An Empirical Study of Coding Style Compliance on Stack Overflow

Qing Mi*, Haotian Bai, Xiaozhou Wang, Wenrui Liu, Xingyue Song
Faculty of Information Technology, Beijing University of Technology, Beijing, China
miqing@bjut.edu.cn,baihaotian20020905@emails.bjut.edu.cn,Wang-xz@emails.bjut.edu.cn,
wenrui.liu@ucdconnect.ie,sxy18386137856@163.com

***Abstract*—Stack Overflow (SO) is one of the world's largest technical Q&A websites, in which many posts contain code snippets. However, these code snippets may not comply with coding style guidelines and result in the problem of low readability and maintainability. To provide a better understanding of this coding style compliance issue for SO users, we plan and conduct an empirical study on SO. Specifically, we collected over 400,000 code snippets from SO in three languages, namely Python, C/C++, and JavaScript. The posts are divided into two types (i.e., question and answer) and analyzed separately. We found that for the question- and answer-type posts, more than 90% and 60% of code snippets contain style violations. The most frequently found violation is syntax errors for Python and indentations for C/C++ and JavaScript. In addition, the results show that with more violations in code snippets, the "Score" of Python and C/C++ posts, the "FavoriteCount" of C/C++ questions, and the "CommentCount" of JavaScript questions tend to be lower. The findings of our research indicate that code snippets on SO do not have good coding style compliance. Users, especially programming beginners are supposed to be wary of the potential problems of reusing code snippets on SO.**

***Index Terms*—Coding Style Compliance, Stack Overflow, Programming Guideline, Stack Exchange**

I. Introduction

Stack Overflow (SO) is a program-related Q&A website. Software developers use SO to initiate, browse, and answer questions. As of 2022, SO has 17.16 million registered users and over 50 million posts, 64% of the posts in SO contain code snippets [3]. However, these code snippets are not fully compliant with coding guidelines and are likely to have quality problems [9], since coding style compliance is highly related to the readability and maintainability of the source code [4]. Simply copying (or reusing) them may cause errors as well as software maintenance issues [10]. Currently, there is still a little research about coding style compliance on SO. Therefore, we carefully plan an empirical study to bridge this research gap.

We first collected 106,248 Python code snippets, 98,723 C/C++ code snippets, and 110,304 JavaScript code snippets on SO. Then, we explore the violation ratio and most frequently found violation type based on the collected dataset. The experimental results illustrate that 93.54% Python code snippets, 98.95% JavaScript code snippets, and 91.13% C/C++ code snippets in question-type posts contain coding style violations, while 89.53% Python code snippets, 100% JavaScript code snippets, and 65.83% C/C++ code snippets in answer-type posts contain coding style violations. The most frequently found coding style violation is non-standard space indentation for JavaScript and C/C++ and syntax error for Python. The Pearson correlation analysis indicates a moderate negative correlation ($-0.8 \leq r \leq -0.6$) between the number of code violations per statement and the "Score" of Python and C/C++ posts, the "FavoriteCount" of C/C++ questions, and the "CommentCount" of JavaScript questions.

The findings of our research suggest that the majority of code snippets on SO do not comply with proper coding style guidelines. As code violations can decrease readability and maintainability, developers should be cautious when reusing code snippets from SO. Especially, users should focus on syntax errors and inconsistent indentation styles, which constitute the majority of the total violations. In addition, when raising questions on SO, we suggest that users should pay more attention to their coding style compliance because a better coding style tends to receive more comments, and code snippets with fewer violations have a greater chance to get a higher "Score".

II. Related Work

Some previous studies have concentrated on coding style compliance on Stack Overflow (SO).

The work of Hart et al. [14] sought to ascertain the influence of social reputation and other aspects on the perception of answer quality. Their findings suggested that social reputation had no substantial effect, while the presentation styles of completeness and conciseness were deemed to be more influential. The research presented herein elucidates the relationship between coding style compliance and the quality of posts, thereby demonstrating the potential for further explorations.

Treude et al. [17] conducted a study of the extent to which developers appraise SO code snippets as self-explanatory. Additionally, they probed the information absent from snippets and judged it not to be self-explanatory. The findings indicated that fewer than half of the code snippets in the sample were deemed self-explanatory. The primary coding style problems that impinge on the understandability of code snippets include incomplete snippets, code quality, missing rationale, code organization, clutter, naming issues, and missing domain information. The findings of their research demonstrate the

*Corresponding author.
DOI reference number: 10.18293/SEKE2023-125

deleterious effect that coding style violations have on the quality of code.

