Understanding Link Sharing Practice in NPM Related Tweets by Package Maintainers Community

Israt Jahan Reshma, Asfak Shahriur, Yusuf Sulistyo Nugroho, Shuvroto Kumar, Dedi Gunawan, and Syful Islam*

Original scientific article

***Abstract*—Twitter is a popular platform for the JavaScript community to share their opinions and thoughts. These tweets contain valuable knowledge about the Node Package Manager (NPM) ecosystem. Links are an example of such knowledge. Therefore, a thorough investigation into the role of links in tweets by NPM maintainers can reveal noteworthy trends and patterns of information dissemination. This study investigates the prevalence, targets, purposes, categories, and decay of links in tweets shared by NPM maintainers to understand how these links connect to the larger NPM ecosystem. To accomplish our goal, we conducted a mixed method analysis of 18,408 links. Our study found that links are prevalent in tweets and majority of the tweets are related to package management. The links are mostly blog posts, tutorials, and articles, and their primary function is to provide tweet context. Surprisingly, 80% of the links are unique, while repeatedly mentioned links make up one-fifth (20%). In addition, github.com was the most frequent domain other than twitter.com, and approximately 770 (5%) of the links shared in tweets are dead. Among the dead links, the domain github.com has the highest number of these links. The results of this study indicate that referencing external resources using links is prevalent practice for the NPM maintainers community on Twitter space. In addition, we identified some research gaps and open challenges that can guide future research efforts.**

***Index Terms*—Documentation, link, NPM Maintainer, tweet.**

I. Introduction

In today's software development world, social media platforms play a crucial role in sharing knowledge [1], solving technical problems [2], building communities [3], and improving the performance of the teams [4]. For NPM maintainers, key contributors to JavaScript's package ecosystem, Twitter serves as an important space for sharing NPM-related information. This includes updates, repository links, tutorials, and other resources that assist with problem-solving, project management, and skill development. These activities contribute to the growth of the NPM community and encourage collaboration among JavaScript developers. However, the specific types of resources shared, the motivations behind sharing them, and the longevity of these links remain unclear. Investigating these aspects can provide a clearer picture of how Twitter supports and grows the NPM community's knowledge base.

On multiple platforms, the importance of link sharing in software development has been extensively documented. Research has demonstrated the significance of link sharing in the software development context. Links that are included in code comments [5], commit messages [6], and other documentation are valuable for directing developers to external resources, such as project documentation, licensing information, or bug-related issues. For example, a prior study revealed that links in source code comments are crucial for accessing external resources, like documentation and licenses for guiding development [5]. In line with it, another work examined links in commit messages and shows that those links are mainly pointing to sources such as Stack Overflow, underscoring the dependence on external resources to clarify and contextualize code changes [6].

Although these studies have shown the importance of links in other domains, their findings may not fully extend to social media platforms, specifically Twitter, a platform characterized by its conciseness and conversational tone. To better understand the implications of link sharing in this setting, it is crucial to explore the specific ways the NPM maintainers utilize links on Twitter.

Numerous studies have laid the groundwork to understand the evolution and interaction between developers and social media [7]. For instance, Storey et al. [8] highlighted the evolving role of social media in software engineering, showing how it has shaped communication, learning, and collaboration, helping the rise of the "social programmer" who actively engages in online communities. Also Zhang et al. (2025) [9] conducted a social network analysis of Twitter use by scientists, educators, and the general public using big data and discovered that Twitter is a collaborative platform for scientific discussion and knowledge sharing on scientific topics. This is in line with our exploration of how NPM maintainers use Twitter to disseminate technical information by means of links. Another study by Ali et al. [4] demonstrated that social media can be used to manage information flow in teams, thus empowering knowledge management and guiding innovation. Their study developed a model to test the influence of social media on transactive memory systems, absorptive capacities, and team innovation performance, revealing a positive impact on these dimensions within software development teams.

Manuscript received October 28, 2025; revised November 19, 2025. Date of publication June 1, 2026. Date of current version June 1, 2026.

I. J. Reshma, A. Shahriur, S. Kumar, and S. Islam are with the Department of Computer Science and Engineering, Gopalganj Science and Technology University, Bangladesh (e-mails: israt.18cse241@gstu.edu.bd, shahrierasfak27@gmail.com, syfulcse@gstu.edu.bd).

Y. S. Nugroho and D. Gunawan are with the Universitas Muhammadiyah Surakarta, Indonesia (e-mails: yusuf.nugroho@ums.ac.id, dedi.gunawan@ums.ac.id).

* corresponding author

Digital Object Identifier (DOI): 10.24138/jcomss-2025-0175

1845-6421/06/2025-0175 © 2026 CCIS

Building on this foundation, Islam et al. [10] conducted an empirical analysis of tweets by NPM maintainers to understand their perceptions and opinions better. Their study identified that most tweets belong to the package management category, followed by notifications and community-related information. It also revealed that the nature of tweets shared by NPM maintainers is predominantly informational, with sentiments ranging from positive in community discussions to neutral in package management topics. While Islam et al. provide a comprehensive overview of tweet topics, nature, and sentiment, their study does not address the specific role of links within these tweets, leaving a critical gap that this research seeks to fill.

In this study, we conduct a mixed-methods analysis of link-sharing behaviors among NPM maintainers on Twitter, focusing on the types, purposes, and decay of these links. By analyzing 18,408 links within 39,426 tweets, this research provides insights into how NPM maintainers contribute to the NPM ecosystem through Twitter and highlights areas where ethical attribution, resource preservation, and mitigation of link decay can be improved. Our findings aim to inform the development of best practices for sustainable and ethical resource sharing on social platforms, enhancing the accessibility and durability of community-driven knowledge within the JavaScript development community.

The major findings of the study are as follows: (1) Prevalence: Links are common in tweets, with 40% of tweets containing at least one link; (2) Link Targets: Most links point to educational resources, including blog posts and tutorials; (3) Link Purpose: The primary purpose of sharing links is to provide additional context for discussions on NPM topics, enhancing knowledge-sharing within the community; (4) Tweet category: Package Management and Notifications are the most frequently shared tweet categories by NPM maintainers; (5) Link Decay: Approximately 5% of links are inaccessible, with GitHub as the primary domain for broken links.

In summary, the key contributions of this paper are:

- We offer an empirical characterization of link-sharing behavior, including the prevalence of links, their targets, and their purposes, and show that the majority of links lead to educational and project-related resources that assist developers in the dissemination and collaboration of knowledge.
- We categorized the tweets according to their content to understand which of the informative categories do developers share the links more. The package management and notification formats list the first two categories where linking is the most popular, provided the high level of informative nature and report-oriented interaction among NPM package maintainer.
- We conduct a dead link analysis prone that the up to 5% of the kept link dies in time, with GitHub being the most affected domain, which proves the necessity of the retention.
- Proposed conduct guidelines for always ethical and responsible link sharing contribute to making the knowledge in the center of the community sustained, accessed, and trustworthy within the software development ecosystems.

We organize the rest of the paper as follows. Section II situates our work to the literature on tweet knowledge sharing and link sharing. Section III structures our six research questions and their motivations, details of the data collection process, and methods in qualitative and quantitative analyses. Section IV presents our research findings by answering the six aforementioned research questions. Section V discusses the role of links in tweets by NPM maintainers, comparing it with link practices in other developer contexts. Section VI acknowledges threats to the validity of our study. Section VII draws conclusions and highlights opportunities for further works.

II. Related Work

In this section, we discuss existing work related to knowledge sharing and link sharing.

A. Knowledge Sharing

Knowledge sharing is vital in software development, where developers often utilize social media to exchange information, such as Twitter, Stack Overflow, and GitHub [11], [12]. According to prior studies, developers use social media mostly to promote projects [10], interact with the community [13], [14], and offer insights on package administration for NPM maintainers [15], [7].

