Noname manuscript No.
(will be inserted by the editor)

How are discussions linked? A link analysis study on GitHub Discussions

Márcia Lima* · **Igor Steinmacher** · **Denae Ford** · **Grace Vorreuter** · **Ludmila Gonçalves** · **Tayana Conte** · **Bruno Gadelha**

Received: date / Accepted: date

Abstract Software development requires collaborative efforts and consensus among developers, emphasizing the need for effective communication and knowledge sharing within the teams. In line with this, GitHub introduced GitHub Discussions, a collaborative communication feature for the community around an open-source or internal project. The user-friendly interface of Discussions facilitates easy sharing of links to different resources. Maintainers highlight this feature's significance, enabling users to maintain a well-informed project environment. We hypothesize that link-sharing activities in Discussions contribute to disseminating project knowledge. To investigate this hypothesis, we conducted a mixed-method study combining qualitative and quantitative analysis based on a convenience sample of ten open-source projects.

*Corresponding Author: Márcia Lima
Federal University of Amazonas (UFAM) & Amazonas State University (UEA)
Tel.: +55 92 3305-1181
Present address: Av. Rodrigo Otavio - 6200. Manaus-AM-Brazil
E-mail: marcia.lima@icomp.ufam.edu.br;msllima@uea.edu.br

Igor Steinmacher
Northern Arizona University (NAU)
E-mail: igor.steinmacher@nau.edu

Denae Ford
Microsoft Research
E-mail: denae@microsoft.com

Grace Vorreuter
GitHub
E-mail: gracevor@github.com

Ludimila Gonçalves
Bemol Digital & Federal University of Amazonas (UFAM)
E-mail: ludimilagoncalves@bemol.com.br

Tayana Conte & Bruno Gadelha
Federal University of Amazonas (UFAM)
E-mail: tayana;bruno@icomp.ufam.edu.br

We aimed to gain insight into the scope and intentions behind these shared links. Our findings indicate that link-sharing activities are common in the Discussions. Users share links to resources directly or indirectly related to the project/repository. Discussions users share links to project documentation, source code, and issues, aiming to clarify concepts, provide supplementary information, offer context to questions, and suggest new features. These findings offer insights for project maintainers to understand Discussions usage better and enable the GitHub Engineering team to promote the feature adoption.

Keywords GitHub Discussions · Link Analysis · OSS communities · Project knowledge · Software Engineering forum

1 Introduction

Global software development teams use different social tools and channels for communication purposes (Storey et al., 2016). These resources enable software teams to collaborate and coordinate tasks, deliver results, connect with other developers, and improve skills (Stray and Moe, 2020; Storey et al., 2014). Code development platforms, Q&A sites, microblogs, chats, feeds and blogs, forums, and modern news aggregators are significant social resources developers use in their work (Aniche et al., 2018; Storey et al., 2016). Mainly, online developer forums are popular for practitioners to collaborate on software development-related issues (Li et al., 2021). These forums are rich sources of information and play a relevant role in software development knowledge sharing (Li et al., 2021; Ahmed and Srivastava, 2017; Barua et al., 2014). Developers count on such platforms (e.g., `Stack Overflow`[1] and `Dev.to`[2]) to figure out how to address technical issues quickly and to share knowledge and expertise regarding bugs, languages, tools, frameworks, libraries, and new technologies (Zhang et al., 2017; Mamykina et al., 2011; Barua et al., 2014).

GitHub Discussions was introduced in 2020 to facilitate collaborative communication for communities involved in open-source software (OSS) or internal projects (Niyogi, 2020). This feature allows maintainers, contributors, and other community members to engage in discussions within a project's repository or at the organizational level. Discussions serve as a central hub for sharing announcements, information, and updates, as well as for asking and answering questions, planning, making decisions, and collecting feedback (Hata et al., 2022; Liu, 2022). It provides a space for creating polls, upvoting discussions and comments, and fostering a welcoming environment for discussions related to project goals, development, administration, and workflows (GitHub, 2021). The feature enhances communication within a project's community and reduces reliance on third-party tools for collaboration (Lima et al., 2023; Hata et al., 2022). On the GitHub platform, each project (a specific GitHub repository) may have its forum, leading to different instances of the Discussions

[1] https://stackoverflow.com/
[2] https://dev.to/

feature, which we refer to as GitHub Discussions forums. In the context of this research, we use the term "community members" to encompass all organization members, collaborators, contributors, repository maintainers, teams, users, and visitors concerning a specific repository inside GitHub. [3]

In addition to the aforementioned advantages, maintainers emphasize the ease with which Discussions allow users to add hyperlinks (or simply "links") between issues and discussions, *"allowing everyone to keep track of the history"* (Liu, 2021). Link-sharing activities are important as previous work has demonstrated its value in GitHub Pull Requests (PRs) and Issues (Chopra et al., 2021; Zhang et al., 2020; Li et al., 2018; Zampetti et al., 2017). Developers share links in Issues and PRs to document PRs (Zampetti et al., 2017), link relevant issues/PRs together (Zhang et al., 2020), and identify information across projects (Li et al., 2018). However, to the best of our knowledge, the investigation of link-sharing activities in GitHub Discussions, considering the different types of linked sources and the users' intentions behind the shared links, lacks investigation. Indeed, GitHub Discussions users (GitHub users [3] who engage in any Discussions forums) share explicit links in discussion threads (Hata et al., 2022). In this sense, we hypothesize that link-sharing activities in Discussions help establish a common understanding and support the project knowledge dissemination.

This work presents an exploratory study about link-sharing activities on GitHub Discussions. We investigate the role of links from the perspective of their purpose and intentions. Specifically, we intend to answer the following Research Questions (RQs):

- **RQ1: How are discussion threads linked to other resources?** This research question aims to investigate the type of linked target resource and determine the extent of the project knowledge dissemination through the shared links. To answer RQ1, we examine:
  - **RQ1.1:** How often do link-sharing activities occur in the Discussions forums?
  - **RQ1.2:** What is the proportion of internal and external GitHub links?
  - **RQ1.3:** What are the types of linked resources?
- **RQ2: What are the users' intentions behind the links shared in the discussion threads?** Besides investigating the type of referred resources, we want to know why Discussions users explicitly share references in discussion threads.

We conducted mixed-method analysis, encompassing both quantitative and qualitative analyses of link references shared on GitHub Discussions forums to answer these research questions. In the quantitative analysis, we investigated the frequency with which 10 OSS communities share links in discussion threads and the types of target resources. We extracted 65,621 links from 29,251 public discussion threads. In the qualitative analysis, we conducted a thematic

[3] https://docs.github.com/en/get-started/quickstart/github-glossary

analysis on 1,106 links selected from the `Next.js` project to investigate the reasons for link sharing.

Our results indicate link references to both internal and external GitHub resources. Many of the internal GitHub links are @-mention type, and most external links connect to the project documentation. In addition, links to the GitHub Wiki feature are rare in the considered sampling. Besides, Discussions users link the project documentation to emphasize they could not find a solution to the problem, communicate that they have been following an official tutorial, and declare the project documentation needs updates. We also noticed that Discussions users find solutions to issues debated in discussion threads on different target resources, such as YouTube, project documentation, Stack Overflow, and GitHub Issues and PRs.

To the best of our knowledge, this work is the first attempt to explore link-sharing activities within the GitHub Discussions. The main contributions are as follows:

1. A detailed report regarding the link-sharing activities on the GitHub Discussions.
2. An exploratory study to identify the types of linked sources of information and the reasons for the link sharing.
3. A comparative overview of users' intentions behind the links shared between Discussions and other sources of information.

We envision that communities can benefit from our results to understand better the impact of GitHub Discussions on the projects' knowledge dissemination and to promote the forum adoption as the official place for communities to interact. Community members can benefit from our results to better understand the relationship between the project entities, such as Issues, PRs, discussion posts, documentation, and project website. Finally, the GitHub engineering team can benefit from the results to better understand the importance and usage of the new feature.

2 Background

This section presents an overview of the GitHub Discussions feature (Section 2.1) and previous work that also investigates link-sharing activities on the GitHub platform (Section 2.2).

2.1 GitHub Discussions

GitHub Discussions is a collaborative communication tool tailored for the community surrounding open-source or internal projects (Hata et al., 2022; Niyogi, 2020).[4] Within GitHub Discussions, community members can create and participate in discussions at both the project's repository and organization levels.

[4] https://docs.github.com/en/discussions

In contrast to GitHub Issues, GitHub Discussions facilitates broader discussions and collaboration, whereas GitHub Issues is specifically designed to track and manage tasks, bugs, or feature requests within a project.[5]

GitHub Discussions facilitates sharing announcements, information, Q&A sessions, updates, open-ended discussions, decision tracking, feedback collection, and planning (Lima et al., 2023; Hata et al., 2022; Niyogi, 2020). Overall, GitHub Discussions cultivates a welcoming environment for community members to engage in discussions concerning project goals, development, administration, and workflows (Niyogi, 2020). Consequently, many open-source or internal projects are embracing the Discussions forum as the primary space for community members to address project-related matters. Among such projects, the `Next.js` project stands out in supporting the launch of the GitHub Discussions since the start and for the number of discussion threads.[6] The maintainer of the `Next.js` project highlighted the relevance of the forum to the project community growth - *"... our community (Next.js) has grown by 900% since moving to GitHub Discussions"* (Liu, 2021). OSS maintainers of `Node`, `Gatsby`, and `Homebrew` projects also highlighted the appropriate use of the Discussions forum for their communities (Liu, 2021).

However, for a specific community to utilize the Discussions forum, the repository administrator or project maintainer must enable it at the repository level.[4] By enabling repository-level Discussions, anyone with access to the repository has the ability to create and participate in discussions related to that specific repository. Moreover, if an organization owner enables GitHub Discussions for an organization, any individual capable of viewing the source repository can initiate discussions at the organizational level.[4] This design choice signifies that some instances of the Discussions forum are tailored to be repository-specific, while others are intended to function as forums for the entire organization.