More recently, Bafatakis et al. [18] investigated coding style compliance in Python code snippets on SO. Their research focuses on Python code snippets. Their results showed that 93.87% of snippets contain style violations, with an average of 0.7 violations per statement. Additionally, they found that user reputation seems to be unrelated to coding style compliance. While there is a strong correlation between the vote "Score" a post received and the average number of violations per statement. The authors also mentioned that the choice of languages and attributes could be expanded in further study.

Our work is an extension of Bafatakis et al.'s study. Based on their work, this study examines coding style compliance on SO as well, but with a particular focus on different types of posts (questions and answers). Furthermore, our research investigates the compliance of code snippets in more languages, as well as the types of violations in each language and the correlation between the violation rate and the attributes of "Score", "ViewCount", "FavoriteCount", "AnswerCount", and "CommentCount".

III. Study Design

In this section, we first present our research questions (RQs). Then, we select representative programming languages and the corresponding analysis tools. Next, we retrieve, sort, and filter Stack Overflow (SO) to construct our dataset. Finally, for each RQ, we provide a brief introduction of our methodology.

A. Research Questions

To determine the coding style compliance on SO, we will analyze JavaScript, Python, and C/C++ code snippets to answer the following RQs:

- RQ1. How many code snippets on SO contain coding style violations?
- RQ2. Which coding style rules are most frequently broken on SO?
- RQ3. Does coding style compliance correlate with SO attributes?

B. Programming Languages and Code Analysis Tools

In our research, Python, C/C++, and JavaScript were selected as the research objects to ensure a comprehensive study, since these languages are among the top ten most popular languages on SO [1], which can provide enough data to support in-depth research.

Corresponding to the three languages, we chose Pylint, Cpplint, and ESLint to check the coding style compliance. In the case of Python, we selected Pylint[1] as it is a widely-used tool that follows the style recommended by PEP 8 (a Python style guide). Pylint provides a comprehensive set of checks that analyze code for potential errors and code smells. For C/C++, we chose Cpplint[2], which is based on Google's coding style guides and has a strict set of rules (i.e., naming conventions and formatting rules) that help developers write consistent and readable code. Finally, for JavaScript, we selected ESLint[3], which is a popular tool for identifying problematic patterns in JavaScript code. We chose these specific code analysis tools based on their ability to analyze code for potential issues and enforce coding standards. Each tool has its own set of configurable rules and guidelines that can be customized to meet the specific needs. Moreover, these tools are widely used in their respective communities and can provide reliable and effective code analysis.

C. Dataset Construction

The code snippets used in this study were acquired from Stack Exchange[4], a network of various Q&A websites, with SO being one of the most actively visited ones.

We extracted question-type posts from July 2008 to December 2021, and their corresponding answers (from July 2008 to April 2022). The posts are distinguished by tags, e.g., posts about Python are tagged with "Python". We retrieved Python, C/C++, and JavaScript code snippets and built our dataset by searching and downloading posts with certain tags.

In total, we extracted 106,248 Python code snippets, 98,723 C/C++ code snippets, and 110,304 JavaScript code snippets from question-type posts, and 40,418 Python code snippets, 44,316 C/C++ code snippets, and 47,463 JavaScript code snippets from answer-type posts.

Note that our research assumed that SO users have different programming experiences and ability levels; beginner programmers tend to ask questions (question-type posts) while experienced programmers prefer providing solutions (answer-type posts). Such differences may lead to discrepancies in the degree of coding style compliance. Consequently, we purposely divided the collected dataset into two subgroups, namely question-type posts and answer-type posts. Our subsequent analyses were conducted on these separate subgroups respectively.

D. Analysis Method

In order to answer RQ1, code analysis tools were utilized to identify coding style violations in the collected Python, C/C++, and JavaScript code snippets. The violation ratio was subsequently calculated, which pertains to the number of coding style violations per statement for each language.

