

# An Exploration of npm Package Co-Usage Examples from Stack Overflow: A Case Study

Syful ISLAM<sup>†a)</sup>, Dong WANG<sup>†</sup>, Raula GAIKOVINA KULA<sup>†</sup>, *Nonmembers*, Takashi ISHIO<sup>†</sup>, *Member*,  
and Kenichi MATSUMOTO<sup>†</sup>, *Fellow*

**SUMMARY** Third-party package usage has become a common practice in contemporary software development. Developers often face different challenges, including choosing the right libraries, installing errors, discrepancies, setting up the environment, and building failures during software development. The risks of maintaining a third-party package are well known, but it is unclear how information from Stack Overflow (SO) can be useful. This paper performed an empirical study to explore npm package co-usage examples from SO. From over 30,000 SO question posts, we extracted 2,100 posts with package usage information and matched them against the 217,934 npm library package. We find that, popular and highly used libraries are not discussed as often in SO. However, we can see that the accepted answers may prove useful, as we believe that the usage examples and executable commands could be reused for tool support.

**key words:** package managers, npm, Stack Overflow

## 1. Introduction

Usage of third-party packages in contemporary software development has become a common practice by developers. For example, npm (i.e., Node.js package manager) is by far the largest package manager, allowing developers to reuse functionality instead of creating their own, saving both time with minimal efforts. The npm ecosystem has provided over 800,000 free and reusable software libraries and is trusted by over 11 million developers around the world\*.

Despite these benefits of using packages, developers constantly face a variety of issues when using them. Dietrich et al. [1] performed a case study and showed that partial package upgrades have high potential to introduce binary incompatibility problems at build time. Raemaekers et al. [2], [3] pointed out that developers are wary of the inherent costs and risks of package incompatibilities when integrating new and unknown packages into their systems. Most prior work have explored package usage [4], [5], developing package recommendation tools [6]–[11].

Previous studies reported that question-answering websites such as Stack Overflow (SO) are useful for communicating developers' issues. Several studies have been conducted on SO resources including source code utilization [12], analogical libraries recommendation [13], fixing runtime exception [14], improving API documentation [15],

API usage scenarios [16] and so forth. Other studies have focused on more interviews and surveys of developers [17], [18]. We hypothesize that the library usage information from SO may also be beneficial for developers. While the risks of maintaining third-party libraries are well known, it is still unclear that whether the library usage information mined from question-answering sites are useful or not in maintaining libraries.

To fill this gap, in this paper, we perform an exploratory study on package usage information from SO in term of co-usage relationship. As defined by Todorov et. al [19], co-usage is the pattern of package dependencies that are used together. The rationale behind refining the co-usage relationship is to study problems caused by npm packages. In particular we investigate (i) whether we can detect package usage (i.e., co-usage) information from SO and (ii) what the developers are looking for to solve problems related to the package. To address these, we study over 2,100 SO posts and matched them to 217,934 npm library packages. We reveal the following valuable lessons along the way:

**Lesson 1:** We find that only three out of the top ten of the most used npm libraries are mentioned in SO. The top-3 discussed npm packages are react, typescript, and webpack. Again, the top-5 libraries that are less frequently discussed in SO are mocha, eslint, chai, babel-core, and lodash. One possible reason is that, well-known libraries are well documented and may have their own forum, chat tools, etc. For this reason, there is no need to discuss them in SO. Furthermore, we find that 87.95% of package co-usage mined from SO exist in the latest npm package release.

**Lesson 2:** Developers post answers provided with usage example or execute command. Results do indicate the potential for a recommendation system, especially with the available execute commands and examples.

Although SO has been a useful resource for finding answers to questions, we find that popular and highly used libraries are not discussed as often. However, we can see that the accepted answers may prove useful, as we believe that the usage examples and executable commands could be reused for tool support.

The remainder of the paper is organized as follows. Section 2 presents motivating example and research questions. Section 3 describes the data preparation. Section 4

Manuscript received February 26, 2021.

Manuscript revised July 14, 2021.

Manuscript publicized October 11, 2021.

<sup>†</sup>The authors are with the Graduate School of Science and Technology, Nara Institute of Science and Technology, Ikoma-shi, 630–0192 Japan.

a) E-mail: islam.syful.il4@is.naist.jp

DOI: 10.1587/transinf.2021MPP0003

\*<https://www.npmjs.com/>

presents the analysis approach. Section 5 reports the results for each research question. Section 6, discusses the implications from this study. Section 7, presents the related works and the research gap. Section 8 discloses the threats to validity of our study. Finally, we conclude the paper in Sect. 9.