2.2 Related Work: link sharing in GitHub

Previous studies have investigated link-sharing activities on the GitHub platform. GitHub users frequently share links in Issues and PRs comments to refer to sources of information within and across the repositories and to call the community members' attention (Zhang et al., 2015; Li et al., 2018; Chopra et al., 2021). By sharing links, users intend to establish common ground knowledge among community members (Chopra et al., 2021). GitHub users create link references using the GitHub auto-linking features (applying special characters such as "@" and "#"), specifying web addresses, using the mailto feature, and writing plain text references (links embedded in texts).

The auto-link feature converts the @-mentions to links automatically. They are frequently used in Issues and PRs to reference community members di-

[5] https://docs.github.com/en/issues

[6] https://github.blog/2022-01-13-how-five-open-source-communities-are-using-github-discussions/

rectly (Zhang et al., 2015; Chopra et al., 2021). Users tend to use @-mention in Issues and PRs comments, making it easier for developers to collaborate on the problem-solving process (Zhang et al., 2015). In addition, @-mentions decrease the delay time for receiving feedback (Zhang et al., 2015). When writing or commenting on a discussion post, the user may use the *"mention a user or a team"* feature available in GUI toolbar for supporting @-mentions use. Moreover, the auto-link feature also converts "#"-references into links to platform-specific entities, such as Issues, PRs, and comments (Chopra et al., 2021). Users share links to different sources to provide additional information about project changes proposed in PRs, positively influencing the quality of PR discussions (Chopra et al., 2021). References help establish a shared understanding, while their absence can lead to uncertainty and delays in the code review process (Chopra et al., 2021).

Analyzing links shared between issues and PRs on GitHub, Li et al. (2018) found out users share links between both issues and PRs to (1) show dependence relationships, (2) detect duplicates, (3) highlight related resources, (4) explain a concept or add information, (5) identify a solution, and (6) make enhancements. Hata et al. (2019) point out that links occur frequently in source code comments. The authors report that such links point to the software home page, tutorial, article, API documentation, forum thread, Q&A forums, code, etc. They also found that users do not update the links; the authors estimate that 10% of the links in source code comments are dead (Hata et al., 2019). Finally, Hata et al. (2022) conducted a study focused on evaluating the impact of GitHub Discussions on other channels within a repository. Their aim was to examine links from GitHub Discussions to Issues and PRs, specifically those opened or closed after the discussion began. The authors manually analyzed a sample of 231 links to determine the relationship between the discussion and the associated Issue or PR.

The existing research inspires our work, knowing that (1) the Discussions forum is a relevant source of project information, (2) the social code activities promote the project knowledge sharing in GitHub (Zhang et al., 2020), and (3) Discussions users share links in discussion threads. Our work differs and expands the scope of that present by Hata et al. (2022) since our research focuses on evaluating the impact of GitHub Discussions' link activities on project knowledge dissemination, regardless of the referenced resource. We examined link references that connect to resources both within and outside the GitHub platform. Additionally, our analysis is not limited to Issues and PRs within the repositories. We intend to investigate the link-sharing activities inside the Discussion forum, how links spread, and what are the users' intentions behind the link sharing.

3 Methodology

To study the link-sharing activities in discussion threads, we conducted a mixed-method analysis, encompassing both quantitative and qualitative data

analysis. Figure 1 shows the overall process of answering our research questions, including data sampling, link type analysis, and link relationships analysis.

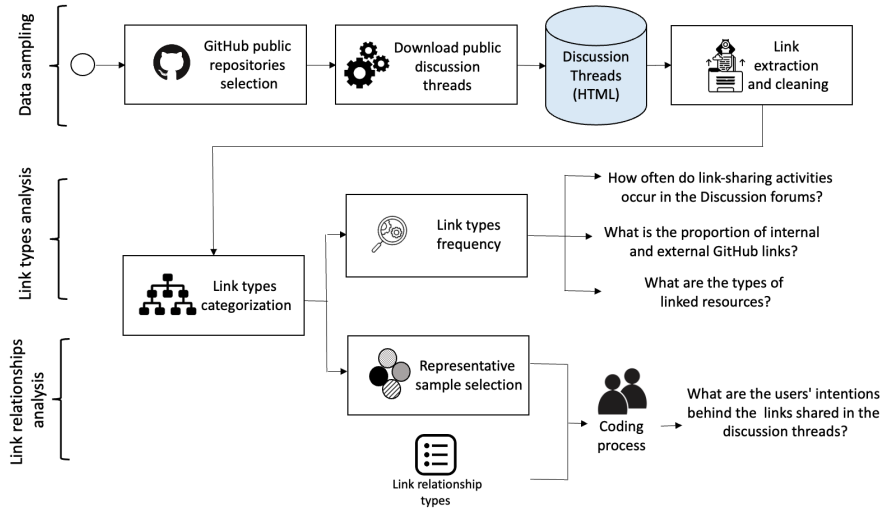

Fig. 1 Research Design

3.1 Data sampling

We analyzed the discussion threads of ten OSS repositories, as shown in Table 1. All ten projects are public. They were selected based on criteria including (1) the use of the Discussions forum (Hata et al., 2022) and (2) recommendations from the GitHub Discussions engineering team. The focus on public repositories was driven by data availability. We analyzed data from all ten repositories independently, acknowledging that some belong to the same organization, which may lead to potential overlap between their communities.

We first collected all repository-level public discussion threads from the selected repositories, totaling 29,251 threads. Then, we used the Python Beautiful Soup library[7] to extract the link references from the discussion threads. Our analysis focused on links at the main discussion level and in comments, explicitly targeting those aligned with the research context to investigate link-sharing activities that facilitate knowledge dissemination within GitHub Discussions. Consequently, we prioritized links referencing web documents and resources contributing to the project's knowledge base or development practices. We excluded certain types of links based on the following rationale:

[7] https://beautiful-soup-4.readthedocs.io/en/latest/

- *Mailto links*: These activate the default mail client for sending emails and are formatted similarly to regular hyperlinks in HTML. The mailto links primarily serve for direct communication purposes. Given our focus on links that promote collaborative knowledge sharing within GitHub Discussions, we considered mailto links, which align more with private communication, outside the scope of this work.
- *Links embedded in email notifications*: These links include unsubscribe links and web push campaign references. Such links often appear in discussions when users incorporate email content directly into their posts. These links do not contribute to our understanding of knowledge sharing as they are not deliberately shared within the community for informational purposes but are byproducts of email communication.
- *Image links*: We specifically excluded links to resources hosted on the "user-images.githubusercontent.com/" domain because, although images can be crucial for illustrating points or issues within discussions, our analysis focused on textual knowledge dissemination through web document references.
- *Search links*: These links, which activate GitHub's search functionality (e.g., "`https://github.com/vercel/next.js/search?q=fields&unscoped_q=fields`"), were excluded because they indicate an action of seeking information within GitHub. They do not constitute direct knowledge sharing or reference specific informational resources.

In total, we extracted 65,621 links that compose the sampling. To answer RQ1 - How are discussion threads linked to other resources? - we performed a quantitative analysis of the 65,621 links extracted from 29,251 discussion threads (Section 3.2). The data sampling is publicly available.[8] To answer RQ2 - What are the users' intentions behind the links shared in the discussion threads? - we conducted a qualitative analysis on a representative sample of 1,106 links extracted from the `Next.js` project (Section 3.3).

3.2 Link types analysis

In this phase, we quantitatively analyzed the link references to determine:

- **RQ1.1:** How often do link-sharing activities occur in the Discussion forums?

We counted the number of discussion threads with links and computed the frequency of link references shared at the main discussion and comment levels. The results help us understand if link sharing is a common practice in the Discussions forum.

- **RQ1.2:** What is the proportion of internal and external GitHub links?

[8] https://zenodo.org/badge/latestdoi/496255427

Table 1 Repositories used in the data sampling creation

Repository	#discussion threads	#links	Age (years)	Prevailing language
`dotnet/csharplang`	2,728	18,273	6	C#
`gatsbyjs/gatsby`	1,666	5,381	8	JavaScript
`laravel/framework`	1,548	1,724	10	PHP
`livewire/livewire`	1,842	3,788	4	Blade
`prisma/prisma`	2,286	3,952	4	TypeScript
`react-hook-form/react-hook-form`	2,522	4,449	4	TypeScript
`tanStack/react-table`	946	929	6	TypeScript
`tailwindlabs/tailwindcss`	3,505	5,345	5	JavaScript
`vercel/vercel/`	1,674	2,386	7	TypeScript
`vercel/next.js`	10,534	19,394	6	JavaScript
sum	**29,251**	**65,621**		

We categorized the links under the *"github.com"* domain as internal links (i.e., the references to target resources inside the GitHub platform). Otherwise, we classified them as external links. Links that the absolute URLs do not explicitly reference the *"github.com"* domain are classified as External.

This categorization is the first step towards identifying the target resource types most frequently referred to in the Discussions forum. Our goal is to determine whether the target resources are internal to the GitHub platform. The proportion of internal and external links allows us to understand the extent of the link sharing. The results determine if knowledge sharing exceeds the limits of the repositories and the GitHub platform.

– **RQ1.3:** What are the types of linked resources?

Once we identified whether the links were internal or external, we subcategorized them according to their URL address. This is a well-known strategy to make inferences based on the link-sharing activities on collaborative forums. Chopra et al. (2021) analyzed the link-sharing activities of developers on GitHub PR. They constructed two reference-oriented taxonomies to describe (1) the types of referenced information and (2) how the information resource is encoded in the PR posts. To do so, the authors conducted a qualitative content analysis on references extracted from public GitHub PRs (Chopra et al., 2021).

