Debugging with Stack Overflow: Web Search Behavior in Novice and Expert Programmers

Annie Li
annieli@umich.edu
University of Michigan, CSE
Ann Arbor, Michigan, USA

Madeline Endres
endremad@umich.edu
University of Michigan, CSE
Ann Arbor, Michigan, USA

Westley Weimer
weimerw@umich.edu
University of Michigan, CSE
Ann Arbor, Michigan, USA

ABSTRACT

Debugging can be challenging for novice and expert programmers alike. Programmers routinely turn to online resources such as Stack Overflow for help, but understanding of debugging search practices, as well as tool support to find debugging resources, remains limited. Existing tools that mine online help forums are generally not aimed at novices, and programmers face varying levels of success when looking for online resources. Furthermore, training online code search skills is pedagogically challenging, as we have little understanding of how expertise impacts programmers' web search behavior while debugging code.

We help fill these knowledge gaps with the results of a study of 40 programmers investigating differences in Stack Overflow search behavior at three levels of expertise: novices, experienced programmers who are novices in Python (the language we use in our study), and experienced Python programmers. We observe significant differences between all three levels in their ability to find posts helpful for debugging a given error, with both general and language-specific expertise facilitating Stack Overflow search efficacy and debugging success. We also conduct an exploratory investigation of factors that correlate with this difference, such as the display rank of the selected link and the number of links checked per search query. We conclude with an analysis of how online search behavior and results vary by Python error type. Our findings can inform online code search pedagogy, as well as inform the development of future automated tools.

CCS CONCEPTS

• **Software and its engineering** → *Software development techniques*; *Software testing and debugging*; • **Social and professional topics** → **Software engineering education**; • **Computing methodologies** → *Online learning settings*.

KEYWORDS

Stack Overflow, Controlled Human Study, Online Search Behavior, Debugging, Programming Experience

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ACM Reference Format:
Annie Li, Madeline Endres, and Westley Weimer. 2022. Debugging with Stack Overflow: Web Search Behavior in Novice and Expert Programmers. In *This is a preprint of a paper that will appear at the 44nd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET '22), May 21–29, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Debugging often is a difficult and time-consuming task [22]. One common debugging strategy involves turning to online forums such as Stack Overflow[1] for help [10, 20, 23, 25, 26]. For example, one 2017 study found that professional software developers spend 11% of their workdays on work-related browsing [27]. Similarly, software developers at Google frequently search for code help online, completing an average of 12 search queries every workday [33]. Studies have found that Stack Overflow can be as effective as traditional communication for conveying debugging information [36].

The Problem: Despite the prevalence of online debugging information on Stack Overflow, developers can still struggle to efficiently find helpful debugging information online [32]. These struggles are particularly evident for novice programmers [7], including those in their first year of programming instruction [4]. Debugging, including debugging using Stack Overflow, can be daunting for novices because they have insufficient knowledge about the program behavior, are unfamiliar with the programming language and environment, or do not know how to apply effective debugging strategies for the task at hand [1, 7, 28].

Various support systems, including tools that automatically mine for helpful Stack Overflow posts [24, 31], have been proposed to augment IDEs and ameliorate finding online debugging information. However, these tools are typically not aimed at novices and often have limited empirical basis in studies of human behavior. As another way to support novices, direct mentoring by experts for training Stack Overflow search query formation has been proposed [4]. For writing Stack Overflow posts, such mentorship has been found to increase the community reception of novices' questions by 50% [14]. However, the effects of mentorship on the post search process (rather than the post writing process) is less explored. Approaches that incorporate multiple factors influencing the behavior of different developers, including experience, have the potential to be even more effective in practice.

This Study: We help close this knowledge gap by conducting and reporting the results of a human study of 40 programmers with varying levels of experience. We investigate, in a controlled manner, experience-related differences in searching for Stack Overflow

[1] https://stackoverflow.com

posts that are relevant to debugging a set of Python bugs. Our goals are to *identify and characterize which search practices and behaviors distinguish programming experts from novices*, as well as to *understand the magnitude of such observed differences*. Our participants have varying levels of both general programming experience (0.5–25 years) and Python experience (0–10 years), admitting analysis of both general and language-specific expertise.

Our key insight is that we can measure and understand student behavior by recording and analyzing whole-screen interactions. This allows us to use an experimental setup with more ecological validity than previous studies: participants search for debugging help using a standard web browser and search engine (Google in our experiment) rather than the built-in Stack Overflow search used by some previous studies [8, 24]. Previous work has indicated that Google is one of the most common search platform for code-related help [42]. Indeed, 37 out of 40 participants in our study reported Google as their preferred search engine. At the same time, we capture aspects of the search process including navigation behavior, search result content and search query text, that can otherwise be difficult to access without intrusive plug-ins.

Research Context: Previous work has identified that novices are more likely to formulate successful Stack Overflow queries when collaborating with an expert [4]. However, we do not yet know the magnitude of experience-related differences overall, making it difficult to know if online code search training should be prioritized. Furthermore, while previous work has investigated how novice programmers write Stack Overflow posts [14], browse a given Stack Overflow post [10], or focused on how experienced programmers search for help with an unfamiliar language [6], we do not yet know what factors during the *search process* correlate with overall expertise, which could help inform pedagogy and training. To the best of our knowledge, this work is the first study to compare the effects of both general and language-specific experience on search behavior for Stack Overflow posts.

Summary of Results: Overall, we find significant and substantial experience-related differences in the efficacy of using Google to search for helpful debugging information on Stack Overflow ($p < 0.05$). In particular, we find that programming experts with Python experience are able to find helpful information on Stack Overflow for a given Python error 63% of the time, compared to 49% of the time for experts without Python experience, and only 44% of the time for programming novices. This represents a 19% difference between Python-familiar experts and novices, a moderate size effect (Cramer's $V = 0.12$). Furthermore, in an exploratory analysis, we find that various search factors, such as the Google search result rank of clicked links and the number of links clicked per search query, are positively correlated with programming experience: experts click on more links further down in the Google search results than do novices, suggesting that novices might be trained to emulate these behaviors, or that they may be supported by tools. Finally, we find that the error type of the given bug has a significant moderate-strong effect on the ability of the participant to find helpful information on Stack Overflow ($p < 0.001$, Cramer's $V = 0.14$): participants are more likely to find helpful Stack Overflow posts for `IndentationErrors` and `AttributeErrors`, and less likely to find helpful posts for `SyntaxErrors` or `TypeErrors`, information that may help focus the design of future automated tools. We conclude with a discussion of the implications of our results.

(a) Buggy code example stimulus

```python
a = [1,2,3,0,3]
for i in a:
    if !i:
        break
    print(i)
```

(b) Error message associated with buggy stimulus

```
File "script.py", line 3
  if !i:
     ^
SyntaxError: invalid syntax
```

Figure 1: Simple example bug from our survey. In this case, the programmer mistakenly used ! rather than the word `not` to indicate logical negation.

2 MOTIVATING EXAMPLE

In this section, we demonstrate the efficacy difference between how experts and novices search for online debugging help, and also show how concrete actions during the search may help explain that difference, using an example from our experiment. Consider the buggy Python program and associated Python error message in Figure 1. This program contains a Python `SyntaxError` caused by the use of "!" rather than "not" to express logical negation. Developers with different levels of programming experience may take different approaches to finding relevant debugging information for this error on a website like Stack Overflow. Decisions made while searching for relevant information can influence whether the activity is helpful for debugging.

Table 1 shows an example of indicative web search processes (including query content and the display rank of links clicked) used by an expert and novice programmer in response to that error. While both sets of queries capture the notion of a syntax error, they vary in prose content. For example, some novice queries spell out "exclamation" rather than using the "!" symbol. On the other hand, all of the expert-written queries include the "if !i" prefix, which none of the novice queries include. These differences in query text result in different candidate Stack Overflow pages being returned by Google. Further differentiating the two programmers, the novice always clicks on the first link displayed in the search result, while the expert is more likely to investigate later links.