## 2. Motivating Example

Recent studies point out that SO is a useful question-answering site among developers to communicate various issues [13]–[16], [20]. In this paper, our motivation is to investigate the following assumptions:

- Package usage information mined from SO is useful to solve developers issues while using libraries.
- Developer responses to package usage information in SO follow some useful patterns that might be reused by the recommendation tools.

Figure 1 shows an example of a package co-usage related question post from SO<sup>†</sup>. As shown in the figure, a developer posts a question on the error issue of node\_module installation, resulting from two dependent packages `babel-loader` and `webpack`. A closer look into the question description, we observe that the successful installation of `babel-loader@7.1.2` requires the package dependency of `webpack@2 || 3`. This issue is solved by a simple installation command (i.e., `npm install webpack -g`) mentioned in the accepted answer of the question. Such a motivating example suggests that package usage information mined from question answering sites may help solve package-related issues.

**Research Questions:** Our goal in this paper is to investigate whether package usage information mined from SO can help maintain the packages. We formulate two research questions to guide our study.


- **RQ<sub>1</sub>: What proportion of package usage information mined from Stack Overflow exist in npm packages?**

**Motivation.** Developers often share package usage information to communicate various software development issues through SO. We would like to understand what is the difference in the package usage information between SO and npm projects.

- **RQ<sub>2</sub>: What kinds of answers are posted in response to questions that include package usage information?**

**Motivation.** This research question investigates the developer's response to package usage information discussed in SO posts. We argue that a closer look at the answers may reveal practical insights to improve real developers' experience dealing with package co-usage issues.

Question


**babel-loader@7.1.2 requires a peer of webpack@2 || 3 but none was installed**  
Asked 3 years, 7 months ago · Active 2 years, 4 months ago · Viewed 12k times

I am having this issue while installing all node\_modules. And this is making me crazy.

6

babel-loader@7.1.2 requires a peer of webpack@2 || 3 but none was installed

2

Here is my package.json file

Throws Error

```

{
  "name": "react-router-firebase-auth",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "babel-core": "^6.26.0",
    "babel-loader": "^7.1.2",
    "babel-plugin-transform-es2015-modules-commonjs": "^6.26.0",
    "babel-preset-es2015": "^6.24.1",
    "babel-preset-react": "^6.24.1",
    "react-scripts": "0.9.5"
  },

```

Package co-usage example

I am using `create-react-app` for this project. So i could not change `webpack.config.js` file. What am i supposed to do here?

javascript reactjs npm babel-loader

Accepted Answer

Please read this post. It describes what a peer dependency is.

6 <https://stackoverflow.com/a/34645112/2379376>

What that means is that you have webpack not installed at all or you have a different version (webpack 1.x) installed. But this plugin needs webpack in version 2 or 3 to function properly.

What you can do is

```
npm install webpack -g
```

So npm will install the newest version of webpack on your system. But now other peer warnings could occur when other loaders need an older version of webpack.

**Fig. 1** A motivating example of npm package co-usage in Stack Overflow. The example shows that a developer encounters a issue when installing all node\_modules, due to two dependent packages `babel-loader` and `webpack`<sup>††</sup>.

## 3. Data Preparation

Our study exclusively examines the npm package usage information from Stack Overflow. Stack Overflow is the largest and most popular question-answering site among developers, which allows them to ask developers and experts for development related questions. In addition, to compare with the package usage from the reality, we collect another dataset from the `libraries.io`<sup>††</sup>. `libraries.io` is pop-