To address RQ2, we first classified hundreds of coding style violations into several categories according to the websites of Pylint, Cpplint, and ESLint, as shown in Table II. Although these code analysis tools have different classification criteria for coding style violations, upon further analysis, we noticed a significant overlap in violation types across different programming languages (e.g., unused-import and reimported in Python, no-duplicate-imports and no-restricted-imports in JavaScript, build/include and build/include-order in C/C++). Based on this finding, we decided to consolidate and discuss some of the most prominent categories of violations, in order

[1]https://Pylint.pycqa.org/en/latest/

[2]https://github.com/google/styleguide/blob/gh-pages/cpplint/cpplint.py

[3]https://eslint.org/

[4]https://stackexchange.com/

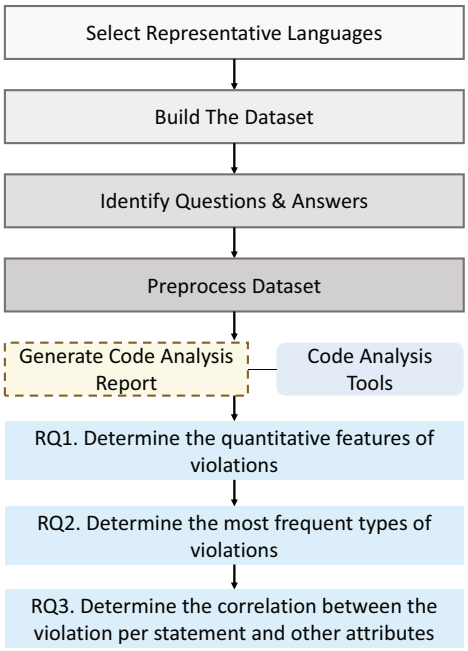

Fig. 1: Methodology Overview

to facilitate fair comparison of common violations across different languages. They are indentation-namespace/name, include/import, and whitespace/space. The distribution of these categories is provided in Table III.

The improper naming of variables or functions can be identified as a "Name" violation. This type of violation is crucial because clear names can help developers understand and work with the code better. Failure to follow standard naming conventions can cause confusion, make debugging harder and sometimes lead to program errors. Examples of these violations in Python include duplicate-argument-name (i.e., duplicate argument names in function definitions), arguments-renamed (i.e., a method parameter has a different name than in the implemented interface or an overridden method), bad-dunder-name (i.e., a dunder method is misspelled or defined with a name, not within the predefined list of dunder names), etc.

"Import" contains violations regarding importing. While many users might not immediately encounter issues due to inadequately importing, those who want to reuse code snippets would likely come up against difficult-to-diagnose bugs caused by such violations. Examples of this type of violations in Python include import-self (i.e., a module is importing itself) and shadowed-import (i.e., a module is aliased with a name that shadows another import). While in JavaScript, there are sort-imports (i.e., enforce sorted import declarations within modules) and no-duplicate-imports (i.e., disallow duplicate module imports).

"Space" contains violations regarding indentations. Indentations play a vital role in readable code, especially in languages like Python where correct indentation impacts code functionality. As a result, Python tends to have fewer indentation errors compared to languages such as C/C++. Common C/C++ indentation violations involve whitespace/braces, whitespace/comma, whitespace/empty-if-body, whitespace/semicolon, etc. Although these violations do not affect program operation and might seem insignificant, they create cluttered code, making it more challenging for colleagues to decipher intent. Thus, the importance of adhering to the standard of indentations should not be underestimated, as it can significantly improve the readability and maintainability of the code.

To answer RQ3, we retrieved five attributes of question-type posts, i.e., "Score" (the number of upvotes minus downvotes a post received), "ViewCount" (the number of times a post was viewed), "FavoriteCount" (the number of times a post was saved by other users), "AnswerCount" (the number of answers a post received), and "CommentCount" (the number of comments a post received). Besides, we retrieved two attributes of answer-type posts, i.e., "Score" and "CommentCount". We first used scatter plots to visualize the relationship between the violation ratio and the attribute values. After that, we preprocessed the collected data and removed the outliers using the interquartile range (IQR) method. Our study partitioned the collected posts into multiple groups based on the variance in violation ratios with increments of 0.2. Subsequently, the mean attribute values of each group were computed. The Pearson correlation coefficient was employed to explore the relationship between the violation ratio and the mean attribute values of each group.

IV. Empirical Results

We present our findings regarding each research question.