Ultimately the search queries written by the novice did not lead to a search process that was helpful in resolving the bug. By settling on the "exclamation point python" query in Table 1 and choosing the first result returned by Google (a common novice tactic observed in our study), the search process leads to a Stack Overflow page

Novice Programmer Search Queries	Links	Rank
! invalid syntax python	0	
python invalid syntax exclamation	0	
python invalid syntax	1	1
exclamation point python	1	**1**
Expert Programmer Search Queries		
if !i: ^SyntaxError: invalid syntax	1	1
if !i: SyntaxError: invalid syntax	1	3
if !i SyntaxError: invalid syntax python not operator	2	1, **8**

Table 1: Queries made by Novice and Expert programmers in response to the error in Figure 1. The literal search query is shown on the left. The "Links" column indicates the number of links clicked per query while the "Rank" column indicates the display rank of each link clicked. The query yielding the information reported as the most helpful is highlighted, while the display rank of the link that contained this information is bolded.

that does not contain relevant information. The link[2] contains text explaining that, "!= means "not equal to" and is a logical comparison," but that statement is not relevant for fixing the bug.

By contrast, the expert's search process in Table 1, with its more complicated queries and use of links displayed further down in the search results, resulted in finding a Stack Overflow post with much more helpful information: "Some of the operators you may know from other languages have a different name in Python. The logical operators `&&` and `||` are actually called `and` and `or`. Likewise the logical negation operator `!` is called `not`."[3]

We hypothesize that these differences in expert and novice behavior, such as the construction of query strings and the consideration of multiple candidate search results, can have a significant impact on the efficacy of using Stack Overflow to help with debugging. The degree to which such behaviors are relevant and influential can impact how we teach the use of search in introductory programming as well as the aspects of this process that may most benefit from tool support. While this motivating example showed how such features can be relevant using the queries and results of two programmers, the remainder of this paper investigates such hypotheses systematically in a controlled human study involving 40 programmers with multiple levels of expertise.

3 BACKGROUND

In this section, we provide a brief summary of related works across three main categories: how developers use and search for online resources like Stack Overflow, existing tools that automatically mine Stack Overflow content, with a focus on different levels of programming expertise, and general processes involved while debugging.

[2]https://stackoverflow.com/questions/22209729/what-does-do-mean-in-python
[3]https://stackoverflow.com/questions/2485466/pythons-equivalent-of-logical-and-in-an-if-statement

3.1 Stack Overflow Usage

The search for new information using online resources is a common and a significant task during software development [10, 20, 23, 26]. Developers commonly search for explanations for unknown terminologies, explanations for error messages, and solutions to common programming bugs [42]. Various studies have investigated how developers interact with crowd-sourced forums, of which Stack Overflow is the most popular, to assist with programming tasks.

There is a body of work that examines how Stack Overflow questions can be categorized, with the goal of understanding the context behind questions being asked, as well as how to leverage this categorization for more efficient retrieval. Stack Overflow is a particularly effective resource for code reviews and conceptual questions [37, 40]: the Stack Overflow community was found to more frequently provide accepted answers to "review," "conceptual," "how-to," and "error" questions compared to other categories.

Researchers have also evaluated the quality of Stack Overflow post answers, discovering that explanations accompanying code examples are as important as the examples themselves [29]. Additionally, one study of the usability of Python snippets on Stack Overflow found that they are 76% parsable and 25% runnable. These numbers are higher for the top Stack Overflow results returned by Google [44].

Social aspects of Stack Overflow have also been explored. For instance, Ford et al. identified barriers women face when contributing to the site (e.g., doubts regarding the necessary level of expertise to contribute) [15]. Further research found that some gender-based barriers may be alleviated through peer parity, a community-focused process that pairs users based on identity such that an "individual can identify with at least one other peer when interacting in a community" [13].

Studies have also investigated how developers formulate programming-related search queries for online resources such as Stack Overflow. In an empirical study on how developers search the web, Hora found that queries typically start with keywords, are short, omit functional words, and are similar to each other [19]. By contrast, students often used familiar terms in searches [6]. Critically for our investigation, such changes (e.g., moving the language name from the beginning to the end of a query) can have non-negligible search result effects [19].

Previous work has identified expertise-related patterns in Stack Overflow interactions. For example, Chatterjee et al. found that novice programmers pay attention to only 27% of the code and 15–21% of the text in a Stack Overflow post, focusing primarily on accepted answers [10]. Additionally, explicitly teaching novice programmers on how to search for online debugging help has been proposed: Al-Sammarraie found that novices are more likely to formulate successful Stack Overflow queries when collaborating with an expert than when working alone [4], indicating expertise is connected with search query efficacy. However, expertise-related differences in how participants *search* for Stack Overflow posts is under-explored. In this study, we seek to fill this gap by carefully considering Stack Overflow search query content and web search browsing behavior as a function of experience.

3.2 Automated Information Retrieval Tools

Several automated tools have been proposed to assist in finding relevant Stack Overflow posts or to facilitate the search query construction process [24, 31, 36, 45]. For example, Thiselton and Treude constructed Pycee, a Sublime Text plugin that enhances Python error messages by automatically summarizing information from Stack Overflow.

There is significant educational potential for tools that augment Stack Overflow. Dondio and Shaheen found that incorporating Stack Overflow as a resource in a university-level programming course was at least as effective as traditional means [11]. Furthermore, previous research has found that expert mentorship can help novices successfully find information on Stack Overflow [4]. However, overwhelmingly created with data from programming experts [24, 31, 36, 45], extant tools do not account for differences between novices and experts.

3.3 General Debugging Processes

Beyond studies and tools specific to Stack Overflow, researchers have also studied the general processes involved in debugging. Gilmore proposed a model of debugging with three stages: program comprehension, mental representations of the behavior of the buggy and correct program, and a process of mismatch correction between the two representations [16]. Similarly, Von Mayrhauser and Vans identified hypothesis generation along with knowledge use as key debugging processes [39]. Researchers have also used medical imaging to investigate debugging processes at the neurological level. For example, Castelhano et al. found that decision areas of the brain activate in the moment when developers discover the cause of a bug [9].

Hypothesis generation is a debugging process of particular interest to the current work: Alaboudi and LaToza found that having a correct hypothesis early was a strong predictor of debugging success [5]. However, even when having access to online resources, they found that developers struggled to generate said debugging hypothesis, correct or otherwise. As they both involve the generation of questions and a search for a solution, Stack Overflow queries may act as a medium for developers to both express and test their debugging hypotheses. Additionally, it may also be that effective Stack Overflow query formation can help developers access resources that inspire new debugging hypotheses more efficiently. In this paper, we consider how programming expertise impacts the the effective formation of such queries.

4 EXPERIMENTAL SETUP

To investigate how Stack Overflow search behavior differs by developer skill level and bug type, we conducted a controlled user study with 40 participants. We observed how developers used a standard browser and search engine (Google) to find relevant Stack Overflow posts while debugging Python programs. All materials regarding our experimental setup and data analysis can be found in our replication package.[4] To increase confidence, a randomly selected subset of all annotations were checked for accuracy by another researcher.

[4]Our replication package can be found at: https://github.com/CelloCorgi/StackOverflow_ICSE_SEET_2022

4.1 Experimental Overview

During the 50-minute study, participants were presented with a series of debugging prompts; each prompt consisted of a buggy Python program and the associated error message (see Section 4.3). Stimuli were shown to participants using the Qualtrics survey platform. Participants were then asked to use a web browser to find a Stack Overflow post that they perceive as helpful for debugging (e.g., a related fix for the given error, or an explanation that identifies why the error occurred). To ensure uniformity in search results, participants were asked to use the Google search engine, and limited their search results to only those from the Stack Overflow website with the search prefix `site:stackoverflow.com`. Participants had at most 10 minutes per prompt and were shown at most 12 prompts.

In total, we collected data regarding 1,201 queries from 332 debugging sessions. Upon finding a helpful post using Google, participants were asked to record the link to the post and to copy those snippet(s) of the post that they found the most helpful. Participants were also asked to describe how to fix each bug and if they thought they knew how to solve the bug before searching. Because our study uses real-world searches and posts, rather than curated information, we conducted a separate expert annotation to provide an assessment of found post helpfulness (see Section 5.2).

All study sessions were conducted over video conference with screen recording. We annotated each recording to collect additional data including search query text, search query timestamps, link click timestamps, and the search order ranking of the each link selected (see Section 5.1). Participants also provided self-reported expertise and demographic information.