Since we are interested in mapping the project knowledge dissemination pattern (1) inside the OSS project repositories, (2) through the GitHub platform, and (3) with external connections, we also conducted a qualitative content analysis on the link references extracted from the discussion threads. Two authors iteratively identified, coded, and refined types of internal and external links. As a result, we defined the subcategories for internal and external links.

Once we had established the (sub)categories, we automatically categorized 65,621 internal and external links. This allowed us to identify the types of resources that Discussions users frequently reference by sharing links in the discussion threads.

Table 2 Conceptualization of relationship types of linked resources

Derived relationship types	Derived definition	Example
Dependent	The resolution of the problem discussed depends on the target resource.	This will be fixed after . this PR is merged #21930
Duplicate	The discussion post and the target resource discuss the same issue.	Originally posted by @Fan..ton0 in #27650 (comment)
Relevant	The discussion post and the target resource discuss similar topics. Their contents are related.	I think this is related to #18419
Reference	The knowledge in the target resource may be useful to understand the discussed issue.	Like it is described in #15453
Fixing Proposal	The target resource provides information that may solve the problem discussed.	You can achieve this behavior today using the loader prop
Enhancement Proposal	The discussion post suggests (or makes) changes to the target resource to make it more robust.	I had posted an initial question here, but it would be nice to...

3.3 Link relationship analysis

To identify the main reasons for linking behavior on GitHub, Li et al. (2018) categorized and conceptualized the relationship types between issue units (issue reports and PRs). The authors identified six relationships for linked units, including *Dependent*, *Duplicate*, *Relevant*, *Referenced*, *Fixed*, and *Enhanced* relationships (Li et al., 2018). The *Dependent* relationship expresses that *the resolution of the source issue unit is dependent on the target one.* The *Duplicate* relationship indicates that the *two issues units discuss the same problem.* The *Relevant* relationship shows that the *two issues units are similar but not the same.* The *Referenced* relationship means *the knowledge in the target issue unit may be useful in understanding the source issue unit.* The *Fixed* relationship indicates *the source issue unit submits a solution to address the problem reported in the target issue unit.* Finally, the *Enhanced* relationship expresses that the *source issue unit conducts some changes on the target issue unit to make the target issue unit robust or meet the requirements.*

Since our investigation focuses on the link-sharing behavior within the GitHub Discussions, we grounded our analysis of link relationships on the findings of Li et al. (2018). To achieve this, we adapted the concepts of the six relationships identified by Li et al. (2018) to fit the context of our research. The revised definitions of the relationship types and corresponding examples are outlined in Table 2.

We conducted a qualitative analysis of the links to answer our RQ2 - What are the users' intentions behind the links shared in the discussion threads? We intended to understand the reasons for the link sharing. To do so, we coded a sample of links based on the six main reasons for linking behavior pointed out in Table 2.

To understand the reasons for the link sharing in discussion posts, we manually analyzed a sample of 1,106 links selected among the `Next.js` project's links. We chose to focus on `Next.js` based on three primary criteria: (1) the notable use of the Discussions forum, where the `Next.js` community has distinguished itself through active participation;[6] (2) the strong support of the community in launching the Discussions feature; [6] and (3) the authors' positive prior experience in collaborating with the `Next.js` community (Lima et al., 2023).

To proceed with the link relationship analysis, we first created a representative sample of link references. To define the sample size, we set the sampling error margin to 10% and the confidence level to 95%. These percentage values were determined by considering various factors: (1) our qualitative analysis of links focused solely on those extracted from the Next.js repository, (2) the substantial number of link references in our database (65,621), and (3) the manual effort involved in analyzing and categorizing intentions associated with a larger sample of link references. This decision was a thoughtful choice aimed at balancing sample representativeness and statistical accuracy.

Then, we coded the sample according to the Derived Relationship Types described in Table 2. Two of the authors independently classified a random sample of links; afterward, they discussed the differences in their classifications until they reached a consensus. We measured the inter-rater agreement using Cohen's kappa coefficient (Cohen, 1960), achieving a value of 0.77. This value indicates substantial agreement according to the interpretation proposed by Landis and Koch (1977). As a result, we could discuss the reasons for the link-sharing activities in the Discussions forum.

4 Results

To the best of our knowledge, this is the first work investigating link-sharing in GitHub Discussions, including the different types of linked sources and the users' intentions behind the shared links. In the following section, we present and observe the results of the research questions.

4.1 How are discussion threads linked to other resources? (RQ1)

To understand how discussion threads link to other resources, we first identified the percentage of discussion threads with links (Section 4.1.1). Then, we categorized the links according to their direction and measured the proportion of internal and external GitHub links (Section 4.1.2). Finally, we determined the types of target resources and computed the frequency of the links in each category (Section 4.1.3). Our sample comprises 65,621 links extracted from 29,251 discussion threads from 10 OSS projects (Table 1).

4.1.1 How often do link-sharing activities occur in the Discussion forums?

Table 3 presents the percentage of discussion threads with links, the rate of the links occurrence in the main post, and comments, respectively. On average, 59.12% of the 29,251 discussion threads have links, considering both the main post and comments (Table 3, column 2). The top-3 projects in the rank of discussion threads with links are `dotnet/csharplang`, `react-hook-form/react-hook-form`, and `gatsbyjs/gatsby` projects, all of which have a percentage value greater than 61%. Table 3 also reveals that, in 7 out of 10 analyzed projects, the median number of links in discussion threads is 2. Specifically, for the `dotnet/csharplang` project, the median is 4; for `gatsbyjs/Gatsby`, it is 3, and for the `TanStack/react-table` project, it is 1.

Table 3, column 2 shows the overall proportion of discussion threads containing links. This percentage was computed for each repository using the following formula: (number of discussion posts with links) / (total number of discussion posts).

As we explored the places where the links were shared, our investigation revealed that, on average, 59.93% of the discussion threads containing links have these links placed in the discussion main post (Table 3, column 3) — % of discussions with a link in main. In addition, around 63.09% of the discussion threads containing links include links within the comments (Table 3, column 4) — % of discussions with a link in comments. These values take into account only the discussions that have links. Given that discussion threads can incorporate links within both the main posts and comments, it is important to note that the cumulative sum of these percentage values can exceed 100%.

Analyzing the discussion threads with links, we found that, on average, discussion threads containing 1 to 10 links are common compared to those with 11 to 20 or 20 to 30 references. In contrast, those with over 30 link references are rare (Table 4). Considering our data sampling, discussion threads with 1 to 10 links correspond to 94.66%, threads with 11 to 20 links correspond to 3.53%, and threads with 21 to 30 links correspond to 0.99% of the sampling. Only 0.82% of discussions have more than 30 links. We also identified repeated links in the same discussion thread (i.e., `Next.js` #12414).

4.1.2 What is the proportion of internal and external GitHub links?

Figure 2 presents the percentage of internal and external links per repository. We found that, on average, 62.17% of links shared in discussion threads are internal, and 37.82% of links are external. These results indicate that Discussions users refer to knowledge resources inside and outside the GitHub platform to support their discussions. We noticed that the knowledge disseminated through shared links exceeds the GitHub platform boundary. However, we found that in some projects, the number of internal links may differ eight times that of external, e.g., the `dotnet/csharplang` project (16,232/2,041). Conversely, we observed that the proportion of external links is 1.4 times

Table 3 Percentage of discussion threads with link

Repository	%discussion threads with link	Discussions with links		
		%in the main post	%in comments	Median number of links
dotnet/csharplang	2208 [80.93%]	1022 [46.28%]	1945 [88.08%]	4
gatsbyjs/gatsby	1078 [64.70%]	797 [73.93%]	656 [60.85%]	3
laravel/framework	687 [44.37%]	496 [72.19%]	293 [42.64%]	2
livewire/livewire	1151 [62.48%]	460 [39.96%]	939 [81.58%]	2
prisma/prisma	1378 [60.27%]	541 [39.25%]	1053 [76.41%]	2
react.../react-hook-form	1703 [67.52%]	1246 [73.16%]	949 [55.72%]	2
TanStack/react-table	430 [45.45%]	287 [66.74%]	211 [49.06%]	1
tailwindlabs/tailwindcss	2077 [59.28%]	1310 [63.07%]	1245 [59.94%]	2
vercel/vercel/	850 [50.77%]	516 [60.70%]	491 [57.76%]	2
vercel/next.js	5838 [55.42%]	3735 [63.97%]	3438 [58.89%]	2
avg	**59.12%**	**59.93%**	**63.09%**	

Table 4 Percentage of links per discussion threads

Repository	#links per discussion threads			
	1—10	11—20	21—30	>30
dotnet/csharplang	77.17%	14.13%	5.11%	3.57%
gatsbyjs/gatsby	90.63%	5.56%	1.39%	2.41%
laravel/framework	98.25%	1.16%	0.29%	0.29%
livewire/livewire	95.74%	3.64%	0.52%	0.08%
prisma/prisma	97.75%	0.94%	0.58%	0.72%
react.../react-hook-form	98.17%	1.35%	0.41%	0.05%
tanStack/react-table	98.60%	1.39%	None*	None*
tailwindlabs/tailWindCss	97.40%	2.11%	0.24%	0.24%
vercel/vercel	96.94%	2.58%	0.35%	0.11%
vercel/next.js	95.83%	2.48%	0.99%	0.68%
avg	**94.66%**	**3.53%**	**0.99%**	**0.82%**

* We haven't identified any link occurrences in this range.

that of internal in the `react-hook-form/react-hook-form` project. The results endorse the singularity of OSS communities. Regarding the balance of internal and external GitHub links, we could not state a standard behavior for link sharing in discussion threads. Therefore, we investigated the types of referenced information resources.