A. Results of RQ1

As shown in Table I, code snippets without any coding style violations are uncommonly found in JavaScript. For Python and C/C++ code snippets, it is noted that the prevalence of compliant code snippets in answer-type posts outweighs that in question-type posts. In the case of C/C++ code snippets, the ratio of compliant code snippets in answer-type posts exceeds that in question-type posts by 25.31%.

We also calculated the violation ratio. Our analysis revealed that the violation ratios are significantly elevated across all three languages, with a particularly pronounced increase in the case of JavaScript.

In conclusion, more than 80% of the code snippets on Stack Overflow (SO) contain coding style violations, indicating that code snippets on SO do not have good coding style compliance, and it is not recommended to reuse them without careful inspection.

B. Results of RQ2

Based on the classification method on the official websites of the code analysis tools, we calculated the proportion of each category of coding style violations in the three languages. As shown in Table II, the most frequently found violation is

TABLE I: Coding Style Violations in Question-type and Answer-type Posts

	Code Snippets with Violations		Violation Ratio	
	Question	Answer	Question	Answer
Python	93.55%	89.53%	0.1196	0.1311
C/C++	91.14%	65.80%	0.3802	0.3227
JavaScript	98.95%	100.00%	0.4457	0.5520

syntax error for Python and whitespace/braces for C/C++. For JavaScript code snippets, the most frequently found violation is indent (i.e., enforce consistent indentation) for questions and semi (i.e., require or disallow semicolons instead of ASI) for answers.

To facilitate further comparison, we identified common violations in the three languages and subsequently categorized these violations into three distinct groups. It can be seen in Table III that a significant number of coding style violations in C/C++ and JavaScript are related to indentations, whereas violations in Python code snippets involve a variety of aspects.

The main reason for this difference is the nature and design of the programming languages. C/C++ and JavaScript are typically written with curly braces where code blocks are enclosed in them. Due to the visual clarity produced by these curly braces, proper indentation is not required for code to be syntactically correct. Nevertheless, consistent indentation remains crucial for code legibility and readability. Therefore, most coding style guidelines for C/C++ and JavaScript emphasize consistent indentation as the primary formatting guideline.

On the other hand, Python is designed to use indentation to define its code blocks. This means that indentation is not just a matter of style, but it is an essential part of the language syntax. In Python, improper indentation can result in syntax errors that make the code unexecutable. As a result, Python coding style guidelines cover multiple aspects of indentation and whitespace usage, such as the number of spaces per indentation level, the use of tabs versus spaces, and the use of whitespace in other contexts such as line breaks and wrapping.

Additionally, Python has a more extensive set of language constructs and syntax features compared to C/C++ and JavaScript. This means that Python coding style guidelines need to cover a wider range of aspects beyond simple indentation rules, such as variable naming conventions, function and class definitions, control structures, etc. The goal of these guidelines is to promote consistency and readability across Python code, especially in larger and more complex projects.

C. Results of RQ3

Figure 2 displays data on Python posts that are questions. Most of these posts received a low "Score" and a low violation ratio. Some posts got a high "Score" with a low violation ratio, while others got a low "Score" with a high violation ratio. Only a very small number of posts in this category got a high "Score" with a high violation ratio. Similar Zipfian distributions were observed for answer-type posts and posts in two other languages.

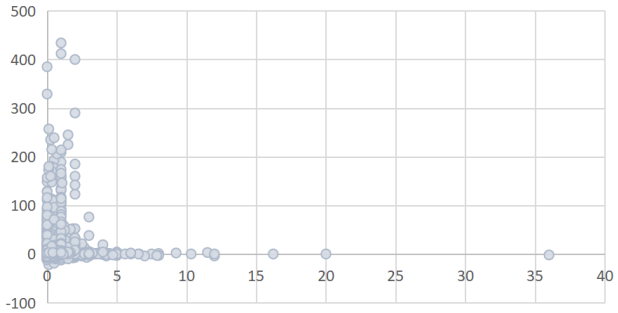

Fig. 2: Python Code Violation Rate vs. "Score"

We posit that a correlation coefficient surpassing the threshold value of $|r| > 0.6, p < 0.01$ represents a significant correlation between the two variables under consideration. Accordingly, we found that whether a code snippet is compliant with coding style rules correlated with some SO attributes. For instance, Figure 3 illustrates that the "Score" for answer-type C/C++ posts exhibits a moderate correlation with the violation ratio. The mean values of "Score" have a Pearson correlation coefficient of $r = -0.789, p = 0.002$ with the violation ratio. The result means that for C/C++ questions on SO, the ones with code snippets following coding style guidelines tend to receive a higher "Score" than other questions. As shown in Table IV, similar results were observed across some of the other attributes.