4.2 Participant Recruitment

To understand expertise-related differences in Stack Overflow search behavior, we recruited participants with a diverse array of programming experiences.

We recruited participants via direct email, university mailing lists, and public forums such as Piazza and Twitter. 92 participants filled out a pre-screening form, of which 40 participated in our study. Table 2 contains a demographic breakdown of participants, and Table 3 provides participants' maximum, minimum, and average years of programming experience. This study's 40 participants had 0.5–25 years of programming experience, 0–9 years of professional development, and 0–10 years of Python. Participants were compensated \$50 for graduate students and professional developers and \$15 for undergraduates.

We call professional software developers or graduate students with three or more years of programming experience and greater than one year of Python experience "Python-familiar-experts". We note that developers with only three years of experience may not be considered experts in the traditional industrial sense. However, as our goal is to understand experience-related differences in search behavior to inform tools supporting introductory programming students, three years of experience is enough for a substantial comparison. "Non-Python-familiar-expert" participants had three or more years of general programming experience but only limited Python experience (one year or less). In our study, both "Python-familiar-experts" and "Non-Python-familiar-experts" consisted of

Table 2: Study participants: Python-familiar-experts (PF), Non-Python-familiar-experts (NPF) and Novices.

	PF-experts	NPF-experts	Novices
Number of Participants	20	9	11
Gender			
Woman	8	5	6
Man	12	3	5
Current Occupation			
Professional in Industry	5	2	0
Professional in Academia	2	0	0
Graduate Student	10	1	0
Undergraduate Student	3	6	11

Table 3: Programming experience of study participants.

Experience in years	*Max*	*Min*	*Average*
General Programming Experience	25	0.5	5.2
Python Specific Experience	10	0	2.5

a mix of advanced undergraduates, graduate students, and professional developers. Finally, "Novice" participants were beginning programmers, having limited experience both with Python and with programming in general (one year or less of experience in both). For our study, these were students enrolled in undergraduate introductory computer science courses at two large public research institutions.

4.3 Programming Stimuli

We presented participants with programs submitted by real users to Python Tutor, a popular web-based program visualization tool [18]. We chose Python Tutor to help study errors that novices encounter, and Python Tutor is widely used in both CS classrooms and also by self-directed learners around the world [17], and has been studied in software engineering contexts (e.g., see [12]).

To select our stimuli, we first filtered all 14,251,337 Python 3 programs submitted to Python Tutor between 2015 and 2017 for those that did not require user or file input, were not likely to have non-deterministic output. This resulted in 9,486,611 remaining programs, of which 3,735,811 produced runtime errors. We identified the five most common categories of errors as determined by the generated Python error message.[5] From most prevalent to least, these were `SyntaxErrors` (56.8%), `TypeErrors` (13.0%), `NameErrors` (14.1%), `IndexErrors` (6.0%), and `AttributeErrors` (2.9%), together accounting for 92.8% of errors in the Python Tutor dataset. From these each of these five error categories, we randomly selected eight programs producing an error of that category. This resulted in a corpus of 40 buggy programs used in our experiment. We note that of the eight `SyntaxError` programs, three produced an `IndentationError`, which is a common `SyntaxError` subclass in our dataset.

Stimuli were presented to participants in a Qualtrics survey. Each program and associated error message was provided in both image and textual format. The image version preserved syntax highlighting and formatting on the Qualtrics platform, while the textual version allowed participants to copy portions of the stimuli to aid in searching should they wish to do so. To ensure we had the statistical power for comparisons, we organized our study to ensure that each participant received at least one randomly-selected stimulus from each error category. Aside from this constraint, stimuli were selected randomly from the 40 programs we included in our study.

5 ANALYSIS METHODOLOGY

We now present our analysis methodology for both annotating the developer interaction screen captures (Section 5.1) and also determining debugging helpfulness of participant-reported Stack Overflow posts (Section 5.2). We conclude with a discussion of our statistical methods in Section 5.3.

5.1 Video Annotations

From the video recordings of participant sessions, the first author manually annotated screen recordings to mark various event timestamps, the contents of search queries, link URLs clicked by the participant, and other relevant information. For example, to collect data associated with search queries, we recorded the timestamps corresponding to when the participant started typing the query into the Google search bar, the timestamp for when the participant finished typing, and the contents of the query itself. In addition, we noted the search result ranking of the clicked link by the participant, as well as the Stack Overflow post ID. There were 12 instances across 6 participants where the participant clicked on an invalid link (marked as "-1" in our replication materials). Video annotations were spot-checked for accuracy by the second author.

Critically, participants did not have to narrate or call out any such information (e.g., "I am clicking on the second link"): debugging search interactions were natural and uninterrupted, and the annotation process was performed *post hoc* on recordings.

5.2 Post Relevance Assessments

Because participants could formulate any search query and visit any Stack Overflow page, we could not pre-assess the helpfulness of query results. We thus conducted an expert annotation with three author annotators to establish a quality and helpfulness assessment for all observed Stack Overflow posts and snippets (informally, a "ground truth" for this aspect of the experiment).

We define the *relevance* of a Stack Overflow link or snippet as a Boolean value. True means the Stack Overflow post content identifies a possible cause, directly provides a solution or example of correct code structure, explains how to resolve the error message for a context specific to that of the debugging prompt, or teaches general Python knowledge necessary for debugging the prompt. False means that the post did not contain any such information. Our full annotation rubric can be found in our replication materials on the project website.

To decide on the relevance of participant survey responses, the first two authors individually assessed each survey answer with relevance categories,[6] compared results, and arrived at a consensus for any discrepancies found. In the 12 of 332 cases where there was still no consensus about relevance, the third author was consulted to reach a final consensus. A similar process was followed to determine

[5] The *category* is the text before the colon in a Python 3.8 error message.

[6] See annotator spreadsheet in our replication package

the correctness of the fix the participant proposed for each stimulus, for which the third author was consulted for 2 out of 332 responses.

5.3 Statistical Methods

In general, our data was stored in a MySQL database, and we conducted our analysis using a Python Jupyter Notebook and `Pandas` [41]. For our statistical analyses, we used methods from both `SciPy` [38] and `Statsmodels` [34].

When testing for a significant difference between two or more categorical values (e.g., different experience levels), we use χ^2 tests of independence. This test is standard when comparing the values of two or more categorical variables for which the data was collected using independent observations, as in our study design [35]. We consider results significant when p is less than 0.05. Beyond statistical significance, to better understand our χ^2 tests (e.g., which categories are driving the significant result), we analyze standardized residuals as suggested by Sharpe [35]. Standardized residuals are a measure of the difference between what was expected and what was actually observed for any given result in the χ^2 test adjusted for differences in the frequency of observations. Generally, standardized residuals with absolute values of at least two are considered to strongly contribute to the significant result [35], while residuals greater than three or four have a very strong effect. Finally, as a measure of the effect size of our χ^2 statistic, we use Cramer's V (see Akoglu [3, Table 2] for a guide for interpreting Cramer's V).

When testing the significance and magnitude of a relation between two continuous variables (e.g., age or years of programming experience), we use Spearman's rank-order correlation coefficient. We use Spearman's rather than the more common Pearson's correlation coefficient because Spearman's captures all monotonic relationships, not just linear ones. Correlations with $r > 0.3$ are considered significant and of weak effect.

Finally, in this study, we investigate multiple research questions and conduct many statistical tests. Thus, it is necessary to consider and correct for multiple comparisons. For our first and third research question, we defined our questions when designing the study to avoid fishing, and we correct for multiple comparisons for each sub-question using a Benjamini-Hochberg False Discovery Threshold of $q = 0.05$. All significant results reported for these research questions pass this threshold. Our second research question, on the other hand, is an exploratory analysis for factors that may explain our primary result in the first research question. Thus, correction for multiple comparisons is not appropriate for this question.

6 EXPERIMENTAL RESULTS

In this section, we report our experimental results. We organize our analysis of the data around three research questions:

- *RQ1—Post Relevance:* Is there a relationship between general programming experience or language-specific experience and successfully selecting relevant and helpful Stack Overflow posts?
- *RQ2—Browsing Features:* What features of browsing behavior correlate with programming experience?
- *RQ3—Error Type:* Does debugging performance with Stack Overflow differ by Python error type?

