5-2022

Does this apply to me? An empirical study of technical context in Stack Overflow

Akalanka GALAPPATHTHI

Sarah NADI

Christoph TREUDE
Singapore Management University, ctreude@smu.edu.sg

Does This Apply to Me? An Empirical Study of Technical Context in Stack Overflow

Akalanka Galappaththi
University of Alberta
Edmonton, Alberta, Canada
akalanka@ualberta.ca

Sarah Nadi
University of Alberta
Edmonton, Alberta, Canada
nadi@ualberta.ca

Christoph Treude
University of Melbourne
Victoria, Australia
christoph.treude@unimelb.edu.au

ABSTRACT

Stack Overflow has become an essential technical resource for developers. However, given the vast amount of knowledge available on Stack Overflow, finding the right information that is relevant for a given task is still challenging, especially when a developer is looking for a solution that applies to their specific requirements or technology stack. Clearly marking answers with their *technical context*, i.e., the information that characterizes the technologies and assumptions needed for this answer, is potentially one way to improve navigation. However, there is no information about how often such context is mentioned, and what kind of information it might offer. In this paper, we conduct an empirical study to understand the occurrence of technical context in Stack Overflow answers and comments, using tags as a proxy for technical context. We specifically focus on *additional context*, where answers/comments mention information that is not already discussed in the question. Our results show that nearly half of our studied threads contain at least one additional context. We find that almost 50% of the additional context are either a library/framework, a programming language, a tool/application, an API, or a database. Overall, our findings show the promise of using additional context as navigational cues.

CCS CONCEPTS

• **Software and its engineering → Reusability**.

KEYWORDS

Stack Overflow, contextual information, navigating information

ACM Reference Format:
Akalanka Galappaththi, Sarah Nadi, and Christoph Treude. 2022. Does This Apply to Me? An Empirical Study of Technical Context in Stack Overflow. In *19th International Conference on Mining Software Repositories (MSR '22), May 23–24, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3524842.3528435

1 INTRODUCTION

Stack Overflow is a popular Q&A website among software developers, with about 22 million questions and 32 million answers. When performing their tasks, developers typically resort to Stack Overflow to quickly find relevant information [1, 22], especially given the pressure of quickly releasing software [15].

Given the vast amount of knowledge on Stack Overflow, finding the right piece of information that is relevant for a developer's task could be challenging though, especially in long threads[1] [23]. Squire and Funkhouser found that 95% of the Stack Overflow answers consist of 75% text [18] while Nadi and Treude found that 37% of all questions on Stack Overflow have more than one answer, with 789 characters as the average length of an answer [12]. Thus, since most threads contain large amounts of text and explanations, a developer could miss important information that is relevant to what they are looking for. This is especially true given the current limited visual indicators that may help developers focus their attention on relevant information. Current visual indicators on Stack Overflow include (1) question tags that typically indicate the technologies related to a question, (2) question and answer scores that show how the community evaluate a given question or answer, (3) the

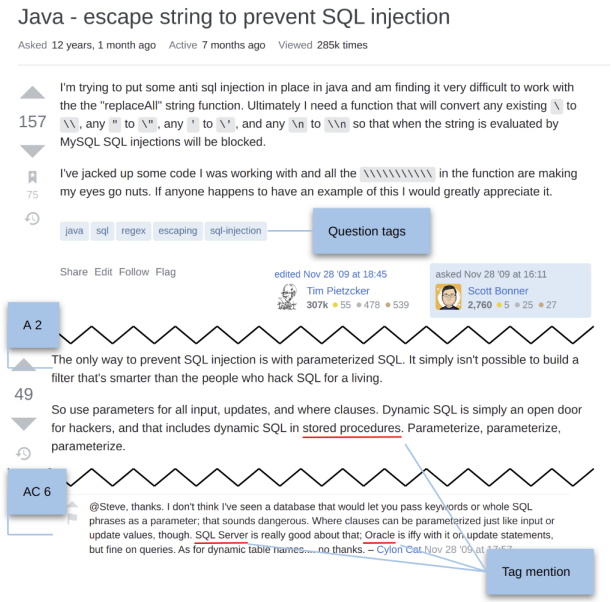

Figure 1: Motivating example based on Stack Overflow Question 1812891. Underlined words are Stack Overflow tags that appear as plain text in sentences.

MSR '22, May 23–24, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-9303-4/22/05…$15.00
https://doi.org/10.1145/3524842.3528435

[1] We use the term thread to refer to a Stack Overflow question and all its answers and comments.

accepted answer checkmark which indicates which answer solved the original poster's problem, and (4) the question/answer activity indicator which shows the change history of the post [16]. None of these visual cues provide quick indications of the content of a given answer or comment to help users navigate the provided information.

Figure 1 shows an example Stack Overflow thread. This question is tagged with the *java*, *sql*, *regex*, *escaping*, and *sql-injection* tags. These question tags typically indicate the technologies related to the question of the original poster and also help with the search functionality on Stack Overflow [17]. By looking at the question tags in Figure 1, users can clearly understand the technologies used in the question without even reading the question body. In other words, these question tags inform users about the context of a question. While context is an overloaded term in computer science [5], in this paper, we define *technical context* (or *context* for short throughout the paper) as information that characterizes the technologies and assumptions of a given question or answer. In this example thread, the question is relevant only to developers who are using Java, and not, for example, any other programming language. While questions present their context information with clear visual cues, other elements of a thread do not have such visual indicators of their contexts. For long threads that have multiple answers, answers with long textual descriptions, or answers with long code snippets, it would be useful for developers to quickly identify the answer context. By knowing the answer context, a developer could make an informed decision on whether to read or skip the answer.

For example, the thread in Figure 1 has fourteen answers. Answer 2 (A 2) mentions context information such as *SQL injection*, *SQL*, and *stored procedures* while Comment 6 on Answer 2 (AC 6) contains *SQL*, *SQL server*, and *Oracle*. Since *SQL* and *SQL injection* are already part of the question context, a user reading this thread already expects that answers will discuss or mention this context. On the other hand, *stored procedures*, *SQL server* and *Oracle* are different contexts from those in the question, which we refer to as *additional context*. Users looking for solutions for these specific techniques or technologies may want to quickly jump to the relevant answers that discuss them. Unfortunately, with such additional context information hidden within plain-text sentences, users will not be able to navigate quickly to or away from these answers or comments. It is worth noting that the accepted answer of the thread in Figure 1 does not have any additional contexts. However, users may focus their attention on it because of the visual indicator (i.e. checkmark) provided by Stack Overflow, and miss relevant additional contexts in other answers.

Overall, identifying the additional context of answers and comments and clearly marking these with visual cues (e.g., answer tags) could be a potential avenue for improving the navigation of Stack Overflow threads. Similar to a user navigating to an accepted answer as it has a checkmark, an answer tag will help a user to navigate to an answer with a technical context that is related to their particular context. However, we first need to understand whether such additional contextual information is present in Stack Overflow answers and comments and what the nature of such information is. Thus, in this paper, we conduct an empirical study of additional context in Stack Overflow answers and comments to answer the following research questions:

RQ1: **How frequently is additional context available in Stack Overflow answers and comments?** If answers and comments never have additional context, then the whole idea of extracting and emphasizing this information is not feasible. On the other hand, if there is too much additional context, then this may result in too many visual cues that make a thread cluttered. Thus, it is important to first understand how often additional context occurs and where it occurs in a thread.