<sup>†</sup><https://stackoverflow.com/questions/46742824>

<sup>††</sup><https://libraries.io/>

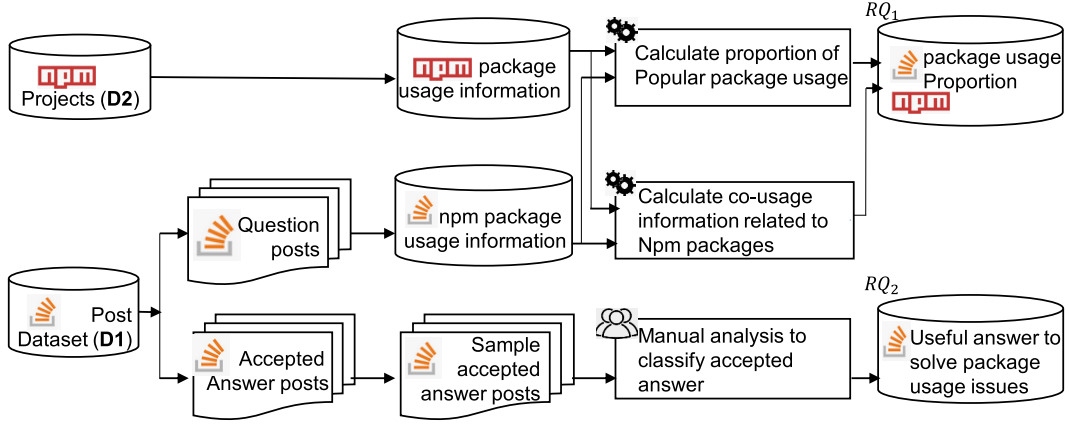


Fig. 2 An overview of the methodology of our study.

ularly known to monitor package releases. Below, we describe the details of two studied datasets.

**(D1) from Stack Overflow posts:** We rely on the SO-Torrent [21] to download the Stack Overflow data dump. We collect npm related question posts using a semi-automatic method. To do so, we first extract all tags from the question posts, and then we manually check whether or not the tags are directly related to the npm packages. After the examination, a list of eight tags that reflect npm packages posts are generated, i.e., ‘npm’, ‘npm-install’, ‘npm-script’, ‘npm-ignore’, ‘pnpm’, ‘npm-shrinkwrap’, ‘npm-start’, ‘npm-build’. We automatically identify all question posts using the defined tag list, resulting in 30,136 questions related to npm packages.

Next we further extract the npm related questions that contain the package usage information. We observed that several package names are as common as the natural language, i.e., `i`, `moment`, `should`, `express`, etc.). Thus, to reduce the bias, we only take those question posts that contain `package.json` files, resulting in 2,805 question posts. As we focus on the relatively popular libraries, we extract all packages from these question posts and sort out 628 npm packages whose frequency are greater than ten.

To ensure that our sample dataset contains most npm libraries, we use the cumulative sampling method to assure that our question posts are saturated to cover all 628 npm packages. Finally, our Stack Overflow npm package usage dataset consists of 2,100 question posts, as shown in Table 1.

**(D2) from npm packages:** To compare the package usage information with SO ones, we construct a dataset consisting of npm packages from `libraries.io`. To do so, we first downloaded the latest history data dump, which is available at <https://libraries.io/data>, resulting in a total number of 1,005,955 npm project release history.

Similar to the (D1), we extract the libraries from these 1,005,955 projects and sort out 23,870 npm packages whose frequency are greater than ten. In the end, our `libraries.io` npm package usage dataset consists of 217,934 npm projects using the cumulative sampling method, as shown in Table 1.

Table 1 Summary of dataset used in the study.

Data Source	Initial dataset	Final dataset
D1: SO (npm question posts)	30,136	2,100
D2: libraries.io (npm projects)	100,5955	217,934

## 4. Data Analysis

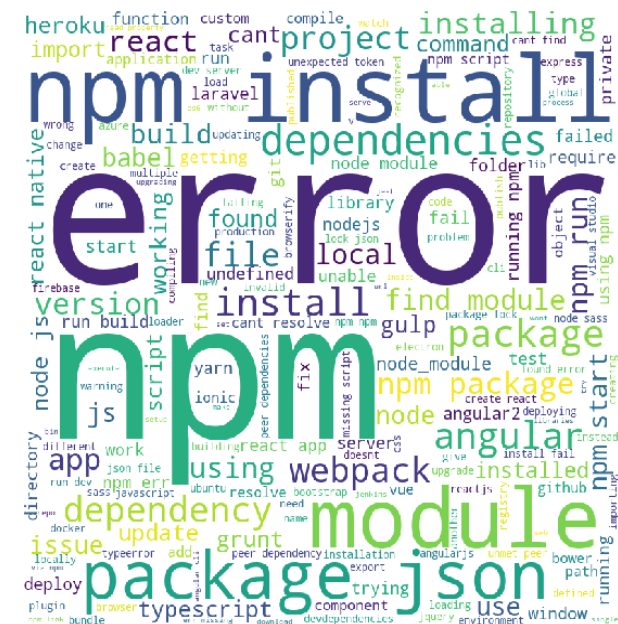
In this section, as shown in Fig. 2, we describe in detail the approaches we use to answer two research questions.

### 4.1 Approach for RQ1

We perform an exploratory study to understand to which extents do developers discuss the package usage information from SO. Below, we describe our approach in detail.