6.1 RQ1: Post Relevance

We first investigate whether general or language-specific experience level impacts a programmer's ability to use a search engine to find relevant and helpful information from Stack Overflow for debugging Python errors. Specifically, we investigate if Python-familiar-experts, Non-Python-familiar-experts, or Novices are more or less likely to find helpful debugging information (see Section 4.2 for an overview of our three experience categories).

We investigate both the ability of participants to find an overall relevant Stack Overflow post and also to their ability to identify a helpful snippet within a post (see Section 4.1). To assess the relevance of posts and snippets provided by participants, we use a multi-annotator relevance notion (see Section 5.2).

Primary Result — Posts: We find a significant relationship between experience level and the ability to find a relevant Stack Overflow post for a given error ($\chi^2(2, N = 332) = 6.68$, $p = 0.035$). By examining the standardized residuals for this result (see Table 5), we find that that this significant χ^2 statistic is primarily caused by Python-familiar-experts performing better and Novices performing worse than predicted by the null hypothesis. Specifically, Python-familiar-experts are more likely to find a relevant Stack Overflow post than both Non-Python-familiar-experts and Novices. Python-familiar-experts find a helpful post 60% of the time (99 / 165) compared to 52% (42 / 81) of the time for Non-Python-familiar-experts and only 43% (37 / 86) of the time for Novices. Beyond being statistically significant, this finding is representative of a moderate size effect (Cramer's $V = 0.10$) [3].

Secondary Result — Snippets: We also find a significant relationship of moderate effect size between programming experience and the helpfulness of the specific snippet within the Stack Overflow post that the participant identified to be most useful for debugging the survey prompt ($\chi^2(2, N = 332) = 9.38$, $p = 0.009$, $V = 0.12$). As with overall post helpfulness, this result is driven by Python-familiar-experts and Novices (see the standardized residuals in Table 5). Python-familiar-expert participants are more likely to identify a helpful snippet than both Non-Python-familiar-experts and Novices: Python-familiar-experts find a helpful post 63% of the time (104 / 165) compared to 49% (40 / 81) of the time for Non-Python-familiar-experts and only 44% (38 / 86) of the time for Novices. These results, along with our overall link helpfulness results, are presented in Table 4.

Implications: Our primary result showing that expertise has a significant and substantial effect on Stack Overflow search efficacy is important because it demonstrates that expert programmers are able to find Stack Overflow posts that contain relevant debugging information at a higher rate than novices. More importantly, this confirms that novices can improve their Stack Overflow search skills over time as they gain experience in programming and debugging. This builds upon previous works, which establishes that expert-novice pairs have more successful web search query rates than novice-only pairs [4]. Furthermore, our results show that both general and language-specific programming experience facilitates search efficacy: while Python-familiar-experts performed the best overall, Non-Python-familiar-experts still performed better than Novices for both overall posts and for snippets. This indicates that while language-specific expertise is helpful, even language-agnostic

Table 4: Percent of links and snippets identified by participants that annotators found relevant for debugging the error, broken down by participant experience level.

Experience Level	% Posts Relevant	% Snippets Relevant	Total Observations
Python-familiar-experts	60.0	63.0	165
Non-Python-familiar-experts	51.9	49.4	81
Novices	43.0	44.2	86

Table 5: Standardized residuals for Chi-square tests of independence for post and snippet relevance by participant experience level. Residuals with an absolute value above 2 are highlighted and considered significant, while those with an absolute value above three are bolded and considered very significant factors.

	Post		Snippet	
	True	*False*	*True*	*False*
Python-familiar-experts	2.3	-2.3	**3.0**	**-3.0**
Non-Python-familiar-experts	-0.4	0.4	-1.1	1.1
Novices	-2.3	2.3	-2.3	-2.3

"Google code search" training may still help Novice programmers find online information more explicitly. This possibility that even language-agnostic training my help Novices is further supported by the importance of hypothesis generation to the debugging process [5, 39]: training developers on code search query formation may help developers to both more effectively communicate their debugging hypothesis and also to access the information they need to test their debugging hypotheses more efficiently.

We also note that our results for finding a helpful snippet are very similar to our results for finding an overall post. This indicates that while participants (experts in particular) do sometimes find a helpful snippet taken out of context from an otherwise unhelpful post, observed differences are primarily driven by browsing behavior for finding a relevant Stack Overflow post rather than within-post differences. As a result, pedagogical training and future automated tools may benefit from focusing on finding a relevant post (i.e., searching) rather than selecting a relevant snippet of the post (i.e., page localization). In the next research question, we investigate what browsing behaviors may drive these experience-based differences.

> We find a relationship between both general and language-specific programming experience and locating relevant debugging help on Stack Overflow when using the Google search engine. For example, Python-familiar-experts are 8% more likely than Non-Python-familiar experts and 17% more likely than Novices to identify helpful links ($p = 0.035$), a moderate size effect. This confirms not only that experts are more effective at searching for relevant debugging information than novices, but also that with experience (and potentially training), programmers can improve at debugging searches.

6.2 RQ2: Browsing Behavior and Expertise

Having established that there is a significant relationship between programming experience and effectively searching for debugging help on Stack Overflow, we now conduct an exploratory investigation into factors that may drive this difference. We explore the relationships between experience and search behavior for two phases of the search process: search query construction, and selecting links from the search query results. By identifying features of Stack Overflow browsing behavior that correlate with experience for each of these phases, we can help direct educators on how to teach novices to use online debugging help, and also provide features relevant for designing future automated tools.

Phase 1 — Search Query Construction: We first investigate expertise-related differences in the textual *content* of constructed queries. In particular, we look at differences in query length, if the query contains common Python keywords, and the edit-distance between the query and the original error message.

We found that the length of users' queries, both in terms of number of characters and number of words, is weakly correlated with years of Python experience ($p = 0.021$ and $r = 0.36$ for characters, and $p = 0.015$ and $r = 0.38$ for words). While not quite reaching significance, we also observed a similar trend between both of these features and years of general programming experience ($p = 0.068$ and $r = 0.29$ for characters, $p = 0.072$ and $r = 0.29$ for words). This indicates that experts, both Python-familiar and Non-Python-familiar, are more likely to write longer queries than novices. Furthermore, it encourages automated Stack Overflow mining tools to not avoid longer queries.

One hypothesis for why we observe this correlation is that programmers with language-specific experience may be more familiar with language-specific vocabulary. As an alternate hypothesis, perhaps experts are more likely to directly copy the error message (which can often be quite long) into the search query, a hypothesis supported by the anecdotal observations of the research team. We note there are other hypotheses that could also explain the observed results. However, to test our two primary hypotheses, we investigate if the number of Python-specific vocabulary in the query (e.g., Python keywords[7]), or the query's Levenshtein distance from the error message also correlated with years of programming or Python experience.

Overall, however, we did not find evidence that the semantic content of search queries mediated experience-related differences: for all six additional tests, p was greater than 0.16 and r was less than 0.25. We encourage future work to investigate search query construction and experience more directly. However, these results may indicate that the bulk of experience-related differences in Stack Overflow search are in a different phase of the search process.

Phase 2 — Selecting Search Results: We next analyze the relationship between programming experience and *how* a participant selects Stack Overflow posts from the search results. Specifically, we investigate the ordered display rank of selected Stack Overflow posts on the Google search page and the number of links investigated per search query.

For the former, we perform a χ^2-test of independence between experience level and whether the Stack Overflow link clicked was

[7] https://docs.python.org/3/reference/lexical_analysis.html#keywords

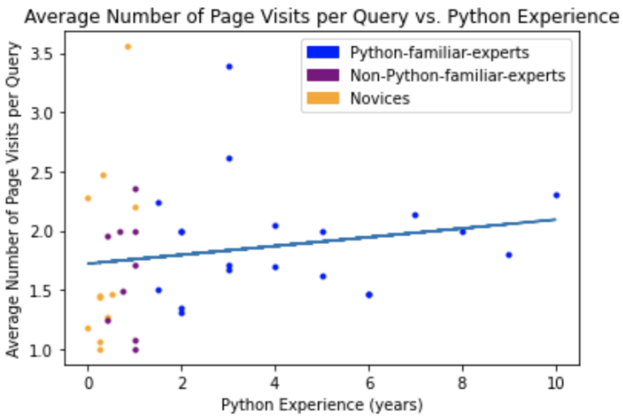

Figure 2: Plot of average number of page visits per unique query over years of Python experience.

the first search result (represented as a Boolean value). We consider all Stack Overflow links clicked, regardless of if the post was the final one reported by the participant. We choose a Boolean representation rather than a correlation with the numerical rank as we are interested in if participants only check the first link in the Google search results or also investigate links further down.