RQ2: **What types of additional context are available in Stack Overflow threads?** In general, there may be various types of context that exist in a thread (e.g., API, programming language, operating system etc.). It is important to understand what types of additional context appear in threads to decide what may be useful for navigation.

RQ3: **What is the purpose of additional context information?** While understanding the general type of additional context is important, we also want to understand why a Stack Overflow answer or comment mentioned the additional context (i.e., what purpose does this additional context serve in this particular thread).

To answer these research questions, we perform a quantitative and qualitative empirical study of 207 threads from three technologies on Stack Overflow, *Json*, *Django*, and *Regex*. Our sample contains 488 answers, 468 answer comments, and 550 question comments, and a total of 3,504 sentences. To scope our study, we use the technologies in existing Stack Overflow tags as a proxy for contextual information and automatically search for the occurrence of these tags in the answers and comments of our studied threads. We then manually verify if the automatically detected tag occurrences are properly used in the marked sentences and whether these tag mentions represent additional contextual information (i.e., do not overlap with the question context). We then use open-coding card sorting to identify categories of additional context as well as their purpose in the thread.

Our results show that approximately half of the studied threads contain at least one sentence with additional context. From those, 50% of the additional contexts belong to categories such as library/framework, programming language, tool/application, API, and database. We also find that 20% of sentences with additional context provide direct solutions or solution conditions that would be valuable for a developer navigating this thread. We provide a discussion of the implications of our results for augmenting Stack Overflow threads with navigational cues based on additional contexts.

2 RELATED WORK

While, to the best of our knowledge, there is no existing research that specifically focuses on finding additional contexts on Stack Overflow, there are several studies and related research that we build our motivation and initial insights on. We review existing work on identifying relevant information and navigating to relevant information on Stack Overflow.

Identifying relevant information on Stack Overflow: Nasehi et al. conducted a qualitative study to find good code examples on Stack Overflow [13]. To find characteristics of good code examples, the authors studied the code snippets, their surrounding text, as well as answer comments. They found that answers often offer alternative solutions and discuss various API performance issues and that comments provide further explanations. These findings motivate us to further study the textual content in Stack Overflow threads in order to identify additional contexts in sentences, with the eventual goal of helping users navigate this information.

Treude and Robillard argued that Stack Overflow users often skip the text surrounding code snippets when they cannot clearly determine whether it explains the code in general or answers the specific requirements in the original question [21]. Even though the authors did not focus on the surrounding text in their study, this argument supports our assumption of the availability of different contextual information. Imtiaz et al. explore Stack Overflow questions related to static analysis tools to understand the challenges developers face when responding to these tools' alerts [9]. While categorizing the questions they studied, the authors found that some of the questions contain different contextual information. For example, direct solutions and conceptual knowledge related to the problem were present in a single thread. Therefore, emphasizing contextual information in threads could help users quickly determine the context and decide whether to skip or read the descriptions.

Treude and Robillard found that sentences extracted from Stack Overflow threads could complement API documentation [20]. Baltes et al. studies links to documentation in Stack Overflow threads [3]. Those two studies have fundamentally different goals, one to enrich software documentation with missing information and the other to improve information diffusion. However, some of the sentences that they found useful in both studies contain technical context that motivates us to identify contextual information in such sentences that could help users to navigate through threads.

Navigating to relevant information on Stack Overflow: Due to the information overload, navigating to relevant information on Stack Overflow threads could be challenging. A recent survey by Chattarjee et al. shows that too much text in answer descriptions is one of the reasons that slows down developers from identifying relevant solutions on Stack Overflow [4]. As an interpretation of their results, the authors suggest that a solution is to highlight the relevant content for users. Our work has a similar motivation but we specifically focus on additional contexts in this information. Along similar lines, Xu et al. also argue that finding relevant information in a long post is difficult, but instead of highlighting relevant information, they generate a summary from the answers [23].

Nadi and Treude extracted essential sentences that help users navigate Stack Overflow threads [12]. They define *essential sentences* as those that allow users to determine whether an answer should be read or skipped. The authors compare four automated techniques for identifying essential sentences, using ratings from survey participants as the ground truth. This is the closest work to ours: we share the end goal of identifying content that could potentially help users to quickly navigate through long threads; additionally, some of the *essential sentences* they found contain additional contexts, although the authors do not recognize it as such. Their results show that some of the highly rated sentences by survey participants were missed by all four techniques, while others were captured by only one out of the four techniques. The authors concluded that there is no single superior technique for capturing essential sentences for navigation. The findings of their work indicate that before automatically capturing navigational cues, we need to better understand what information can potentially be used as navigational cues. Therefore, in this paper, we take a step back and focus on one category of well-defined information, technical context, analyze its presence in Stack Overflow threads, and its potential to be used as navigational cues.

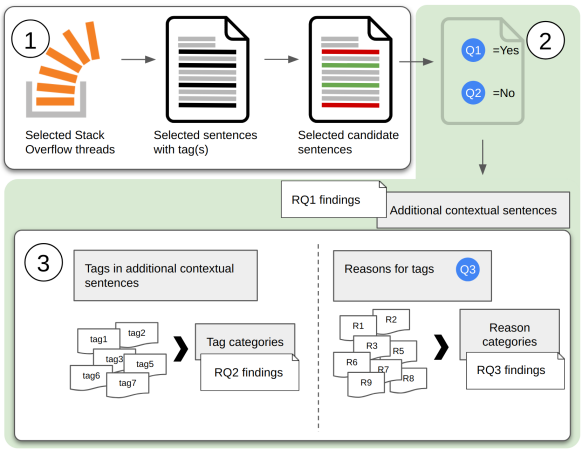

Figure 2: Overview of the research methodology.

3 METHODOLOGY

Figure 2 illustrates the three phases of collecting the required data and information for our empirical study. In Phase ①, we use automated criteria to analyze the selected Stack Overflow threads and identify sentences that potentially contain additional context.

In Phase ②, we manually review the candidate sentences from Phase ① and confirm those that contain additional context, which allows us to answer RQ1.

In Phase ③, we perform open card sorting to categorize the types of tags that appear as additional context as well as the reasons for mentioning these additional contexts, which allows us to answer RQ2 and RQ3.

3.1 Phase ① Automatically select sentences for manual review

The goal of Phase ① is to automatically identify sentences that potentially contain additional context. We will then manually review these candidate sentences in Phase ②. Phase ① has three steps. First, we select a sample of Stack Overflow threads for our empirical study. Second, we automatically identify sentences in answers and comments that contain technical context. We define *technical context* as information that characterizes the technologies

Table 1: Statistics of number of questions

Tag	Questions	Questions w/answers & no duplicates
Json	264,898	226,287
Django	212,850	168,384
Regex	168,969	158,490

and assumptions of a given question or answer. We use Stack Overflow tags as a proxy for technical context. Third, we automatically compare the technical context identified in the answer/comment sentences to the set of tags that exist in the question; this allows us to filter out sentences whose context already overlaps with that of the question. The output of Phase ① is a set of candidate sentences with potential additional context that we manually review in Phase ②.