**Proportion of popular package usage:** To analyze the proportion of package usage in SO and npm projects, we extracted libraries from SO posts obtained in datasets D1 and D2, separately. Afterward, we count and sort these packages based on the frequency.

**Co-usage information related to npm package:** To analyze the frequency of package usage information, we first need to identify npm package co-usage. To do so, we follow the two steps: (I) we extract all the target packages appearing in the code snippets from 2,100 SO post related to npm package obtained in Sect. 3 (D1). Then we generate all possible binary combinations for co-usage of npm packages. For example, if a project contain three package dependencies ( $A, B, C$ ), then the generated list of binary package co-usage will be: ( $A, B$ ), ( $A, C$ ), ( $B, C$ ). After this step, we were able to retrieve 68,750 npm package co-usage information from SO. (II) we then extract the npm package co-usage information based on the latest release, referring to 217,934 npm projects in Sect. 3 (D2). Finally, we check the occurrences of SO npm package co-usage information in the generated package co-usage from latest npm projects using the following formula:  $\frac{\alpha}{\beta} \times 100$  where  $\alpha$ =Number of SO npm package co-usage found in the latest npm project release, and  $\beta$ =Total npm package co-usage extracted from



so.

To visually understand the package related issues asked in the SO, we generate a word cloud based on the constructed title corpus.

## 4.2 Approach for RQ<sub>2</sub>

**Representative sample construction:** As the full set of our constructed data is too large to manually examine their accepted answers, we then draw a statistically representative sample. The representative sample consists of 286 randomly selected accepted answer, with a confidence level of 95% and a interval of 5<sup>††</sup>.

Why don't you use angular-cli to generate angular 4 project: Here is the link: <https://cli.angular.io/> ➡ Usage examples

Steps that you need to follow:

1. Install the Angular CLI:  
Step by step instruction  
`npm install -g @angular/cli` ➡ Execute command
2. Generating and serving an Angular project :  

```
ng new my-project
cd my-project
ng serve
```
3. open <http://localhost:4200/> in your browser

**Manual coding:** We adopt three rounds to do our manual coding. First, the three authors independently sampled 25 random questions and conducted an open discussion to establish an initial code taxonomy. Hence, the following three codes emerged from our manual analysis in the first round.

- In the second round, to assure that there is no new cases, the three authors coded another 25 samples and we found that the initialized codes can totally fit. In the third round, to test the comprehension understanding, we coded another 20 samples and used the Kappa score to measure the agree-

<sup>††</sup><https://www.surveysystem.com/sscalc.htm>



ment. The score is 0.82, which is implied as nearly perfect [22]. Based on this encouraging result, the first author then coded the rest of the representative sample independently.

## 5. Results

In this section, we provide the results for each research question. First, we describe the result analysis, and then highlight our findings and answer each question.

### 5.1 Answering RQ<sub>1</sub>

To show the proportion of popular package usage, we depict tables to statistically show the package usage between SO and npm projects. Then, to analyze the frequency of So npm package co-usage in the latest npm projects, we calculated the ratio using formula (i.e.,  $\frac{\alpha}{\beta} \times 100$ ) discussed in the approach.

**Observation 2- Only three out of top-10 npm packages are mostly discussed in SO.** Table 2 shows the top-15 packages discussed in SO with their proportion and ranks in the latest npm projects. The top-3 discussed npm packages are `react`, `typescript`, and `webpack`. Again, Table 3 shows the top-15 package usage extracted from the latest npm projects. We observe that, the top-5 packages in the latest npm projects which are less frequently discussed in SO are `mocha`, `eslint`, `chai`, `babel-core`, and `lodash`. One possible reason is that, such well-known libraries are well documented and may have their own forum, chat tools, etc. For this reason, there is no need to discuss them in SO.

**Observation 3- 87.95% of the SO package co-usage information exist in the latest npm project release.** Furthermore, Table 4 shows the top-15 SO package co-usage mentioned by developers. We observed that most of the package co-usage mentioned by developers are related to `angular` followed by (`'typescript'`, `'zone.js'`). The top co-

**Table 2** Top-15 npm packages extracted from SO posts with their proportion and rank in the latest npm projects. Result shows that Only three out of top-10 npm packages are mostly discussed in SO.

npm packages	Count (SO)	Count (npm projects)	Rank (SO)	Rank (npm projects)
react	586	42,591	1	9
typescript	548	57,864	2	4
webpack	489	52,453	3	5
rxjs	471	12,339	4	69
zone.js	462	8,570	5	119
react-dom	461	32,941	6	16
@angular/core	434	10,050	7	95
@angular/common	434	9,406	8	103
@angular/compiler	426	9,170	9	110
@angular/platform-browser	424	8,482	10	120
@angular/platform-browser-dynamic	419	7,634	11	132
jquery	413	9,263	12	108
@angular/forms	401	6,608	13	147
@angular/http	388	5,433	14	169
@angular/router	380	5,583	15	166

usage patterns from SO and their rank hints that developers face most error type issues when they use angular packages.