We find a small but significant relationship: compared to both Novices and Non-Python-familiar-experts, Python-familiar-experts are more likely to click on a link other than the first result ($\chi^2(2, N = 1201) = 11.43$, $p = 0.003$, $V = 0.07$). In particular, 59% (374 / 639) of links clicked by Python-familiar-experts had a rank greater than one, compared to 50% (140 / 279) of links clicked by Non-Python-familiar-experts, and 48% (135 / 283) of links clicked by Novices.[8]

Notably, compared to search success rate where language-specific and general experience contributed similar overall gains (see Section 6.1), rank differences are most apparent for Python-familiar-experts: Non-Python-familiar-experts and Novices are more similar. Programmers with language-specific expertise are more likely to scan further down the search result, perhaps as a function of greater familiarity with language-specific vocabulary.

We also find that the number of posts investigated per search query is weakly, but significantly, positively correlated with years of Python experience ($r = 0.32$, $p = 0.045$), as shown in Figure 2. This result is connected with the previous one: Python experts tend to spend more time with each search query by clicking on more links that are further down the page. In a sense, this is more of a "depth first" approach, rather than the "breadth first" approach observed in novices who are more likely to make a search query, look at the first result, and then refine the search query rather than look at another result. As experts have higher success rates for finding relevant information on Stack Overflow, this indicates that explicitly teaching novices to check more posts per search query may help increase their success. Based on our results, we also hypothesize that selecting from the search page (Phase 2) may better explain observed experience-related differences than search query construction (Phase 1). We encourage future research to more directly investigate this hypothesis.

[8]We note that a similar pattern emerges when considering a rank higher than 1, though generally with a slightly smaller χ^2 statistic. For example, Python-familiar-experts are also significantly more likely to click on results with a rank greater than three, also with small effect ($\chi^2(2, N = 1201) = 9.72$, $p = 0.008$, $V = 0.06$).

> In an exploratory analysis, we find evidence that experience-related differences in Stack Overflow search efficacy are connected to differences in how programmers select links from the search result (Phase 2). Experts tend to click more search results than novices. We also observe limited evidence that query construction (Phase 1) may also facilitate some experience-related difference. Our results encourage teachers to focus training on the second phase of the search process.

6.3 RQ3: Error Type Effects

Beyond expertise-related differences in Stack Overflow searching behavior, we also investigate differences associated with Python error types (e.g., `SyntaxError` vs. `TypeError`). Understanding the intersection between Stack Overflow post-relevance and error types may help direct instructors who are teaching students *when* (i.e., in which error contexts) to search for help on Stack Overflow. It may also help scope future generations of automated tools that mine Stack Overflow. To determine whether debugging search outcomes vary by Python error type, we compare how characteristics such as post and snippet relevance, correctness of the fix proposed by the participant, and Stack Overflow link uniqueness varied by the error type of the debugging stimulus.

Post and Snippet Relevance: For our primary result regarding this research question, we analyze if there were significant differences in the ability of participants to find relevant Stack Overflow posts depending on the Python error type of the bug. We conduct a χ-square test of independence between stimulus error type and link relevance, and find evidence that a significant relationship does exist between the error type and whether a participant's Stack Overflow post provided in the survey was relevant to debugging the error ($\chi^2(5, N = 332) = 31.59$, $p < 0.001$). This difference is of strong-moderate effect ($V = 0.14$). Table 6 contains a breakdown of the likelihood of finding a relevant Stack Overflow post by error type. We find that relevant posts are most likely to be found for `IndentationErrors` (73% success rate) and `AttributeErrors` (70% success rate), but least likely to be found for `SyntaxErrors` (only 25% success rate). These results indicate that finding relevant and helpful information on Stack Overflow may be more challenging for general `SyntaxErrors` and less challenging for `IndentationErrors` and `AttributeErrors`, both for programming students and potentially also for automated tools.

Similarly, we also find a significant relationship exists between error type and relevance of the particular snippet the participant identified as the most helpful from the post ($\chi^2(5, N = 332) = 13.37$, $p = 0.02$). As shown in Table 6, we find that our snippet results largely align with the results for the overall post: the three error types with the highest success rate for finding an overall post are also the three highest for finding a relevant snippet. However, the magnitude of the difference is smaller and only of small effect ($V = 0.09$). In particular, we note that while participants were able to find a relevant post for `SyntaxErrors` only 25% of the time, they

Error Type	*Found Relevant Post*				*Found Relevant Snippet*				*Able to Fix the Bug*			
	Yes	No	% Yes	Rank	Yes	No	% Yes	Rank	Yes	No	% Yes	Rank
Syntax Error	14	41	**25.4**	6	30	25	54.5	4	45	10	81.8	2
Type Error	46	47	49.5	5	43	50	46.2	5	59	34	63.4	4
Index Error	28	26	51.9	4	23	31	42.6	6	23	31	42.6	6
Name Error	34	18	65.4	3	34	18	65.4	2	36	16	69.2	3
Attribute Error	31	13	70.4	2	31	13	70.5	1	25	19	56.8	5
Indentation Error	25	9	73.5	1	21	13	61.8	3	30	4	88.2	1

Table 6: Percentage of relevant posts and snippets found, broken down by error type, as well as the percentage of responses for which the participant correctly fixed the bug. For each of these three catagories, the rank column indicates which error type had the lowest (6) and the highest (1) success rate. Cells with over a 70% success rate are underlined and highlighted in green, while cells with a success rate under 30% are bolded and highlighted in red.

were able to identify a relevant snippet 54% of the time. Anecdotally, this difference was driven by participants finding a correct example of the buggy syntax in the question or answer of an otherwise not-relevant Stack Overflow post.[9] This suggests that for many Syntax Errors, resources such as language documentation or tutorials with example code may be more helpful than Stack Overflow posts, and thus a better focus of pedagogical training.

Fix Correctness: We further contextualize our results regarding post and snippet relevance by comparing with the likelihood a participant proposed a correct fix for the bug. As with post and snippet relevance, we find a moderate significant difference between error type and correctly fixing the bug ($\chi^2(5, N = 332) = 28.82$, $p < 0.001$, $V = 0.13$). The full results for our fix correctness analysis can be found in Table 6. However, we emphasize that while the error type rankings for finding a relevant post and finding a relevant snippet were relatively similar, the rankings for fix correctness are less so, especially for `SyntaxErrors` and `AttributeErrors`. For example, while `SyntaxErrors` were the error type with the lowest likelihood for finding a relevant post (25%), they were the second most common for the participant to be able to fix the bug (82%). This may indicate that even though `SyntaxErrors` are challenging to find help for on Stack Overflow, they are still generally easier to fix for participants. This indicates that future automated tools that mine Stack Overflow will likely be more beneficial to users if they focus on error types that are not `SyntaxErrors`.

Stack Overflow Post Uniqueness: Finally, we investigate the uniqueness of Stack Overflow posts for each debugging stimulus to see if participants were more likely to "converge" on a single Stack Overflow post for bugs of a particular error type. In this context, a post is defined to be *unique* if its Stack Overflow link ID, extracted from the URL, appears only once within the set of participant responses for that stimulus.[10] Understanding post uniqueness may be an important factor for directing future automated tools that

Error Type	% Posts Unique	Residual
Syntax Error	75.6	**4.7**
Name Error	60.0	1.7
Index Error	57.1	1.3
Attribute Error	34.1	-1.7
Type Error	33.3	**-3.5**
Indentation Error	30.3	-2.3

Table 7: Analysis of the uniqueness of selected Stack Overflow posts, broken down by Error type. Residual contains the standardized residual from our χ-square test. Residuals with an absolute value above 2 are highlighted and considered significant, while those with an absolute value above three are bolded and considered very significant factors.

mine Stack Overflow: if participants tend to converge on a single helpful post for errors of a given error type, it may be more likely for an automated tool to be constructed to do the same.