3.1.1 Select threads for empirical study. To investigate the availability of additional contexts, we will need to eventually manually examine the sentences in Stack Overflow answers and comments. Manual examination of Stack Overflow threads is time-consuming and requires domain expertise. Since Stack Overflow has over 21 million questions, to balance manual effort and representativeness, we limit our study to questions that belong to one of three tags: *json*, *regex*, and *django*. We selected these tags to diversify the types of threads we study. *Json* is a general data exchange format supported by many programming languages. *Regex* is also supported by many programming languages for pattern matching in strings. *Django* is a popular open-source Python web framework. We use the Stack Overflow dump[2] from September 7, 2021, to collect all question IDs that belong to these three tags. The second column of Table 1 shows the total number of questions available in the Stack Overflow dump for our three selected tags.

From the pool of available question IDs, we remove questions that do not have any answers. Note that a question could be tagged with more than one of our selected three tags (e.g., Stack Overflow Question[3] 8586346 has both *django* and *regex* as tags). In such cases, we remove duplicates by attaching this question ID to the set of questions related to only one of the tags. The third column of Table 1 shows the corresponding number of considered questions after filtering and removing duplicates. Given the large number of threads available, we randomly select a representative sample of question IDs for our study. When calculating the sample size, we choose 90% confidence interval and 10% margin of error, so that our sample size is practical for manual examination. The resulting sample size is 69 threads from each tag, for a total of 207 threads. These 207 threads contain 488 answers with 1,644 sentences, 468 answer comments with 1,064 sentences, and 550 question comments with 776 sentences.

3.1.2 Identify sentences with technical context. To reduce the burden of manually reading through each sentence in our sample to determine if it contains contextual information, we design an automated approach to identify candidate sentences for review.

We use Stack Overflow tags as a proxy for technical context. However, using tags to capture contexts poses two problems. The first problem is that Stack Overflow tags are user created content. Therefore, there is no guarantee that a tag has a description or that the tag description has enough information to determine what kind of technology it represents. If a tag does not have a description it is not possible for us to determine whether that tag is used as intended in a sentence. The second problem is that Stack Overflow currently has over sixty thousand tags. While some of these tags represent technical contexts, the same tag can be used as a regular word in a sentence. *Flow*, *monitor*, and *back* are a few examples that fall under this category.

To solve the above problems, we leverage the *Witt* taxonomy of Stack Overflow tags by Nassif et al. [14]. The authors automatically analyzed tag descriptions, tag wikis, and Wikipedia pages of a tag to identify a tag's category [14]. Using this information, they then created a taxonomy of Stack Overflow tags. Table 2 shows examples of tags and corresponding categories extracted from the *Witt* taxonomy. However, given the automated process, some of the categories in the taxonomy are not meaningful. For example, "project", "software", "developed", "framework", "deprecated", and "abandoned" are the categories for *moonlight* in *Witt*[4]. According to the Stack Overflow tag description, *moonlight* is the Linux port for Microsoft's cross-browswer plugin called Silverlight. Considering this tag description, "project", "software", and "framework" are more suitable as categories while "developed", "deprecated", and "abandoned" are more suitable as attributes. After investigating the taxonomy, we observe that some tag categories such as *programming-language*, *framework*, and *library* have more than 200 corresponding tags, while other categories have very few tags. In general, we find that the category frequency distribution is right skewed with the median of 2 tags per category and that low count categories in the long tail of the distribution were not meaningful (e.g., *abandoned* and *end-user*). To ensure that we consider only meaningful categories, we discard categories that describe only a limited number of tags; this amounted to discarding 99% of the categories. The remaining 1% contains 29 categories in which each category has more than 230 tags. This subset of Witt categories contains 14,536 tags, representing ∼35% of the total number of tags in the taxonomy. We find that our selected subset of tags does not capture tags like *push*, *drop*, and *background* as technical contexts in sentences, but is still able to capture tags that represent technologies such as *java*, *node*, and *oracle*.

Stack Overflow tags can contain one word (e.g., *java*) or multiple words (e.g., *sql-server*). Thus, to capture tags mentioned in sentences, we have to consider different word combinations. We generate unigrams, bigrams, and trigrams from the words in sentences to compare with the filtered tag list. To demonstrate this, Table 3 contains two sentences from AC 6 of our motivating example in Figure 1. Generally, multiword Stack Overflow tags use hyphens to separate each word (e.g. *java-collection-api*, *python-3.6*). However, some tags do not follow the same format (e.g *android4.0.3*). To capture tags with different formats, we first replace

[2]https://archive.org/details/stackexchange
[3]https://stackoverflow.com/questions/8586346
[4]https://witt.herokuapp.com/

Table 2: Selected tags and their categories extracted from *Witt* taxonomy [14]

Tag	Category	Tag	Category
django-1.10	web-framework	jackson	api
json	format	r-tree	data-structure
linuxmint	os	oracle	database
gson	library	notepad++	editor

Table 3: Capturing tags mentioned in sentences

Sentence	SQL Server is really good about that
unigram	sql, server, is, really,...
bigram	sql-server, server-is, is-really, ...
trigram	sql-server-is, server-is-really, ...
Sentence	Oracle is iffy with it on update ...
unigram	oracle, is, iffy, ...
bigram	oracle-is, is-iffy, ...
trigram	oracle-is-iffy, is-iffy-with, ...

all hyphens with spaces. Then, we use the following regular expression `([a-z]+)(\d+(\.\d+)*)([a-z]+)?` to capture version numbers that appear after text or in the middle of text. For example, occurrences of *java 7* or *java-7* or *java7* will all be treated as *java 7* during the matching process. If any unigram, bigram, or trigram matches a tag, we capture it as a technical context. The underlined unigram and bigram in Table 3 are the tags mentioned in the two sentences from AC 6 in Figure 1.

Identify sentences with additional context. Recall from the introduction that we are interested in identifying sentences with additional context, that is contextual information that is not already included in the question. When posting questions, Stack Overflow users can include tags to indicate the context of the question. Our running example (Figure 1) has five tags, *java*, *sql*, *regex*, *escaping*, *sql-injection*. Thus, identifying that an answer or comment sentence mentions java, for example, is not particularly useful for navigation since it is natural that the whole thread is about java. However, there may also be technical contextual information mentioned in the question text without a corresponding question tag. For example, Figure 3 shows Question[5] 2551933 tagged with *django*. The question body also contains *session-variables*, which is a Stack Overflow tag and thus would be considered technical context based on our proxy. Thus, similar to how the question tag *django* is expected to be mentioned in many places in this thread, it is also likely that many answers and comments will mention *session-variables*. The red underlines in Figure 3 shows the tag mentioned as plain text in the question body.