**RQ<sub>1</sub> Summary:** Our analysis result shows that, only three out of top-10 npm packages are mostly discussed in SO. In addition, 87.95% of the SO npm package co-usage information exist in the latest npm project release.

### 5.2 Answering RQ<sub>2</sub>

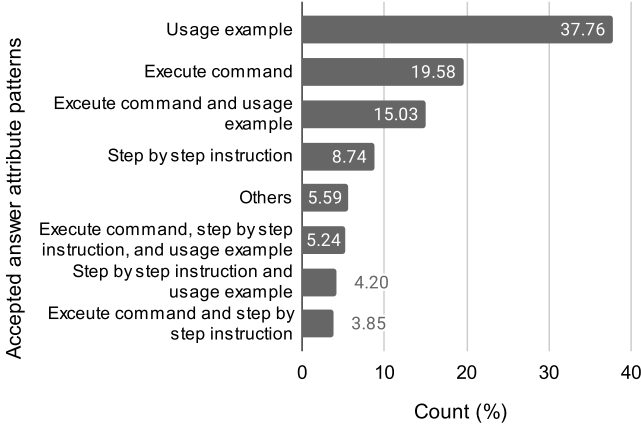
To show the useful accepted answer attributes pattern in response to the package usage question by developers, we prepare all possible combinations for three manually curated

**Table 3** Top-15 package usage extracted from the latest npm projects with their proportion and rank in the SO posts. The top package usage patterns from npm projects shows that application developers top usage packages are different from SO.

npm package	Count (npm projects)	Count (SO)	Rank (npm projects)	Rank (SO)
mocha	101898	126	1	65
eslint	81767	199	2	45
chai	58368	114	3	78
typescript	57864	548	4	4
webpack	52453	489	5	5
babel-core	51351	309	6	26
lodash	45618	348	7	19
babel-loader	44398	340	8	22
react	42591	586	9	1
jest	41188	115	10	76
babel-eslint	39336	134	11	68
babel-cli	38038	102	12	83
eslint-plugin-import	36019	94	13	90
@types/node	35197	320	14	24
rimraf	34466	139	15	63

**Table 4** Top-15 package co-usage extracted from SO posts except angular since rest of the top co-usage are related to angular. The top co-usage patterns and their rank in SO indicate that developers discuss most package dependency issues related to `angular` followed by (`'typescript'`, `'zone.js'`).

Package Co-usage	Rank	Count
( <code>'typescript'</code> , <code>'zone.js'</code> )	9	317
( <code>'zone.js'</code> , <code>'rxjs'</code> )	15	290
( <code>'react-dom'</code> , <code>'react'</code> )	17	288
( <code>'typescript'</code> , <code>'rxjs'</code> )	21	283
( <code>'karma'</code> , <code>'karma-jasmine'</code> )	31	258
( <code>'zone.js'</code> , <code>'core-js'</code> )	33	251
( <code>'core-js'</code> , <code>'rxjs'</code> )	39	233
( <code>'webpack'</code> , <code>'babel-loader'</code> )	40	230
( <code>'typescript'</code> , <code>'core-js'</code> )	40	230
( <code>'jasmine-core'</code> , <code>'karma-jasmine'</code> )	43	225
( <code>'karma-jasmine'</code> , <code>'karma-chrome-launcher'</code> )	44	223
( <code>'babel-core'</code> , <code>'babel-loader'</code> )	45	220
( <code>'typescript'</code> , <code>'tslint'</code> )	46	216
( <code>'typescript'</code> , <code>'karma'</code> )	47	210
( <code>'typescript'</code> , <code>'karma-jasmine'</code> )	48	209



**Fig. 5** Analysis of accepted answers posted in response to questions that include package usage information. Result shows that 37.76% accepted answers contain usage example followed by execute command 19.58%.

attributes: Execute command, Step by step instruction, and Usage example, respectively. Thus, we obtain eight distinct combinations (i.e., subsets), including the others (i.e., empty set). Finally, we calculate the percentage of each variety in the manually analyzed representative sample.