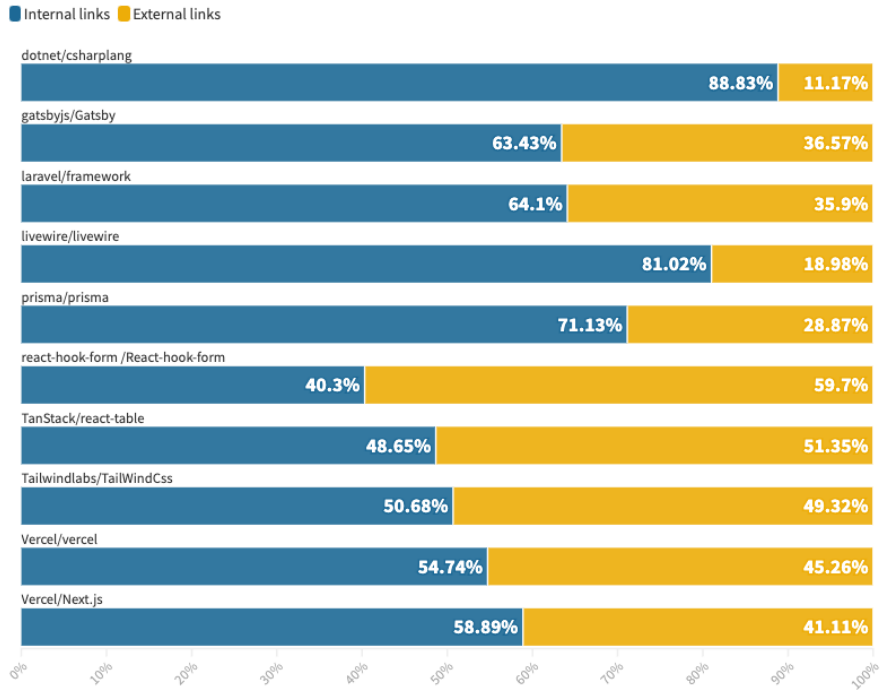

Fig. 2 Proportion of internal and external links

4.1.3 What are the types of linked resources?

To know the target of the links and the type of information it represents, we categorized the links according to their URL address. We organized the internal links into three top-level categories and seven subcategories (Section 3.2). We classified the external links into eight subcategories (Section 3.2). Figure 3 shows the categories of link types considered in this research. The following categories emerged from a manual analysis of the extracted links.

- *Internal links.* As presented in Section 3.2, Internal links are those under the GitHub domain (*"github.com"*). This top-level category comprises three sub-categories:
  1. *@-mentions.* The @-mentions category emerges from the auto-link GitHub feature. The feature detects references prefixed with "@" and converts them into links automatically (Chopra et al., 2021).
  2. *Within-repo.* This category emerges by grouping links to information resources within the repository. To map the behavior of the links inside the OSS projects' repositories, we classified the within-repo type of links according to some GitHub activities, such as Issues, Discussions, Wiki, Code, and Pull Requests (PRs). We split the within-repo subcategory into five others:

Code. This category groups the links to code entities inside the repository (branches, hooks, code containers, libraries, and elements). We applied a regular expression to identify links that match the repository URL address, followed by the keywords "commit/," "tree/," "blob/," or "releases/."
Issues. This category groups the links to issues within the repository. To this end, we applied a regular expression to identify links that match the repository URL address and the keyword "issues/".
Discussions. Links to discussion threads or discussion comments within the repository to a specific repository. We identified these links by applying a regular expression to search for links that match the source repository URL address, followed by "discussions/."
Pull Requests (PRs). This category groups the links to PRs discussions for a particular repository. To identify such links, we also applied regular expressions searching for links that match the repository URL address, followed by the string "pull/."
Wiki. This category groups the links to Wiki pages to a given repository. To identify such links, we also applied regular expressions searching for links that match the repository URL address, followed by the string "wiki/."

3. *Cross-repo.* Cross-repo category category encompasses links to resources located outside the URL address of a specific repository. To map the linked GitHub entities (repositories, issues, PRs, etc.) outside the repository, we split the cross-repo sub-category into two other:
Within-organization. This category groups the links to resources within the organization's URL address of a specific repository. For example, one link from the `Next.js` project to the `vercel` project is a within-organization link since both projects are under the `vercel` organization domain (Table 1).
Cross-organization. The links from this category link to GitHub entities outside the organization domain of a specific repository.

– *External links.* To map the types of information resources more frequently linked outside the GitHub platform, we specialized the External links top-level category into eight sub-categories as follows:

1. *Project Documentation.* This category groups the links to the project documentation hosted outside the GitHub platform. To extract these links, we first identified the URL address of the project documentation by reading the project README.md file. Then, we applied a regular expression to search for the links that match the documentation domain. Frequently, documents are under "https://<project_site>/docs/*" URL.

2. *Online code editor / Playground.* The links classified in this category link to code snippets shared in cloud-based code tools such as CodeSandbox,[9] CodePen,[10] and TypeScript Playground.[11]
3. *Project website.* The links from this category link to the project website outside the GitHub platform. We identified the project website URL in the README.md file of each analyzed project.
4. *Stack Overflow.* This category groups the Stack Overflow questions. We found the links to Stack Overflow are frequently among the External links. We also use a regular expression to search for links under the "stackoverflow.com/" domain.
5. *X/Previous Twitter.* *X*/Previous Twitter is a popular social media platform among software developers. Since the platform impacts software developers' work day (Fang et al., 2020), we defined a specific subcategory to group links to *X*/Previous Twitter posts.
6. *Video platforms.* This category emerges by grouping the links to online video-sharing platforms, such as YouTube.[12] We identified the links to video platforms by searching for links under the YouTube and Loom[13] (online screen recording tool) domains. These platforms were prominently identified through a manual inspection of our dataset, where links predominantly led to YouTube and Loom.
7. *Wikipedia.*[14] Wikipedia is a general-purpose online encyclopedia. We also found that Discussions users also share links to Wikipedia pages. We identified these references by searching for links under the encyclopedia domain.
8. *LinkedIn.*[15] LinkedIn is a networking platform for individuals and businesses to connect, share professional information, and explore career opportunities. We found link references to LinkedIn pages. We identified these references by searching for links under the "linkedin.com" domain.
9. *Blogs.* This category includes link references to blog pages. To extract these links, we applied a regular expression to search for links matching domains containing "/blogs/," "/blog/," "//blog.,' and "//blogs."
10. *Third-party Documentation.* This category groups link references to third-party documentation not hosted on the project website. To extract these links, we applied a regular expression to search for the links matching common documentation domains, such as */doc/*, */docs/*, and */documentation/*.

[9] https://codesandbox.io/
[10] https://codepen.io/
[11] https://www.typescriptlang.org/play
[12] https://www.youtube.com/
[13] https://www.loom.com/
[14] https://en.wikipedia.org/
[15] https://www.linkedin.com/

11. *Example Domains*. This category groups link references used to exemplify URL addresses, such as "https://mydomain.com/" and "https://example.com/."
12. *Cloud platform for storage service*. This category groups link references to online services that allow users to store and access data across multiple devices. To extract these links, we used a regular expression to identify links associated with specific platforms like Google Drive, Dropbox, and Microsoft OneDrive platforms.
13. *Collaborative platform for software developers*. This category groups links associated with online environments that facilitate teamwork, communication, and code sharing, exemplified by platforms like "dev.to." and "npmjs.com."
14. *Cloud platform for app deployment*. This category groups links associated with an online service that provides infrastructure, tools, and resources for deploying, managing, and scaling software applications in a cloud computing environment, such as "heroku.com."
15. *Other*. The category refers to links that do not fit the other External links categories.

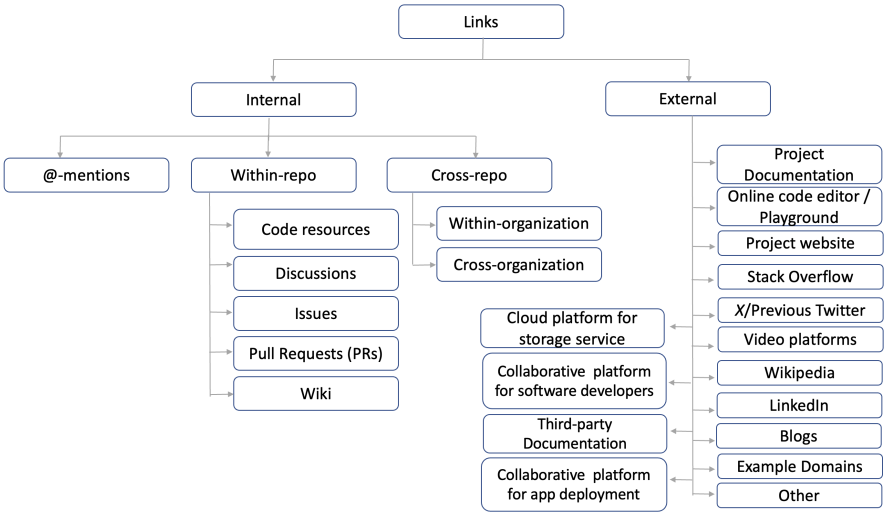

Fig. 3 Types of linked resources

After defining the categories, we built a Python script to classify the extracted links automatically. Table 5 shows the distribution of the links extracted according to their URL address. We notice that:

– @-mentions are the most frequent links among internal ones, 49.81%. Within-repo and cross-repo links represent 28.35% and 21.84%, respectively. The @-mentions link differs from others because they primarily link to user

profiles. However, we also noticed that Discussions users use @-reference notations to link to repository pages within GitHub and programming language features.