Thus, to identify sentences with additional context, we first compile the question's technical context by combining any tags

[5]https://stackoverflow.com/questions/2551933

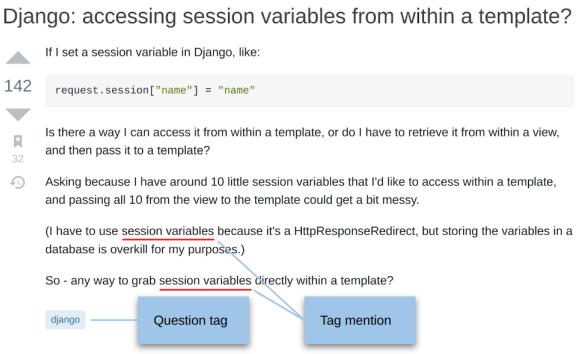

Figure 3: Stack Overflow 2551933 has only one question tag, *Django*. However, the question body contains *session variables* which is a Stack Overflow tag.

mentioned in the question body as well as the explicitly included question tags. We use the same subset of tags extracted from *Witt* to capture tags mentioned in the question body. Given the identified answer and comment sentences with technical context, we filter out any sentences where the identified context overlaps with the question's context.

At the end of Phase ①, we are left with a set of automatically identified sentences that likely contain additional context. However, we still need to manually review these sentences for confirmation.

3.2 Phase ② Manual Confirmation of Additional Context

While Phase ① automatically identifies candidate sentences that match our definition of additional context, the automated detection process can flag sentences that do not actually provide additional context.

For example, the Stack Overflow tag `super` refers to the programming language keyword used to invoke members of a super class. Our automated detection process would flag a sentence such as *In the super worst case, ...* However, this sentence does not actually contain any technical context. Thus, we need to first manually analyze each flagged sentence to make sure that the tag is used as intended and that this sentence actually contains additional context. Additionally, in this manual review phase, we perform a qualitative analysis on the reason why the additional context is mentioned. Specifically, we answer the following questions during the manual review phase:

(Q1) Is the tag used as intended in the tag description? *Yes/No*
(Q2) Does this tag overlap with the immediate parent's context? *Yes/No*
(Q3) Why is this "tag/context" mentioned in this thread? *Free form text*

We use Q1 and Q2 to confirm sentences with additional context, which we then use to answer RQ1 as shown in Figure 2. We first explain the coding guide for answering these questions and then describe the process we used to distribute sentences across multiple authors for review. In Phase ③, we perform open card sorting [19] of the reviewers' answers of Q3 (collected in Phase ②) to determine

categories of reasons. We then ask the reviewers to go through the sentences again and use these fixed categories to answer Q3.

Coding guide for annotating question 1 (Q1): We use the Stack Overflow tag description to determine whether a tag is used as intended in a sentence. We create a guideline to help annotators when they face ambiguous cases. If a tag has no description, we answer *No* to Q1. For example, *defaults* and *data-lake* have no description. When those tags appear in sentences, we select *No* to avoid subjective interpretation of the tag's intention. If the tag is part of a URL (e.g.*php* in `owasp.org/index.php/`), part of a file name (e.g. *jackson* in `jackson-module-jaxb-annotations-2.3.2`), or a path (e.g. *html* in `/var/www/html/sandbox/sandbox/wsgi.py`), we choose *No* to Q1 as well. Table 4 shows three sentences extracted from the sample of threads that we annotated for this study. We only show the responses for the first two questions (Q1 and Q2) due to space limitation and clarity. The tag column shows the tag in the sentence as automatically captured by Phase ①. For example, the captured tag is *django 1.10* in sentence 1. Since this sentence discusses the settings file content in a particular Django version (i.e. 1.10), an annotator would answer 'Yes' to whether the tag is used as intended. In sentence 2, the detected *dot* tag in the sentence does not match with the tag description. Therefore, the answer to Q1 would be *No*. Note that if a reviewer answers *No* to Q1, they do not need to answer the rest of the questions, since the rest of the questions only apply if the tag is used as intended.

Coding guide for annotating question 2 (Q2): When answering Q2, we consider the content hierarchy of a Stack Overflow thread. Our automated process in Phase ① already does a certain amount of filtering. For example, if a tag mentioned in a sentence is in the question context, then that sentence is filtered out. For simplicity and precision, we did not automatically process the code snippets or images in the question, because we focused on textual content. However, since there may be cases where a technical context is mentioned in a code snippet in the question, we need to manually verify if there is an overlap.

Figure 4 shows the hierarchical structure of a thread where the question is at the top level, question comments and answers are at the next level, and answer comments are at the bottom of the hierarchy. Assume we extracted the contexts *a, b, c* and *d* in the various parts of the thread as shown in the Figure 4. When we are reviewing comment *ac2*, we would select *Yes* for the overlap question since *d* exists in answer *a1*, which is an immediate parent of the comment. On the other hand, when reviewing the same context *d* for answer *a1*, an annotator would mark *No* overlap since *d* was not mentioned in any of *a1*'s parents. Following this guide, in Figure 6, a reviewer would mark no overlap for sentence 1 but yes for sentence 3 since *static* does appear in the code snippet posted in the question even though the tag is used as intended. Note that the answers to Q1 and Q2 determine whether a sentence is considered as containing additional context or not. Specifically, we consider a sentence as containing additional context if the tag is used as intended (Q1=Yes) and there is no overlap with any of its parents (Q2=No). In Table 4, Sentence 1 is the only sentence containing additional context.

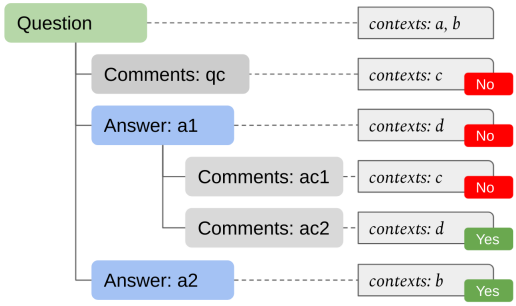

Figure 4: Stack Overflow thread hierarchy to determine tag overlap. *Yes* means this context is already previously mentioned in a parent, while *No* indicates no overlap.

Coding guide for annotating question 3 (Q3): To understand why the identified tags are mentioned as additional context, we ask annotators to provide a free-form text response for Q3. Annotators read the surrounding text to understand the purpose of the tag in the sentence. We did not provide any guidance for answering this question, because we did not want to influence the annotators' response. We later use this information in Phase ③.

Distributing threads between annotators: As mentioned earlier, we manually review a statistically representative sample of 69 threads from each tag, totalling 207 threads. To ensure the reliability of study, we use an iterative process for manually coding the data, starting with a general guideline for capturing additional contexts. Instead of annotating all threads at once, we conduct two pilot rounds using only a subset of threads from each tag. In the first round, three authors annotate the same 30 threads. We calculate the Fleiss' kappa score [6] for Q1 and Q2 to determine the ambiguity of the coding task. We then discuss any disagreements and use the insights from our discussions to remove ambiguities in the coding guide in each round. The three authors then use the refined coding guide to annotate another 30 threads. Table 5 shows the statistics of the first two rounds of annotations. Note that we randomly select ten threads from each tag for the first two rounds (a total of sixty threads). However, Table 5 only counts the threads and sentences that the automated detection process from Phase ① flagged as candidates for review. For example, in round 1, only six out of ten *Json* threads had potential additional contexts. After round 2, we were satisfied with the moderate agreement level [2] (0.416). This level of agreement is not surprising since "additional context" is a new concept without existing coding schema. Also, we annotated questions Q1 and Q2 at the same time in each round. Therefore, disagreement in Q1 results in disagreement in Q2. We then distribute the remaining 147 among the three authors to review individually.

The result of Phase ② is a set of confirmed sentences with additional context, which allows us to answer RQ1.