For JavaScript posts, it can be observed that the correlation between attribute values and violation ratio is commonly found to be insignificant, except the "CommentCount" attribute in questions. The findings of RQ2 indicate that the majority of violations in JavaScript code do not significantly affect how the program operates. This may lead users to perceive coding style violations in JavaScript posts as less important.

It can be noted that violations have a greater impact on "Score" than other attributes. Users tend to upvote more on questions and answers with fewer coding style violations and believe them to be of higher quality. Besides, we noticed that the violation ratio of question-type C/C++ posts is negatively correlated with "FavoriteCount".

At this time, there is no conclusive evidence proving that coding style violations significantly affect users' browsing, commenting and answering behaviors. As all "ViewCount", "AnswerCount" and answers' "CommentCount" exhibit weak correlation ($|r| < 0.6$) with the violation ratio, as shown in Table IV, which means that under certain circumstances, users seem to pay less attention to the coding style compliance. It is possible that users are willing to overlook minor coding style violations if the code is otherwise useful or valuable to them. They may prioritize the benefits of the code over its compliance with coding style guidelines.

TABLE II: Proportion of Each Category of Coding Style Violations

Python Violation Types			C/C++ Violation Types			JavaScript Violation Types		
1	Refactor for a "good practice" metric violation	5.70%	1	build: builds when the error	3.16%	1	Possible Errors: Rules related to possible syntax or logic errors in JavaScript code	0.32%
2	Convention for coding standard violation	28.14%	2	readability: readability error	4.16%	2	Best Practices: Rules related to better ways of doing things to help you avoid problems	4.07%
3	Warning for stylistic problems, or minor programming issues	32.53%	3	runtime: runtime error	3.70%	3	Parsing error: parsing error	5.76%
4	Error for important programming issues	33.61%	4	whitespace: space indentation error	88.98%	4	Variables: Rules related to variable declarations	0.01%
5	Fatal for errors which prevented further processing	0.02%				5	Node.js and CommonJS: Rules related to code running in Node.js, or in browsers with CommonJS	0.01%
6	Informational messages that Pylint emits	0.00%				6	Stylistic Issues: Rules related to style guidelines, and are therefore quite subjective	82.75%
						7	ECMAScript 6: Rules related to ES6, also known as ES2015	7.09%

TABLE III: Proportion of Each Reclassified Category of Coding Style Violations

	Question			Answer		
	name	import	space	name	import	space
C/C++	0.05%	0.68%	89.07%	0.92%	0.78%	89.26%
Python	14.01%	3.46%	8.88%	9.45%	1.79%	5.18%
JavaScript	0.01%	0.00%	44.78%	0.01%	0.00%	37.91%

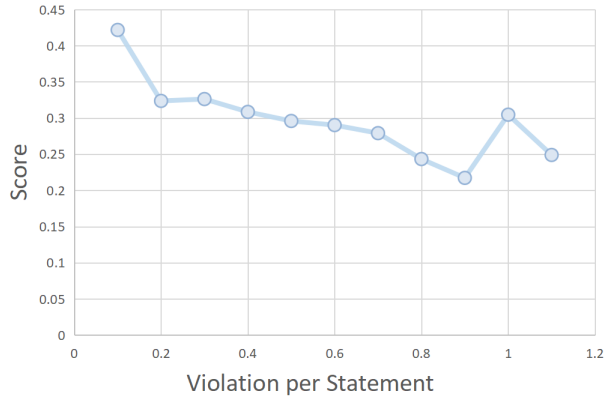

Fig. 3: Mean value of "CommentCount" for question-type C/C++ posts with the same range of violation ratio

TABLE IV: Pearson Correlation Coefficients between Different Attributes and the Violation Ratio

Attribute		Python	C/C++	JavaScript
Question	Score	**-0.615**	**-0.789**	-0.01
	ViewCount	0.305	0.517	0.595
	FavoriteCount	0.348	**-0.627**	0.130
	AnswerCount	0.186	0.127	0.483
	CommentCount	0.086	0.329	**-0.649**
Answer	Score	**-0.607**	**-0.745**	-0.156
	CommentCount	0.487	0.521	0.347

V. Discussion

A. Implications

Our findings have important implications for both developers and software organizations.