- We found that, in total, Issues are the most frequent type of link. However, we observed that the number of links to Issues extracted from the `dotnet/csharplang` project affects the sum value. Based on the project's README.md file,[16] the `dotnet/csharplang` community opens issues to propose new C# language features, and they use the Discussions forum to discuss the proposals. This community behavior supports the high number of links between discussion threads and issues in the project. Upon dismissing the `dotnet/csharplang` project, we observed a shift in the data: links to code resources emerged as the most frequent type of link. This shift underscores the variability in community engagement practices across different projects and highlights the unique role of the `dotnet/csharplang` community in utilizing Discussions. The exclusion of the `dotnet/csharplang` data reveals that the prevalence of issue links may be influenced by specific community practices. This observation suggests the need to further explore how various communities differ in using Discussions and raises questions about the diversity of usage patterns of Discussions across repositories.
- The number of links indicates that it is common to find links between discussion threads. Out of 10 projects, 5 have more links to discussion posts than other within-repo target resources. This result suggests users are aware of the within-repo discussion posts' topics. In addition, OSS community members unconsciously promote the project knowledge reuse by sharing links to discussion posts they consider relevant to the source discussion.
- Regarding the links to code entities, we found links to commits, branches, source code packages, source code files, source code lines, releases, and resources that support using the source codes, such as example files.
- When we compare the occurrence of links of within-organization with cross-organization links, the results suggest that users cite resources outside the organization domain of the source project. Conversely, the percentage rate of within-organization type links of the `dotnet/csharplang` project is four times greater than that of cross-organization links. Nearly 80% of the links within the `dotnet/csharplang` project that point to other resources within the organization (within-organization) direct to the `dotnet/roslyn` repository. This result suggest a close relationship between `csharplang` and `dotnet/roslyn` repositories. The `dotnet/csharplang` README.md file indicates the `dotnet/ roslyn` repository contains the reference implementation of the C# language, being a relevant source of information for the `dotnet/csharplang` project. This result suggests the effectiveness of the `dotnet/csharplang` project knowledge dissemination through Discussions and that the project knowledge dissemination extends beyond the repository boundaries.

[16] https://github.com/dotnet/csharplang/#readme

- External links frequently refer to the project documentation. This result corroborates the lack of explicit references to the GitHub Wiki (within-repo). Only the `vercel/vercel` project has a Wiki page that links to the external project documentation page.
- The projects' documentations are not always under "https://<project _site> /docs/*" URL. That is why we report the value zero to the frequency of the Project Documentation link type for `react-hook-form/react-hook-form` and `dotnet/ csharplang` projects in Table 5.
- The Discussions' users share links to online code editors or playgrounds to show code snippets they implemented, e.g., CodeSandbox,[17] CodePen.[18] The majority of the `react-hook-form/react-hook-form` and `tanStack/ react-table` projects' external links (69% and 49%, respectively) link to online code editors. Besides Codesandbox and CodePen, we found links shared to laravelplayground.com, play.tailwindcss.com, jsfiddle.net, stackblitz.com, sharplab.io, and dotnetfiddle.net.
- We also found that users share links to video-sharing platforms. Code videos may provide introductory or in-depth software engineering topics, helping developers to learn new technical skills (Poche et al., 2017; Ponzanelli et al., 2017).
- In `Next.js` data sampling, it is common to find links used to exemplify a URL address. These links are not necessarily valid, they may result in errors. They are used to illustrate or represent the format of a URL but do not guarantee that the referenced page or resource is actually available or functions correctly.
- In the "other" sub-category type, we found links to the Spectrum platform, which GitHub communities used to ask questions, report bugs, and chat.
- In the "other" sub-category type, we also found links to job description pages, Slack channels, tutorial webpages, online learning platforms, etc.
- Finally, we also identified the occurrence of dead links. Users may share dead links to exemplify a bug occurrence or even request a feature regarding the use of URLs. Dead links do not reference a valid webpage, e.g. *"http://example.ca/fr"*.

Answering RQ1: Based on the analysis of our data sampling, link sharing is a common practice in the Discussions forum. Discussions users share links to target resources hosted both internally and externally on the GitHub platform. Although most within-repo links reference issues, link references to other discussion threads are common. When analyzing reference links pointing to resources outside the GitHub platform, project documentation and websites constitute the majority, excluding the 'Other' category.

[17] https://codesandbox.io/
[18] https://codepen.io/

Table 5 Link types frequency

INTERNAL LINKS

	Within-repo						Cross-repo		#Internal links
	@-mention	Code	Issues	PRs	Discussions	Wiki	Within-organization	Cross-organization	
dotnet/csharplang	9517 [58.63%]	596 [3.67%]	1626 [10.02%]	67 [0.41%]	1395 [8.59%]	1 [0.01%]	2447 [15.08%]	583 [3.59%]	16232
gatsbyjs/gatsby	1406 [41.20%]	440 [12.89%]	344 [10.08%]	195 [5.71%]	155 [4.54%]	0 [0.00%]	48 [1.41%]	825 [24.17%]	3413
laravel/framework	326 [29.50%]	283 [25.61%]	126 [11.40%]	120 [10.86%]	20 [1.81%]	0 [0.00%]	58 [5.25%]	172 [15.57%]	1105
livewire/livewire	2150 [70.06%]	187 [6.09%]	123 [4.01%]	188 [6.13%]	147 [4.79%]	0 [0.00%]	22 [0.72%]	252 [8.21%]	3069
prisma/prisma	1552 [55.21%]	64 [2.28%]	675 [24.01%]	8 [0.28%]	79 [2.81%]	0 [0.00%]	192 [6.83%]	241 [8.57%]	2811
react.../react-hook-form	1086 [60.57%]	124 [6.92%]	164 [9.15%]	65 [3.63%]	176 [9.82%]	0 [0.00%]	25 [1.39%]	153 [8.531%]	1793
tanStack/react-table	249 [55.09%]	16 [3.54%]	26 [5.75%]	12 [2.65%]	33 [7.30%]	0 [0.0%]	0 [0.0%]	116 [25.66%]	452
tailwindlabs/tailWindCss	1188 [43.85%]	140 [5.17%]	96 [3.54%]	174 [6.42%]	227 [8.38%]	0 [0.0%]	104 [3.84%]	780 [28.79%]	2709
vercel/vercel	637 [48.77%]	41 [3.14%]	26 [1.99%]	16 [1.23%]	147 [11.26%]	0 [0.0%]	87 [6.66%]	352 [26.95%]	1306
vercel/next.js	3959 [34.66%]	1421 [12.44%]	888 [7.78%]	519 [4.54%]	1411 [12.35%]	0 [0.0%]	231 [2.02%]	2992 [26.20%]	11421
Total	**22,070**	**3,312**	**4,094**	**1,364**	**3,790**	**1**	**3,214**	**6,466**	**44,311**
%	49.807	7.474	9.239	3.078	8.553	0.002	7.253	14.592	

EXTERNAL LINKS #1

	Project Website		Stack Overflow	Videos Platform	*X* / Previous Twitter	Online Code Editor/ Playground	Wikipedia	LinkedIn	#External links
	/docs/	other							
dotnet/csharplang	—*	—*	159 [7.79%]	25 [1.22%]	22 [1.08%]	296 [14.50%]	113 [5.54%]	1 [0.04%]	...
gatsbyjs/gatsby	437 [22.21%]	428 [21.75%]	56 [2.85%]	19 [0.97%]	24 [1.22%]	26 [1.32%]	2 [0.10%]	0 [0.0%]	...
laravel/framework	221 [35.70%]	38 [6.14%]	43 [6.95%]	8 [1.29%]	16 [2.58%]	3 [0.48%]	4 [0.65%]	1 [0.16%]	...
livewire/livewire	213 [29.62%]	33 [4.59%]	23 [3.20%]	25 [3.48%]	43 [5.98%]	77 [10.71%]	3 [0.42%]	0 [0.0%]	...
prisma/prisma	620 [54.34%]	52 [4.56%]	38 [3.33%]	36 [3.16%]	18 [1.58%]	1 [0.09%]	0 [0.0%]	0 [0.0%]	...
react.../react-hook-form	—*	452 [17.02%]	32 [1.20%]	26 [0.98%]	18 [0.68%]	1843 [69.39%]	1 [0.04%]	0 [0.0%]	...
tanStack/react-table	94 [19.71%]	1 [0.21%]	19 [3.98%]	1 [0.21%]	0 [0.0%]	238 [49.90%]	1 [0.21%]	0 [0.0%]	...
tailwindlabs/tailWindCss	765 [29.02%]	27 [1.02%]	76 [2.88%]	80 [3.03%]	45 [1.71%]	544 [20.64%]	1 [0.04%]	1 [0.03%]	...
vercel/vercel	283 [26.20%]	195 [18.06%]	16 [1.48%]	3 [0.28%]	12 [1.11%]	0 [0.0%]	0 [0.0%]	1 [0.09%]	...
vercel/next.js	2031 [25.47%]	310 [3.89%]	255 [3.20%]	62 [0.78%]	91 [1.14%]	214 [2.68%]	18 [0.23%]	63 [0.79%]	...
Total	**4,664**	**1,536**	**717**	**285**	**289**	**3,242**	**143**	**67**	
%	21.88	7.20	3.36	0.13	1.35	15.21	0.67	0.31	

EXTERNAL LINKS #2

	Blogs	Third-party Documentation	Example Domains	Cloud Platform for storage Service	Collaborative Platform for Software Developers	Cloud Platform for application Deployment	other	#External links
dotnet/csharplang	178 [8.71%]	540 [26.44%]	0 [0.0%]	0 [0.0%]	3 [0.14%]	0 [0.0%]	705 [34.52%]	2041
gatsbyjs/gatsby	77 [3.91%]	147 [7.46%]	7 [0.35%]	1 [0.05%]	60 [3.04%]	1 [0.05%]	683 [34.70%]	1968
laravel/framework	11 [1.77%]	54 [8.70%]	6 [0.96%]	0 [0.0%]	0 [0.0%]	3 [0.48%]	212 [34.19%]	619
livewire/livewire	7 [0.97%]	62 [8.61%]	16 [2.22%]	2 [0.27%]	1 [0.13%]	0 [0.0%]	215 [29.86%]	719
prisma/prisma	13 [1.13%]	106 [9.29%]	6 [0.52%]	1 [0.08%]	24 [2.10%]	20 [1.75%]	206 [18.05%]	1141
react.../react-hook-form	8 [0.30%]	54 [2.03%]	3 [0.11%]	1 [0.03%]	6 [0.22%]	0 [0.0%]	212 [7.98%]	2656
tanStack/react-table	1 [0.20%]	22 [4.61%]	1 [0.20%]	2 [0.41%]	12 [2.51%]	0 [0.0%]	85 [17.81%]	477
tailwindlabs/tailWindCss	41 [1.55%]	231 [8.75%]	8 [0.30%]	0 [0.0%]	83 [3.14%]	0 [0.0%]	735 [27.87%]	2636
vercel/vercel	28 [2.59%]	93 [8.60%]	21 [1.94%]	1 [0.09%]	23 [2.12%]	14 [1.29%]	391 [36.17%]	1080
vercel/next.js	200 [2.50%]	776 [9.73%]	426 [5.34%]	66 [0.82%]	345 [4.32%]	26 [0.32%]	3153 [39.54%]	7973
Total	**564**	**2085**	**494**	**74**	**557**	**64**	**6597**	**21,310**
%	2.65	9.78	2.32	0.35	2.61	0.30	30.95	

* We could not determine this value.