**Observation 4-** Our results show that, accepted answers posted by developers in response to questions that include package usage information mostly contain usage examples (i.e., includes real-life examples, external links, source code, build configuration files, etc.). Figure 5 shows the analysis result of accepted answers posted by developers in response to questions that include package usage information. We observe that usage example (36.76%) is most dominant attribute in accepted answer, followed by execute command (19.58%). The third most frequent (15.03%) attribute combination in the accepted answer is execute command and usage example. These findings hint that application developers are suffering from a lack of technical depth in managing third-party libraries in their applications.

**RQ<sub>2</sub> Summary:** Result shows that 37.76% accepted answers posted by developers in response to questions that include package usage information contain usage example followed by execute command 19.58%.

## 6. Discussion

In order to aid application developers faced with package usage issues, we conducted an empirical study to understand the usefulness of package usage information mined from SO. We learned two lessons along the way:

**Lesson 1-** Many of the library usage information on SO is not from the popular npm package. We find that only three out of the top ten of the most used npm libraries are mentioned in SO. The top-3 discussed npm packages are react, typescript, and webpack. Again, the top-5 li-

braries that are less frequently discussed in SO are mocha, eslint, chai, babel-core, and lodash. Furthermore, we find that 87.95% package co-usage mined from SO exist in the latest npm package release. The npm post that discuss package usage information are mostly relate to different type of errors.

**Lesson 2-** Developers tend to post answers that are usage example or execute command. The good news is that maybe the answers can be useful for any npm developer that suffers from similar issues. There is potential for a recommendation system, especially with the available execute commands and examples available.

## 7. Related Work

In this section, we discuss the related works. First, we discuss on third-party package usage issues faced by developers. Second, we discuss on mining useful information from question-answering sites.

**Third-party libraries usage issues:** software package is a reusable component developed by a body other than the original vendor of the development platform. The usage of third-party libraries provides developers with unique opportunity to integrate pre-tested, reusable software that saves development time and cost<sup>†</sup>. Recent empirical studies have found that 93.30% of modern OSS project use third-party libraries, with an average of 28 libraries per project [6].

In recent years, analyzing the characteristics of software ecosystem has gained much attention. Decan et al. [23] investigated package dependency network for seven software ecosystems. Their findings reveals that, software ecosystems grew over time in term of number of published libraries. Bogart et al. [24] investigate the challenges of reusing libraries from software ecosystem. They reported that developers struggle with changing versions of the libraries as the changes might potentially break dependent codes. Bavota et al. [25] examine the evolution of dependencies in Apache ecosystem and found that developers were reluctant to upgrade their dependencies considering that changes of a package might break its dependent libraries. Xavier et al. [26] performed a large scale study on 317 real-life Java libraries with 9K releases and 260K projects. Their analysis results show that 14.78% of API changes are incompatible with previous versions. Kula et al. [27] also reported that, developers do not update their dependencies even though the updates are related to new features, fix vulnerabilities. Several research was done on package recommendation tools like LibRec [6], LibCup [7], CrossRec [11]. In this research, we empirically investigate usefulness of npm package usage information mined from question-answering site.

**Mining SO:** Recent studies point out that SO is a useful source for developers to meet their information needs.

<sup>†</sup><https://tinyurl.com/y5b2fajq>

For instance, Chen et al. [13] reported that SO is useful in recommending analogical libraries across different programming languages. Mahajan et al. [14] proposed a recommendation tool to fix Runtime Exceptions based on knowledge from SO posts. Similarly, Treude et al. [15], Rubei et al. [20], and Uddin et al. [16] showed that SO posts are useful knowledge source to support software developers. Previous studies suggest that SO can be useful to solve package usage related issues. There is no existing work that studies package usage information mined from SO help to improve developers' experience.

## 8. Threats to Validity

In this section, we discuss threats to validity that might influence our study.

**Internal Validity:** Threats to internal validity refer to experimental bias. In this study, we found two main internal threads that could affect our results. First, is the pre-processing of the dataset we decide the number of posts (2100) and npm packages (217,934) based on cumulative extraction of npm libraries and the generated co-usages. We continue the cumulative extraction until all the libraries and the co-usage cover. Second, in RQ<sub>2</sub> we perform manual analysis on random sample since the dataset size is large. To mitigate this challenge, we prepare representative sample consists of 286 randomly selected accepted answer, with a confidence level of 95% and a interval of 5.