For developers, our study highlights the importance of compliance with established coding styles and conventions in order to produce readable and maintainable code. The results show that code that does not conform to a standard style is more likely to receive negative feedback and be downvoted on SO, which can impact the reputation of the developer and reduce the visibility of their code. Therefore, developers should carefully follow established coding styles in order to improve the quality and maintainability of their code, as well as their position within the developer community. Automated tools such as linters and code formatters can help developers capture coding style violations in the early stage of the development process. Besides, developers should collaborate with their peers to improve their coding style. Code reviews can provide feedback on how to improve code readability and maintainability.

For software organizations, our study highlights the importance of establishing and enforcing coding standards across development teams. By ensuring that all developers adhere to a consistent coding style, organizations can improve the quality and readability of their code, as well as increase the efficiency and effectiveness of their development processes. Additionally, our study suggests that organizations should introduce tools and resources to help developers adhere to coding standards, such as automated code review tools and training programs.

Finally, our study has implications for the software development community. The findings suggest that the use of standardized coding styles and conventions can improve the quality and maintainability of code, which can ultimately lead to better software products and a more robust software industry.

B. Threats to Validity

Many code snippets used in our experiments are incomplete, which is a possible threat to the conclusion of our study. As many questions on SO only contain partial code, several coding style rules cannot be applied. To address this issue, we had to disable these rules, as illustrated in Table V. While this approach may have some impact on the results obtained, it is unlikely to introduce a significant bias in the study's findings. Nonetheless, caution is warranted when interpreting the results, and future research could benefit from using more complete code snippets.

TABLE V: Disabled Rules for Three Code Analysis Tools

Pylint	Cpplint	ESLint
1.Unused wildcard import 2.Missing docstring 3.Undefined variable 4.Missing final newline 5.Allow constants and modules using any named style 6. Import error	1.Missing copyright information in file 2.No newline at the end of the code block 3.No space at the end of the code line	1.no-unused-vars 2.no-use-before-define 3.no-undef 4.no-trailing-spaces

In the process of gathering data, code snippets were selectively extracted through the utilization of pertinent tags. Nonetheless, it is plausible that a subset of posts might contain erroneously labeled snippets, or code snippets written in Python, C/C++, and JavaScript are not labeled by those tags. As a mitigation strategy, we processed all code snippets with code analysis tools and found a small minority, less than 10%, of snippets that proved to be beyond the bounds of the tools. These snippets were deemed invalid and were consequently excluded by way of manual intervention. For instance, within the 116,000 snippets that were extracted with the "Python" tag, 9,752 snippets (8.41%) were identified as unsuitable for analysis and were consequently eliminated. Nonetheless, it is conceivable that some mislabeled code snippets may still exist.

VI. Conclusions

To explore the coding style compliance on Stack Overflow, we first collected over 400,000 Python, C/C++, and JavaScript code snippets from July 2008 to April 2022. Then, we conducted empirical experiments on the collected dataset. Our findings indicate that: 1) 93.54%, 91.13%, and 98.95% Python, C/C++, and JavaScript code snippets in questions and 89.53%, 65.83% and 100% Python, C/C++, and JavaScript code snippets in answers contain violations of coding style guidelines. 2) The most frequently broken rule is whitespaces in C/C++, undefined syntax errors in Python, inconsistent indentation and semicolon usage in JavaScript questions and answers. 3) Posts in Python and C/C++ with more violations have lower "Score". For C/C++ questions, "FavoriteCount" is negatively correlated with violation per statement, while for JavaScript questions, "CommentCount" is negatively correlated with violation per statement.

In this study, we analyzed code snippets in Python, C/C++, and JavaScript on Stack Overflow to explore the problem of coding style compliance. In the future, we will further investigate other programming languages and Q&A websites to provide additional insights.

Acknowledgments

This work was supported by the Spark Project of the Beijing University of Technology (Project No. XH-2023-02-35).