4.2 What are the users' intentions behind the links shared in the discussion posts? (RQ2)

To answer RQ2, we conducted a qualitative analysis on 1,106 links extracted from the `Next.js` project (Section 3.3). We manually classified the links into the derived relationship categories described in Table 2. We also provide user quotations extracted from discussion posts that support the results.

– Dependent relationship

Dependent relationships are rare. Out of the 1,106 links we analyzed, we classified three links under this category. Users shared links to PRs to highlight that the solution to the problem discussed depends on the PR merge — "*(The RFC) will be no longer required when #36436 becomes available on stable in Next.js 13*" (#8207).

– Duplicate relationship

Discussions users duplicate discussion posts on different platforms to get different feedback about their questions — "*I also asked this question on SO but didn't get any logical answer*" (#24452). Users also duplicate discussions on different platforms because they have doubts about the best place to post it — " *I didn't know where best to post it*" (#20502). Besides, users duplicate discussion posts to emphasize their need for help — "*I'm basically asking that same question here: #24659* " (#23737).

We also found that Discussions users intentionally duplicate discussion posts on/from Stack Overflow — "*This is cross-posted to Stack Overflow...*" (#22715). Users may triplicate the discussion content on third-party websites —"*Thanks for asking this question. I noticed your Reddit and Stack Overflow posts also have zero responses...*"(#35554). In addition, we found that users may also highlight the occurrence of duplicates — "*Same as: #12226*"(#14337).

– Relevant relationship

The Relevant relationship is the most frequent among links to discussion threads, achieving 48% of the analyzed *within-repo* discussions type of links. Discussions users share links to other discussion posts that may discuss similar feature requests, share the same topic, or address related bugs — "*I believe that the request is pretty similar to this one: #34567*" (#34543).

Users frequently share links between discussion posts to identify related ones. In addition, users link to relevant discussions to highlight already created RFCs (Request For Comments) that share similar topics with the discussion thread — "*we already have similar RFC #8981.*" (#16854) —, call attention to other people who met similar problems — "*And I search people met this problem too*" (#24437) —, and fulfill the discussion template with well-known similar discussions — "*Related*" (#11822).

– Reference relationship

Links classified in the Reference relationship category are the most frequent ones. Among the 1,106 links we analyzed, 78% link to information resources that help clarify the discussion post content.

Users share links to Wikipedia to conceptualize terms and clarify the ideas proposed in the discussion threads. Although it is not a programming-restricted purpose media, software engineering researchers use Wikipedia pages to train word Embedding models (Mishra and Sharma, 2021), detect ambiguity in requirements engineering (Ferrari and Esuli, 2019), and study bots' activities (Zheng et al., 2019). Users also share links to Wikipedia to challenge the

correctness of the project documentation —"*In my view, this is certainly not a pure function (and indeed in Wikipedia's view), as it violates the following:...*" (#20015).

Discussions users add references to online code editors to explain the bugs they have faced, allowing people to reproduce them. Using online code editors, the users avoid copying and pasting code snippets in the discussion body content, providing executable code.

In addition, users link the project documentation to emphasize they could not find a solution to the problem, communicate they have been following an official tutorial, and declare the project documentations need updates — "*I cannot find any official doc*" (#25091); "*I'm using NextJS + Electron and trying to follow ¡link¿ but as soon as I add ..., things start to break*" (#10829); "*I think that it would also be nice to add examples on how to customize ... to the official docs*" (#24115).

When users do not find their need for information on the project documentation, they may recur to Programming Community Question and Answer (PCQA) forums, such as Stack Overflow — "*I cannot find any official doc. So I follow this topic: https:// stackoverflow.com/questions/60459747/connect-apollo-with-mongodb*" (#25091).

Some users prefer to create a Stack Overflow question and reference it in the discussion post. In this case, users detail issues or bugs in a Stack Overflow question and start a discussion post to declare they need help on that issue. In this case, users do not duplicate the description of the problems in both Discussions and Stack Overflow —"*Hi. I have posted a question on Adding a script with onload in Next.js on Stackoverflow. Would anyone be able to help/comment? Thanks!*" (#27197); "*Here are the details of my bug.*" (#2884).

We also found Discussions users use @-mentions type of link to catch the community members' attention or emphasize a repository —"*@jnv, @nevnein Using it in 9.5.3 (not canary), without as. Works just fine!*" (#8207); "*Cool. This RFC issue can be finally closed. Am I right, @Timer?*" (#8207).

We noticed that users might share links to exemplify issues regarding the use of links. Users create references to clarify proposed solutions, illustrate a bug occurrence, present a question, or ask for a new feature —"*How nextjs do this https://nextjs.org/showcase and https://nextjs.org/showcase/?*" (#23988).

Finally, Discussions users may add references to target sources to ask for help managing them —"*Can some help me to fix this issue. #21890*" (#23184).

– Fixing Proposal

The fixing proposal type relationship is frequently among links to PRs, ranking the second position right after the Reference type relationship. However, Discussions users find potential solutions to issues discussed in discussion threads on different target resources: YouTube, documentation, Stack Overflow — "*Check this answer [on Stack Overflow], I hope it help you*" (#11351)

—, Issues, PRs, and *X*/Previous Twitter — "*There was an interesting Twitter thread on this, involving ... and containing some points to potential solutions...*" (#22715).

– Enhancement Proposal

Like the Dependent relationships, the Enhancement Proposal type relationships are rare. Out of the 1,106 links we analyzed, we classified three links in this category (two links to discussion threads and one to a cross-organization resource).

Discussions users create a new discussion thread to add information to the target resource — "*I had posted an initial question here, which the suggestion was basically "use gSSP", which is also a possible solution but it would be nice to not regenerate the page from scratch every time when it's basically static.*" (#17111). In all three cases, users did not invalidate the original posts.

In addition to the mentioned purposes, we found that users may reference the same target resource with different intentions. In the same discussion thread, we found duplicate links, which we classified the first occurrence as Duplicate and the second as Fixing Proposal.

1. Duplicate relationship: "*I posted this question here (vercel/ swr#362) 2 days ago but didn't receive any response, hence posting here.*" (#12414).
2. Fixing Proposal: "*Was answered here: vercel/swr#362*" (#12414).

We also identified a different reason for link sharing: to identify the correct place for a discussion. We call this relationship a Localization relationship —"*I suggest adding your configuration to #30174 so it can be prioritized*" (#30862); "*If you're a company looking to post a role, use #32937*" (#32938).

Answering RQ2: Based on the analysis of our data sampling, most link-sharing activities reference resources that clarify the topic under discussion. Discussions users also share links to identify issue dependencies, highlight duplication, point out related topics, suggest fixing proposals, and identify enhancement proposals. We found that dependent and enhancement proposal relationships are rare. Users may intentionally duplicate their posts in Stack Overflow. The fixing proposal relationships may occur in different categories, such as videos, documentation, Stack Overflow, Issues, PRs, and *X*/Previous Twitter.

5 Discussion

This section compares our findings against previous link analysis studies on the OSS software development domain. We discuss the types of linked resources we found and the Discussions users' intentions behind the link sharing. In addition, we present the implications of this research from the perspective of OSS communities and software engineering researchers.

5.1 Types of linked resources

As we reported in Section 4.1, users frequently leave explicit links in discussion threads — as well as Modern Code Review platforms (Wang et al., 2021), source code repositories (Hata et al., 2019), GitHub PR discussions (Chopra et al., 2021), and GitHub issues reports (Li et al., 2018). Almost 59% of the discussion threads in our sample have at least one link. Discussions' creators share links to resources they judge relevant to the topic they have been discussing. In addition, users who add comments to the discussion threads also share links to support their rationale regarding the discussion's main topic. So, we can declare that Discussions users share links in discussion posts to support the correct knowledge understanding among community members.

In contrast with Modern Code Review (MCR) users (Wang et al., 2021), Discussions users frequently share links directly and indirectly related to the projects. They share links to resources hosted outside and inside the repository domain. In addition, they link to resources outside the GitHub domain. The difference between the proportion of internal and external GitHub links shows that Discussions users push the platform's boundaries to promote the projects' knowledge sharing.

We observed similarities between our categorizations of link types (Figure 3) and the taxonomy of referent types created by Chopra et al. (2021) as follows.