It is well-documented that knowledge flows across platforms like Twitter, GitHub, and Stack Overflow. Baltes et al.in [13] examined this interaction, illustrating how technical information travels across these platforms and reported that there is remarkable consistency in how NPM maintainers link to GitHub repositories in their tweets, engaging in a feedback loop from social media to code hosting platforms. According to Baltes et al. [13], there is a recurring problem of improper attribution, though. Similarly, our work reveals that NPM maintainers frequently fail to provide consistent attribution to the source of the resources they share, as it challenges ethical standards and emphasizes the importance of a greater attribution practice.

Dabbish et al. [16] observed that GitHub's transparency has been identified as an incentive for collaborative learning. On GitHub, developers can learn new skills by perusing other people's contributions. This is supported by our research, which demonstrates that NPM maintainers regularly exchange links to GitHub repositories, promoting collaboration among JavaScript developers. Wattanakriengkrai et al. [17] explored how GitHub interacts with academic work and found that the relationship between practical development and academic research remains weak. Our work also finds that, similar to NPM maintainers, academic insights can help facilitate greater development, but the integration of academic insights with practical development is an area of opportunity.

Expertise detection on platforms like Quora has also been explored. Patil et al. [18] discovered that certain activity patterns and language usage might be used to identify expert behaviors. Support for this is our results, which show that

NPM maintainers regularly post to Twitter links to share their knowledge and help teach the community at large. In line with our finding that NPM maintainers improve discussions by adding context and clarity through the links they post, Maity et al. [19] discovered that the language used in Quora queries could predict whether they would receive answers.

The ethical side of sharing knowledge, especially when it comes to giving credit, is important. Baltes et al. [13] talked about how not giving proper credit for code on platforms like GitHub can lead to ethical problems. Additionally, our study demonstrates that NPM maintainers commonly share resources from other authors, possibly without giving credit. This highlights the necessity of greater awareness and improved procedures to guarantee that those who produce or transmit information receive the credit they deserve.

Content decay dynamics are a central issue in the persistence of online information. Here, Kaleel et al. in [20] conducted a comparative study of data ephemerality in Twitter, Reddit and TikTok concluding their "platforms' moderation policies. Based on their observations, Twitter is characterized by the fastest decay as a result of intense moderation and user account deletions. These results are of direct relevance to our study on link decay as they highlight how reachability and persistence characteristics of common resources — like Twitter links — are governed by the platform dynamics.

In conclusion, NPM maintainers are key facilitators of knowledge sharing across platforms like Twitter, GitHub, and Stack Overflow. By sharing links, they promote collaboration and provide access to educational resources, contributing to the technical development of the JavaScript ecosystem. However, there are still areas that require attention, particularly regarding ethical sharing practices and the integration of academic research into practical development.

B. Link Sharing

Link sharing is integral to knowledge exchange in software communities, providing resources, context, and solutions across platforms. Previous studies offer valuable perspectives on how developers exchange links to satisfy diverse information requirements in diverse contexts.

Hata et al. [5] studied links embedded in source code comments, showing that developers often reference essential resources such as licenses and project homepages to support ongoing work. Extending this, Xiao et al. [6] analyzed links in commit messages, observing that most links point to external resources like Stack Overflow and other technical sites, underscoring the utility of external content to contextualize code changes. Also previous research on Stack Overflow users found that they frequently share and reuse external links to supplement crowdsourced knowledge [21]. Similarly, our study finds that NPM maintainers on Twitter frequently share external links to repositories and tutorials, which provide additional layers of context and knowledge beyond what is available within the code itself.

In code reviews, link sharing serves as a tool for providing context and references to past discussions, bug reports, and other project files. Wang et al. [22] studied this in the OpenStack and Qt projects, finding that shared links in code reviews are predominantly internal, streamlining understanding within teams. In our study, NPM maintainers on Twitter often use links to GitHub repositories and documentation to give further context to their tweets, allowing followers to access a broader perspective on technical matters.

Social media platforms, particularly Twitter, support knowledge-sharing behaviors among developers and researchers. Schmitt et al. [23] observed that computer scientists on Twitter share professional content, creating a network of quality information readily accessible to peers. Nizam et al. [24] found that during popular events, shared links guide users toward relevant topics, underscoring social media's role as a navigation tool for information. In parallel, our research shows that NPM maintainers' tweets often feature links that point to tutorials and technical articles, serving as resources for the developer community to enhance understanding and collaboration.

Within Q&A platforms, link sharing is a vital mechanism for distributing technical insights. Ye et al. [25] found that developers on Stack Overflow share URLs to support specific answers and solutions, enriching discussions with targeted resources. In a related study, Gómez et al. [26] found that links on Stack Overflow also promote new tools and libraries, facilitating the spread of innovations within developer circles. Our findings reflect a similar trend, with NPM maintainers frequently sharing educational resources that address specific package management questions, extending knowledge to other developers facing related challenges.

In large software ecosystems, links frequently serve to connect related projects, enhancing cross-project collaboration. For instance, Zhang et al. [27] analyzed link-sharing patterns within the Rails community and found that linking issues across projects supports collaborative work and improves issue tracking without slowing down resolution processes. In a similar vein, our research shows that NPM maintainers actively exchange resources on Twitter from other projects, enhancing interaction and information exchange across the broader NPM and JavaScript communities.

In summary, the existing literature reveals that link sharing is an essential method for knowledge dissemination, support, and collaboration across developer networks. Our research adds to this by examining NPM maintainers' link-sharing practice on Twitter, where links to documentation, repositories, and educational content help promote knowledge exchange and community-driven problem-solving.

III. Materials and Methods

In this section, we discuss the methodology of our study. In detail, we will introduce the research questions along with their goal, data collection process, and the mixed-methods analysis techniques.

A. Research Goal and Questions

The study aims to understand the target, purpose, and decay of links shared by NPM maintainers on Twitter. We designed our research questions to systematically investigate link-sharing behaviors. We now present each of these questions, along with the motivation for each.

(RQ1) *How prevalent are links in tweets?* To gain an initial intuition about links and understand the usage of links in Tweet messages, we conducted a quantitative exploration of the regularity, distribution, diversity, and spread of these links shared by NPM maintainers.

(RQ2) *What types of link targets are referenced in tweets?*

(RQ3) *What purpose do links in tweets serve?* The key motivation of RQ2 is to identify the types of link targets that NPM maintainers are likely to refer to in tweets, and RQ3 is to determine the reason why links are used in tweets. In RQ2 and RQ3, we conducted qualitative analysis to understand the nature and purpose the links serve.

(RQ4) *What are the main topics on Twitter that NPM maintainers share?*

In order to identify the topics discussed by NPM maintainers on Twitter that contain a link, we conducted qualitative analysis on a sample dataset.

(RQ5) *How many links in tweets are dead?*

To assess the long-term sustainability of shared knowledge, RQ5 investigates link decay by examining HTTP status codes, highlighting potential risks to knowledge preservation and resource reliability.

B. Data Collection

In this section, we describe the process of building the tweet link dataset. This involves two steps: Tweet text collection and link identification and recovery.

1) Tweet Text Collection: For an in-depth examination of links shared by developers, we primarily focused on NPM maintainers' Tweets. Here we used the previous Tweet dataset [10] as a source of links posted by NPM maintainers. This dataset consists of 39,426 tweets spanning from March 2010 to August 2021.

2) Link Identification and Recovery : Twitter automatically shortens all links shared in tweets to make them more concise and easier to share. For example, in the tweet: "@caitp69 @RReverser Sorry about that, I think that is either a bug in the NPM client or incorrect documentation on the website because `--offline` is meant to make no network requests at all `https://t.co/OLqp1BsxAz`."

To extract shortened URLs generated by Twitter, first, we used the following regular expression:

```
(\https://t\.co/\ S+)
```

Lastly, in order to get the original URLs from tweets, we sent HTTP queries using the Python requests package. Thus, we find that 15,896 tweets contained at least one link, contributing to the total of 18,408 identified links as shown in Table I. These 18,408 links serve as the foundation for our analysis, offering insights into link-sharing practice among NPM maintainers on Twitter. These data highlight the frequency and context in which the NPM community includes links in their tweets, offering valuable insights into the role of link-sharing in online communication and information dissemination.