We find that there is indeed a significant and strong relationship between error type and post uniqueness ($\chi^2(5, N = 332) = 38.67$, $p < 0.001$, $V = 0.15$). Table 7 contains the percentage of unique links broken down by error type, as well as standardized residuals from our χ-square test (see Section 5.3). This significant difference is largely driven by `SyntaxErrors`, `TypeErrors`, and `IndentationErrors`, all of which have residuals with an absolute value of over 2, indicating they contribute strongly to the significant χ-square statistic. Specifically, 76% of posts selected for `SyntaxErrors` were unique, compared to only 33% and 30% of `TypeErrors` and `IndentationErrors` respectively. Even though the residuals are not quite as large, they still also approach significance for both `NameErrors` and `AttributeErrors`, with 60% of `NameError` posts unique compared to only 34.1% of `AttributeError` posts.

These results indicate that future automated Stack Overflow tool design may want to focus on `TypeErrors`, `IndentationErrors`, and `AttributeErrors` rather than `SyntaxErrors` and `NameErrors`: if humans are more likely to converge on a single helpful post for these error types, it may also be easier for an automated tool mimicking human online search behavior to do the same.

[9] As an example, for one `SyntaxError` caused by not having a closing quote at the end of a string in an `input` call, one participant selected a Stack Overflow post about using f-strings in Python (https://stackoverflow.com/questions/58463606/how-do-you-format-user-input-as-an-f-string-in-python). Even though the given stimulus did not contain an f-string, the snippet selected by the participant coincidentally had code showing matching string quotes in an input call.

[10] Posts marked as duplicate questions on Stack Overflow are treated as their own post in our analysis as long as they have their own URL. Such questions typically have distinct question wording or answers, even if they were later marked as a duplicate of another question by Stack Overflow users.

We find that significant differences in Stack Overflow search outcomes exist between Python error types for the ability to find relevant stack posts ($p < 0.001$), relevant snippets ($p = 0.02$), and correctly fix the bug ($p < 0.001$). We also find that the likelihood of participants to converge on the same Stack Overflow post also differs by error type ($p < 0.001$). Taken together, these results indicate that Stack Overflow may be more helpful for `IndentationErrors`, `TypeErrors`, and `AttributeErrors`, but less helpful for general `SyntaxErrors` and `NameErrors`, information that may help the development of future automated tools.

7 DISCUSSION

In this discussion, we first contextualize our exploratory results regarding search behavior from RQ2. We then discuss the implications of our results, focusing on impacts on pedagogy and on future automated tools.

Contextualization of our exploratory results: In our second research question (see Section 6.2), we focused our analysis on connections between programming experience and behavior during two phases of the search process: search query construction and selecting search results. However, there is an additional third phase of the search process: locating helpful snippet(s) within a selected Stack Overflow post. In this paper, we focus on the first two phases as they are the most understudied in previous work. Also, our results in RQ1 indicate that the majority of experience-related differences are apparent by the time the programmer identifies a relevant post.

However, we also explored differences in the location of helpful snippets ("Phase 3") as doing so admits comparison to previous work [10, 30]. To do so, we coded the location of the helpful snippet provided by the participant (e.g., Answer, Question, Comment, etc.) using our same inter-annotator agreement procedure (see Section 5.2). Table 8 contains our full location information breakdown. As anticipated, we found no significant relationships between programming experience and the location of helpful snippets. However, our results do align and support prior work in that developers are more likely to focus on post text than titles [30], and within that text, they primarily focus on accepted answers [10]. For example, Chatterjee et al. found 42% of the time, information novices highlighted as helpful for debugging in a Stack Overflow post was in the post's Accepted Answer [10]. This is very similar to the 40% of novices in our study who did the same, an observation which helps us gain confidence in the repeatability of our results.

Pedagogical Implications: First and foremost, we provide evidence of how expertise influences Stack Overflow debugging. However, it is an open question if the correlations we observed result from a causal relationship. We encourage future work to investigate if training that induces behaviors similar to those displayed by experts (e.g., clicking more search results, framing longer queries, etc.) can help novices. To the best of our knowledge, however, training online code search is not yet standard in computing curricula. Our results provide evidence that instigating such training may have a noticeable effect: we observed a significant, moderate-sized effect of programming expertise on Stack Overflow search efficacy.

Table 8: Page locations of snippets identified as helpful for debugging, by experience: Python-familiar-experts (PF), Non-Python-familiar-experts (NPF) and Novices.

	PF-Experts	NPF-Experts	Novices
Total Snippets	161	80	75
Post location			
Answer	145 (90%)	72 (90%)	70 (93%)
Question	7 (4%)	4 (5%)	4 (5%)
Comment	7 (4%)	4 (5%)	1 (1%)
Post Title	2 (1%)	0 (0%)	0 (0%)
Additional Answer Breakdown			
Top Answer (Accepted)	73 (45%)	37 (46%)	30 (40%)
Top Answer (Not Accepted)	46 (29%)	19 (24%)	23 (31%)
Accepted Answer (Not Top)	0 (0%)	2 (3%)	0 (0%)
Other Answer	26 (16%)	14 (18%)	17 (23%)

Additionally, there is evidence for general debugging that training can help performance. For example LaToza et al. found that providing developers with explicit debugging strategies improved debugging success [21]. It may be possible to adapt this "explicit stratagy" approach to hep guide novice developers with their debugging query formation. For example, novices could be given explicit instructions to copy parts of the error message or to check multiple results from a single search query. Such an approach could complement the expertise-based mentoring approach proposed by Al-Sammarraie [4].

Furthermore, we provide evidence that general and Python-specific expertise effect search efficacy in different ways, a result which may influence pedagogy. For instance, we found that experience-related differences in the page-rank of selected posts was primarily observed as a function of Python-specific expertise rather than general expertise (see Section 6.2). Thus, different pedagogical strategies may be effective depending on the student's type of programming experience (e.g., explicitly teach about the rank order in which results appear primarily when training students who lack Python-specific experience). We encourage future work to investigate if these expertise-type differences generalize to languages other than Python.

Implications for future automated Stack Overflow tools: Our results also have implications for the design of future automated tools that interface with Stack Overflow. Experience-based variance in search efficacy emphasizes the potential of such tools to provide support to less experienced developers. Furthermore, our results regarding error types can be used to inform which errors to focus on when developing automated information retrieval tools.

In particular, our result that programmers are more likely to find helpful information on Stack Overflow for certain error types (e.g., `AttributeErrors`) and less likely for others (e.g., `SyntaxErrors`) indicates that the same may be true for some automated tools (e.g., those based on mined user data). Future tools that automatically integrate Stack Overflow into a developer's debugging workflow may want to focus on providing help for those error types for which it has the highest chance of providing helpful (rather than potentially distracting) information. This finding is particularly relevant as `SyntaxErrors`, the error type for which finding help on Stack Overflow was the most challenging, are often the errors most approachable by non-online automated tools such as language compilers and error correcting parsers (e.g., [2]). For example, recent versions of Python (e.g., Python 3.10.0[11]) contain updated compiler error messages, including clarifications for `SyntaxErrors`.

Furthermore, our finding that general and language-specific experience effect search success in different ways also has implications for automated tools. In particular, it indicates that it may be beneficial for some future automated tools that mine Stack Overflow to focus on augmenting language-specific or general expertise, rather than both simultaneously; if online search outcomes vary depending on the type of experience, it may follow that such programmers would prefer different types of automated support.

8 THREATS TO VALIDITY

We now discuss some of the threats to validity and limitations of our study, focusing on threats to generalizability, ecological validity, and the reliance on manual annotation.

One potential limitation is generalizability. While we mitigate this threat by recruiting participants from multiple institutions and included participants from a number experience levels, bias may remain. For instance, we recruited novices primarily from two large public university computer science programs. Thus, our population is not a random sample of programming novices (e.g., self-taught, small liberal arts, and high school level novices are not included), and so our results may not generalize to other populations. Additionally, our study used a corpus of 40 relatively small programs, so our results may not generalize to other program types.

A second potential threat is our reliance on manual annotation during our analysis. To retain as much ecological validity as possible, we had developers complete the study using their own computer and web browser. Doing so, however, necessitated using manual annotation for event timestamps and search query contents to prepare the data for analysis. Unfortunately, manual annotation can be prone to error. To mitigate errors due to manual annotation and gain confidence in our results, we had a second annotator randomly check a subset of the annotated data. Furthermore, when using manual annotation to establish values for post and snippet relevance, we used a process involving three independent annotators to establish a grounded consensus (see Sections 5.1 and 5.2).