[6]Answer of this column determined by the outcome of Q1 and Q2. It is shown here only for the demonstration, but never used during the annotation.

Table 4: Examples of annotation process

#	Sentence	Tag	Used as intended (Q1)	Overlap (Q2)	Additional context[6]
1	in Django 1.10 , the `django.template.context_processors.request` was already in the setting file : D	django 1.10	Yes	No	Yes
2	Ultimately, use the querystring method to get around the dot in the route	dot	No	-	-
3	Because you are deploying on Heroku, you need to use something like white noise to serve your static files.	static	Yes	Yes	No

Table 5: Fleiss kappa scores for Q1 and Q2 of first two annotation rounds. Since Q2 is answered only if Q1=*Yes*, we show the number of sentences annotated in each round for each question.

Round	Tag	Threads	Q1 annotation		Q2 annotation	
			#	Kappa	#	Kappa
1	Json	6	35	0.91	23	0.137
	Django	8	46	0.613	30	0.273
	Regex	10	96	0.809	64	0.106
	Overall	24	177	0.783	117	0.223
2	Json	7	73	0.69	39	0.319
	Django	5	17	0.664	8	0.7
	Regex	6	37	0.881	11	0.542
	Overall	18	127	0.757	58	0.416

3.3 Phase ③ Qualitative Analysis of Additional Context

In Phase ③, we answer RQ2 and RQ3. Once we determine that a tag mentioned in a sentence is an additional context in Phase ②, we analyze the tags to determine their high-level technological category (RQ2) and why the additional context is mentioned (RQ3).

3.3.1 Identifying categories of additional context. To answer RQ2, we conduct open card sorting [19] of the additional contexts identified in Phase ②. This technique allows us to iteratively arrange cards (tags) into groups so that we can understand the types of additional context. We create cards for all tags that are identified as additional contexts. Three authors then work together to group tags that have similar characteristics, based on their tag descriptions. They then provide a descriptive name for each category. For example, *java*, *r*, and *c++* are categorized as programming language. Note that we did not rely on tag categories in *Witt* for this task, because there is more than one possible category for each tag. For example, in Witt, the tag *django* is categorized as "software", "framework", and "web-framework".

3.3.2 Identifying reasons for mentioning additional context. To determine categories of reasons for mentioning additional context, we use the annotators' free-text answers to Q3 from Phase ②. We collect all free-form responses of Q3 from all annotation rounds and the three authors perform an open coding card sorting of these reasons. Once we determine categories of reasons, we ask each author to answer Q3 again, but this time using closed coding where they have to select one of the pre-determined categories. To ensure our categories and their definitions are clear for the closed-coding analysis, we first conduct a pilot round using only a subset of threads from our sample set of threads and using three authors for each sentence. Then we refine the coding guide by providing examples of when to use each code, because we observed that annotator agreement is low during the pilot round. After refining the coding guide, we assign two authors to annotate each of the remaining sentences. Disagreements in round 2 were handled through discussions involving all three authors. Table 6 shows the agreement rate between raters in each round.

Table 6: Kappa scores for reasons for using "tag/context" in a sentence (Q3, Phase ③). We use Fleiss Kappa Score for annotations in both rounds.

Round	Raters	# Sentences	Kappa Score
1	1-2-3	105	0.354
2	1-2	144	0.83
	2-3	146	0.847
	1-3	144	0.777

4 EMPIRICAL STUDY RESULTS

4.1 RQ1: Frequency of additional context in Stack Overflow threads

Overall, we analyze 207 threads in our empirical study, containing a total of 3,504 sentences. The "Processed" (dark grey) bars in Figure 5 show the distribution of these processed sentences across answers, answer comments, and question comments. Out of these 3,504 sentences, the automatic identification process of Phase ① flagged 595 sentences for review. We then manually reviewed these 595 sentences in Phase ②. To answer RQ1, we count the sentences that were annotated with *Yes* to Q1 and *No* to Q2 in Phase ②.

Figure 5 shows how many sentences contain additional context (light grey bars) compared to the total number of processed sentences. Overall, out of the 3,504 processed sentences, only 288 (~8%) sentences contain additional contexts. As shown in Figure 6, the

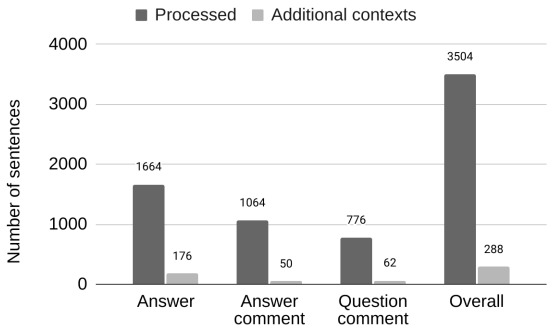

Figure 5: Comparison of sentences with additional contexts to total number of sentences processed.

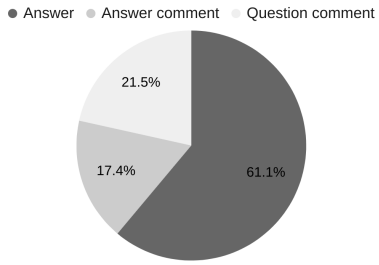

Figure 6: Location of the 288 confirmed sentences with additional context.

majority of the 288 sentences with additional context were found in answers (61.1%). The remaining sentences with additional context appear in question comments (21.5%) and answer comments (17.4%), respectively. Note that these 288 confirmed sentences belong to 96 unique threads. This means ~46% of the 207 threads that we study contain at least one sentence with additional context.

RQ1 Summary: While only ~8% of the total sentences processed contain additional context, 46% of the threads we analyze contain at least one sentence with additional context. The majority of sentences with additional context (61.1%) occur in answers.

4.2 RQ2: Types of additional context in Stack Overflow threads

The 288 confirmed sentences with additional contexts in RQ1 contain 142 unique tags. Through open card sorting, we create 16 technical categories to describe these tags. Table 7 shows the categories, the number/proportion of tags that belong to each category, and their distribution in sentences with additional contexts. The complete list can be found in our artifact page[7].

The top five categories in Table 7 account for ~79% of the tags we found as additional contexts. The rest of the tags are spread across the remaining eleven categories. Nearly 38% of the additional contexts that we find are either a "computing concept", such as *command*, or a "programming concept", such as *package*. The third column of Table 7 shows the distribution of tag categories in sentences with additional contexts. Computing concepts and programming concepts account for approximately 46% of the sentences with additional contexts while the other top four categories account for 45% of the sentences. Apart from concepts, the most common types of additional contexts are programming languages (17% of sentences) and library/frameworks (11% of sentences). Overall, the categories library/framework, programming language, tool/application, API and database cover 50% of the additional contexts and are found in 45% of the sentences with additional contexts.

Table 7: The 16 categories of tags that appear in the 288 sentences with additional context. We show the number (proportion) of tags in each category, as well as the number (proportion) of sentences.