TABLE I
COLLECTED TWEETS AND LINKS.

category	count	(%)
total tweets	39,426	100%
tweets with links	15,896	40.4%
tweets without links	23,530	59.6%
total identified links	18,408	-

3) Online Appendix: Our online appendix[1] contains 18,408 links associated with the information of tweets, the NPM maintainer's ID, and the date of sharing tweets.

C. Research Method

In this section, we describe the mixed-method procedure that includes quantitative analysis (RQ1 and 5) and qualitative analysis (RQ2, 3, and 4).

1) Quantitative Analysis: We performed quantitative analysis of tweets and 18,408 links in order to gain an idea of links and their usages in tweets (RQ1) and link decay (RQ5).

Link Prevalence (RQ1) In RQ1, we conducted quantitative analysis on three aspects of tweets, including tweet regularity, link existence, and popular domains in our dataset. For tweet regularity, we calculate the number of tweets shared per year, distinguishing between tweets that contain links and those that do not. For link existence, we calculate the ratio of tweets based on how many links are shared by tweets. The frequency of links in each domain is calculated for popular domains.

Link Decay (RQ5) In RQ5, we evaluated the number of accessible links and error links in the link-sharing practice of NPM Maintainers on Twitter, and we also analyzed which types of error links are most frequently shared. In detail, we examined 14,857 distinct links from our dataset. To investigate the number of inaccessible links, we accessed the web content of these unique links using Python's requests library. We identified the HTTP status codes for each link, considering 2xx codes as an indication of link accessibility.

2) Qualitative Analysis: We performed qualitative analysis of a statistically representative and stratified sample to understand the types of link targets referenced in tweets (RQ2), the purpose that links in tweets serve (RQ3), and the topics of tweets (RQ4).

Similar to the previous study [5], we divided the data into three strata: (1) links to commonly referenced domains, (2) links to sometimes referenced domains, and (3) links to rarely referenced domains. For this, the distribution was displayed on a log scale to accommodate the wide range of link frequencies as shown in Figure 1. Through visual inspection, we identified a threshold of 20 links as the boundary between commonly and sometimes referenced domains, based on an observable inflection point in the distribution pattern. We considered domains that account for a single link are regarded as rarely shared.

Table II shows the number of domains and links in each stratum. With 18,408 links in total, we used a 2 percent margin of error and a 95 percent confidence level to compute the sample size required. Using these parameters, we then

[1]https://github.com/IsratJahanR/LinkInTweetMessage

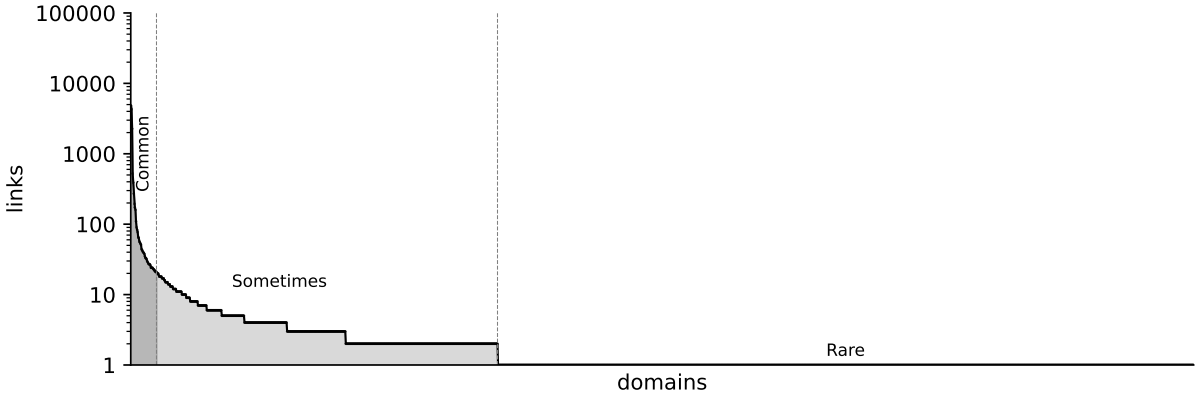

Fig. 1. Distribution of links per domain.

TABLE II
CONSTRUCTION OF THE STRATIFIED SAMPLE.

strata	#domains	#links	#links in sample
common	47	14,594	715
sometimes	616	2,557	706
rare	1,257	1,257	704
sum	1,920	18,408	2,125

drew a statistically representative and nearly equal number of randomly sampled links from each stratum. The calculation of statistically significant sample sizes is well-established (first published by Krejcie and Morgan in 1970 [28]) and is based on the population size, margin of error, and confidence level. In the results, we obtained 715 (from commonly linked domains), 708 (from domains sometimes linked) and 704 links (from rarely linked domains).

Link Target (RQ2) In RQ2, we conducted a qualitative study on the stratified sample to understand the types of link targets referenced in tweets. The qualitative analysis was carried out in multiple iterations. In the first iteration, the first, second , and fourth authors randomly selected a sample of 25 tweets and independently coded the target of the link in each tweet, refining the coding guide by merging existing codes and adding missing ones. Previous study [5] introduced the 19 coding guide for link targets in comments of source code files. We categorized link targets in our study by using the same coding guide. However, we found that not every link target in a tweet was covered by the coding guide. Therefore, we added 'repository- online storage location of software packages' as an additional coding guide for the link target. Thus, we obtained 20 coding guidelines for target links, as follows:

- 404: link does not exist (anymore) or cannot be accessed. For example "https://www.npm-stats.com/~packages/metismenu".
- license: license of a software project. For example "https://license.md/".
- software homepage: main web presence of a library or software project. For example "https://bubblin.io/fyodordostoyevsky-fyodor-dostoyevsky".
- specification: anything that resembles a requirements document or a technical standard. For example "https://styled-components.com/docs/tooling".
- organization homepage: main web presence of an organization or company. For example "http://mermaid.js.org/".
- other: anything that does not fit the other codes, including if sign-in is required. For example "https://REDACTED.s3-us-west-2.amazonaws.com/packageb-1.0.0.tgz".
- tutorial or article: technical article or tutorial, without commenting section (blog post otherwise). For example "https://www.theregister.com/2016/03/23/npm_left_pad_chaos/".
- API documentation: documentation of an API element. For example "https://getanalytics.io/plugins/amplitude/".
- blog post: technical content with a commenting section. For example "https://twitter.com/revelto/status/966710086805704704/photo/1".
- application: interactive application (e.g., web application, online utility). For example "https://feather.netlify.app/".
- bug report: bug report or issue in an online bug/issue tracker. For example "https://github.com/microsoft/node-pty/releases/tag/0.7.1".
- research paper: academic paper. For example "https://www.researchgate.net/publication/330541820_On_the_Impact_of_Outdated_and_Vulnerable_Javascript_Packages_in_Docker_Images".
- personal homepage: personal homepage of one individual. For example "https://build.sh/".
- code: a source code file. For example "https://runkit.com/gr2m/octokit-rest-js-1808".
- forum thread: thread in a forum or entire forum. For example "https://issuetracker.google.com/issues/147259775".
- GitHub profile: profile of a GitHub contributor. For example "https://github.com/47ng".
- book content: chapter/section of a book or entire book. For example "https://v2.vuejs.org/v2/cookbook/packaging-sfc-for-npm.html?redirect=true".
- Q&A thread: question-and-answer thread, but not Stack

Overflow. For example "https://github.com/insin/nwb/issues/422#issuecomment-374422125".
- Stack Overflow: question-and-answer thread on Stack Overflow. For example "https://survey.stackoverflow.co/2019#technology".
- repository: github repository. For example "https://github.com/nfroidure/ttf2woff2".