Furthermore, while we aimed to get close to capturing an ecologically-valid search process (e.g., using a personal computer or using Google as the search engine rather than providing search results), we acknowledge that this may pose a threat to reproducibility: Google's search algorithm is not public, but it may provide different search results at different times or for different users. Additionally, there are still some constraints imposed by the controlled study environment. For instance, as we focus on analyzing the most helpful post found by a given participant, our study structure does not capture situations when information from multiple Stack Overflow posts is used to fill in gaps that one post alone cannot encompass. Similarly, as we focus only on Stack Overflow, we are not able to capture interactions between Stack Overflow and other sources such as API documentation or tutorials. This pattern of combining Stack Overflow posts either with other such posts or with other sources has been observed in previous studies of the platform [43]. Even so, our results still give insight into how individuals of different programming experience levels search for and navigate to helpful Stack Overflow posts.

[11] https://docs.python.org/3/whatsnew/3.10.html#better-error-messages

9 CONCLUSION

Debugging can be a challenging task for programmers at all experience levels. Programmers often use online resources, such as Stack Overflow, for help. However, expert and novice programmers use Stack Overflow in different ways and with varying levels of success. Tools that automatically search for posts to help developers have been explored. However, these tools are generally aimed only at experts, and often are designed without a firm understanding of how programmers with different types of expertise search for and use online debugging information. Such limited knowledge may also make designing and implementing training for finding and using online resources pedagogically challenging.

In this paper, we help fill these knowledge gaps with the results of human study with 40 programmers *investigating differences in Stack Overflow search behavior at three levels of programming experience*: Novices, experienced programmers who are novices in Python, and experienced Python programmers. We observe *significant and substantial differences between all three experience levels*. Python-familiar-experts are able to find helpful debugging information at a higher rate than Non-Python-familiar-experts, and Non-Python-familiar-experts have more success than than Novices programmers. This demonstrates that both general and language-specific expertise impact success. We also conduct an exploratory analysis of search behaviors that correlate this difference. In this analysis, we find that features during the post selection phase of the search (e.g., display rank of the selected post) are generally more predictive of programming experience than features during the query construction or within-post phases. Finally, we find that there are strong significant differences in Stack Overflow search efficacy depending on the error type of the bug. Our findings can both guide teaching of the skill of searching for online debugging help and also inform future automated tools.

ACKNOWLEDGEMENTS

We acknowledge the partial support of the National Science Foundation (CCF 1908633, CCF 1763674) and a Google Faculty Research Award. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NSF or Google. Additionally, we extend our thanks to Ben Cosman and Steve Oney who both kindly agreed to allow us to recruit novice participants from their university courses.

REFERENCES

[1] Marzieh Ahmadzadeh, Dave Elliman, and Colin Higgins. 2005. An analysis of patterns of debugging among novice computer science students. *ACM SIGCSE Bulletin* 37, 3 (2005), 84–88. https://doi.org/10.1145/1151954.1067472
[2] Alfred V. Aho and Thomas G. Peterson. 1972. A Minimum Distance Error-Correcting Parser for Context-Free Languages. *SIAM J. Comput.* 1, 4 (1972), 305–312. https://doi.org/10.1137/0201022
[3] Haldun Akoglu. 2018. User's guide to correlation coefficients. *Turkish journal of emergency medicine* 18, 3 (2018), 91–93.
[4] Mareh Fakhir Al-sammarraie. 2017. *An Empirical Investigation of Collaborative Web Search Tool on Novice's Query Behavior.* Master's thesis. University of

North Florida. https://digitalcommons.unf.edu/cgi/viewcontent.cgi?article=1810&context=etd

[5] Abdulaziz Alaboudi and Thomas D. LaToza. 2020. Using Hypotheses as a Debugging Aid. In *IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2020, Dunedin, New Zealand, August 10-14, 2020*, Michael Homer, Felienne Hermans, Steven L. Tanimoto, and Craig Anslow (Eds.). IEEE, 1–9. https://doi.org/10.1109/VL/HCC50065.2020.9127273

[6] Gina R. Bai, Joshua Kayani, and Kathryn T. Stolee. 2020. How Graduate Computing Students Search When Using an Unfamiliar Programming Language. In *ICPC '20: 28th International Conference on Program Comprehension, Seoul, Republic of Korea, July 13-15, 2020*. ACM, 160–171. https://doi.org/10.1145/3387904.3389274

[7] Brett A. Becker, Paul Denny, Raymond Pettit, Durell Bouchard, Dennis J. Bouvier, Brian Harrington, Amir Kamil, Amey Karkare, Chris McDonald, Peter Michael Osera, Janice L. Pearce, and James Prather. 2019. Compiler error messages considered unhelpful: The landscape of text-based programming error message research. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE* (2019), 177–210. https://doi.org/10.1145/3344429.3372508

[8] Kaibo Cao, Chunyang Chen, Sebastian Baltes, Christoph Treude, and Xiang Chen. 2021. Automated Query Reformulation for Efficient Search based on Query Logs From Stack Overflow. November (2021), 1273–1285. https://doi.org/10.1109/icse43902.2021.00116 arXiv:2102.00826

[9] Joao Castelhano, Isabel C Duarte, Carlos Ferreira, Joao Duraes, Henrique Madeira, and Miguel Castelo-Branco. 2019. The role of the insula in intuitive expert bug detection in computer code: an fMRI study. *Brain imaging and behavior* 13, 3 (2019), 623–637.

[10] Preetha Chatterjee, Minji Kong, and Lori Pollock. 2020. Finding help with programming errors: An exploratory study of novice software engineers' focus in stack overflow posts. *Journal of Systems and Software* 159 (2020), 110454. https://doi.org/10.1016/j.jss.2019.110454

[11] Pierpaolo Dondio and Suha Shaheen. 2019. Is stackoverflow an effective complement to gaining practical knowledge compared to traditional computer science learning? *ACM International Conference Proceeding Series* (2019), 132–138. https://doi.org/10.1145/3369255.3369258

[12] Madeline Endres, Georgios Sakkas, Benjamin Cosman, Ranjit Jhala, and Westley Weimer. 2019. InFix: Automatically Repairing Novice Program Inputs. In *34th IEEE/ACM International Conference on Automated Software Engineering, ASE 2019, San Diego, CA, USA, November 11-15, 2019*. IEEE, 399–410. https://doi.org/10.1109/ASE.2019.00045

[13] Denae Ford, Alisse Harkins, and Chris Parnin. 2017. Someone like me: How does peer parity influence participation of women on stack overflow?. In *2017 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2017, Raleigh, NC, USA, October 11-14, 2017*, Austin Z. Henley, Peter Rogers, and Anita Sarma (Eds.). IEEE Computer Society, 239–243. https://doi.org/10.1109/VLHCC.2017.8103473

[14] Denae Ford, Kristina Lustig, Jeremy Banks, and Chris Parnin. 2018. "We Don't Do That Here": How Collaborative Editing with Mentors Improves Engagement in Social Q&A Communities. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, Montreal, QC, Canada, April 21-26, 2018*, Regan L. Mandryk, Mark Hancock, Mark Perry, and Anna L. Cox (Eds.). ACM, 608. https://doi.org/10.1145/3173574.3174182

[15] Denae Ford, Justin Smith, Philip J. Guo, and Chris Parnin. 2016. Paradise unplugged: identifying barriers for female participation on stack overflow. In *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2016, Seattle, WA, USA, November 13-18, 2016*, Thomas Zimmermann, Jane Cleland-Huang, and Zhendong Su (Eds.). ACM, 846–857. https://doi.org/10.1145/2950290.2950331

[16] David J. Gilmore. 1990. Models of Debugging. In *Proceedings of the 2nd Annual Workshop of the Psychology of Programming Interest Group, PPIG 1990, Wolverhampton, UK, January 4-6, 1990*. Psychology of Programming Interest Group, 8. http://ppig.org/library/paper/models-debugging