**External validity:** Threats to external validity refer to the generalizability of our findings. Our datasets consist of npm packages from libraries.io and SO posts. SO is a popular platform for question and answers from developers with various domains and experts. Hence, our observations and results can not be generalized for other package managers like Maven, NuGet, and others. Besides, we consider only those SO posts that contain package.json file. Selecting more question posts may cause variation of top package co-usage results.

**Construct validity:** Threats to construct validity refers to the suitability of our evaluation measure. In our qualitative analysis of classifying accepted answers (RQ<sub>2</sub>), the answer patterns may be miscoded due to the subjective nature of our coding approach. To mitigate this threat, we took a systematic approach to validate the taxonomy and the comprehension understanding by the three authors in several rounds. Only until the Kappa score reaches 0.82, indicating that the agreement is almost perfect (0.81–1.00), we were able to complete the rest of the sample dataset.

## 9. Conclusion

In this paper, we examine the usefulness of package usage information mined from SO. We perform a case study on npm package co-usage information from SO question posts

(2100) and libraries.io (217,934 npm projects) dataset. Although SO has been a useful resource for finding answers to questions, we find that unfortunately popular and highly used libraries are not discussed as often. However, we can see that the accepted answers may prove useful, as we believe that the usage examples and executable commands could be reused or be used for tool support. In our future work, we will develop tool support that will utilize SO usage examples and executable commands extracted from accepted answers to assist npm application developers.

## Acknowledgments

This work has been supported by JSPS KAKENHI Grant Numbers JP8H04094, JP20K19774, and JP20H05706.

## References

- [1] J. Dietrich, K. Jezek, and P. Brada, "Broken promises: An empirical study into evolution problems in java programs caused by library upgrades," 2014 Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE), pp.64–73, IEEE, 2014.
- [2] S. Raemaekers, A. van Deursen, and J. Visser, "Semantic versioning and impact of breaking changes in the maven repository," J. Syst. Softw., vol.129, pp.140–158, 2017.
- [3] S. Raemaekers, A. van Deursen, and J. Visser, "Semantic versioning versus breaking changes: A study of the maven repository," 2014 IEEE 14th International Working Conference on Source Code Analysis and Manipulation, pp.215–224, 2014.
- [4] F.L. De La Mora and S. Nadi, "Which library should i use?: A metric-based comparison of software libraries," Proc. 40th International Conference on Software Engineering: New Ideas and Emerging Results, pp.37–40, 2018.
- [5] J. Dietrich, D. Pearce, J. Stringer, A. Tahir, and K. Blincoe, "Dependency versioning in the wild," 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR), pp.349–359, 2019.
- [6] F. Thung, D. Lo, and J. Lawall, "Automated library recommendation," 2013 20th Working conference on reverse engineering (WCRE), pp.182–191, 2013.
- [7] A. Ouni, R.G. Kula, M. Kessentini, T. Ishio, D.M. German, and K. Inoue, "Search-based software library recommendation using multi-objective optimization," Inform. Softw. Tech., vol.83, pp.55–75, 2017.
- [8] M.A. Saied, A. Ouni, H. Sahraoui, R.G. Kula, K. Inoue, and D. Lo, "Improving reusability of software libraries through usage pattern mining," J. Syst. Softw., vol.145, pp.164–179, 2018.
- [9] H. Alrubaye, M.W. Mkaouer, I. Khokhlov, L. Reznik, A. Ouni, and J. McGoff, "Learning to recommend third-party library migration opportunities at the API level," Applied Soft Computing, vol.90, p.106140, 2020.
- [10] H. Yu, X. Xia, X. Zhao, and W. Qiu, "Combining collaborative filtering and topic modeling for more accurate android mobile app library recommendation," Proc. 9th Asia-Pacific Symposium on Internetware, pp.1–6, 2017.
- [11] P.T. Nguyen, J. Di Rocco, D. Di Ruscio, and M. Di Penta, "Cross-Rec: Supporting software developers by recommending third-party libraries," J. Syst. Softw., vol.161, p.110460, 2020.
- [12] Y. Wu, S. Wang, C.-P. Bezemer, and K. Inoue, "How do developers utilize source code from stack overflow?," Empir. Softw. Eng., vol.24, no.2, pp.637–673, 2019.
- [13] C. Chen and Z. Xing, "Similartech: automatically recommend analogical libraries across different programming languages," Proc. 31st

IEEE/ACM International Conference on Automated Software Engineering, pp.834–839, 2016.