Then we calculated the Fleiss's Kappa[2] [29] for each iteration to assess inter-rater reliability among the three coders. After each iteration, we reviewed and discussed the identified targets to improve the kappa agreement in the next round. Although the initial iteration yielded a result of 0.58, the kappa score steadily improved with each iteration, ultimately exceeding the overall agreement of 0.81 or "almost perfect" in the fourth and final round [30]. Building on these promising results, we divided the 2,125 selected samples among the three authors. In case an author was unsure to determine which code was correct for a particular link—which happened in 20 cases—we discussed the classification together and jointly decided on the code. After all samples had been processed, we cross-checked each other's results to ensure consistency and accuracy, thereby ensuring that all links were coded correctly and uniformly.

Link Purpose (RQ3) In RQ3, we used the same sample and process as RQ2. Only this time, focusing on the purpose of a link (in a tweet) rather than the target of the link. We have identified 5 coding guides where 3 are taken from a previous study [22]: providing context, elaborating, and clarifying. Hence, this paper cannot cover all purposes; we added two additional purposes: suggesting solutions and others. For link purposes, the 5 coding guidelines are as follows:

- providing context: The category refers to the links that are shared to provide additional information related to the topic of discussion in tweets. For example, "npm ERR! peerinvalid The package node-inspector does not satisfy its siblings' peerDependencies requirements http://t.co/9xtGXejnnj."
- elaborating: The category refers to the links that are shared to provide additional information related to the topic of discussion in tweets. For example, "@bitandbang this action can turn on automerge for a pull request which meets whatever rules you want. e.g., a scheduled action running npm audit and making a pr, then use this action to merge when the checks pass –https://t.co/8LmG6q85er."
- clarifying: This category refers to the links that are shared to clarify some doubts or to correct the developer's understanding about the topic of discussion in the tweet. For example, "How to check for #outdated #packages via #npm #nodejs #node.js https://t.co/bNJ8aiGBmm."
- suggesting solution: We define this category as the links are shared to point out a solution/expert (other developers). For example, "When navigating through JavaScript projects, I often want a quick peek on the scripts in the package.json. Here's a quick guide on how to write a command-line script to check your npm scripts: https://t.co/4T3b3KSRh8."
- others: Links that do not fit in the above categories. For example, "In honor of Valentine's Day, I'm announcing my latest project: The npm Book. Consider it a love letter to @npmjs. https://t.co/pM6Jr4rr."

[2]Kappa agreement was calculated using http://justusrandolph.net/kappa/agreement

This time leading to an overall kappa agreement of 0.76, which indicates "substantial agreement" [30].

Tweet Category (RQ4) In RQ4, we used the same sample and process as RQ2 and RQ3 focusing on the tweet rather than links. We identified all 4 major codes from paper [10]. This coding guide covers all link categories in a tweet. The 5 coding guides are:

- Notification: Tweets that contain notification related to software package, such as media sharing, new release/update, general news, software product promotion, etc. For example "Today is my last day at @npmjs I wrote about the journey and some of the team's accomplishments leading up to the acquisition: https://t.co/I5VwHgKEgI https://t.co/INp1nLMhzu."
- Package Management: Tweets that contain package management related information such as usage scenarios, software comparison, dependency, configuration, feature information/requests, bug reports, build issues, etc. For example "@bitandbang @_developit @npmjs This is similar to OS package registries that have two kinds of packages: binary package and source package. If npm added these, I think it would solve the problem for end users. https://t.co/4Xs2WZOMlp."
- Community related: Tweets that contain community-related information such as crowd sourcing requests, community events/information, personal promotion, etc. For example: "LIVE on #Periscope: Insight into npm3 at Boston ember meetup https://t.co/d6XrSXDSWr."
- Other: Tweets that contain other information that can not be classified in the above categories, such as satire or not in English text, etc. For example:"@NinjaSquad Sur https://t.co/Oz3HCieiKO, le npm install fail Ã cause de fsevents qui n'est compatible que sur OSX (je suis sur linux)."

We validated the coding guide using the same methodology as previous RQ2 or RQ3, and obtained an overall kappa agreement of 0.80, indicating "almost perfect" [30].

IV. Results

In this section, we present the outcomes of each research question.

A. Prevalence of Links (RQ1)

To explore the prevalence of links referenced in tweets, we conducted quantitative analyses of our collected dataset in terms of tweet regularity, existence of links, diversity of domains, and popularity of domains.

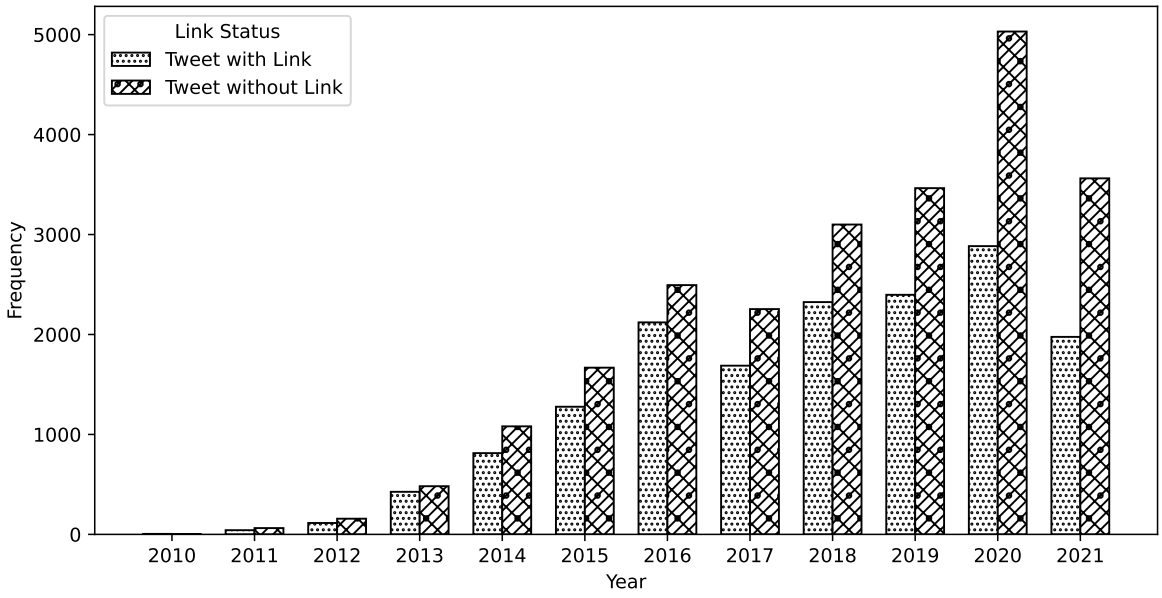

Fig. 2. Yearly shared link in tweet.

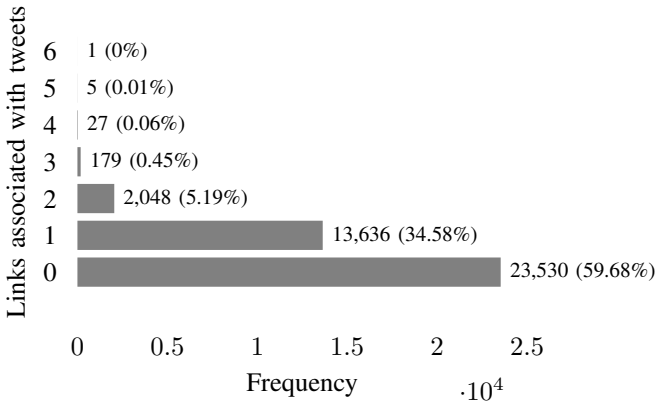

Fig. 3. Distribution of links associated with tweets.