Tag category	Tag count (%)	Sentence Count (%)	Example
Computing concept	32 (23%)	96 (31%)	command
Library/framework	26 (18%)	35 (11%)	asp.net
Programming concept	22 (15%)	47 (15%)	package
Programming language	16 (11%)	52 (17%)	perl
Tool/application	16 (11%)	26 (8%)	browser
API	8 (6%)	17 (5%)	tostring
Database	5 (4%)	11 (4%)	oracle
Standard	3 (2%)	6 (2%)	wsgi
Protocol	3 (2%)	9 (3%)	http
Data format	2 (1%)	3 (1%)	cbor
IDE	2 (1%)	2 (1%)	pyscripter
Web server	2 (1%)	4 (1%)	iis
Template engine	2 (1%)	2 (1%)	razor
Technology stack	1 (1%)	1 (0%)	xampp
Project	1 (1%)	1 (0%)	gnu
Hosting service	1 (1%)	2 (1%)	github

RQ2 Summary: 38% of the tags used as additional context are either computing concepts or programming concepts. Categories such as library/framework, programming language, tool/application, API and database cover 50% of the types of additional contexts and are found in 45% of the sentences with additional contexts.

4.3 RQ3: Purpose of additional contexts

Our open-coding of the free-text reasons provided for A3 resulted in ten reasons as to why additional contexts are mentioned in a give thread. Table 8 shows the distribution of reasons across the 288 sentences. We first explain each category, along with examples from Stack Overflow threads.

We find that in many cases, the additional context is mentioned as a direct solution to the posted problem. For example, *Solution - suggest API/library/framework* includes sentences where a user suggests an API/library/framework as a direct solution to a problem, e.g., "*Use a StringBuilder instead*" from Question[8] 25803443

[7] https://figshare.com/s/97007e09150ca97046a6

[8] https://stackoverflow.com/questions/25803443

contains the tag *stringbuilder*. This sentence directly suggests the `StringBuilder` API as a solution to the posed problem. Similarly, in the *Solution - suggest programming language* category, the purpose of the additional context is to suggest a programming language to directly solve the problem. The tag *javascript* used in "*Or go full javascript and add <p> element...*" from Question[9] 24261644 is an example of this case. We use the category *Solution - own experience* when the technical context is a direct solution that is simply based on personal experience, not necessarily a unique solution. For example, *PyScripter* in "*PyScripter works for me ..*" from Question[10] 11699407 states a solution based on a personal experience. Finally, we use *Solution - other* for any other additional contexts that is mentioned as a direct solution but that does not fall into any of the above solution categories. For example, Question[11] 26981520 contains the sentence "*As for the type of DBMS, a NoSQL would be recommended, like MongoDB*" which has the additional context *mongodb* that provides a direct solution but that does not fit the above categories.

We also find that several of the additional contexts are mentioned as solution conditions, i.e., the suggested solution works only under specific condition. Note that these categories are different from the *Solution* categories as they are not direct solutions. The first of these categories is *Solution condition - programming language* where the solution holds only for a specific programming language. *javascript* in "*Example in JavaScript:*" from Question[12] 33783429 is an example of a sentence in this category. The category *Solution condition - version* contains sentences where the tag is mentioned to denote a specific version (typically a programming language or library/framework) that is a condition for the answer. For example, the tag *django 1.6* in *"For django 1.6, in settings..."* (Question[13] 2551933) suggests that this answer applies only to this specific version of django. We use the category *Solution condition - environment* when the tag is used to denote the expected environment for this solution to work, such as operating system, browser, or technology stack. For example, "*In case you are dealing with a legacy system...*" (Question[14] 1812891) states a solution that only holds for old systems. Finally, *Solution condition - other* encompasses other solution conditions that do not fall into the above cases. For example, *postgresql* in *"If you use a PostgreSQL database, you can do this simpler ..."* (Question[15] 56201268) is a solution condition; using PostgreSQL as the database is not essential to answering this question, but the given solution in this answer only works with PostgreSQL. We argue that *solution condition* is the most interesting type of additional context since it indicates that a solution only applies in a certain situation. While it is possible to merge some of the subcategories of *solution condition* (or *solution*), the purpose of this analysis is to identify the intent of the tag mentioned in sentences.

When analyzing the reasons for mentioning additional contexts in sentences, we find that there are many cases that mention the tag as part of a technical explanation that describes a solution or elaborates details, whether directly or through external references. We also find cases where the tag represents essential or trivial terms that are necessary to the discussion of the thread (e.g., "*valid JSON format requires keys to be strings in double quotes*"). We group all these cases under one category *Technical explanation/discussion/resource*. Finally, we include an *Other* category for sentences with additional contexts that did not match with any of the previous categories.

Table 8: Distribution of reason categories in sentences with additional contexts.

Reason category	Sentence count (%)
Technical explanation/discussion/resource	224 (79%)
Solution - suggest API/library/framework	16 (6%)
Solution condition - version	13 (5%)
Solution - own experience	8 (3%)
Solution condition - environment	7 (2%)
Solution - other	7 (2%)
Other	4 (1%)
Solution condition - programming language	3 (1%)
Solution - suggest programming language	2 (1%)
Solution condition - other	1 (0%)

We find that most of the additional context in the 288 sentences is mentioned as part of a technical explanation/discussion or as an additional resource. This reason alone covers 79% of the 288 sentences. We find that 17% of the time, the additional context is a direct solution to the thread topic, including an API/library/framework or programming language. Such additional context would be valuable for developers navigating the thread. In 8% of the sentences, the additional context provides specific conditions that need to be satisfied for this solution to be useful to a developer. These include specific APIs or configuration environments, which developers are searching for [10].

We also compare the reason a specific tag is mentioned and the category of this tag from Table 7 from RQ2. We find that 94% of sentences whose tags belong to the programming concepts or computing concepts categories are categorized as technical explanation/discussion/resource in Table 8. We also find that 40% of the tags in the library/framework, programming language, tool/application, API, database, and web server categories in Table 7 are in one of the solution categories in Table 8. This suggests that the tag category may often help in determining the purpose of mentioning a specific tag in a sentence, which in turn can help determine if it can serve as a navigational cue. We discuss these implications more in Section 5.

RQ3 Summary: 79% of additional contexts are used to indicate necessary technical explanation or resource. The remaining approximately 20% are direct solutions or solution conditions.

[9]https://stackoverflow.com/questions/24261644
[10]https://stackoverflow.com/questions/11699407
[11]https://stackoverflow.com/questions/26981520
[12]https://stackoverflow.com/questions/33783429
[13]https://stackoverflow.com/questions/2551933
[14]https://stackoverflow.com/questions/1812891
[15]https://stackoverflow.com/questions/56201268

5 DISCUSSION

The goal of our empirical study is to understand how often additional context occurs in Stack Overflow answers/comments, as well as the types of additional context and their purpose in a thread. The results of such a study provide insights into whether additional context, i.e., technical information mentioned elsewhere in a thread and not already in the question, can potentially be used as navigational cues.

Implications for Navigation. Our findings show that approximately half of the threads we analyzed contain at least one sentence with additional context. This means that in almost half of the threads, there is potential for providing additional navigational cues. Additionally, those sentences account for only ~8% of all sentences in these threads. This is a positive indication for navigational cues; if a large proportion of sentences contained additional context, using all of them as cues could clutter the thread instead of helping users identify the information they are looking for. Overall, we envision that one way for adding navigational cues is by representing the additional context we studied as answer/comment tags as shown in Figure 7. Once a user finds a relevant question/thread, they can use these answer/comment tags to manually navigate to relevant information on the thread (i.e. quickly scrolling to a relevant answer). That is, we do not intend to make use of answer/comments tags for searching threads. Currently, Stack Overflow uses question tags to help users search for relevant threads. Therefore, the added answer/comment tags should not divert the search engine as they are not part of the question tags.