References

[1] Stack Overflow trends – most popular languages. [Online]. Available: https://insights.stackoverflow.com/trends

[2] YANG, DI, HUSSAIN, AFTAB, LOPES, CRISTINA. From Query to Usable Code: An Analysis of Stack Overflow Code Snippets[J]. 2016.

[3] 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.

[4] Lee, T., J. B. Lee, and H. P. In. "A Study of Different Coding Styles Affecting Code Readability." International Journal of Software Engineering & Its Applications 7.5(2013):413-422.

[5] Themistoklis Diamantopoulos and Andreas L. Symeonidis. Employing source code information to improve question-answering in Stack Overflow. In MSR '15, pages 454–457, 2015.

[6] Luca Ponzanelli, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, and Michele Lanza. Mining StackOverflow to turn the IDE into a self-confident programming prompter. In MSR '14, pages 102–111,2014.

[7] Kathryn T Stolee, Sebastian Elbaum, and Daniel Dobos. Solving the search for source code. Transactions on Software Engineering and Methodology, 23(3):1–45, 2014.

[8] Siddharth Subramanian and Reid Holmes. Making sense of online code snippets. In MSR '13, pages 85–88, 2013.

[9] C. Ragkhitwetsagul, J. Krinke, M. Paixao, G. Bianco and R. Oliveto, "Toxic Code Snippets on Stack Overflow," in IEEE Transactions on Software Engineering, vol. 47, no. 3, pp. 560-581, 1 March 2021, doi: 10.1109/TSE.2019.2900307.

[10] Toshihiro Kamiya, Shinji Kusumoto, and Katsuro Inoue. CCFinder: a multilinguistic token-based code clone detection system for large scale source code. Transactions on Software Engineering, 28(7):654–670, 2002.

[11] Yasemin Acar, Michael Backes, Sascha Fahl, Doowon Kim, Michelle L Mazurek, and Christian Stransky. You get where you're looking for: The impact of information sources on code security. In SP '16, pages 289–305, 2016.

[12] Felix Fischer, Konstantin Bottinger, Huang Xiao, Christian Stransky, Yasemin Acar, Michael Backes, and Sascha Fahl. Stack Overflow considered harmful? the impact of copy&paste on Android application security. In SP '17, pages 121–136, 2017.

[13] Tianyi Zhang, Ganesha Upadhyaya, Anastasia Reinhardt, Hridesh Rajan, and Miryung Kim. Are online code examples reliable? an empirical study of API misuse on Stack Overflow. In ICSE'18, 2018.

[14] K. Hart and A. Sarma, "Perceptions of answer quality in an online technical question and answer forum," in Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), 2014, pp. 103–106.

[15] S. M. Nasehi, J. Sillito, F. Maurer and C. Burns, "What makes a good code example?: A study of programming Q&A in StackOverflow," 2012 28th IEEE International Conference on Software Maintenance (ICSM), 2012, pp. 25-34, doi: 10.1109/ICSM.2012.6405249.

[16] M. Duijn, A. Kučera, and A. Bacchelli, "Quality questions need quality code: classifying code fragments on stack overflow," in Proceedings of the 12th Working Conference on Mining Software Repositories. IEEE Press, 2015, pp. 410–413.

[17] C. Treude and M. P. Robillard, "Understanding stack overflow code fragments," in International Conference on Software Maintenance and Evolution (ICSME). IEEE, 2017, pp. 509–513.

[18] N. Bafatakis et al., "Python Coding Style Compliance on Stack Overflow," 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR), 2019, pp. 210-214, doi: 10.1109/MSR.2019.00042.

[19] F. Fischer et al., "Stack Overflow Considered Harmful? The Impact of Copy&Paste on Android Application Security," 2017 IEEE Symposium on Security and Privacy (SP), 2017, pp. 121-136, doi: 10.1109/SP.2017.31.

[20] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and B. Hartmann, "Design Lessons from the Fastest Q&A Site in the West," in Proceedings of the 2011 annual conference on Human factors in computing systems, New York, NY, USA, 2011, pp. 2857–2866.

[21] M. Squire and C. Funkhouser, "A bit of code': How the stack overflow community creates quality postings, " in Proc. ofHICSS 2014, pp. 1425–1434.

[22] J. Yang, C. Hauff, A. Bozzon, and G.-J. Houben, "Asking the right question in collaborative Q&A systems, " Proc. of Hypertext 2014, pp. 179–189.