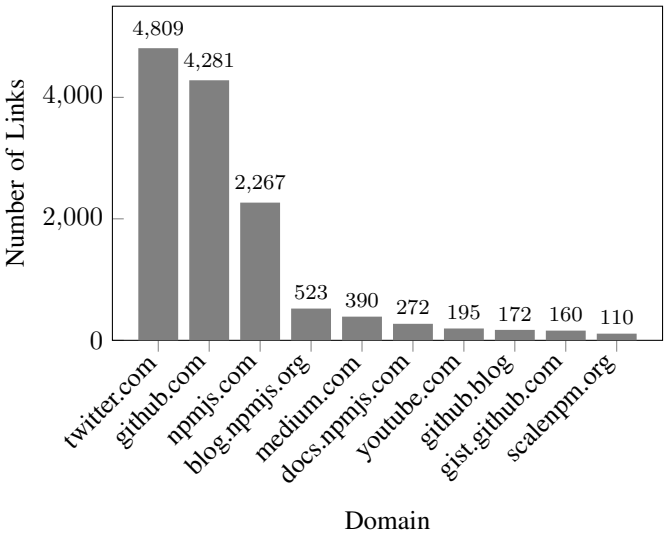

Fig. 4. Frequency of top ten most referenced domains.

Tweet Regularity: Figure 2 displays the sharing of tweets with links and without links from March 2010 to August 2021. Overall, it shows a yearly increasing trend in tweets with link sharing practice by NPM maintainers. From 2010-2013, there was a slight increase, followed by a noticeable growth in tweet frequency from 2014-2016. However, in 2017, there was a decrease, and from 2018-2019, both categories showed significant increases. The highest frequency was observed in 2020, especially for tweets without links, peaking at 5081, and tweets with links peaked at 2834. Although there was a decline in 2021, the frequency remained higher than in previous years. It is crucial to remember that the 2021 data only covers the first seven months of the year. These findings indicate that link-sharing practice by the NPM maintainers community is becoming prevalent over time.

Link Existence: We obtained 18,408 links from 39,426 tweets. Figure 3 presents the percentages of links in a tweet. We observed that 40% of tweets shared at least one link, whereas 60% of tweets did not contain any links at all. 34% of tweets have one link, whereas 5% of tweets contain two, three, four, or five links, and a maximum of six links is shared, which is from only one tweet.

Popular Domains: We obtained 1,920 distinct domains or internet hostnames out of 18,408 links. Figure 4 depicts the frequency of the top ten most referenced domains on tweets. Based on the frequency of links from the domains, we rank the top ten domains with a cutoff frequency of 110. The top 3 domains are Twitter, GitHub, and npmjs.

TABLE III
FREQUENCY OF LINK TARGET TYPES IN OUR SAMPLE.

	common		sometimes		rare	
tutorial or article	112	(16%)	209	(29%)	139	(20%)
blog post	215	(30%)	117	(17%)	78	(11%)
software homepage	58	(8%)	116	(16%)	164	(23%)
404	52	(7%)	77	(11%)	110	(16%)
API documentation	73	(10%)	26	(4%)	24	(3%)
other	27	(4%)	42	(6%)	54	(8%)
application	18	(3%)	35	(5%)	64	(9%)
repository	100	(14%)	6	(1%)	0	(0%)
code	24	(3%)	25	(4%)	12	(2%)
organization homepage	5	(1%)	17	(2%)	26	(4%)
specification	4	(1%)	10	(1%)	14	(2%)
Q&A thread	18	(2%)	4	(1%)	2	(0%)
personal homepage	0	(0%)	6	(1%)	5	(1%)
book content	1	(0%)	5	(1%)	4	(1%)
licence	0	(0%)	6	(1%)	2	(0%)
bug report	3	(0%)	1	(0%)	2	(0%)
Stack Overflow thread	4	(1%)	0	(0%)	1	(0%)
research paper	0	(0%)	3	(0%)	1	(0%)
forum thread	0	(0%)	1	(0%)	2	(0%)
GitHub profile	1	(0%)	0	(0%)	0	(0%)
sum	715	(100%)	706	(100%)	704	(100%)

TABLE IV
FREQUENCY OF LINK PURPOSES IN OUR SAMPLE.

	common		sometimes		rare	
providing context	370	(52%)	313	(44%)	307	(44%)
elaborating	129	(18%)	166	(23%)	169	(24%)
suggesting solution	78	(11%)	82	(12%)	122	(17%)
clarifying	85	(12%)	90	(13%)	65	(9%)
other	53	(7%)	55	(8%)	41	(6%)
sum	715	(100%)	706	(100%)	704	(100%)

B. Link Targets (RQ2)

Table III shows the result of our qualitative analysis for all types of domains. The table illustrates the frequency distribution of different link target types across common, sometimes, and rare occurrences in the sampled tweets. Blog posts are the most frequently linked in the common category, representing 30% of links. In the ”sometimes” category, tutorials or articles dominate with 29%, while software homepages are the most frequent in the ”rare” category, accounting for 23%. Notably, links leading to 404 errors—indicating missing or broken links—are significant across all categories, comprising 7%, 11%, and 16% of the links in common, sometimes, and rare categories, respectively. This analysis highlights that while blog posts, tutorials, and software homepages are mostly shared as key link targets in tweets.

C. Link Purpose (RQ3)

Table IV summarizes the result of different purposes of links in sampled tweets, showing that providing context is the most common use, especially in the “common” category, where it makes up 52% of all links. It also remains significant in the “sometimes” and “rare” categories, each with 44%. Elaborating is the second most frequent purpose, appearing more in the “sometimes” (23%) and “rare” (24%) categories. Suggesting a solution and clarifying are less common purposes of sharing a link in a tweet.

TABLE V
ASSOCIATIONS BETWEEN LINK TARGET TYPE AND LINK PURPOSE.

strata	association rule			conf.	supp.
common	(tutorial or article)	⇒	(providing context)	0.76	85
common	(404)	⇒	(providing context)	0.71	37
common	(application)	⇒	(providing context)	0.83	15
common	(Q&A thread)	⇒	(clarifying)	0.77	14
sometimes	(tutorial or article)	⇒	(providing context)	0.73	153
sometimes	(code)	⇒	(elaborating)	0.92	23
sometimes	(organization homepage)	⇒	(elaborating)	0.76	13
sometimes	(personal homepage)	⇒	(providing context)	0.83	5
rare	(application)	⇒	(providing context)	0.75	48
rare	(specification)	⇒	(providing context)	0.71	10
rare	(book content)	⇒	(providing context)	1.00	4

Patterns in the Relationship between Link Targets and Purposes: Based on the qualitative analysis conducted to answer RQ2 and RQ3 about the targets and purposes of links in tweets, we can now investigate the relationships between the different types of link targets and the different purposes that emerged from our qualitative analysis. To do so, we applied association rule learning using the Apriori algorithm (as implemented in the R package arules) to our data, treating each link as a transaction containing two items: its target type and its purpose. We used four as the support threshold and 0.7 as the confidence threshold, i.e., all the rules that we extracted are supported by at least four data points, and we have at least a 70% confidence that the left-hand side of the rule implies the right-hand side.

As seen in Figure 5, which contrasts link target versus link purpose, we find that research papers (100%), tutorials or articles (66%), applications (60%), book content (60%), and personal homepages (54%) related links are shared frequently for the purpose of providing context. For clarification, relevant links from Q&A threads (58%) and forum threads (67%) are frequently shared.