Implications for Automated Creation of Navigational Cues. Ideally, the navigational cue tags we envision in Figure 7 would be automatically captured and created without the need for their authors to add them. This allows the addition of navigational cues to existing Stack Overflow content. In our work, we used tags as proxies for identifying additional context. Based on the findings of RQ2 and RQ3, there are clearly certain types of tags that are more likely to serve as useful navigational cues than others. Specifically, tags in the programming concept and computing concept categories are typically mentioned as part of a general discussion/explanation that does not warrant tagging the whole answer with that tag. On the other hand, tags in categories such as API, library/framework, programming language, or web server were more likely to be about a direct solution or condition that is necessary for this solution to work. A developer navigating a thread might already be looking to solutions related to a specific technology or those that match their technology stack and requirements. Thus, creating navigational cues for such tag categories is more promising. Future research into automatically, and precisely, categorizing tags into the categories we identified can be helpful for adding navigation cues. Such automatic categorization along with our automated identification of additional context (Phase ①) can then be used to augment Stack Overflow threads with add navigational cues.

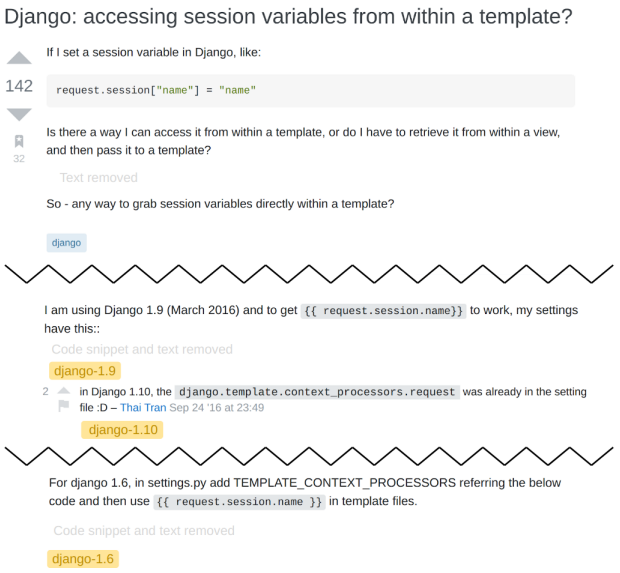

Figure 7: Emphasizing additional context through answer/comment tags that can serve as navigational cues

6 THREATS TO VALIDITY

Internal validity: We automatically capture tags mentioned in sentences to reduce the manual review burden on annotators. Instead of using the complete tag set from Stack Overflow, we use a subset of tags extracted from the *Witt* taxonomy [14]. Since the *Witt* taxonomy was created in 2019, our automated approach could miss newer tags mentioned in sentences that are not available in *Witt*. However, the goal of our study is to understand the usage of technical context and the general phenomena of additional context, not to automatically identify them. Therefore, using *Witt* does not affect our understanding of additional context; our results generally represent a lower bound of the occurrence of additional context.

We use the Stanford CoreNLP toolkit [11] to extract sentences from answers and comments. Any limitations of the tools affect the sentence extraction step of Phase ①. For example, the CoreNLP sentence processor may not properly parse poorly written or poorly punctuated sentences, leading to some of the reported sentences not being complete sentences. Since our goal is to understand the occurrence of additional context, searching for it in sentences with grammatical mistakes does not affect our results, especially since we manually validate all data.

When identifying additional context, three authors first annotate ~29% of the threads from the sample. The rest of the threads were divided among each author to annotate individually, which is standard practice when annotating large data sets [7, 8, 13]. Doing the initial round of annotation with three authors ensures common understanding of the coding guidelines. We also calculate and report inter-rater agreement for two of the annotation questions (Q1 and Q2). In each pilot annotation round, we observed substantial (>0.61) [2] agreement for Q1. Even though we observed a fair (0.223) score for Q2 in round 1, there was an improvement in agreement score in round 2 (0.416) once we removed the ambiguity in the annotation guide. In the pilot round, when all three authors annotate Q3, our inter-rater agreement for close coding was moderate (0.354) [2]. After refining the coding guide, the remaining sentences were divided among three authors such that each sentence is coded by

two authors. Our inter-rater agreement between each author was substantial (>0.7) [2]. The systematic process we followed and the increase in agreement scores gives us confidence in the reliability of the individually annotated data.

Construct validity: We use Stack Overflow tags as a proxy for technical context. Since Stack Overflow tags are user created contents, there could be multiple token-level representation for the same tag (e.g. *JS* and *Javascript*). The intention of automatically capturing tag mentions is to reduce the manual review burden of annotators from reading all sentences. While this proxy could exclude some of the sentences that contain additional contexts from being manually examined, our goal is to conduct an initial empirical study to understand the new concept of additional context before investing in efforts that capture all possible representations of technical contexts.

Some Stack Overflow tags may be used with a different intention in the sentences. This is why we did not rely solely on the automatic identification of technical context but chose to manually confirm all results. However, there may also be other important contextual information that can be used for navigation that goes beyond technical context (e.g., non-functional properties such as performance or security being mentioned in threads). We choose to limit the scope of this study to technical context to avoid the subjectivity of deciding what other contextual information may be [12]. Future work that includes human participants is needed to conclude that the additional navigational cues based on technical context will help developers find information faster; we do not claim to measure this in our work. Instead, we study how additional context is used to understand its potential to serve as navigational cues.

Some of the accepted answers in our data did not contain additional contexts, which can lead to the argument that adding visual cues for additional context can mislead users to unverified answers. However, answers other than the accepted one are not necessarily incorrect. Similar to the checkmark indicating that an answer is the one selected by the original poster (not necessarily the only correct answer [24]), an additional context visual cue will alert users that an answer/comment is specific to a particular context (e.g., an OS) and allow users to dismiss irrelevant content (e.g., about other OSs).

External validity: When collecting threads for annotation, we limit our study to threads from three tags: `json`, `regex`, and `django`. Even though we focused on threads from only these tags, the threads in our sample contain 142 unique technical contexts from different programming languages, databases, libraries/frameworks, and APIs. We do not claim that our study encompasses *all* possible additional context, but it provides first insights into the occurrence and usage of additional context in Stack Overflow threads.

7 CONCLUSION

Given the vast amount of knowledge available on Stack Overflow, finding the right information that applies to developers' specific requirements or technology stack is challenging. In this paper, we define *context* as information that characterizes the technologies and assumptions of a given question or answer. We define *additional context* as context in a thread's answer or comment that does not overlap with the question context. We argue that such additional context may provide useful navigational cues for Stack Overflow users. In this paper, we conduct an empirical study of the occurrence and nature of additional context on Stack Overflow to understand its potential for being used as navigational cues. We find that the majority of the additional contexts are computing concepts that discuss necessary technical details of the thread. However, technology categories such as library/framework, programming language, API, and tool/application usually provide a solution to the original problem. Since only a small percentage of threads contain additional context that provide a solution, we could leverage these findings to automatically identify such technical contexts to generate navigational cues.

