developers about GitHub work

We analyze tweets containing links to GitHub repositories, combining automatic characterization of tweet authors in terms of their relationship to the GitHub items linked in their tweets with [qualitative analysis](https://en.wikipedia.org/wiki/qualitative_analysis) of the tweet contents.

The following six major themes emerged from our qualitative analysis.

Question: These tweets ask about technical details of a repository, or about an open issue.Answer: These tweets respond to questions by other Twitter users.They may explain technical details of the linked repository,or point other store sources on GitHub.Call for Action: The authors of these tweets actively seek or request actions from others.This includes collective actions like staringa repository,helping to solve anissue, or requesting someone to merge a pull request. Most of those tweets link to either an issue or are pository home page.Repository Advertisemen: Tweets advertising or advocating for the use of the linked repository. Most such tweets link to are pository home page.Work Discussion: Tweets which are part of a discussion on Twitter,but are not a suggestion to use a certain repository. This usually includes discussing the solution of a technical problem,or how to implement certain features.Most of those tweets link to issues,repository home pages,or files.Information Sharing : Tweets which inform the public or specific individuals about the existence of are pository,a file,an issue discussion,or other work-related news,but with no explicit intention to advocate for the use.Tweets in this category are mostly not part of a Twitter conversation.Most link to are pository,release page,or as pecific commit or pull request.Our results are partially aligned with Yasir et al. who found that Python core contribut or mostly tweet about software related (corresponding to Work Discussion, Question, and Answer) and community-related topics(Sharing Information); the main difference between our results and theirs is the bulk of tweets on repository advertisement, which may be explained by the fact that Python is a well-knownlanguage which needs less advertisement. Info share or advocating for the use of the linked repository. Most such tweets link to a repository homepage. Work Discussion .We hypothesize that people knew about those repositories from other places, and posted about them on Twitter to further inform othersThis suggests that there are channels where people can learn about repositories without engaging in their development or actively following their developers.Cross-linking data across online platforms where open source developers participate is feasible and it can help provide deeper insights into how the activities of developers on one platform are moderated by the roles they play on the other.We provide a large dataset mapping 70,427 open-source developer GitHub and Twitter accounts, as basis for future empirical research in these direcsion.

Paper#02

Software-related Slack Chats with Disentangled Conversations

Preetha Chatterjee , Kostadin Damevski , Nicholas A. Kraft , Lori Pollock

Link: program-MSR 2020

More than ever, developers are participating in public chat communities to ask and answers of are 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

Software developers are engaging in conversations via online chat services such as Slack, IRC, Gitter, Microsoft Teams, and Flowdock.

Software-related Slack Chats with Disentangled Conversations.

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 .

Different from many other sources of software development related communication,the information on chat for ums is shared in an unstructured, informal, and asynchronous manner

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 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 .

Background: The most popular chat communities used by software developers include Slack, IRC, Microsoft Teams, and Flowdock.

Figure 1 presents an overview of our process for automatic data collection, preprocessing, disentanglement and storage of Slack developer chats.

We download daily chat transcripts from each Slack channel in JSON format.

For each Slack community, we downloaded all of the discussion data from each channel incrementally, every day for two years (July 2017- Jun 2019).

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.

Our dataset originates from public Slack channels, focusing on conversations that start with a question followed by a discussion with answers.

The code of our modified disentanglement algorithm may need to be adapated to work well on other chat platforms or developer communications.

Researchers have mined the knowledge embedded in Q&A forums, such as Stack Overflow, for supporting IDE recommendation , learning and recommendation of APIs , automatic generation of comments for source code , and in building thesauri and knowledge graphs of software-specific terms and commonly-used terms in software engineering.

Presence of similar information in Slack Q&A chats suggests that it can serve as a resource for several mining-based software engineering tools.

The widespread use of chat communication platforms such as Slack provides a thriving opportunity to build new conversationbased tools and integrations, such as chatbots.

Paper#03

Detecting Video Game-Specific Bad Smells in Unity Projects

Antonio Borrelli, vittoria Nardone, Giuseppe A. Di Lussa, Gerardo Canfora, Massimiliano Di Penta

Link ; MSR 2020

This paper proposes UnityLinter ,a static analysis tool that supports Unity video game developers to detect seven types of bad smells we have identified as relevant in videogame development. Such smell types pertain to performance, maintainability and incorrect behavior problems. Video games represent a conspicuous and increasing share of the [software development](https://en.wikipedia.org/wiki/software_development) market.While the video [game market](https://en.wikipedia.org/wiki/Game_Market) is increasing, development skills in this area still represent a niche.Just to give an idea, Stack Overflow features over 1.5M discussions tagged [java] and 1.2M tagged Android, while only 50k are about Unity3DIt is clear how in this context developers may need suitable support while creating their video games, helping them to avoid introducing performance bottlenecks, or making the game difficult to maintain and evolve.A scene contains a camera (Main Camera), which is the perspective from which the user sees the scene, a light (Directional Light), and a Cube objectWe apply UnityLinter on a set of projects, to (i) validate its accuracy, and (ii) study the magnitude of the investigated smells.As for the precision, we extracted a stratified random sample of 359 smells among the 5,461 detected one, where strata are computed based on the proportions of different smell types.This sample ensures a ±5% margin of error with a confidence level of 95%.Since one smell (Getting a GameObject reference finding it by name) has a fairly limited number of instances (61), reaching a number of 420 smells to be manually analyzed in total This paper described UnityLinter, a static analyzer capable of identifying 7 types of bad smells in Unity projects.Such smells cover various aspects of [video game development](https://en.wikipedia.org/wiki/video_game_development), including performance issues (e.g., use of Instantiate and Destroy in Update() methods without relying on object pools), maintainability issues, and behavioral issues.We reported and discussed the diffuseness of Unity smells on 100 open source video games, showing how the studied smells practice affect a relatively large proportion of the analyzed projects, i.e., between 39% and 97%.