D. Link Category (RQ4)

Table VI details the distribution of tweet categories, highlighting how often each category appears within the sample across different frequency levels. Package Management emerges as the most dominant category, consistently leading in all three groups—41% of tweets in the “common” category, 47% in the ”sometimes,” and 46% in the “rare” category. Notifications are the second most frequent, appearing in 33% of “common” tweets, 29% in “sometimes”, and 30% in “rare” instances. The Other category, encompassing various miscellaneous topics, ranges from 17% to 20% across all frequency levels. Community-related tweets are the least represented, accounting for only 5% to 7% of the tweets. This analysis highlights that NPM maintainers primarily share links related to package management and notifications, with less emphasis on community-related topics

E. Link Decay (RQ5)

Link Retrieval Responses: There are 14,857 distinct links out of the 18,408 links that were retrieved. Table VII shows the obtained HTTP Status codes for all unique links. 92% of the connections were successfully reached, meaning that the

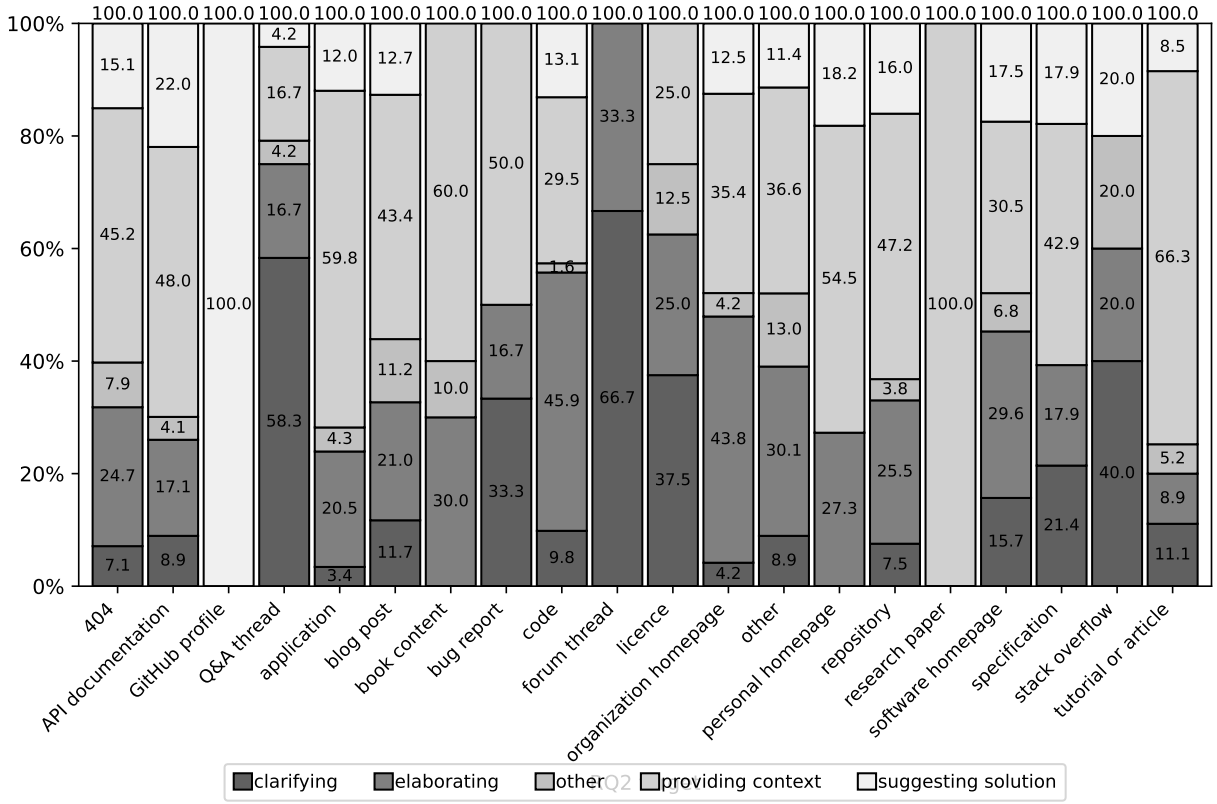

Fig. 5. Distribution of link targets vs. link purposes.

TABLE VI
FREQUENCY OF TWEET CATEGORY.

	common		sometimes		rare	
package management	292	(41%)	328	(47%)	322	(46%)
notification	237	(33%)	206	(29%)	212	(30%)
other	141	(20%)	121	(17%)	132	(19%)
community related	45	(6%)	51	(7%)	38	(5%)
sum	715	(100%)	706	(100%)	704	(100%)

TABLE VII
HTTP STATUS CODES FOR ALL UNIQUE LINKS

status	#links	(%)
2xx success	13,671	(92.01%)
404 not found	770	(5.18%)
Request failed	301	(2.02%)
410 requested resource is no longer available	36	(0.24%)
408 timeout error	36	(0.24%
others	42	(0.29%)
sum	14,857	(100%)

TABLE VIII
FREQUENCY OF '404 NOT FOUND' LINKS FOR ALL UNIQUE DOMAINS

domain	#links	(%)
github.com	213	(28%)
npmjs.com	69	(9%)
blog.npmjs.org	65	(8%)
npm-stats.com	20	(3%)
docs.npmjs.com	16	(2%)
others	387	(50%)
sum	770	(100%)

majority of the links are accessible. About 5% of the links are currently unavailable (404 Not Found), despite the fact that some HTTP status codes do not explicitly indicate that a link is broken. We observed certain trends among these 404 links: page not found, relocated or removed web content, lost Internet host, JSON-formatted response, site shut down, bogus URL, or typo in the link.

Inaccessible Links: Table VIII shows how many inaccessible unique links are generated from each domain. We identified 770 inaccessible links from 14,857 unique links (5%). Among these, Github.com is the domain with the most 404 errors, with 213 inaccessible links. Other domains have significantly fewer inaccessible links, whereas only 5 domains have more than 10 inaccessible links, and others frequency is under 10. The frequency of inaccessible links with under 10 is taken as others.

V. DISCUSSION

In this section, we have characterized how these links contribute to knowledge sharing in the JavaScript ecosystem by addressing six research questions. We have also compared our findings to practices observed in other developer contexts, such as source code comments and commit messages, and other literature. Here, we contextualize our results, discuss the implications, and reflect on future research opportunities. We also provide recommendations to developers and researchers based on our results.

A. Comparisons

To understand link-sharing behaviors, we compare our results for each research question with practices observed in source code comments, commit messages, and related literature.

Prevalence of Links (RQ1): We found that 40% of tweets by NPM maintainers contain at least one link, underscoring Twitter's role as a hub for knowledge dissemination in the JavaScript community. This prevalence is lower compared to links in source code comments (89%, [5]) and commit messages (83%, [6]) , suggesting that developers share links less frequently on social media than within development artifacts.

Link Targets (RQ2): The primary targets of links in tweets were educational resources, such as blog posts (11-30%), tutorials (16-29%), and software homepages (8-23%). By comparing this, we see that the blog post decreased from 0-6% in both source code comments and commit messages. Like this, tutorials and software homepages are also decreased in both source code comments and commit messages. The emphasis on educational content aligns with Twitter's informal and conversational nature, where NPM maintainers share resources to help others learn and solve technical problems.

Purpose of Links (RQ3): Providing context was the most frequent purpose for sharing links (44-52%), followed by elaboration (18-24%) and suggesting solutions (11-17%). This pattern mirrors findings from Wang et al. (2021) [22], who identified similar purposes for links shared during code reviews. However, metadata-focused links, which dominate commit messages, were rarely observed in tweets, indicating a shift in link-sharing practice depending on the medium and audience.

Main Topics Shared by NPM Maintainers (RQ4): Tweets by NPM maintainers primarily revolve around package management, which ranges from 41 to 47%, notifications, ranging between 29 to 33%, and community-related topics, ranging between 5 to 7%, which are in line with the results of Islam et al. 2024 Islam et al. (2024) [10]. Compared to prior studies, our analysis indicates a greater focus on technical problem-solving, reflecting the dynamic use of Twitter for real-time issue discussion.

Link Decay (RQ5): Around 5% could not be reached; of these, 28% were located on GitHub. This is a lower decay compared to source code comments (18.8%) and older SVN-linked repositories (70%) [6]. The fact that the links to GitHub are relatively stable already reflects an improvement in terms of reliability, while other resources, like personal blogs and older tutorials, remain in a fragile position. These findings therefore point out the necessity of archival practices and sustainable strategies to preserve shared knowledge over time.

Despite its constraints, this paper did provide insight into links shared by NPM maintainers on Twitter. Since the analysis targets only NPM maintainers, results cannot be generalized into other developer communities or platforms. Moreover, link accessibility was considered based solely on HTTP status codes. This may fail to consider transient issues such as server outages and geographic restrictions. Future research should investigate cross-platform link-sharing practices, conduct longitudinal studies on link accessibility, and develop automated tools that predict and mitigate link decay.