- [14] S. Mahajan, N. Abolhassani, and M.R. Prasad, “Recommending stack overflow posts for fixing runtime exceptions using failure scenario matching,” Proc. 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp.1052–1064, 2020.
- [15] C. Treude and M.P. Robillard, “Augmenting API documentation with insights from stack overflow,” Proc. 38th International Conference on Software Engineering, pp.392–403, 2016.
- [16] G. Uddin, F. Khomh, and C.K. Roy, “Mining API usage scenarios from stack overflow,” Inform. Softw. Tech., vol.122, p.106277, 2020.
- [17] E.L. Vargas, M. Aniche, C. Treude, M. Bruntink, and G. Gousios, “Selecting third-party libraries: The practitioners’ perspective,” Proc. 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp.245–256, 2020.
- [18] L. Xavier, A. Hora, and M.T. Valente, “Why do we break APIs? First answers from developers,” 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), pp.392–396, 2017.
- [19] B. Todorov, R.G. Kula, T. Ishio, and K. Inoue, “SoL Mantra: Visualizing update opportunities based on library coexistence,” 2017 IEEE Working Conference on Software Visualization (VISSOFT), pp.129–133, 2017.
- [20] R. Rubei, C.D. Sipio, P.T. Nguyen, J.D. Rocco, and D.D. Ruscio, “PostFinder: Mining stack overflow posts to support software developers,” Inform. Softw. Tech., vol.127, p.106367, 2020.
- [21] S. Baltes, L. Dumanı, C. Treude, and S. Diehl, “SOTorrent: reconstructing and analyzing the evolution of stack overflow posts,” Proc. 15th International Conference on Mining Software Repositories, pp.319–330, 2018.
- [22] A.J. Viera, J.M. Garrett, et al., “Understanding interobserver agreement: The kappa statistic,” Fam med, vol.37, no.5, pp.360–363, 2005.
- [23] A. Decan, T. Mens, and P. Grosjean, “An empirical comparison of dependency network evolution in seven software packaging ecosystems,” Empir. Softw. Eng., vol.24, no.1, pp.381–416, 2019.
- [24] C. Bogart, C. Kästner, J. Herbsleb, and F. Thung, “How to break an API: cost negotiation and community values in three software ecosystems,” Proc. 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, pp.109–120, 2016.
- [25] G. Bavota, G. Canfora, M.D. Penta, R. Oliveto, and S. Panichella, “How the Apache community upgrades dependencies: an evolutionary study,” Empir. Softw. Eng., vol.20, no.5, pp.1275–1317, 2015.
- [26] L. Xavier, A. Brito, A. Hora, and M.T. Valente, “Historical and impact analysis of API breaking changes: A large-scale study,” 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), pp.138–147, IEEE, 2017.
- [27] R.G. Kula, D.M. German, A. Ouni, T. Ishio, and K. Inoue, “Do developers update their library dependencies?,” Empir. Softw. Eng., vol.23, no.1, pp.384–417, 2018.



**Syful Islam** He received the M.E. degree in Information Science from Nara Institute of Science and Technology, Japan. He is currently working toward the Ph.D degree in the same institute. At present, he is on study leave from Noakhali Science and Technology University, Bangladesh. His research interests include software ecosystem, mining Stack Overflow, etc.



**Dong Wang** He received the M.E. degree in Information Science from Nara Institute of Science and Technology, Japan. He is currently working toward the Doctor degree in Nara Institute of Science and Technology, Japan. His research interests include code review and mining software repositories.



**Raula Gaikovina Kula** is currently an assistant professor at Nara Institute of Science and technology. In 2013, he graduated with a PhD. from Nara Institute of Science and Technology, Japan. He is currently an active member of the IEEE Computer Society and ACM. His research interests include repository mining, code review, software libraries and visualizations.



**Takashi Ishio** received the Ph.D degree in information science and technology from Osaka University in 2006. He was a JSPS Research Fellow from 2006-2007. He was an assistant professor at Osaka University from 2007-2017. He is now an associate professor of Nara Institute of Science and Technology. His research interests include program analysis, program comprehension, and software reuse. He is a member of the IEEE, ACM, IPSJ and JSSST.



**Kenichi Matsumoto** received the B.E., M.E., and PhD degrees in Engineering from Osaka University, Japan, in 1985, 1987, 1990, respectively. Dr. Matsumoto is currently a professor in the Graduate School of Information Science at Nara Institute Science and Technology, Japan. His research interests include software measurement and software process. He is a senior member of the IEEE and a member of the IPSJ and SPM.