REFERENCES

[1] Rabe Abdalkareem, Emad Shihab, and Juergen Rilling. 2017. On code reuse from StackOverflow: An exploratory study on Android apps. *Inf. Softw. Technol.* 88 (2017), 148–158. https://doi.org/10.1016/j.infsof.2017.04.005

[2] Douglas G Altman. 1990. *Practical statistics for medical research.* CRC press.

[3] Sebastian Baltes, Christoph Treude, and Martin P. Robillard. 2020. Contextual Documentation Referencing on Stack Overflow. *IEEE Transactions on Software Engineering* (2020), 1–1. https://doi.org/10.1109/TSE.2020.2981898

[4] Preetha Chatterjee, Minji Kong, and Lori L. Pollock. 2020. Finding help with programming errors: An exploratory study of novice software engineers' focus in stack overflow posts. *J. Syst. Softw.* 159 (2020). https://doi.org/10.1016/j.jss.2019.110454

[5] Anind K. Dey. 2001. Understanding and Using Context. *Pers. Ubiquitous Comput.* 5, 1 (2001), 4–7. https://doi.org/10.1007/s007790170019

[6] Rosa Falotico and Piero Quatto. 2015. Fleiss' kappa statistic without paradoxes. *Quality & Quantity* 49, 2 (2015), 463–470.

[7] Hideaki Hata, Raula Gaikovina Kula, Takashi Ishio, and Christoph Treude. 2021. Same File, Different Changes: The Potential of Meta-Maintenance on GitHub. In *43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021*. IEEE, 773–784. https://doi.org/10.1109/ICSE43902.2021.00076

[8] Hideaki Hata, Christoph Treude, Raula Gaikovina Kula, and Takashi Ishio. 2019. 9.6 million links in source code comments: purpose, evolution, and decay. In *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, Joanne M. Atlee, Tevfik Bultan, and Jon Whittle (Eds.). IEEE / ACM, 1211–1221. https://doi.org/10.1109/ICSE.2019.00123

[9] Nasif Imtiaz, Akond Rahman, Effat Farhana, and Laurie A. Williams. 2019. Challenges with responding to static analysis tool alerts. In *Proceedings of the 16th International Conference on Mining Software Repositories, MSR 2019, 26-27 May 2019, Montreal, Canada*, Margaret-Anne D. Storey, Bram Adams, and Sonia Haiduc (Eds.). IEEE / ACM, 245–249. https://doi.org/10.1109/MSR.2019.00049

[10] Hongwei Li, Zhenchang Xing, Xin Peng, and Wenyun Zhao. 2013. What help do developers seek, when and how?. In *20th Working Conference on Reverse Engineering, WCRE 2013, Koblenz, Germany, October 14-17, 2013*, Ralf Lämmel, Rocco Oliveto, and Romain Robbes (Eds.). IEEE Computer Society, 142–151. https://doi.org/10.1109/WCRE.2013.6671289

[11] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, System Demonstrations*. The Association for Computer Linguistics, 55–60. https://doi.org/10.3115/v1/p14-5010

[12] Sarah Nadi and Christoph Treude. 2020. Essential Sentences for Navigating Stack Overflow Answers. In *27th IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2020, London, ON, Canada, February 18-21, 2020*, Kostas Kontogiannis, Foutse Khomh, Alexander Chatzigeorgiou, Marios-Eleftherios Fokaefs, and Minghui Zhou (Eds.). IEEE, 229–239. https://doi.org/10.1109/SANER48275.2020.9054828

[13] Seyed Mehdi Nasehi, Jonathan Sillito, Frank Maurer, and Chris Burns. 2012. What makes a good code example?: A study of programming Q&A in StackOverflow. In *28th IEEE International Conference on Software Maintenance, ICSM 2012, Trento, Italy, September 23-28, 2012*. IEEE Computer Society, 25–34. https://doi.org/10.1109/ICSM.2012.6405249

[14] Mathieu Nassif, Christoph Treude, and Martin P. Robillard. 2019. Witt: querying technology terms based on automated classification. In *Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*. IEEE / ACM, 63–66. https://doi.org/10.1109/ICSE-Companion.2019.00039

[15] Cornelius Ncube, Patricia A. Oberndorf, and Anatol W. Kark. 2008. Opportunistic Software Systems Development: Making Systems from What's Available. *IEEE Softw.* 25, 6 (2008), 38–41. https://doi.org/10.1109/MS.2008.153

[16] Stack Overflow. [n.d.]. *Ask questions, get answers, no distractions.* https://stackoverflow.com/tour

[17] Stack Overflow. [n.d.]. *How do I ask a good question?* https://stackoverflow.com/help/how-to-ask

[18] Megan Squire and Christian Funkhouser. 2014. "A Bit of Code": How the Stack Overflow Community Creates Quality Postings. In *47th Hawaii International Conference on System Sciences, HICSS 2014, Waikoloa, HI, USA, January 6-9, 2014.* IEEE Computer Society, 1425–1434. https://doi.org/10.1109/HICSS.2014.185

[19] Klaas-Jan Stol, Paul Ralph, and Brian Fitzgerald. 2016. Grounded theory in software engineering research: a critical review and guidelines. In *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016*, Laura K. Dillon, Willem Visser, and Laurie A. Williams (Eds.). ACM, 120–131. https://doi.org/10.1145/2884781.2884833

[20] Christoph Treude and Martin P. Robillard. 2016. Augmenting API documentation with insights from stack overflow. In *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016*, Laura K. Dillon, Willem Visser, and Laurie A. Williams (Eds.). ACM, 392–403. https://doi.org/10.1145/2884781.2884800

[21] Christoph Treude and Martin P. Robillard. 2017. Understanding Stack Overflow Code Fragments. In *2017 IEEE International Conference on Software Maintenance and Evolution, ICSME 2017, Shanghai, China, September 17-22, 2017.* IEEE Computer Society, 509–513. https://doi.org/10.1109/ICSME.2017.24

[22] Mario Linares Vásquez, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, and Denys Poshyvanyk. 2014. How do API changes trigger stack overflow discussions? a study on the Android SDK. In *22nd International Conference on Program Comprehension, ICPC 2014, Hyderabad, India, June 2-3, 2014*, Chanchal K. Roy, Andrew Begel, and Leon Moonen (Eds.). ACM, 83–94. https://doi.org/10.1145/2597008.2597155

[23] Bowen Xu, Zhenchang Xing, Xin Xia, and David Lo. 2017. AnswerBot: automated generation of answer summary to developersź technical questions. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering, ASE 2017, Urbana, IL, USA, October 30 - November 03, 2017*, Grigore Rosu, Massimiliano Di Penta, and Tien N. Nguyen (Eds.). IEEE Computer Society, 706–716. https://doi.org/10.1109/ASE.2017.8115681

[24] Haoxiang Zhang, Shaowei Wang, Tse-Hsun Chen, and Ahmed E. Hassan. 2021. Reading Answers on Stack Overflow: Not Enough! *IEEE Trans. Software Eng.* 47, 11 (2021), 2520–2533. https://doi.org/10.1109/TSE.2019.2954319