Additionally, issues such as broken links and inconsistent attribution raise concerns about long-term management and preservation strategies for shared knowledge. This makes it beneficial for the developer community to proactively focus on improving the longevity and reliability of shared resources in addition to platform-specific enhancements that support continued growth and sustainability of the JavaScript ecosystem.

B. Recommendations

Based on our findings, we make the following recommendations for developers and researchers. First, we recommend to developers:

- Adopt Archival Practices. To guarantee long term access to shared information, developers need to practice archival practices. Using technologies as the Wayback Machine, perma.cc, or DOI services can create this process more efficient by creating persistent links. The links we posted must be reviewed and possibly updated to prevent the link decay and improve the reliability of the resource. There are automated solutions such as using NPM workflows to add archival tools. Such initiatives helps in keeping the quality contributions for the bigger community.
- Enhance Awareness of Link Decay. Archiving critical resources like GitHub repositories and documentation using permanent URLs can reduce the risk of link decay. Furthermore, going for workshops or community discussion to educate people on sharing sustainable attitudes will help with the conscious act of using resources.
- Promote Ethical and Purposeful Link Sharing. Developers should align shared links with clear objectives to ensure transparency and build on trust among community members. Inspired by the findings of [6], prioritizing stable and authoritative resources enhances both the longevity and impact of shared links. For example, using permanent links for tutorials or official documentation instead of dynamically generated URLs ensures reliability over time. Adding annotations like "Tutorial" or "BugFix" improves clarity and engagement, guiding users effectively. Developers prefer the use of credible and trustworthy sources over that of transient or untrustworthy URLs.
- Boost Community Engagement. By providing links to events, guides, and cooperative projects, Twitter provides a great forum for encouraging community engagement. By actively promoting comments and discussions on shared resources [14], developers may foster a stronger sense of community. Because Twitter is itself dynamic, developers can leverage that to further amplify the visibility of their projects and build deeper, richer connections with peers and contributors to create an even more vibrant, more engaged npm ecosystem.
- Standardize Link Sharing Procedures. Establishing tweet templates or guidelines for effective link sharing will

ensure consistency and clarity in communication. In order to increase the usefulness of shared links, these templates ought to promote the addition of annotations or metadata.

Second, we recommend to researchers:

- Investigate Link Traceability Across Platforms. Researchers should explore the lifecycle of sharing links and their traceability across platforms like GitHub [17], Reddit, and Stack Overflow [21]. Identifying cross-platform patterns can uncover common practices and platform-specific behaviors. These investigations demonstrate larger implications of link sharing practices and propose possibilities for improved cross platform integration.
- Evaluate the Impact of Link Decay, Pattern and Cause. Link decay is a process studied by future research through longitudinal analysis, which identifies the pattern of decay and its underlying causes. It would also be important to study how these broken or unreachable links affect community interaction, such as hindering discussions, lessening collaboration, or reducing access to historical context. Quantifying these impacts will help both researchers and platform designers develop targeted strategies for minimizing decay and enhancing the durability of shared information.
- Create Link Decay Predictive Tools. Using machine learning models to predict and prevent link decay would offer proactive mechanisms for developers with a view to preserving shared resources. By identifying degradation before it affects users, such predictive tools improve the way platforms and developers deal with external references. Ensuring information continuity in such a way would, therefore, contribute to the long-term sustainability of the NPM ecosystem.
- Work with Platform Developers. Working with social media and repository platform designers can result in features that encourage automated preservation and sustainable link sharing. Creating tools to annotate links with metadata (such as purpose, target, and expiration date) and notify maintainers of decaying links can enhance resource management. Cooperation leads to the innovative solution, which enhances the ecosystem's resilience to resource inaccessibility.
- Examine Link Sharing Patterns. Comparing how npm maintainers use Twitter to other platforms for knowledge dissemination can reveal best practices and guide recommendations for improving link-sharing tactics. Recurring or viral links give things as to how efficient communication is and root to what researchers can provide as customized approaches for developer communities.

  Integration of these recommendations by developers and researchers together aids in extending the longevity, clarity, and utility of links shared in tweets, in a way that strengthens positive connections and maximizes social good within our npm community.

VI. Threats to Validity

In this section, we discuss the threats to the validity of our study.

Construct Validity: Threats to construct validity concern potential errors that may arise when extracting links present in tweets. For instance, using Python's requests library for link extraction and validation may have been influenced by issues like temporary server downtime, rate limits, or network restrictions, potentially affecting the accuracy of link status assessments. The downside of using regular expressions for detecting URLs is that they can miss nonstandard or uncommon URLs.

Content Validity: We manually classified the sample in our analysis, which increases the possibility of unidentified link targets and link purposes. We noticed that 5% of the links were broken, which represents how quickly content may change online. To handle this, we marked these broken links as "unknown," but this may impact the results reported for the purpose or target of these links.

Internal Validity: Errors in the analysis and experimental bias are threats to internal validity. To answer RQ2–4, we conducted qualitative studies of a sample of all links in our dataset. These qualitative studies are manual analyses that were conducted according to our coding guides. These codes may be inadequate since coding guides are subjective in meaning. To reduce this threat, we require Fleiss's kappa agreements [29] of at least "Substantial agreement" for the understanding of the coding guides in RQ2–4, following previous work (Wang et al., 2021 [22]; Xiao et al., 2021 [6]). Relying on HTTP status codes for link checking might have led to mislabeling some links if there were temporary server issues or regional access limits. Thus, we believe that there was less experimental bias and analysis error.

External Validity: Although we analyzed the links in tweets shared by the NPM maintainer, the findings may not generalize to other industries or social media platforms. Nevertheless, our study is consistent with previous research that has also relied on Twitter data and the practice of sharing links.

VII. Conclusions and Future Works

In this study, we investigate the link-sharing practice of NPM maintainers on Twitter space, directed by six research questions to understand the role, types, and existence of links in tweets. Through quantitative analysis, we explore the prevalence of links within tweets (RQ1) and how the link decay occurs (RQ6). In contrast, through qualitative studies of a stratified sample, we determine the specific targets of links (RQ2), purposes of links(RQ3), or types of tweets(RQ4) shared by NPM maintainers. These questions help characterizing how NPM maintainers use Twitter to share and maintain knowledge and the difficulties they encounter in maintaining the accessibility of information over time.

Our findings reveal that NPM maintainers use links extensively, with approximately 40% of tweets containing links, primarily to GitHub and NPM-related domains (RQ1). The targets of these are to share knowledge, including blog posts, tutorials, and documentation pages, with a predominant purpose of providing context and additional information to discussions (RQ2, RQ3). This behavior highlights the NPM maintainers' role for the JavaScript community. In terms of tweet content,

package management and notifications emerged as the most common topics, indicating a focus on sharing technical updates and resources (RQ4). However, link decay presents a challenge, with 5% of links among 14,857 unique links found to be inaccessible (RQ6).

We open up exciting opportunities for future exploration, such as expanding this study to other software ecosystems, link traceability, link decay prediction, tool support for better link maintenance, etc. Finally, based on the key findings, we discover future research areas, by which we hope, together with the findings, can offer valuable insights to practitioners of this research area.

REFERENCES

[1] A. Masood, Q. Zhang, M. Ali, G. Cappiello, and A. Dhir, "Linking enterprise social media use, trust and knowledge sharing: paradoxical roles of communication transparency and personal blogging," *Journal of Knowledge Management*, vol. 27, no. 4, pp. 1056–1085, 2023.

[2] Y. Zhang, H. Wang, G. Yin, T. Wang, and Y. Yu, "Social media in github: the role of@-mention in assisting software development," *Science China Information Sciences*, vol. 60, pp. 1–18, 2017.

[3] M.-A. Storey, C. Treude, A. Van Deursen, and L.-T. Cheng, "The impact of social media on software engineering practices and tools," in *Proceedings of the FSE/SDP workshop on Future of software engineering research*, 2010, pp. 359–364.