[17] Philip Guo. 2021. Ten Million Users and Ten Years Later: Python Tutor's Design Guidelines for Building Scalable and Sustainable Research Software in Academia. In *UIST '21: The 34th Annual ACM Symposium on User Interface Software and Technology, Virtual Event, USA, October 10-14, 2021*, Jeffrey Nichols, Ranjitha Kumar, and Michael Nebeling (Eds.). ACM, 1235–1251. https://doi.org/10.1145/3472749.3474819

[18] Philip J. Guo. 2013. Online python tutor: embeddable web-based program visualization for cs education. In *The 44th ACM Technical Symposium on Computer Science Education, SIGCSE 2013, Denver, CO, USA, March 6-9, 2013*, Tracy Camp, Paul T. Tymann, J. D. Dougherty, and Kris Nagel (Eds.). ACM, 579–584. https://doi.org/10.1145/2445196.2445368

[19] Andre Hora. 2021. Googling for software development: What developers search for and what they find. *Proceedings - 2021 IEEE/ACM 18th International Conference on Mining Software Repositories, MSR 2021* 2 (2021), 317–328. https://doi.org/10.1109/MSR52588.2021.00044

[20] Amy J. Ko, Robert DeLine, and Gina Venolia. 2007. Information needs in collocated software development teams. *Proceedings - International Conference on Software Engineering* (2007), 344–353. https://doi.org/10.1109/ICSE.2007.45

[21] Thomas D. LaToza, Maryam Arab, Dastyni Loksa, and Amy J. Ko. 2020. Explicit programming strategies. *Empir. Softw. Eng.* 25, 4 (2020), 2416–2449. https://doi.org/10.1007/s10664-020-09810-1

[22] Lucas Layman, Madeline Diep, Meiyappan Nagappan, Janice Singer, Robert DeLine, and Gina Venolia. 2013. Debugging Revisited: Toward Understanding the Debugging Needs of Contemporary Software Developers. In *2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, Baltimore, Maryland, USA, October 10-11, 2013*. IEEE Computer Society, 383–392. https://doi.org/10.1109/ESEM.2013.43

[23] Hongwei Li, Zhenchang Xing, Xin Peng, and Wenyun Zhao. 2013. What help do developers seek, when and how? *Proceedings - Working Conference on Reverse Engineering, WCRE* (2013), 142–151. https://doi.org/10.1109/WCRE.2013.6671289

[24] Mingwei Liu, Xin Peng, Qingtao Jiang, Andrian Marcus, Junwen Yang, and Wenyun Zhao. 2018. Searching stackoverflow questions with multi-faceted categorization. In *ACM International Conference Proceeding Series*. Association for Computing Machinery. https://doi.org/10.1145/3275219.3275227

[25] Sonal Mahajan, Negarsadat Abolhassani, and Mukul R. Prasad. 2020. Recommending stack overflow posts for fixing runtime exceptions using failure scenario matching. *ESEC/FSE 2020 - Proceedings of the 28th ACM Joint Meeting European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (2020), 1052–1064. https://doi.org/10.1145/3368089.3409764 arXiv:2009.10174

[26] Lena Mamykina, Bella Manoim, Manas Mittal, George Hripcsak, and Björn Hartmann. 2011. Design lessons from the fastest Q&A site in the west. *Conference on Human Factors in Computing Systems - Proceedings* (2011), 2857–2866. https://doi.org/10.1145/1978942.1979366

[27] Andre N. Meyer, Laura E. Barton, Gail C. Murphy, Thomas Zimmermann, and Thomas Fritz. 2017. The Work Life of Developers: Activities, Switches and Perceived Productivity. *IEEE Transactions on Software Engineering* 43, 12 (2017), 1178–1193. https://doi.org/10.1109/tse.2017.2656886

[28] Laurie Murphy, Gary Lewandowski, Renée McCauley, Beth Simon, Lynda Thomas, and Carol Zander. 2008. Debugging: the good, the bad, and the quirky – a qualitative analysis of novices' strategies. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2008, Portland, OR, USA, March 12-15, 2008*, J. D. Dougherty, Susan H. Rodger, Sue Fitzgerald, and Mark Guzdial (Eds.). ACM, 163–167. https://doi.org/10.1145/1352135.1352191

[29] 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. *IEEE International Conference on Software Maintenance, ICSM* (2012), 25–34. https://doi.org/10.1109/ICSM.2012.6405249

[30] Cole S. Peterson, Natalie M. Halavick, Jonathan A. Saddler, and Bonita Sharif. 2019. A gaze-based exploratory study on the information seeking behavior of developers on stack overflow. *Conference on Human Factors in Computing Systems - Proceedings* (2019), 1–6. https://doi.org/10.1145/3290607.3312801

[31] Luca Ponzanelli, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, and Michele Lanza. 2014. Mining stackoverflow to turn the IDE into a self-confident programming Prompter. *11th Working Conference on Mining Software Repositories, MSR 2014 - Proceedings* (2014), 102–111. https://doi.org/10.1145/2597073.2597077

[32] Md Masudur Rahman, Jed Barson, Sydney Paul, Joshua Kayani, Federico Andrés Lois, Sebastián Fernandez Quezada, Christopher Parnin, Kathryn T. Stolee, and Baishakhi Ray. 2018. Evaluating how developers use general-purpose web-search for code retrieval. *Proceedings - International Conference on Software Engineering* (2018), 465–475. https://doi.org/10.1145/3196398.3196425 arXiv:1803.08612

[33] Caitlin Sadowski, Kathryn T. Stolee, and Sebastian Elbaum. 2015. How developers search for code: A case study. *2015 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2015 - Proceedings* (2015), 191–201. https://doi.org/10.1145/2786805.2786855

[34] Skipper Seabold and Josef Perktold. 2010. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*. SciPy, Austin, TX, US, 92–96.

[35] Donald Sharpe. 2015. Chi-square test is statistically significant: Now what? *Practical Assessment, Research, and Evaluation* 20, 1 (2015), 8.

[36] Emillie Thiselton and Christoph Treude. 2019. Enhancing Python Compiler Error Messages via Stack Overflow. *International Symposium on Empirical Software Engineering and Measurement* 2019-September (2019). https://doi.org/10.1109/ESEM.2019.8870155 arXiv:arXiv:1906.11456v1

[37] Christoph Treude, Ohad Barzilay, and Margaret Anne Storey. 2011. How do programmers ask and answer questions on the web? (NIER track). *Proceedings - International Conference on Software Engineering* (2011), 804–807. https://doi.org/10.1145/1985793.1985907

[38] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero,

Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272. https://doi.org/10.1038/s41592-019-0686-2

[39] Anneliese von Mayrhauser and A Marie Vans. 1997. Program understanding behavior during debugging of large scale software. In *Papers presented at the seventh workshop on Empirical studies of programmers*. 157–179.

[40] Shaowei Wang, David Lo, and Lingxiao Jiang. 2013. An empirical study on developer interactions in StackOverflow. *Proceedings of the ACM Symposium on Applied Computing* March (2013), 1019–1024. https://doi.org/10.1145/2480362.2480557

[41] Wes McKinney. 2010. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman (Eds.). SciPy, Austin, TX, US, 56–61. https://doi.org/10.25080/Majora-92bf1922-00a

[42] Xin Xia, Lingfeng Bao, David Lo, Pavneet Singh Kochhar, Ahmed E. Hassan, and Zhenchang Xing. 2017. What do developers search for on the web? *Empirical Software Engineering* 22, 6 (2017), 3149–3185. https://doi.org/10.1007/s10664-017-9514-4

[43] 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

[44] Di Yang, Aftab Hussain, and Cristina Videira Lopes. 2016. From query to usable code: An analysis of Stack Overflow code snippets. *Proceedings - 13th Working Conference on Mining Software Repositories, MSR 2016* (2016), 391–401. https://doi.org/10.1145/2901739.2901767 arXiv:1605.04464

[45] Xuejiao Zhao, Hongwei Li, Yutian Tang, Dongjing Gao, Lingfeng Bao, and Ching Hung Lee. 2018. A Smart Context-Aware Program Assistant Based on Dynamic Programming Event Modeling. *Proceedings - 29th IEEE International Symposium on Software Reliability Engineering Workshops, ISSREW 2018* (2018), 24–29. https://doi.org/10.1109/ISSREW.2018.00-36