1. The @-*mentions* subcategory equates Chopra et al.'s "*Actor*" top-level category. Similar to Chopra et al., our subcategory comprises links to community members, project repositories, or organizations that affect the software project.
2. Our definition of the *within-repo issues* and *within-repo PRs* sub-categories is similar to the definition proposed by Chopra et al. (2021). However, since we were interested in comprehending the link-sharing activities inside the source repository, we first filtered the *within-repo* and the *cross-repo* links. For example, considering the link "`https://github.com/vercel/next.js/issues/37793`" extracted from a discussion within the `Next.js`, we classify it into the *within-repo issues* category. However, we classify the link "`https://github.com/vercel/next-plugins/issues/507`", also extracted from a `Next.js` discussion, in the *cross-organization* category because the repository domain differs from the source repository.
3. Our *within-repo code* category definition merges the definitions of Chopra et al.'s "*Version Control Entity*" and "*Source Code*" categories. According to the authors, the category "*Version Control Entity*" comprises entities related to the version control system of the software (branches, hooks, and merge conflicts). Besides, the category "*Source Code*" includes any artifacts in the project representing source code (code containers, code elements, code libraries, API, test).

4. Our *external documentation* subcategory comprises links to resources detailing how to use the projects' functionalities. It is similar to the "*Client Documentation*" subcategory defined by Chopra et al. (2021).

Just like Modern Code Review users (Wang et al., 2021) and source code repositories users (Hata et al., 2019), Discussions users also share links to Stack Overflow questions and the software homepage. As well as in source code repositories (Hata et al., 2019), we also observed the occurrence of dead links. However, we found that Discussions users might intentionally share dead or broken links to describe URL problems related to the projects' context.

5.2 The users' intentions behind the link sharing

Most shared links provide additional information, helping users understand the discussed topic. However, we identified that the link-sharing intentions might differ according to the link type.

We only found *Dependent relationships* occurrence between discussion posts and PRs. In this case, users referred to PRs to make clear that the PR merge would fix the problem described in the discussion post.

Duplicate relationships are frequently between discussion posts and Stack Overflow questions. Since Stack Overflow is a popular Programming Community-based Question Answering (PCQA) forum (Wang et al., 2020; Pei et al., 2021; Ford et al., 2018), users duplicate discussion posts in Stack Overflow to increase the question visibility and the chance of getting an adequate answer. Discussions users highlight they also intentionally repeat questions in both forums because they do not know where is the best place to post them — "*I have also post this on Stack Overflow as I didn't know where best to post it*" (`Next.js` #20502). While both GitHub Discussions and Stack Overflow resemble a Q&A forum, they differ in the intentions behind the posts. GitHub Discussions focuses on a software project ecosystem, where each project has its own forum (Hata et al., 2022). In contrast, Stack Overflow questions cover a broader context, aiming to answer developers' technical questions regardless of a specific software project.

The Relevant relationships are frequent between discussion posts. Users often identify discussion threads or discussion comments related to the source discussion. We noticed that Discussion authors share links to related discussions when they (1) fulfill discussions' templates, (2) emphasize people who face the same problem, and (3) indicate they have found similar posts that do not consider their need for information.

The Fixing Proposal relationships are frequently between discussion posts and PR questions. We noticed that all fixing proposal links to PR were in the discussion comments. It means OSS community members know the proposed project changes and point out the solutions to previously known problems.

The Enhancement Proposal relationships are unusual. We found this type of relationship between discussion posts. Discussions creators enhanced the discussion posts by adding information to a previous question.

Regarding the most frequent type of internal links, users use @-mentions to catch the attention of a specific community member or to highlight a repository. GitHub users use the @-mentions auto-link feature to reference the projects' stakeholders (Zhang et al., 2015; Chopra et al., 2021). Zhang et al. (2015) found out that @-mentions tend to reduce the problem-solving time and the time delay between comments in issue and PR reports. They claim that @-mentions facilitate the developers' collaboration, positively affecting the problem-solving process of the issues. Moreover, Discussions users use @-mentions to locate repositories, e.g. "*I've created @ceteio/next-layout-loader which is a workaround to support nested _app.js /_layout.js files*" (`Next.js` #26389). The @-mentions links rarely link to resources to disseminate information and establish a shared understanding.

Discussions users frequently reference external sources of information. The number and diversity of linked resources suggest that (1) OSS communities build valuable project knowledge networks by sharing links in discussion posts, and (2) the project knowledge-sharing activities go beyond the project repository and the GitHub platform.

5.3 Research implications

We envision this research bringing opportunities for OSS communities, software engineering researchers, and the GitHub engineering team as follows.

OSS communities: Our findings yield valuable insights into the practice of sharing links within discussion threads. This research is the first step towards comprehending the dynamics of project knowledge sharing within the Discussions forum. Maintainers can benefit from the results to better understand the Discussions forums' usage, their role in disseminating project knowledge, the tools and resources that community members employ to express their ideas and needs for help, and the patterns of user collaborations. For instance, we noticed that Discussions users use online code editors or playgrounds. However, this practice is limited (Table 5). We envision maintainers benefit from this finding to stimulate Discussions users to use code playgrounds to create code snippets and reproduce coding errors, making the code errors easier to replicate for easier troubleshooting.

Research Opportunities: Considering the diverse behavior of OSS communities within the Discussion forum, our findings open up promising avenues for future work. Our results show that the Discussions users share links to multiple sources of information to contextualize or clarify the content of the discussion threads. We encourage researchers to develop strategies that can identify pertinent links for specific discussion topics. We also expect this research brings evidence to motivate researchers to develop a Issues and PRs recommendation system. Such systems could suggest the most relevant issues or PRs based on the subject matter of a given discussion thread, aiding in focused collaboration. Moreover, we plan to investigate if and how projects' context (business models, age, programming languages) affect link-sharing activities on GitHub

Discussions. In addition, based on our results, which suggest that the use of the GitHub Wiki feature is relatively rare, we see this as a promising research opportunity: understanding the use of the GitHub Wiki feature. Furthermore, our results present opportunities for a more in-depth investigation into the number of link references in discussion threads. Specifically, researchers can investigate why some threads have numerous links.

A particularly rich area for future exploration is understanding the relation and interplay between Discussions, PRs, and Issues mechanisms since they all provide a means of communication and discussion that may promote the project knowledge sharing. Based on the goals of GitHub discussions, focused research in this domain could include: (1) analyzing the similarities and differences in the nature of discussions in the Issue and Discussions mechanisms, (2) exploring the dynamic interactions between Issues, PRs, and Discussions to understand their collective impact in the project collaboration, (3) investigating the phenomena of convert Issues and PRs into Discussions, and (4) investigating the role these mechanisms play in facilitating effective communication within discussion threads. For instance, Wang et al. (2023) conducted a study to understand how developers maintain and manage GitHub communication channels in OSS projects, focusing on the reasons and implications of converting discussions into issues and vice versa in two repositories. These gaps bring opportunities for future work, which can ultimately contribute to a deeper understanding of knowledge-sharing dynamics within OSS communities.

Lastly, we acknowledge the importance of examining how communities' communication extends when utilizing the Discussions feature on GitHub. This is because discussions can happen at both the repository and organizational levels, given that GitHub's design for Discussions allows for enabling instances of the forum at both the repository-specific and organizational levels (GitHub, 2021). This perspective could offer valuable insights into how organizational structures influence engagement and interaction patterns within OSS communities' discussions.

In summary, our study lays the groundwork for broad research opportunities aimed at deepening our comprehension of OSS community engagement and collaboration mechanisms. We encourage the research community to explore these topics further.

GitHub Engineering team: Our findings bring evidence regarding the use of the GitHub Discussions, some benefits of the forum to OSS communities, and how communities integrate the Discussions forum with other GitHub features (repositories, Issues, PRs, discussions, code). The findings may also bring insights to the GitHub engineering team on promoting the Wiki feature usage, integrating code playgrounds to the GitHub platform, planning strategies to host the projects' websites, and detecting duplicates automatically. These opportunities, in turn, could enhance the platform and better support software teams, for example.

6 Limitations

Although we conducted a mixed-method analysis and selected a statistically significant sampling in the qualitative study, this research may present limitations.

Threats to the internal validity relate to the link extraction step. In this research, we highlight two events that may affect the internal validity: (1) our sampling may not contain all links shared in discussion threads, (2) we assume that all extracted links impact the project knowledge dissemination, and (3) our sampling includes data from public discussion threads at the repository level.

Threats to construct validity exist in our manual classification to identify users' intentions behind link sharing. To minimize this limitation, we analyzed a representative sample of `Next.js` project links and included the inter-rater agreement rate of our classification process to reduce biases. To construct the representative sample, we set the margin of error to 10% with a 95% confidence level. This decision was a thoughtful choice aimed at balancing sample representativeness and statistical accuracy. Opting for a 10% margin of error, as opposed to smaller percentages, facilitates a realistic and feasible analysis without significantly compromising result validity. We manually examined and categorized 1,106 links, offering preliminary insights into users' intentions when sharing links in Discussions. Moreover, we investigated the project knowledge dissemination from the perspective of link sharing. We are aware of the existence of different ways to support knowledge dissemination in open-source communities. The automated classification may also threaten construct validity as we utilized specific tokens to create regular expressions, which may not cover all occurrences within a specific category. Another limitation of our study is the focus on YouTube and Loom as primary examples of video-sharing platforms, based on their prevalence in our dataset through manual inspection. This approach may not fully represent the diversity of video platforms available, potentially overlooking the presence of other video-sharing platforms in different contexts or communities.

Finally, our results are not generalizable, which may pose a threat to external validity. Activities and discussions can vary significantly across different communities, so we cannot guarantee that our findings apply to all projects hosted on GitHub. We acknowledge that we only analyzed data from 10 GitHub repositories that focus on coding (mainly in JavaScript and TypeScript), even though many repositories are available.[19] Given the use of convenience sampling, we cannot aim for representativeness. However, using convenience sampling provided advantages such as speed in creating the sample (Baltes and Ralph, 2022).