[4] A. Ali, W. Bahadur, N. Wang, A. Luqman, and A. N. Khan, "Improving team innovation performance: Role of social media and team knowledge management capabilities," *Technology in Society*, vol. 61, p. 101259, 2020.

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

[6] T. Xiao, S. Baltes, H. Hata, C. Treude, R. G. Kula, T. Ishio, and K. Matsumoto, "18 million links in commit messages: purpose, evolution, and decay," *Empirical Software Engineering*, vol. 28, no. 4, May 2023. [Online]. Available: http://dx.doi.org/10.1007/s10664-023-10325-8

[7] M. Amable and A. Dávila, *Knowledge Sharing in Software Development: A Tertiary Study*. Springer Nature Switzerland, 2024, pp. 175–187.

[8] M.-A. Storey, L. Singer, B. Cleary, F. Figueira Filho, and A. Zagalsky, "The (r) evolution of social media in software engineering," *Future of software engineering proceedings*, pp. 100–116, 2014.

[9] M. Zhang, L. Lundgren, and H. Nguyen, "An online scientific twitter world: Social network analysis of# sciencetwitter,# scicomm, and# academictwitter," *Journalism and Media*, vol. 6, no. 4, p. 159, 2025.

[10] S. Islam, Y. S. Nugroho, C. M. Shahrear, N. Wahed, D. Gunawan, E. W. Pamungkas, M. H. Kabir, Y. I. Kurniawan, and M. K. Uddin, "An empirical study of software ecosystem related tweets by npm maintainers," *PeerJ Computer Science*, vol. 10, p. e1669, 2024.

[11] P. Sarka and C. Ipsen, "Knowledge sharing via social media in software development: a systematic literature review," *Knowledge Management Research & Practice*, vol. 15, no. 4, pp. 594–609, 2017.

[12] W. Okong'o and J. R. A. Ndiege, "Knowledge sharing in open-source software development communities: a review and synthesis," *VINE Journal of Information and Knowledge Management Systems*, vol. 55, no. 3, pp. 622–649, 2023.

[13] S. Baltes, L. Dumani, C. Treude, and S. Diehl, "Sotorrent: reconstructing and analyzing the evolution of stack overflow posts," in *Proceedings of the 15th international conference on mining software repositories*, 2018, pp. 319–330.

[14] K. Dworatzyk and T. Schlauch, "Sharing is caring: The effect of knowledge exchange workshops on community commitment among research software engineers," in *International Conference on Human-Computer Interaction*. Springer, 2025, pp. 157–174.

[15] H. Fang, D. Klug, H. Lamba, J. Herbsleb, and B. Vasilescu, "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*, 2020, pp. 322–326.

[16] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in github: transparency and collaboration in an open software repository," in *Proceedings of the ACM 2012 conference on computer supported cooperative work*, 2012, pp. 1277–1286.

[17] S. Wattanakriengkrai, B. Chinthanet, H. Hata, R. G. Kula, C. Treude, J. Guo, and K. Matsumoto, "Github repositories with links to academic papers: Public access, traceability, and evolution," *Journal of Systems and Software*, vol. 183, p. 111117, 2022.

[18] S. Patil and K. Lee, "Detecting experts on quora: by their activity, quality of answers, linguistic characteristics and temporal behaviors," *Social network analysis and mining*, vol. 6, pp. 1–11, 2016.

[19] S. K. Maity, A. Kharb, and A. Mukherjee, "Analyzing the linguistic structure of question texts to characterize answerability in quora," *IEEE Transactions on Computational Social Systems*, vol. 5, no. 3, pp. 816–828, 2018.

[20] A. Kaleel, B. K. Alhajri *et al.*, "Platform-specific data decay patterns: A comparative study of twitter, reddit, and tiktok," *MEDAAD*, vol. 2025, pp. 8–26, 2025.

[21] J. Liu, H. Zhang, X. Xia, D. Lo, Y. Zou, A. E. Hassan, and S. Li, "An exploratory study on the repeatedly shared external links on stack overflow," *Empirical Software Engineering*, vol. 27, no. 1, p. 19, 2022.

[22] D. Wang, T. Xiao, P. Thongtanunam, R. G. Kula, and K. Matsumoto, "Understanding shared links and their intentions to meet information needs in modern code review: A case study of the openstack and qt projects," *Empirical Software Engineering*, vol. 26, pp. 1–32, 2021.

[23] M. Schmitt and R. Jäschke, "What do computer scientists tweet? analyzing the link-sharing practice on twitter," *Plos one*, vol. 12, no. 6, p. e0179630, 2017.

[24] N. Nizam, C. Watters, and A. Gruzd, "Link sharing on twitter during popular events: Implications for social navigation on websites," in *2014 47th Hawaii International Conference on System Sciences*. IEEE, 2014, pp. 1745–1754.

[25] D. Ye, Z. Xing, and N. Kapre, "The structure and dynamics of knowledge network in domain-specific q&a sites: a case study of stack overflow," *Empirical Software Engineering*, vol. 22, pp. 375–406, 2017.

[26] C. Gómez, B. Cleary, and L. Singer, "A study of innovation diffusion through link sharing on stack overflow," in *2013 10th working conference on mining software repositories (MSR)*. IEEE, 2013, pp. 81–84.

[27] Y. Zhang, Y. Yu, H. Wang, B. Vasilescu, and V. Filkov, "Within-ecosystem issue linking: a large-scale study of rails," in *Proceedings of the 7th international workshop on software mining*, 2018, pp. 12–19.

[28] R. V. Krejcie and D. W. Morgan, "Determining sample size for research activities," *Educational and psychological measurement*, vol. 30, no. 3, pp. 607–610, 1970.

[29] J. L. Fleiss, "Measuring nominal scale agreement among many raters." *Psychological bulletin*, vol. 76, no. 5, p. 378, 1971.

[30] A. J. Viera, J. M. Garrett *et al.*, "Understanding interobserver agreement: the kappa statistic," *Fam med*, vol. 37, no. 5, pp. 360–363, 2005.

Israt Jahan Reshma is an ABAP Programmer at Eitekh ERP Ltd, where she works on developing and maintaining SAP-based enterprise solutions. She received her B.Sc. degree in Computer Science and Engineering from Gopalganj Science and Technology University, Gopalganj-8105, Bangladesh. Her professional and research interests include Software Ecosystems, Machine Learning, and Human–Computer Interaction.

Asfak Shahriur received the B.Sc. degree in Computer Science from Gopalganj Science and Technology University. He is currently a Software Engineer at Trickcel, where he contributes to software development projects. His research interests include social media analytics, software ecosystems, and knowledge-sharing practices in developer communities.

Yusuf Sulistyo Nugroho is an associate professor at the Department of Informatics Engineering, Universitas Muhammadiyah Surakarta, Indonesia. He received his Ph.D degree from Nara Institute of Science and Technology, Japan, in 2020. His research interests include Empirical Software Engineering, Software Documentation, and Mining Software Repositories.

Shuvroto Kumar received his B.Sc. in Computer Science and Engineering from Gopalganj Science and Technology University. His research interests focus on data science, machine learning, and modern software development practices. He is passionate about exploring innovative solutions to real-world problems through computational approaches.

Dedi Gunawan is an Associate Professor in the Department of Informatics Engineering, Universitas Muhammadiyah Surakarta, Indonesia. He received a Ph.D degree from Kanazawa University, Japan, in 2019. Currently, his research interests include data mining, machine learning, privacy-enhancing technology, data anonymity, privacy-preserving data mining, and privacy-preserving data publishing.

Syful Islam is an assistant professor in the Department of Computer Science and Engineering, at Gopalganj Science and Technology University (Former Name: Bangabandhu Sheikh Mujibur Rahman Science and Technology University), Gopalganj-8105, Bangladesh. He received his M.E. and Ph.D. degrees from the Nara Institute of Science and Technology, Japan. His research interests include Software Ecosystems, mining Stack Overflow, mining Software Repositories, etc.