In addition, we selected GitHub communities (Table 1) based on recommendations from the GitHub engineering team and the number of discussion threads in each forum. Our sample comprises 29,251 discussion threads, from

[19] https://github.com/

which we extracted 65,621 links. Moreover, the number of links extracted from `vercel/next.js` and `dotnet/csharplang` repositories deviates from the others in the sample. The use of these repositories may affect the analysis and bias the final results. We acknowledge that more studies are required to assess the results in different communities to understand the similarities and differences across domains and communities (e.g., coding communities vs. non-coding communities).

7 Conclusion

This paper investigates the link-sharing activities within the GitHub Discussions. We conducted a mixed-method study to determine the forums' impact on knowledge dissemination of OSS projects. Our findings lead us to the conclusion that the links shared within discussion threads play a significant role in promoting knowledge sharing among members of the OSS community. The Discussions users create discussions to ask questions, make announcements, discuss ideas, report bugs, or request features. Additionally, they share explicit links to resources that offer context, support, examples, clarification, and conceptualization for the discussed subjects.

Users of Discussions share links to both internal and external GitHub information resources. These resources are not always directly related to the project development process; nevertheless, they hold significance in fostering shared understanding among users. These resources can encompass personal homepages, LinkedIn profiles, and even intentionally shared broken links to address URL-related concerns. We noticed that link-sharing activities can vary among projects. Depending on the project's community code of conduct, practices, and organization, users tend to share links to a specific resource more frequently.

We envision that the outcomes of this research can provide valuable opportunities for OSS communities, Software Engineering researchers, and GitHub Engineering teams. For OSS communities, our results help maintainers better comprehend forum usage. This includes (1) the extent to which community members link to other information resources, (2) the types of information resources referenced by community members to support project usage and development, and (3) encouraging forum adoption, avoiding the use of third-party forums. Software Engineering researchers can benefit from our results to explore different aspects of the link-sharing activities in the Discussions forum, such as examining the role of link creators within the community. Finally, the GitHub Engineering team can use our results to promote the forum's adoption.

The upcoming phases of our research will center around investigating features that may distinguish relevant links to a particular discussion topic, considering the intentions for the link sharing. To achieve this, we plan to employ deep learning strategies to classify the links according to the types of link relationships. OSS communities can benefit from the results to decrease the reading time of the discussion threads and find relevant information resources.

Furthermore, we aim to explore the consequences of sharing inaccurate references in disseminating project knowledge. This investigation will help shed light on how such instances can impact the project's overall understanding and spread of knowledge.

Acknowledgements We thank GitHub for supporting this research. We also would like to express our gratitude to the reviewers for their valuable comments and suggestions. This research was carried out within the scope of the Samsung-UFAM Project for Education and Research (SUPER), according to Article 48 of Decree number 6.008/2006(SUFRAMA). We also thank the financial support granted by CNPq through processes number 314797/2023-8 and 443934/2023-1. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES-PROEX) - Finance Code 001. In addition, this work was partially supported by FAPEAM through the POSGRAD 22-23 project. Finally, the present work is the result of the Research and Development (R&D) project 001/2020, signed with UFAM and FAEPI, Brazil, which has funding from Samsung, using resources from the Informatics Law for the Western Amazon (Federal Law nº 8.387/1991), and its disclosure is in accordance with article 39 of Decree No. 10.521/2020.

Declaration of competing interest

We declare that Grace Vorreuter (coauthor of this manuscript) participates in the GitHub Discussions engineering team. Denae Ford works for Microsoft Research. The other authors have no competing interests to declare relevant to this article's content.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the author(s) used ChatGPT in order to improve the manuscript's readability and language. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

References

Ahmed, T. and Srivastava, A. (2017). Understanding and evaluating the behavior of technical users. A study of developer interaction at StackOverflow. *Human-centric Computing and Information Sciences*, 7(1):1–18.

Aniche, M., Treude, C., Steinmacher, I., Wiese, I., Pinto, G., Storey, M.-A., and Gerosa, M. A. (2018). How Modern News Aggregators Help Development Communities Shape and Share Knowledge. ICSE '18, page 499–510, New York, NY, USA. Association for Computing Machinery.

Baltes, S. and Ralph, P. (2022). Sampling in software engineering research: A critical review and guidelines. *Empirical Software Engineering*, 27(4):94.

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

Chopra, A., Mo, M., Dodson, S., Beschastnikh, I., Fels, S. S., and Yoon, D. (2021). @ alex, this fixes# 9: Analysis of Referencing Patterns in Pull Request Discussions. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW2):1–25.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Fang, H., Klug, D., Lamba, H., Herbsleb, J., and Vasilescu, B. (2020). Need for tweet: How open source developers talk about their GitHub work on Twitter. In *Proceedings of the 17th International Conference on Mining Software Repositories*, pages 322–326.

Ferrari, A. and Esuli, A. (2019). An NLP approach for cross-domain ambiguity detection in requirements engineering. *Automated Software Engineering*, 26(3):559–598.

Ford, D., Lustig, K., Banks, J., and Parnin, C. (2018). We Don't Do That Here: How Collaborative Editing with Mentors Improves Engagement in Social Q&A Communities. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–12, New York, NY, USA. Association for Computing Machinery.

GitHub, I. (2021). Github discussions. URL: https://docs.github.com/en/discussions. Accessed: January 12, 2023.

Hata, H., Novielli, N., Baltes, S., Kula, R. G., and Treude, C. (2022). GitHub Discussions: An exploratory study of early adoption. *Empirical Software Engineering*, 27(1):1–32.

Hata, H., Treude, C., Kula, R. G., and Ishio, T. (2019). 9.6 million links in source code comments: Purpose, evolution, and decay. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 1211–1221. IEEE.

Landis, J. R. and Koch, G. G. (1977). The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.

Li, L., Ren, Z., Li, X., Zou, W., and Jiang, H. (2018). How are issue units linked? empirical study on the linking behavior in GitHub. In *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*, pages 386–395. IEEE.

Li, T., Louie, E., Dabbish, L., and Hong, J. I. (2021). How Developers Talk About Personal Data and What It Means for User Privacy: A Case Study of a Developer Forum on Reddit. *Proceedings of the ACM on Human-Computer Interaction*, 4(CSCW3):1–28.

Lima, M., Steinmacher, I., Ford, D., Liu, E., Vorreuter, G., Conte, T., and Gadelha, B. (2023). Looking for related posts on GitHub Discussions. *PeerJ Computer Science*, 9:e1567.

Liu, E. (2021). GitHub Discussions is out of beta. URL: https://github.blog/2021-08-17-github-discussions-out-of-beta/. Accessed: November 21, 2023.

Liu, E. (2022). 7 unique software collaboration features in GitHub Discussions. URL: https://github.blog/2021-11-10-7-unique-software-collaboration-features-in-github-discussions/. Accessed: November 21, 2023.

Mamykina, L., Manoim, B., Mittal, M., Hripcsak, G., and Hartmann, B. (2011). Design lessons from the fastest Q&A site in the west. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 2857–2866.

Mishra, S. and Sharma, A. (2021). Crawling Wikipedia Pages to Train Word Embeddings Model for Software Engineering Domain. In *14th Innovations in Software Engineering Conference (formerly known as India Software Engineering Conference)*, pages 1–5.

Niyogi, S. (2020). New from Satellite 2020: GitHub Discussions, Codespaces, securing code in private repositories, and more. URL: https://github.blog/2020-05-06-new-from-satellite-2020-github-codespaces-github-discussions-securing-code-in-private-repositories-and-more/. Accessed: September 17, 2023.

Pei, J., Wu, Y., Qin, Z., Cong, Y., and Guan, J. (2021). Attention-based model for predicting question relatedness on Stack Overflow. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, pages 97–107. IEEE.

Poche, E., Jha, N., Williams, G., Staten, J., Vesper, M., and Mahmoud, A. (2017). Analyzing user comments on YouTube coding tutorial videos. In *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*, pages 196–206. IEEE.

Ponzanelli, L., Bavota, G., Mocci, A., Oliveto, R., Di Penta, M., Haiduc, S., Russo, B., and Lanza, M. (2017). Automatic identification and classification of software development video tutorial fragments. *IEEE Transactions on Software Engineering*, 45(5):464–488.

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

Storey, M.-A., Zagalsky, A., Figueira Filho, F., Singer, L., and German, D. M. (2016). How social and communication channels shape and challenge a participatory culture in software development. *IEEE Transactions on Software Engineering*, 43(2):185–204.

Stray, V. and Moe, N. B. (2020). Understanding coordination in global software engineering: A mixed-methods study on the use of meetings and Slack. *Journal of Systems and Software*, 170:110717.

Wang, D., Xiao, T., Thongtanunam, P., Kula, R. G., and Matsumoto, K. (2021). Understanding shared links and their intentions to meet information needs in modern code review. *Empirical Software Engineering*, 26(5):1–32.

Wang, L., Zhang, L., and Jiang, J. (2020). Duplicate question detection with deep learning in Stack Overflow. *IEEE Access*, 8:25964–25975.

Zampetti, F., Ponzanelli, L., Bavota, G., Mocci, A., Di Penta, M., and Lanza, M. (2017). How developers document pull requests with external references.

In *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*, pages 23–33. IEEE.

Zhang, W. E., Sheng, Q. Z., Lau, J. H., and Abebe, E. (2017). Detecting duplicate posts in programming QA communities via latent semantics and association rules. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1221–1229.

Zhang, Y., Wang, H., Yin, G., Wang, T., and Yu, Y. (2015). Exploring the use of @-mention to assist software development in GitHub. In *Proceedings of the 7th Asia-pacific Symposium on Internetware*, pages 83–92.

Zhang, Y., Wu, Y., Wang, T., and Wang, H. (2020). iLinker: a novel approach for issue knowledge acquisition in GitHub projects. *World Wide Web*, 23(3):1589–1619.

Zheng, L., Albano, C. M., Vora, N. M., Mai, F., and Nickerson, J. V. (2019). The roles bots play in Wikipedia. *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW):1–20.